Freescale ZigBee™ Application

User's Guide for ZigBee 2007

Document Number: ZAUGZB2007
Rev. 1.8
07/2011

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006, 2007, 2008, 2009, 2010, 2011. All rights reserved.

Contents

About This Book

The *ZigBee Application User's Guide for ZigBee 2007* explains how to install and run the ZigBee Feature Set example applications that are included in Freescale's BeeKit Wireless Connectivity Toolkit for ZigBee 2007 compliant applications.

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 for ZigBee 2007* to understand what the applications do and how they do it.

Organization

This document is organized into the following chapters.

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	Creating a Smart Energy Network Application - shows how to create a Smart Energy Network consisting of three different applications using the Freescale BeeKit Wireless Connectivity Toolkit.
Chapter 7	Example Applications – provides several examples to allow users to configure and run ZigBee wireless applications.
Chapter 8	Creating a BeeStack BlackBox Binary – shows how to create a BeeStack BlackBox binary using the Freescale BeeKit Wireless Connectivity Toolkit, how to download the image to the target board and how to use the BeeStack BlackBox.
Chapter 9	Creating a ZigBee BlackBox Host Application – shows how to create the ZigBee BlackBox Host Application using the Freescale BeeKit Wireless Connectivity Toolkit.
Chapter 10	Using the Over the Air (OTA) Upgrade Cluster- shows how to use the OTA programming cluster to update a firmware image over the air on an OTA cluster client node from a ZTC-based OTA server node.

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
NN	1322x Network Node, 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
SN	1322x Sensor Node, 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 053474r17, *ZigBee Specification*, ZigBee Alliance, October 2007
3. Document 075123r00, *ZigBee Cluster Library Specification*, ZigBee Alliance, July 2007
4. Document 053520r24, *Home Automation Profile Specification*, ZigBee Alliance, September 2007
5. ZigBee Alliance document 075360
6. The data sheets for the MC13193, MC13203, MC1321x, MC1323x radios
7. Freescale *MC9S08GB/GT Data Sheet*, Document MC9S08GB60, December 2004

Revision History

The following table summarizes revisions to this manual since the previous release (Rev. 1.7).

Revision History

Location	Revisions
Entire document	Updated CodeWarrior steps and screen shots. Added and updated MC1323x 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 and Health care application profiles, supplemented by Accelerometer and Wireless UART example applications. This guide also describes the ZigBee Pro Feature Set that uses the Smart Energy application profile. 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 either Freescale CodeWarrior or IAR Embedded Workbench
- Build the applications in CodeWarrior or Embedded Workbench Integrated Development Environments (IDEs)
- Load the applications into Freescale development boards using either a BDM or JLink pod
- Run the applications on the boards

These applications require the installation of the BeeKit Wireless Connectivity Toolkit, including the BeeStack Codebases, and either CodeWarrior for the HCS08 version 10.1 for HCS08 based development or IAR Embedded Workbench version 5.20 for Codebases supporting only ARM7 MC13224/MC13225 based development or version 5.40 or 5.50 for codebases supporting both ARM7 MC13226 and MC13224/MC13225. They also require a P&E Multilink Background Debug Mode pod for the HCS08 or a JLink JTAG pod for ARM7 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 the IDEs to get the applications loaded to the development boards. These much larger topics are explained in the appropriate reference manuals and ZigBee/802.15.4 specifications.

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 the IDEs
- 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

2.1 HCS08 MCU Development Boards

The following development boards are based on the Freescale HCS08 based ZigBee transceivers:

- Evaluation Board for QE128 MCUs (1320x-QE128EVB)
- Network Coordinator Board (MC1321x-NCB)
- Sensor Reference Board (MC1321x-SRB)
- 1323x-RCM (Remote Controller Motherboard) with a 1323x-MRB (Modular Reference Board) plugged in
- 1323x-REM (Remote Extender Motherboard) with a 1323x-MRB (Modular Reference Board) plugged in

NOTE

The MC1321x and MC1323x based boards have a limited flash memory capacity compared to the 1320x-QE128EVB. Application developers should use the MC1321x and MC1323x boards for ZigBee End Device applications instead of ZigBee Coordinators, Routers, or Combo Devices. See the *BeeStack Application Development Guide* for details about how to reduce application FLASH size.

Table 2-1. Freescale HCS08 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
1323x-RCM + 1323x-MRB	MC13233 (PiP using a MC9S08 QE core)	USB	MCU and IR CMT connector are on the MRB daughter card; LCD with two 16-character rows; 3-axis accelerometer; PWM-driven speaker; matrix remote control-style keyboard; touchpad
1323x-REM + 1323x-MRB	MC13233 (PiP using a MC9S08 QE core)	USB	MCU and IR CMT connector are on the MRB daughter card

2.1.1 HCS08 MCU Board Details

Each of these boards has:

- a Freescale HCS08 MCU:
  - 60KB flash for MC1321x MCUs, 82KB for MC13233, or 128 KB flash for MC9S08QE128
  - 4KB static RAM for MC1321x MCUs, 5KB for MC13233 or up to 8KB for MC9S08QE128
  - Two Serial Communication Interface (SCI) ports for MC1321x boards and 1320x-QE128EVB
    - 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 Communication Interface (SCI1) port for MC1323x boards
  - One Serial Peripheral Interface (SPI) port connected to the ZigBee radio for MC1321x boards and 1320x-QE128EVB
- Four push button switches (S1-S4 for MC1321x boards or SW2-SW5 on 1320x-QE128EVB) on MCU input pins. Pin assignments are not identical across all boards
- Matrix Remote Control Keyboard for MC1323x boards; BeeStack ZigBee applications use only switches SW1-SW4 and the MCU Reset Switch on the 1323x-MRB.
- 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 MC1323x-RCM, MC1323x-REM, 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.

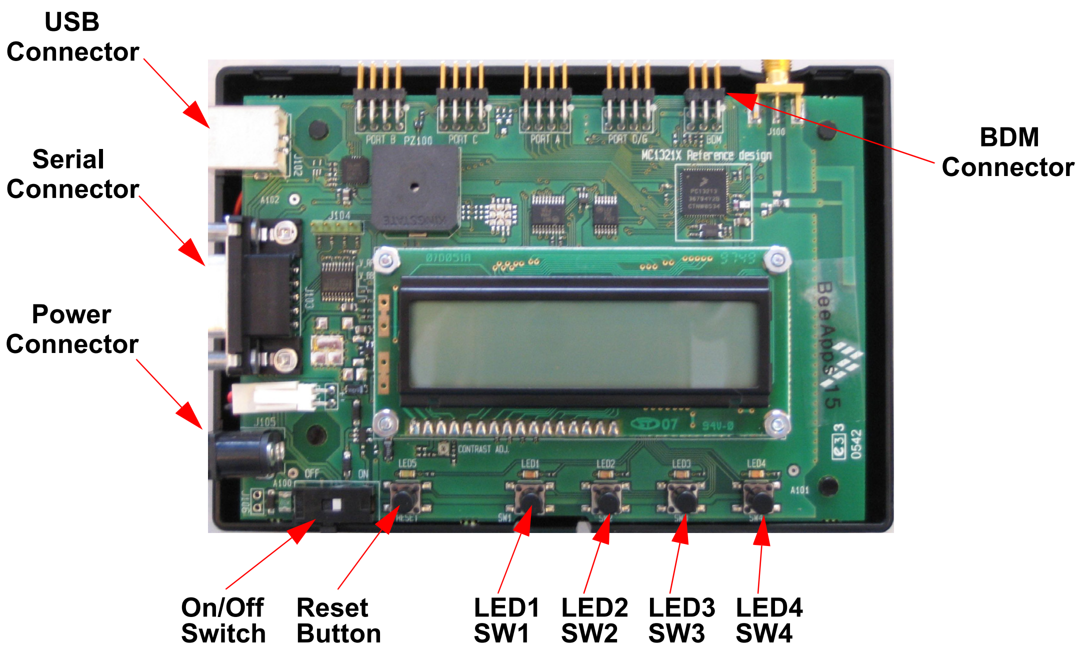


Figure 2-1. MC1321x-NCB

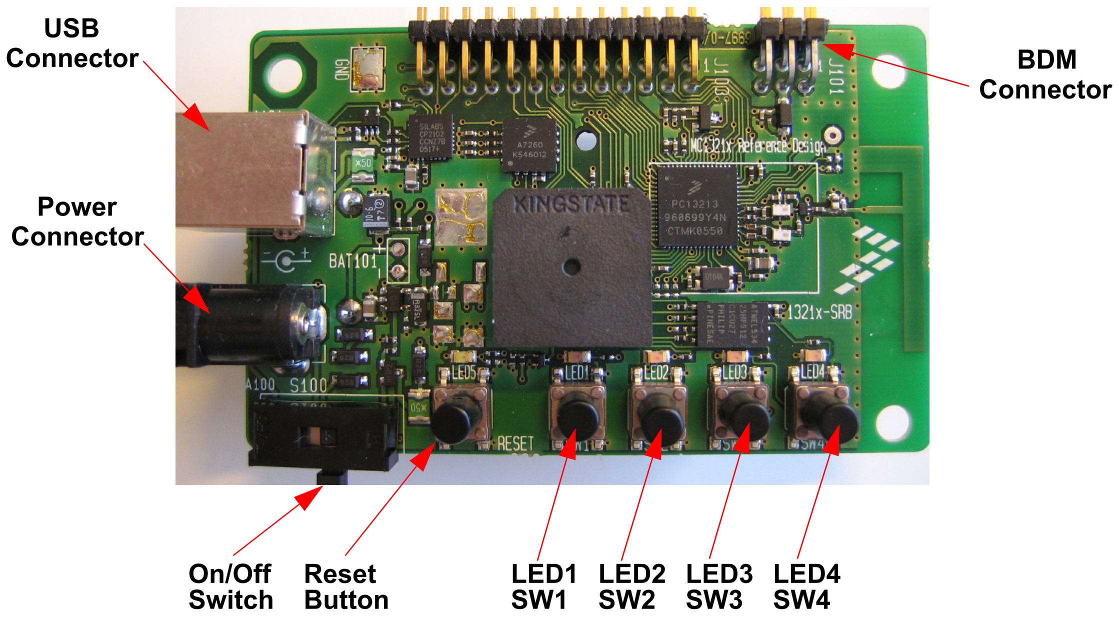


Figure 2-2. SRB Evaluation Board

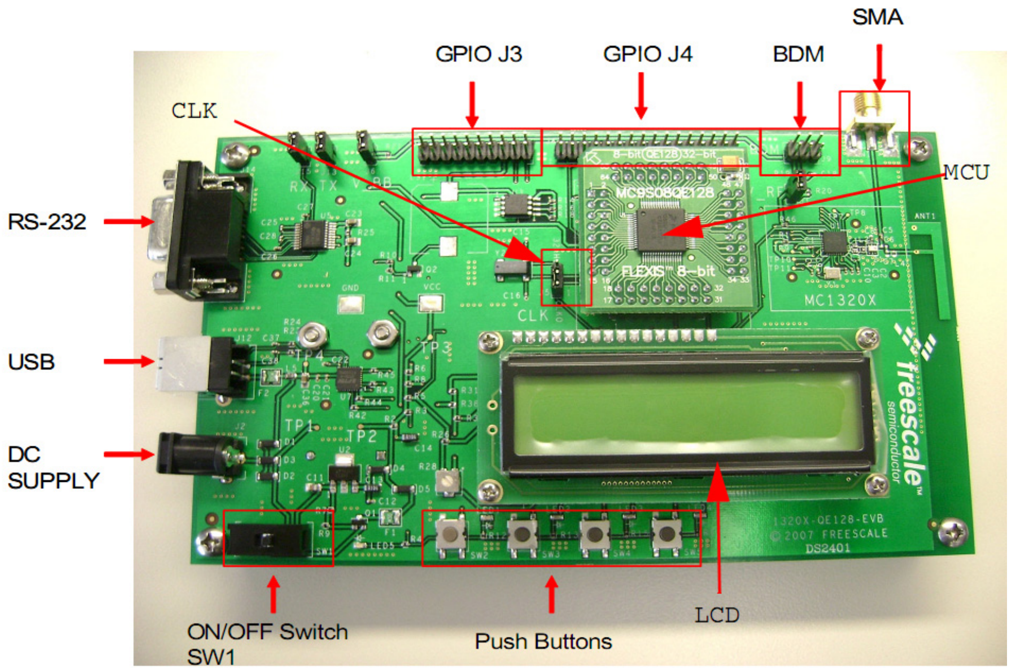


Figure 2-3. 1320x-QE128EVB

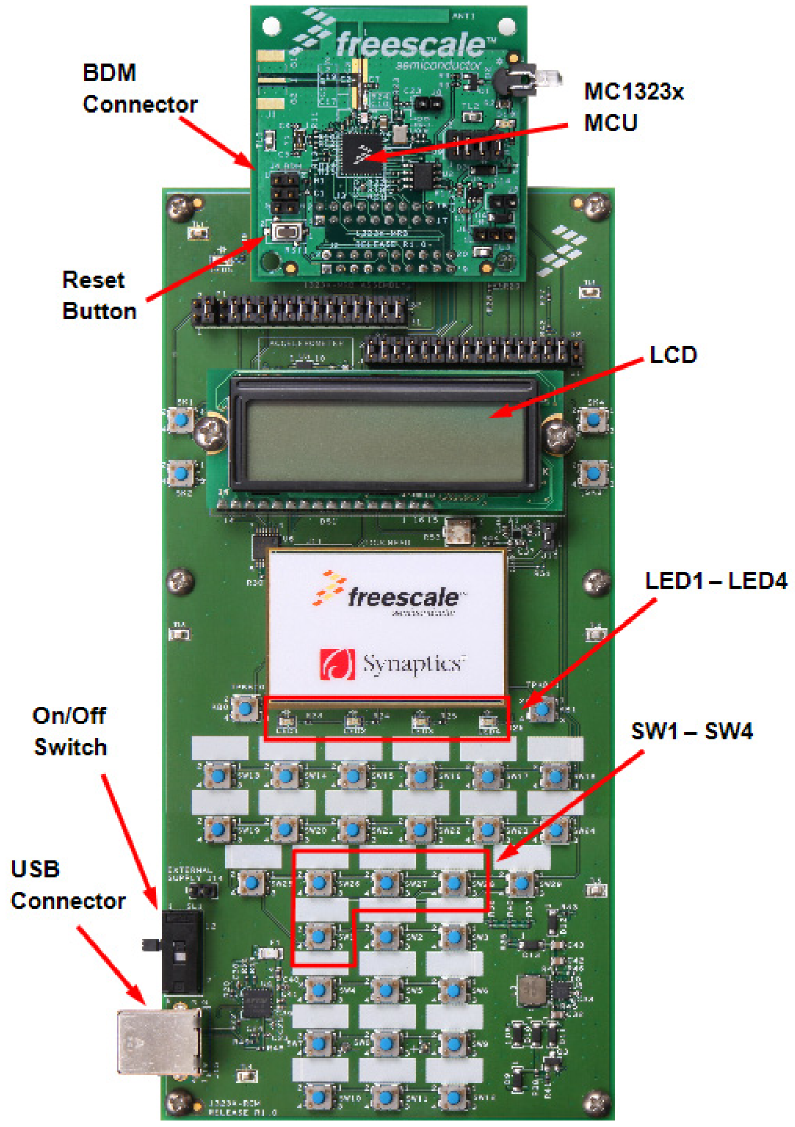


Figure 2-4. 1323x-RCM with 1323x-MRB Plugged In

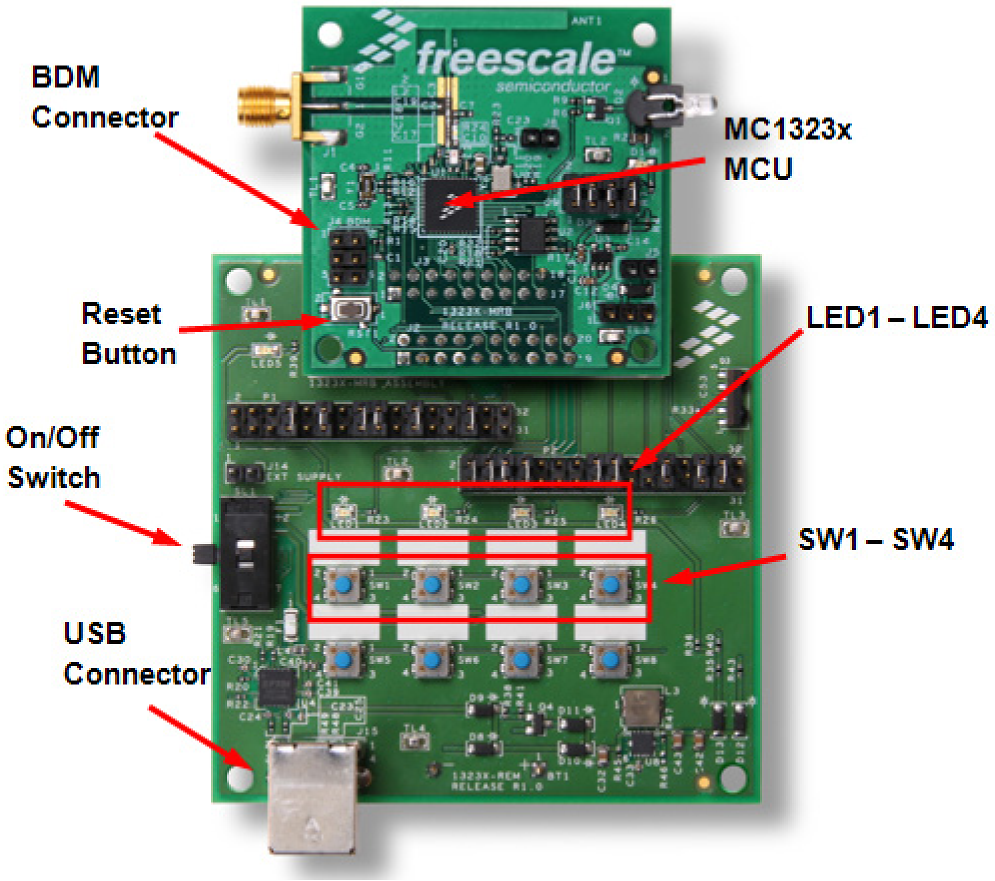


Figure 2-5. 1323x-REM with 1323x-MRB Plugged In

2.2 MC1322x (ARM7) MCU Development Boards

The following development boards are based on the Freescale ARM7 based ZigBee transceivers:

- MC1322x Network Node (MC1322x-NN)
- MC1322x Sensor Node (MC1322x-SN)
- MC13226 Network Node (MC13226-NN)
- MC13226 Sensor Node (MC13226-SN)
- MC1322x Low Power Node (MC1322x-LPN)
- MC1322x USB Dongle (MC1322x-USB)

NOTE

The MC1322x Network Node is available with either the MC13224 or the MC13226. The MC13226 version is identified by exception from the MC13224 version. For the MC13226, the node PCB has a special “13226-NCB” label located between the lower part of the LCD and the push button switch array.

The MC1322x Sensor Node is available with either the MC13224 or the MC13226. The MC13226 version is identified by exception from the MC13224 version. For the MC13226, the node PCB has a special “13226-SRB” label located over the F-antenna on the left side of the PCB.

The MC13226 version of the SRB does NOT contain the XYZ tri-axis accelerometer.

2.3 Available Devices

The MC1322x family is available as two part numbers. These device types differ only in their ROM contents, all other device hardware, performance, and specifications are identical:

- MC13224V - this is the original version and is the generic part type.
  - The MC13224V is intended for most IEEE 802.15.4 applications including MAC-based, ZigBee-2007 Profile 1, and ZigBee RF4CE targets.
  - It has a more complete set of peripheral drivers in ROM.
- MC13226V - this is a more recent version and is provided specifically for ZigBee-2007 Profile 2 (Pro) applications. Only the onboard ROM image has been changed to optimize ROM usage for the ZigBee Pro profile and maximize the amount of available RAM for application use.
  - The IEEE MAC/PHY functionality has been streamlined to include only that functionality required by the ZigBee specification. The MAC functionality is 802.15.4 compatible.
  - For a typical application, up to 20 kbytes more of RAM is available versus the M13224V
  - Some drivers present in the MC13224 ROM have been removed and these include the ADC, LCDfont, and SSI drivers. These drivers are still available as library functions, but now compile into the RAM space.
  - The Low Level Component (LLC) functionality has also been streamlined for the ZigBee specification

NOTE

- When running the Freescale IEEE 802.15.4 MAC (or a related stack) on the MC1322x platform, neither beaconing or GTS are supported.
- See the MC1322x Reference Manual (Document No MC1322xRM), for information on using applications on these devices.

2.3.1 Development Board Features

Table 2-2. Freescale ARM7 Development Boards

Board Number	Device Type	Serial Interface	Board Features
MC1322x-NN	MC13224/MC13226	USB	128x64 graphic monochrome LCD, on-board speaker, 2.5mm audio jack, 4-directional joystick
MC1322x-SN	MC13224/MC13226	USB	on-board speaker, 2.5mm audio jack, 4-directional joystick, pressure sensor, temperature sensor, 3-axis accelerometer (MC13224 only)
MC1322x-USB	MC13224/MC13226	USB	USB dongle form factor
MC1322x-LPN	MC13224	JTAG	Small form factor and low power usage.

The boards have the following common features:

- A Freescale MC1322x MCU:
  - 128KB serial flash
  - 96KB static RAM
  - 80KB ROM
  - Hardware Acceleration for IEEE 802.15.4
- Push button switches: 4 (SW1-SW4) for MC1322x-NN and MC1322x and 2 (SW1, SW2) for MC1322x-LPN
- 1 Reset switch for MC1322-NN, MC1322x-SN and MC1322x-USB and 1 Reset jumper switch for MC1322x-LPN
- 1 (MC1322x-USB), 2 (MC1322x-LPN) or 4 LEDs (MC1322x-NN, MC1322x-SN) for application purposes
- A JTAG connector for general purpose debugging (add-on socket needed for MC1322x-USB).

The boards are shown in the following figures:

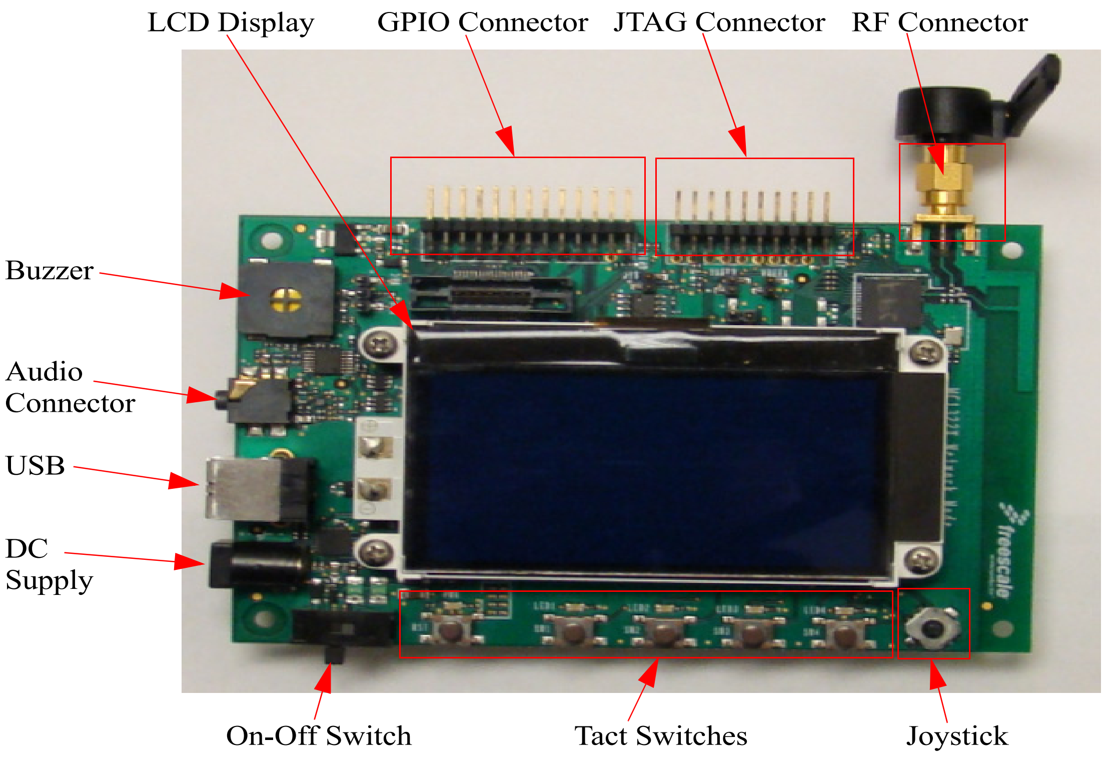


Figure 2-6. MC1322x-Network Node

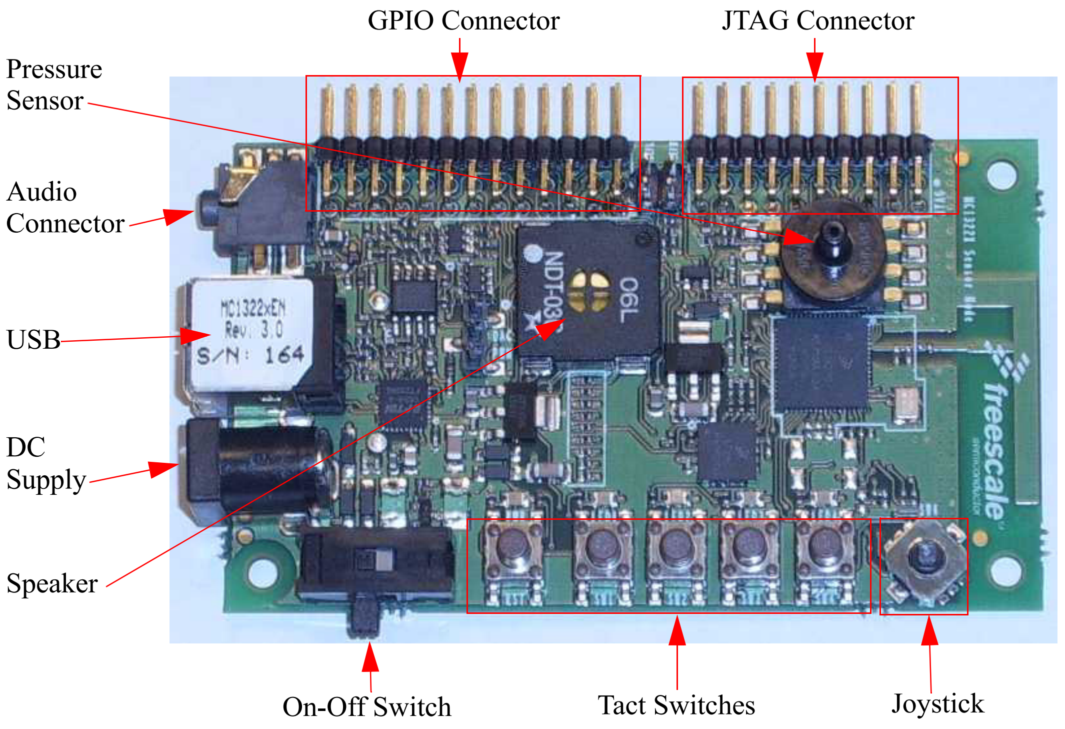


Figure 2-7. MC1322x-Sensor Node

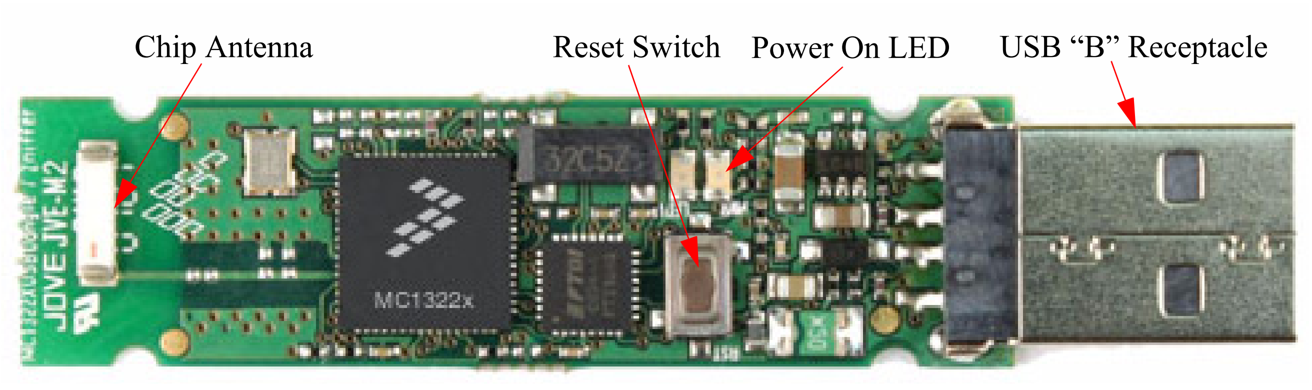


Figure 2-8. MC1322x-USB

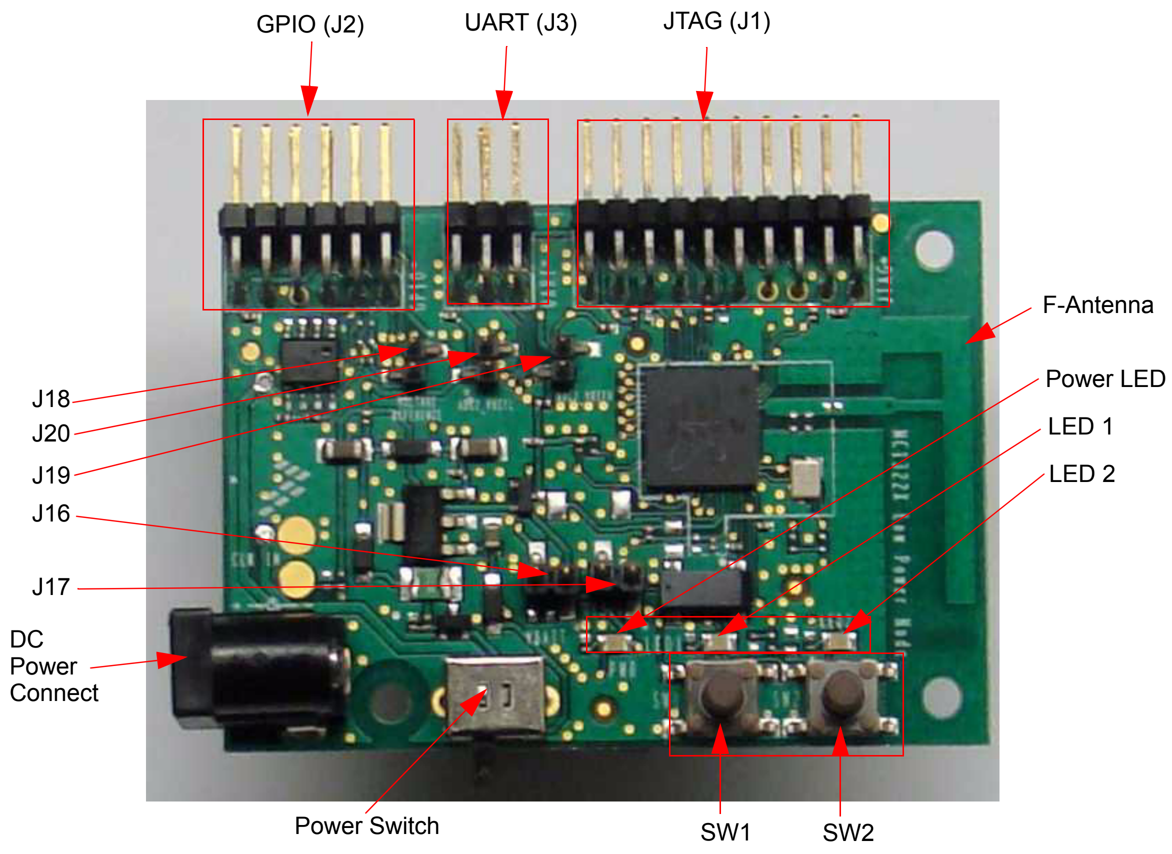


Figure 2-9. MC1322x-Low Power Node

Chapter 3
Using BeeKit to Create BeeStack Applications

The Freescale BeeKit Wireless Connectivity Toolkit is a comprehensive package 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 users have configured a networking project, BeeKit can generate an XML file that can be imported by an IDE such as Freescale's CodeWarrior, used for HCS08 projects, or the IAR Embedded Workbench (IAR EWB) used for ARM7 based projects. Once the IDE has the project contents, users can build the target files and load them into the appropriate Freescale development boards.

3.1 Creating a BeeKit Project

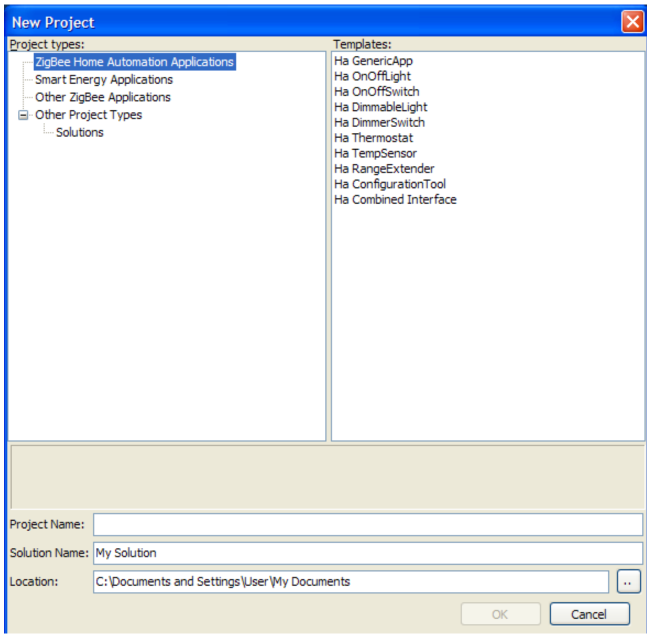
Perform the following tasks to create a Home Automation network project and configure the individual devices.

NOTE

The tasks outlined in this chapter are valid for both the HCS08 and ARM7 based BeeStack Codebases. However, because the ARM7 based BeeStack Codebase also supports the ZigBee Pro Feature Set (Stack Profile 2), configuration options as shown in some the figures, may be different than those of the HCS08 based Codebase.

1. Start BeeKit.
2. If another Codebase (MAC, SMAC or Synkro) is selected, perform the following:
   a) Select File -> Select Codebase… or click the "Select Other Codebase..." link.
   b) Select the BeeStack Codebase version to use from the codebase list.
   c) Click Set Active.
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.



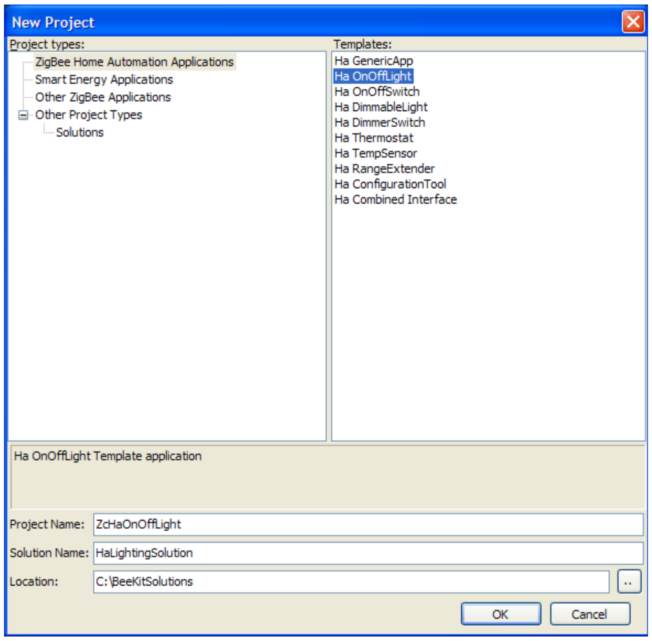
Project Name:
Solution Name: My Solution
Location: C:\Documents and Settings\User\My Documents

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:** ZcHaOnOffLight
   > **Solution Name:** HaLightingSolution
   > **Location:** BeeKitSolutions (sub directory on host PC)



Ha OnOffLight Template application

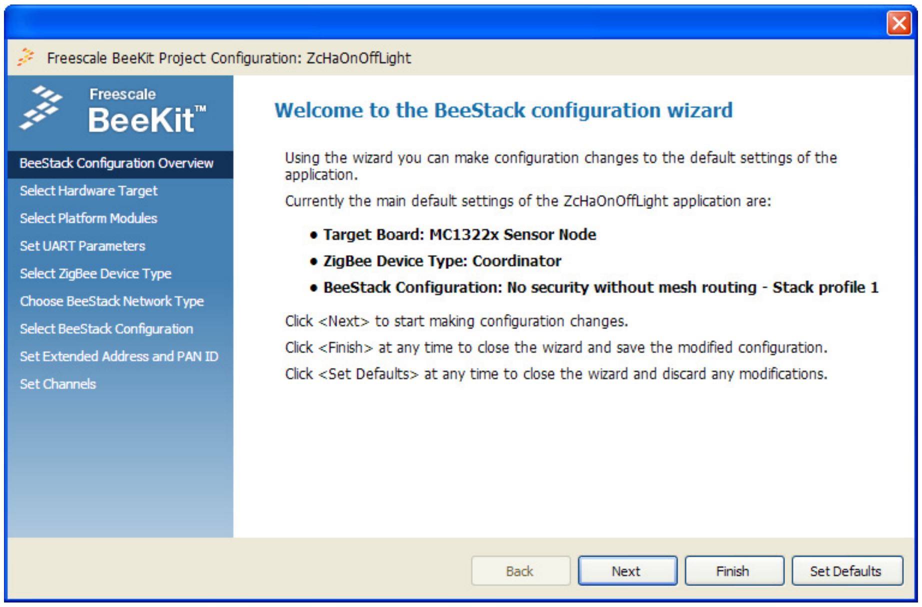
Project Name: ZcHaOnOffLight
Solution Name: HaLightingSolution
Location: C:\BeeKitSolutions

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.


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Welcome to the BeeStack configuration wizard

Using the wizard you can make configuration changes to the default settings of the application.

Currently the main default settings of the ZcHaOnOffLight application are:

- **Target Board: MC1322x Sensor Node**
- **ZigBee Device Type: Coordinator**
- **BeeStack Configuration: No security without mesh routing - Stack profile 1**

Click <Next> to start making configuration changes.

Click <Finish> at any time to close the wizard and save the modified configuration.

Click <Set Defaults> at any time to close the wizard and discard any modifications.

Figure 3-3. BeeKit Project Wizard Welcome Page

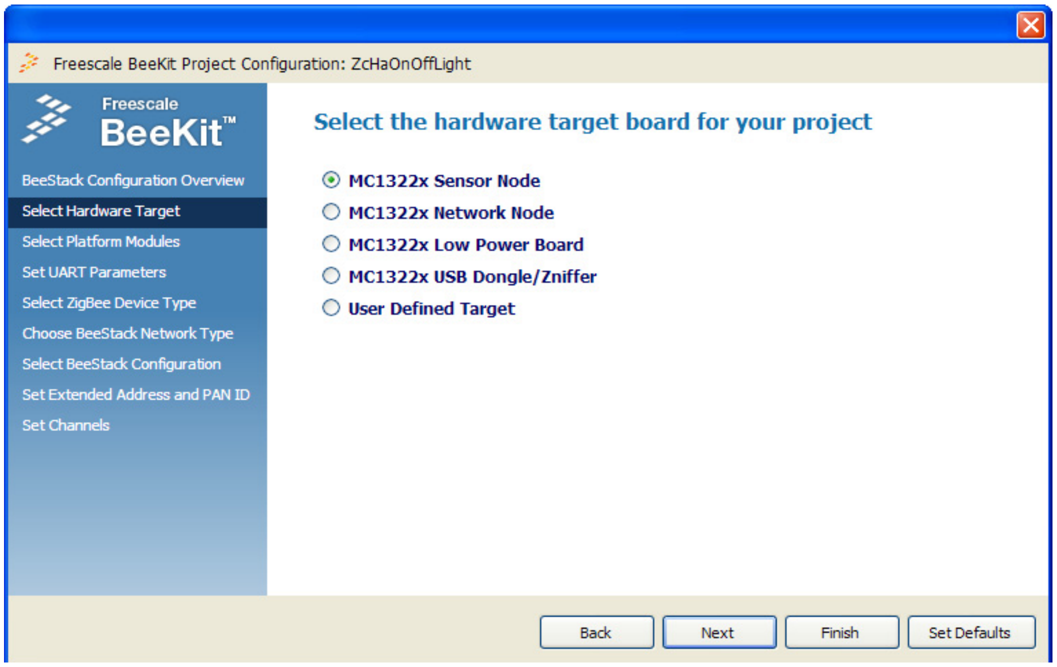
Review the current project settings. In this example, the board is a MC1322x Sensor Node, the ZigBee node type is coordinator, and there is no security or mesh routing enabled. The selected stack profile is Stack profile 1 which is the default setting for HCS08 BeeStack codebase (The default settings depend on codebase platform and the project type).

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

- 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"
- If users wish to discard the modification they have made to the default configuration and close the wizard, they can click the Set Defaults button.

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 (ARM7 codebase version) and Figure 3-5 (HCS08 codebase version).

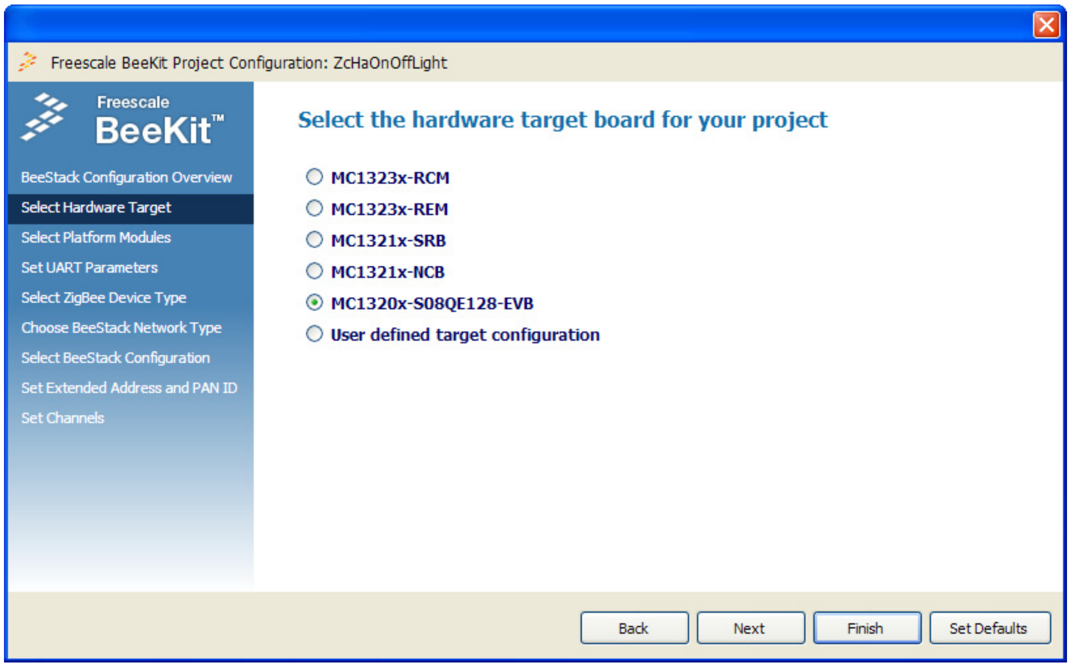

Freescale BeeKit Project Configuration: ZcHaOnOffLight

Select the hardware target board for your project

- (✓) MC1322x Sensor Node
- ( ) MC1322x Network Node
- ( ) MC1322x Low Power Board
- ( ) MC1322x USB Dongle/Zniffer
- ( ) User Defined Target

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Figure 3-4. Hardware Target Page for ARM7 BeeStack Codebase


Freescale BeeKit Project Configuration: ZcHaOnOffLight

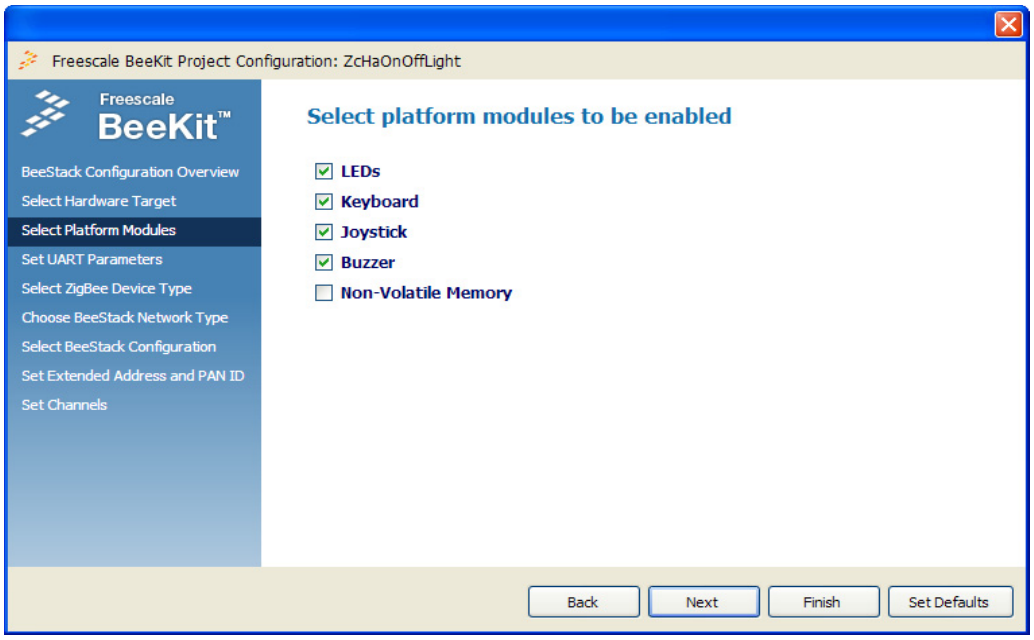
Select the hardware target board for your project

- ( ) MC1323x-RCM
- ( ) MC1323x-REM
- ( ) MC1321x-SRB
- ( ) MC1321x-NCB
- (✓) MC1320x-S08QE128-EVB
- ( ) User defined target configuration

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Figure 3-5. Hardware Target Page for HCS08 BeeStack Codebase

1. Change the device to a specific platform. This example uses the MC1322x Sensor Node for ARM7 codebases and the MC1320x-QE128-EVB for the HCS08 codebases.
2. Select the MC1322x Sensor Node or MC1320x-QE128-EVB radio button and click on the Next button. The Platform Modules page appears as shown in Figure 3-6.

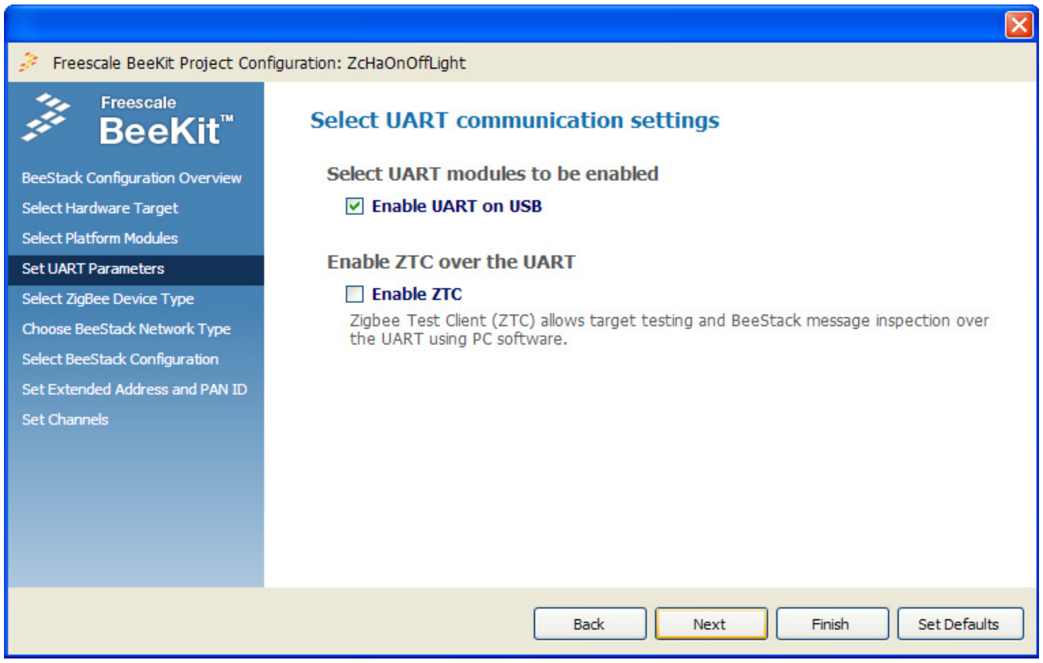

Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select platform modules to be enabled

- [✓] LEDs
- [✓] Keyboard
- [✓] Joystick
- [✓] Buzzer
- [ ] Non-Volatile Memory

Figure 3-6. Platform Modules Page

4. Leave the default platform modules settings unchanged. LEDs and Keyboard modules on the board will be enabled. Click Next to go to the UART Parameters page shown in Figure 3-7.


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select UART communication settings

Select UART modules to be enabled

- [✓] Enable UART on USB

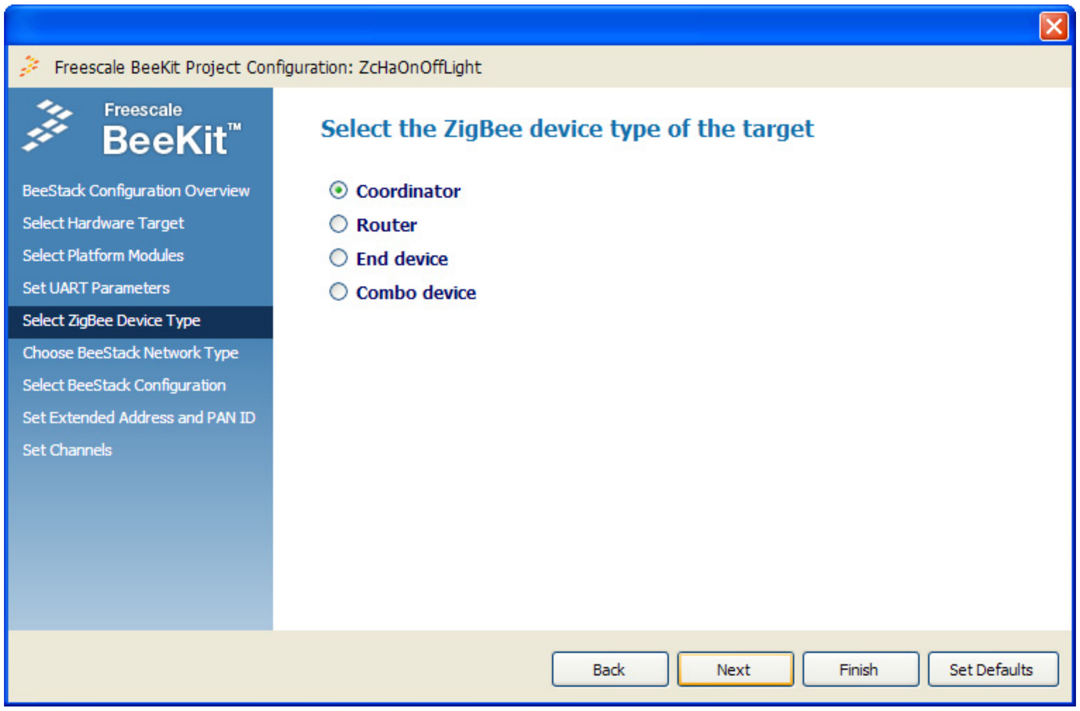
Enable ZTC over the UART

- [ ] Enable ZTC

Zigbee Test Client (ZTC) allows target testing and BeeStack message inspection over the UART using PC software.

Figure 3-7. UART Parameters Page

6. Leave the default UART settings as is. The UART module will be enabled on the USB port of the board and ZTC will be disabled. Click Next to go to the ZigBee Device Type Selection wizard page shown in Figure 3-8.


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select the ZigBee device type of the target

[✓ Coordinator]
Router
End device
Combo device

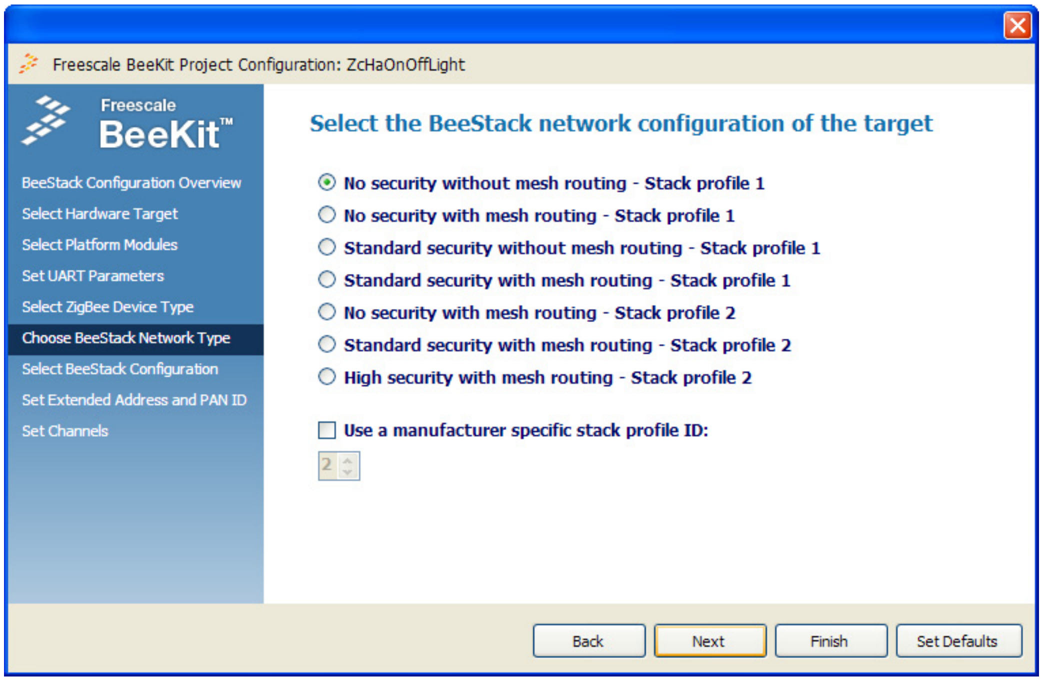
Figure 3-8. ZigBee Device Type Selection Page

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 3-9 (ARM7 codebase version).


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

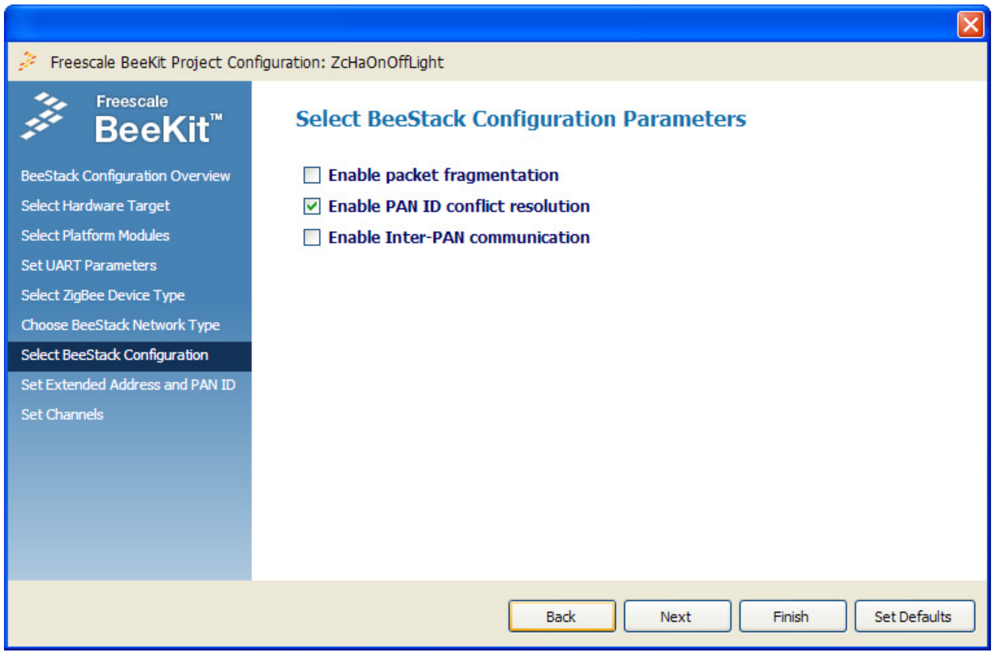
Select the BeeStack network configuration of the target

- [✓ No security without mesh routing - Stack profile 1]
- No security with mesh routing - Stack profile 1
- Standard security without mesh routing - Stack profile 1
- Standard security with mesh routing - Stack profile 1
- No security with mesh routing - Stack profile 2
- Standard security with mesh routing - Stack profile 2
- High security with mesh routing - Stack profile 2

- [ ] Use a manufacturer specific stack profile ID: 2

Figure 3-9. BeeStack Network Type Selection Page Codebases

9. Make sure the “No security without mesh routing - Stack Profile1” option is selected as the BeeStack network configuration.
10. Click Next.
11. The window as shown in Figure 3-10 displays additional BeeStack configuration options.
12. Click Next to keep these default options and move to the next window.

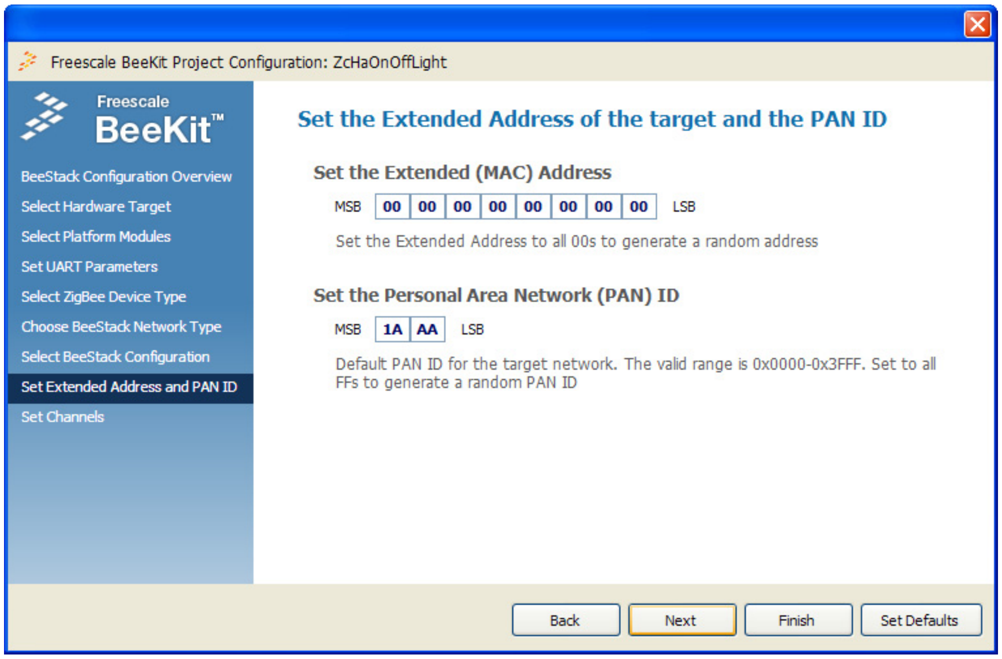

Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select BeeStack Configuration Parameters

- [ ] Enable packet fragmentation
- [x] Enable PAN ID conflict resolution
- [ ] Enable Inter-PAN communication

Figure 3-10. BeeStack Configuration Parameters Codebases

14. The Extended address and PAN ID page appears as shown in Figure 3-11.


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Set the Extended Address of the target and the PAN ID

Set the Extended (MAC) Address

MSB 00 00 00 00 00 00 00 00 LSB

Set the Extended Address to all 00s to generate a random address

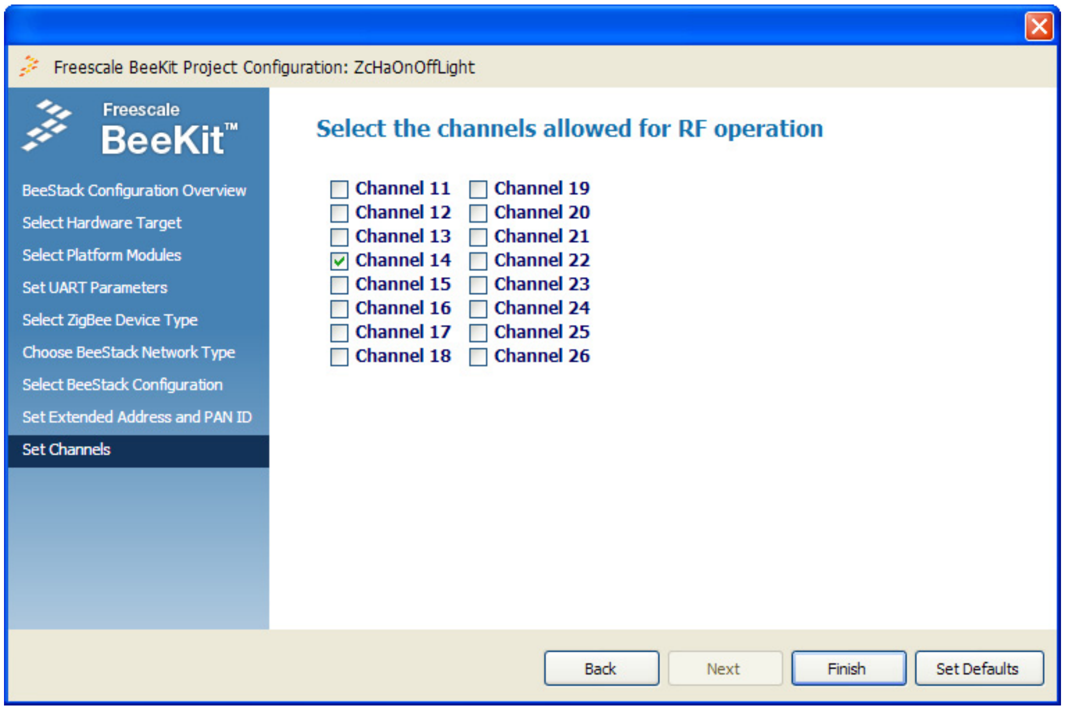
Set the Personal Area Network (PAN) ID

MSB 1A AA LSB

Default PAN ID for the target network. The valid range is 0x0000-0x3FFF. Set to all FFs to generate a random PAN ID

Figure 3-11. Extended Address and PAN ID Selection Page

15. For this example, leave the Extended Address option with all zeros. BeeStack automatically generates 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.
16. Click Next. The Channels page appears as shown in Figure 3-12.


Freescale BeeKit Project Configuration: ZcHaOnOffLight
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select the channels allowed for RF operation

☐ Channel 11	☐ Channel 19
☐ Channel 12	☐ Channel 20
☐ Channel 13	☐ Channel 21
✓ Channel 14	☐ Channel 22
☐ Channel 15	☐ Channel 23
☐ Channel 16	☐ Channel 24
☐ Channel 17	☐ Channel 25
☐ Channel 18	☐ Channel 26

Figure 3-12. 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 BeeStack issue; this is a real-world ZigBee issue. User products must be able to cope with this.

17. 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 3-13.

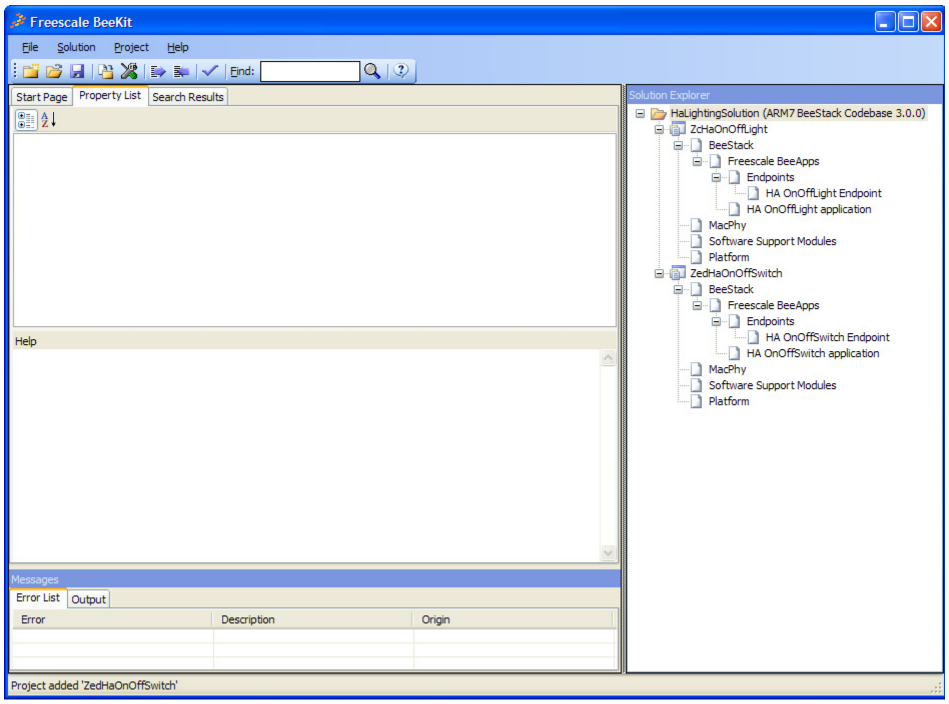


Figure 3-13. BeeKit Main Window

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,

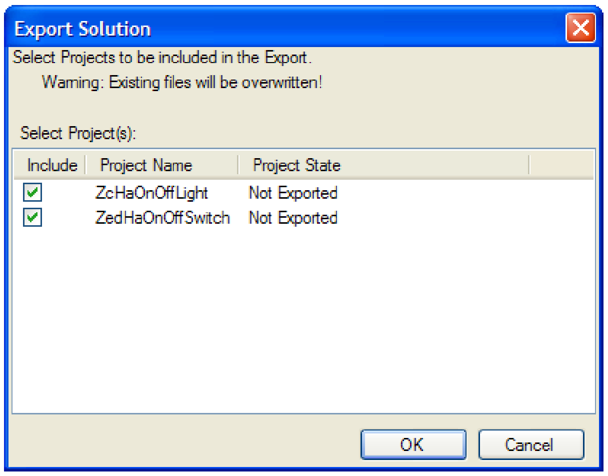
- To change the development board:
  - Select the Platform component in the Explorer window
  - Select the Hardware Target property in the Property List
  - Use the pull-down menu to select the board
- To change the device type:
  - Select the MacPhy component in the Explorer window
  - Select the Device Type property in the Property List
  - Use the pull-down menu to select coordinator, router, or end device
- To change the BeeStack default configuration regarding routing and security:
  - Select the BeeStack component in the Explorer window
  - Select the Stack Library Configuration in the Property List
  - For HCS08 BeeStack codebases use the pull-down menu to select the configuration
  - For ARM7 BeeStack codebases use the BeeStack Configuration Editor by clicking on the [...] button on the right of the property and use the options in the Editor window to change configuration

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 either the IAR Embedded Workbench or 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 3-14.



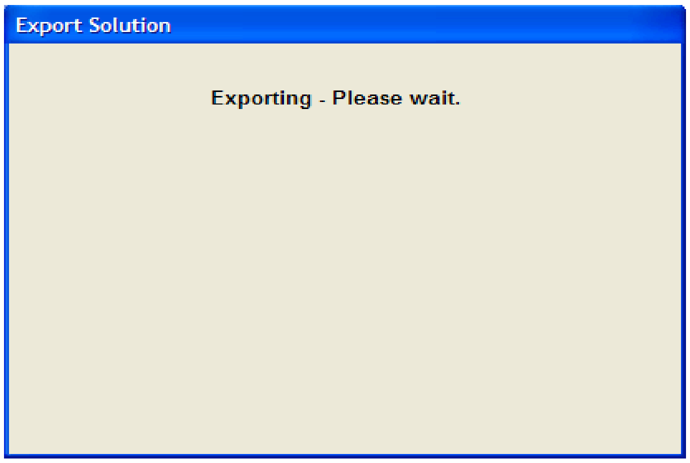
Select Projects to be included in the Export.
Warning: Existing files will be overwritten!

Select Project(s):

Include	Project Name	Project State
✓	ZcHaOnOffLight	Not Exported
✓	ZedHaOnOffSwitch	Not Exported

Figure 3-14. 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 3-15, and the steps it takes scroll by in the Messages window pane.



Exporting - Please wait.

Figure 3-15. 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 Automatically Importing Projects into an IDE

The project files created in BeeKit and saved as Solutions must be imported into CodeWarrior or Embedded Workbench to build the binary output file suitable for programming into each MCU’s FLASH.

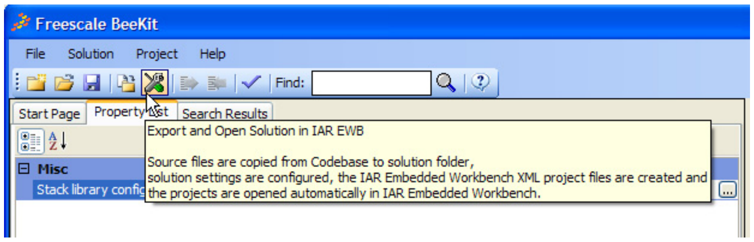
The export and import option is available only for Embedded Workbench.

To export and import the solution into Embedded Workbench automatically perform the following task:

From the BeeKit tool bar, click on Solution -> Export -> Open Solution in IAR EWB

Or

Click the IDE icon in the toolbar as shown in Figure 3-16 and export the solution in IAR EWB.



Export and Open Solution in IAR EWB

Source files are copied from Codebase to solution folder,
solution settings are configured, the IAR Embedded Workbench XML project files are created and the projects are opened automatically in IAR Embedded Workbench.

Figure 3-16. Exporting and Opening the Solution in Embedded Workbench

BeeKit exports the solution as shown in Section 3.1, “Creating a BeeKit Project”, automatically opens the IDE and loads the selected projects that were exported. Figure 3-17 shows the projects of the solution previously created and imported into the IAR EWB.

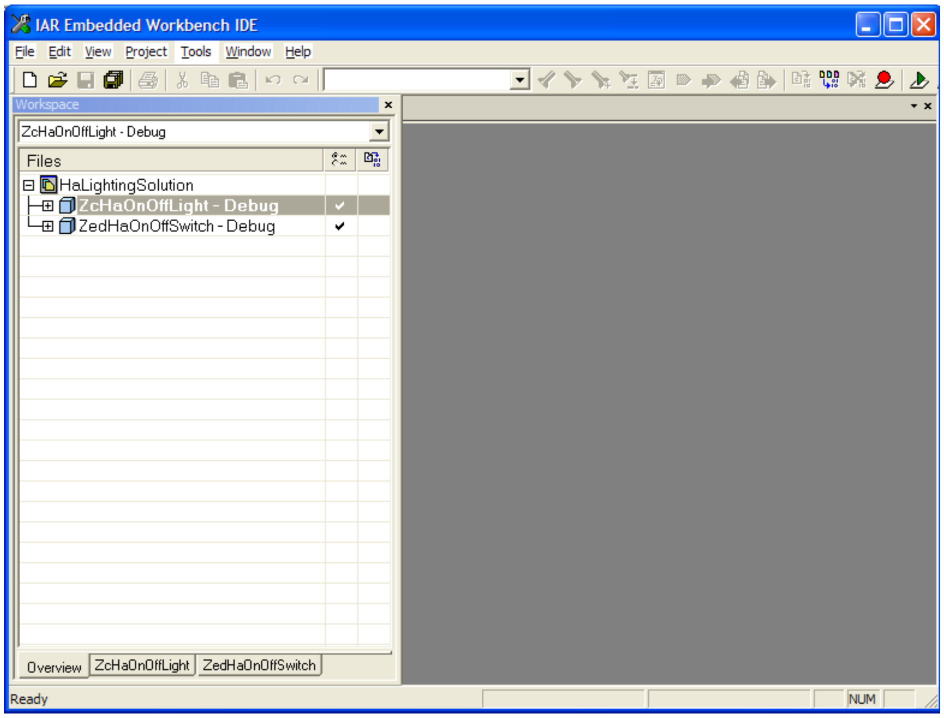


Figure 3-17. Solution Projects Imported in IAR Embedded Workbench

3.3 Manually Importing Projects into an IDE

When BeeKit exports the projects of a solution it creates a `*.xml` file in the exported project folder that can be used to manually import and open the projects in the corresponding IDE (IAR Embedded Workbench or CodeWarrior).

To open the projects manually in IAR Embedded Workbench go to the exported project folder and launch the `*.eww` workspace file created during solution export.

To open the projects manually in CodeWarrior, follow these steps:

1. Start CodeWarrior, which opens to a blank window or to a start page with the menu at the top.
2. Select File -> Import… as shown in Figure 3-18.

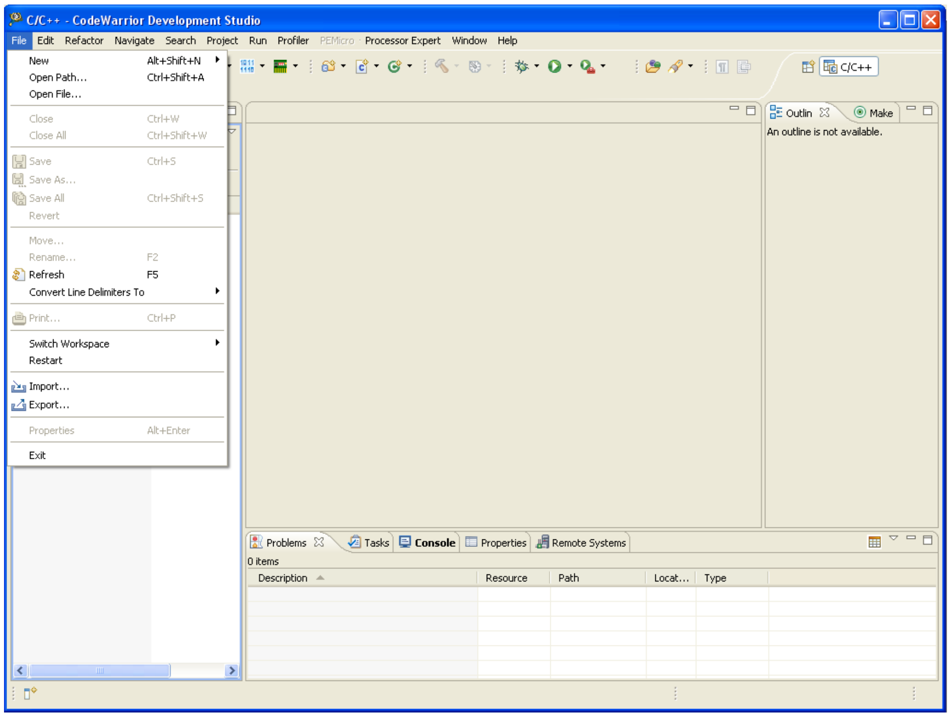
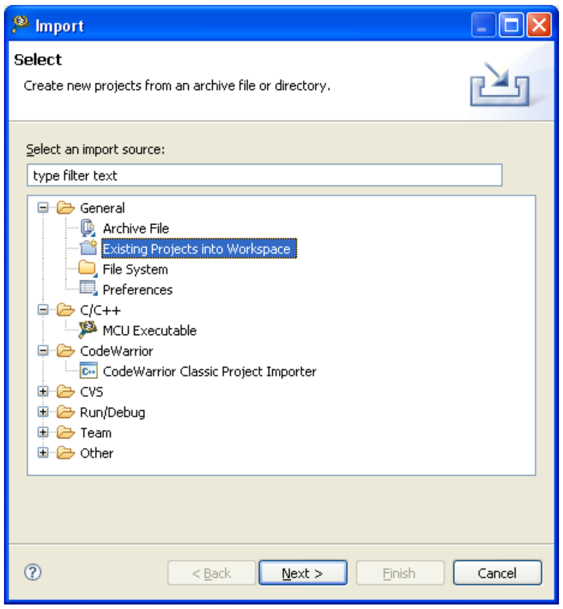


Figure 3-18. CodeWarrior Import Project…

3. Select Existing Projects into Workspace as shown in Figure 3-19.


Import

Select

Create new projects from an archive file or directory.

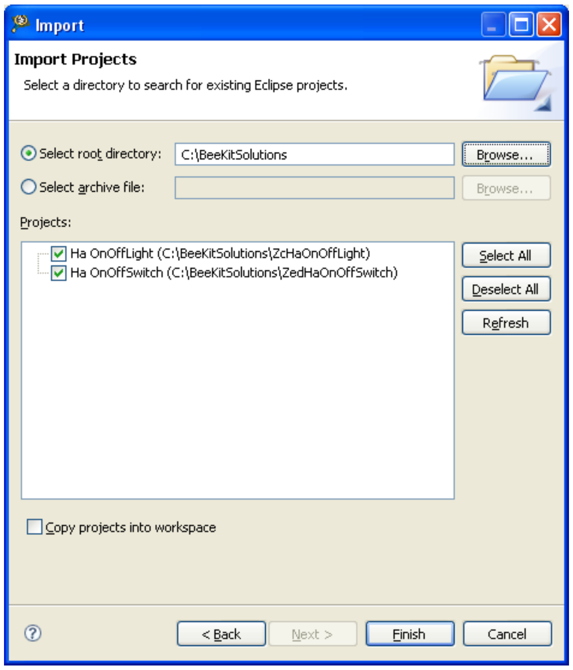
Select an import source: type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
  - MCU Executable
- CodeWarrior
  - CodeWarrior Classic Project Importer
- CVS
- Run/Debug
- Team
- Other

Figure 3-19. CodeWarrior Import Existing Project into Workspace

4. Click Next.
5. Navigate to the directory created in the BeeKit export procedure. This directory contains a directory for each device created earlier in BeeKit. This example uses the directories `ZcHaOnOffLight` and `ZedHaOnOffSwitch`. Users can import both projects simultaneously by selecting the parent directory of `ZcHaOnOffLight` and `ZedHaOnOffSwitch` directories.
6. In this example as shown in Figure 3-20, it is:

`C:\BeeKitSolutions.`


Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] C:\BeeKitSolutions

Select archive file:

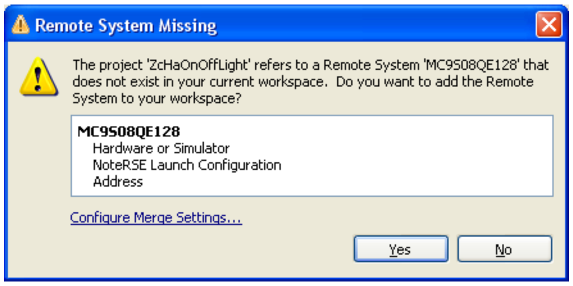
Projects:

- [✓] Ha OnOffLight (C:\BeeKitSolutions\ZcHaOnOffLight)
- [✓] Ha OnOffSwitch (C:\BeeKitSolutions\ZedHaOnOffSwitch)

[ ] Copy projects into workspace

Figure 3-20. CodeWarrior Select Project

7. Click Finish.
8. The first time you import an MC1320x-QE128-EVB project into the current workspace the Remote System Missing window opens and displays a warning as shown in Figure 3-21.


Remote System Missing

The project 'ZcHaOnOffLight' refers to a Remote System 'MC9S08QE128' that does not exist in your current workspace. Do you want to add the Remote System to your workspace?

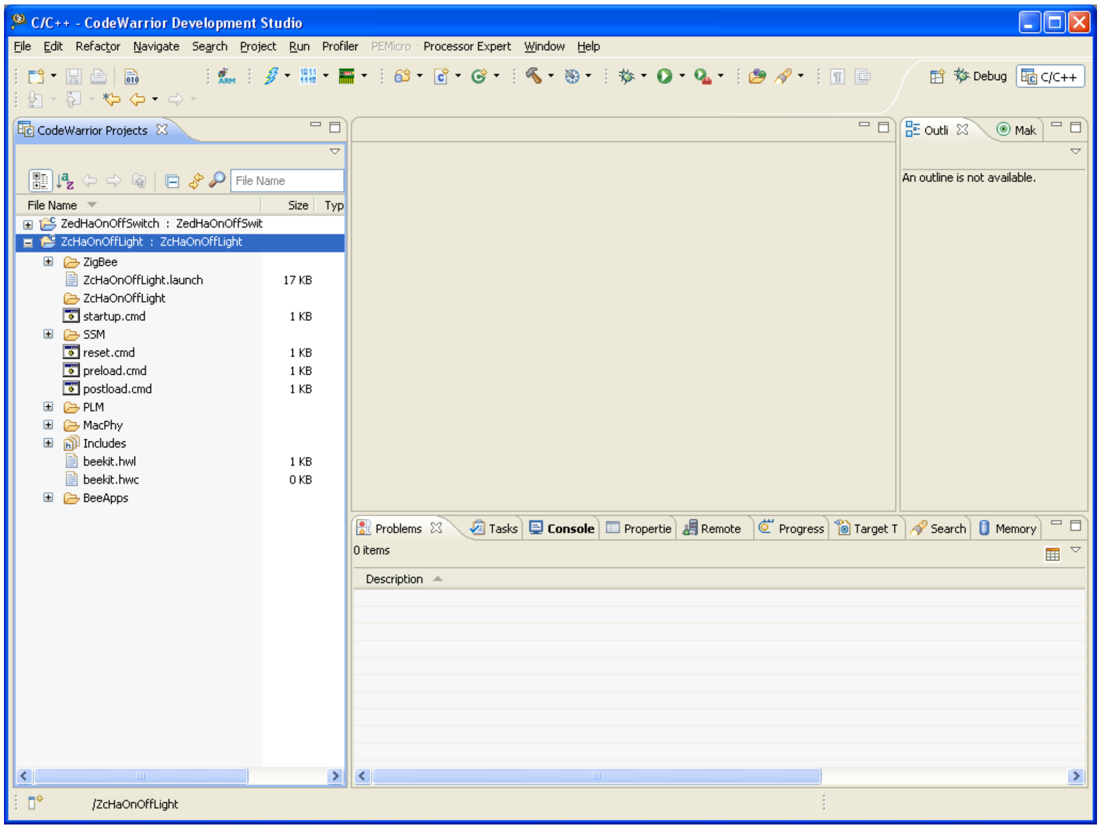
MC9S08QE128
Hardware or Simulator
NoteRSE Launch Configuration
Address

Configure Merge Settings...

Yes No

Figure 3-21. Remote System Missing Warning

9. Click Yes to add the Remote System from the current project to your workspace.
10. CodeWarrior loads the project files as shown in Figure 3-22.


C/C++ - CodeWarrior Development Studio
File Edit Refactor Navigate Search Project Run Profiler PEMicro Processor Expert Window Help
Debug C/C++
CodeWarrior Projects
File Name
An outline is not available.
Problems Tasks Console Propertie Remote Progress Target T Search Memory
0 items
Description
/ZcHaOnOffLight

File Name	Size
ZedHaOnOffSwitch : ZedHaOnOffSwit	
ZcHaOnOffLight : ZcHaOnOffLight	
ZigBee	
ZcHaOnOffLight.launch	17 KB
ZcHaOnOffLight	
startup.cmd	1 KB
SSM	
reset.cmd	1 KB
preload.cmd	1 KB
postload.cmd	1 KB
PLM	
MacPhy	
Includes	
beekit.hwl	1 KB
beekit.hwc	0 KB
BeeApps	

Figure 3-22. CodeWarrior Project Window

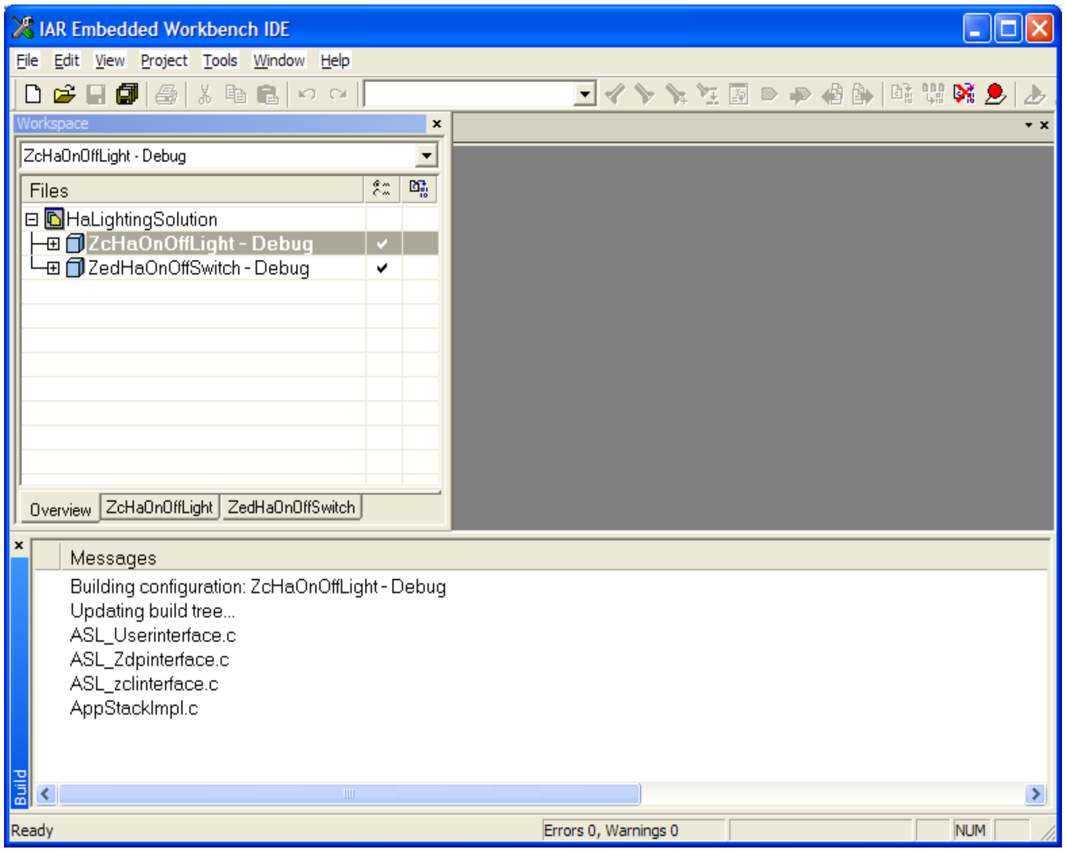
3.4 Building a Code Image Using the IDEs

3.4.1 Building a Code Image Using Embedded Workbench

To build the application binary in IAR Embedded Workbench:

1. In the Workspace panel, select the Overview tab.
2. Right click on the project to build.
3. Select “Set as Active” from the menu.
4. Build the binary in one of three ways:
    a) Press the Make hot key F7
    b) Click the “Make” icon
    c) From the menu select Project -> Make

Embedded Workbench reports the build progress in the Build Messages panel as shown in Figure 3-23.



```
Building configuration: ZcHaOnOffLight - Debug
Updating build tree...
ASL_Userinterface.c
ASL_Zdpinterface.c
ASL_zclinterface.c
AppStackImpl.c
```

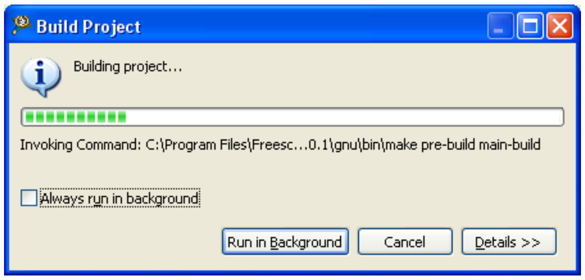
Figure 3-23. IAR EWB Showing the Build Messages Panel

3.4.2 Building a Code Image Using CodeWarrior

Build the application binary in CodeWarrior in one of three ways:

1. Click the “Build” icon (Looks like a hammer.)
2. From the menu select Project -> Build Project
3. Right click on the project -> Build Project

CodeWarrior reports the build progress in a window it opens and then closes as shown in Figure 3-24.



Building project...

Invoking Command: C:\Program Files\Freesc...0.1\gnu\bin\make pre-build main-build

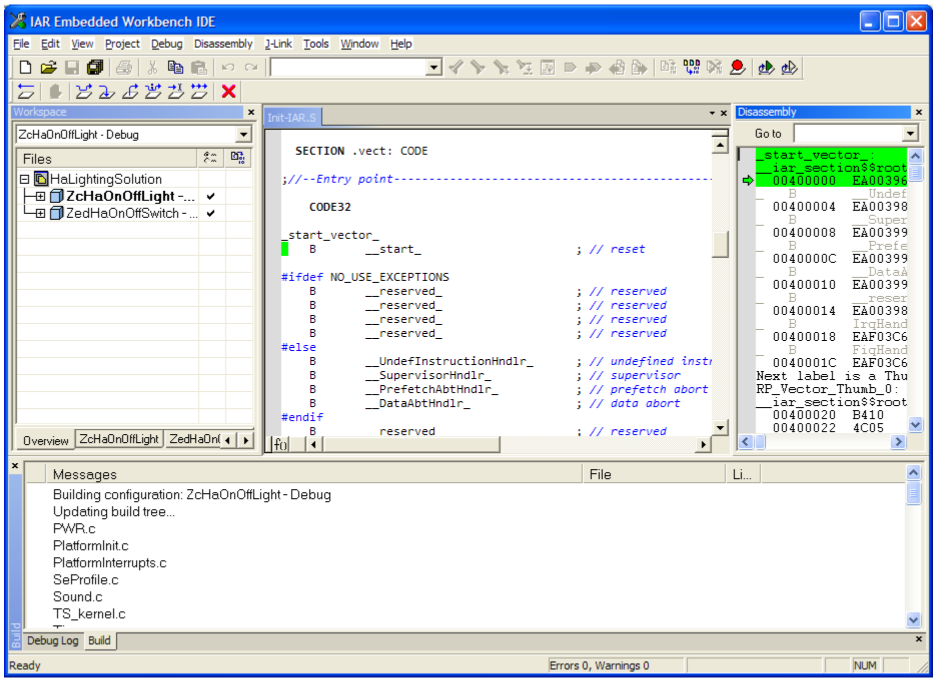
Figure 3-24. CodeWarrior Build Window

3.5 Loading the Code Image into a ZigBee Device

3.5.1 Loading the Code Image into a MC1322x Board via JLink

Perform the following steps to load the code image into a MC1322x, ARM7 based evaluation board from within Embedded Workbench using the JLink JTAG debugger pod.

1. Connect the JLink pod to the computer using a USB cable.
2. Turn on the MC1322x evaluation board.
3. Connect the JLink ribbon cable to the JTAG pins on the evaluation board. Align pin 1 of the JTAG port which is marked with a white dot with the blue ribbon wire of the JLink.
4. Download the compiled image to the board in one of three ways:
   a) Press the Debug hot key Ctrl+D.
   b) From the Project menu select Download and Debug (Embedded Workbench 5.20 (no later) or Debug (previous versions).
   c) Click the Debug icon (a green triangle icon) in the main toolbar.
5. Wait until the board’s FLASH memory is written. When this is complete, the IAR EWB debugging window appears as shown in Figure 3-25.



```
  SECTION .vect: CODE

;//--Entry point-----------------------------------------

    CODE32

_start_vector_
    B       __start_                    ; // reset

#ifdef NO_USE_EXCEPTIONS
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
#else
    B       __UndefInstructionHndlr_    ; // undefined instr
    B       __SupervisorHndlr_          ; // supervisor
    B       __PrefetchAbtHndlr_         ; // prefetch abort
    B       __DataAbtHndlr_             ; // data abort
#endif
    B         reserved                  ; // reserved
```

```
_start_vector_:
__iar_section$$root
   00400000  EA00396
_    B        __Undef
   00400004  EA00398
_    B        __Super
   00400008  EA00399
_    B        __Prefe
   0040000C  EA00399
_    B        __DataA
   00400010  EA00399
_    B        __reser
   00400014  EA00398
_    B        IrqHand
   00400018  EAF03C6
_    B        FiqHand
   0040001C  EAF03C6
Next label is a Thu
RP_Vector_Thumb_0:
__iar_section$$root
   00400020  B410
   00400022  4C05
```

Messages	File	Li...
Building configuration: ZcHaOnOffLight - Debug		
Updating build tree...		
PWR.c		
PlatformInit.c		
PlatformInterrupts.c		
SeProfile.c		
Sound.c		
TS_kernel.c		

Figure 3-25. IAR EWB Debugging Window

7. Stop the debugger by pressing Ctrl+Shift+D or by choosing “Stop Debugging” from the Debug menu.
8. Right click on the HaOnOffSwitch application in the Workspace Overview panel and select “Set as Active” from the context menu.
9. Repeat the steps as shown in Section 3.5.1, “Loading the Code Image into a MC1322x Board via JLink” to load the code for the other application into another board.

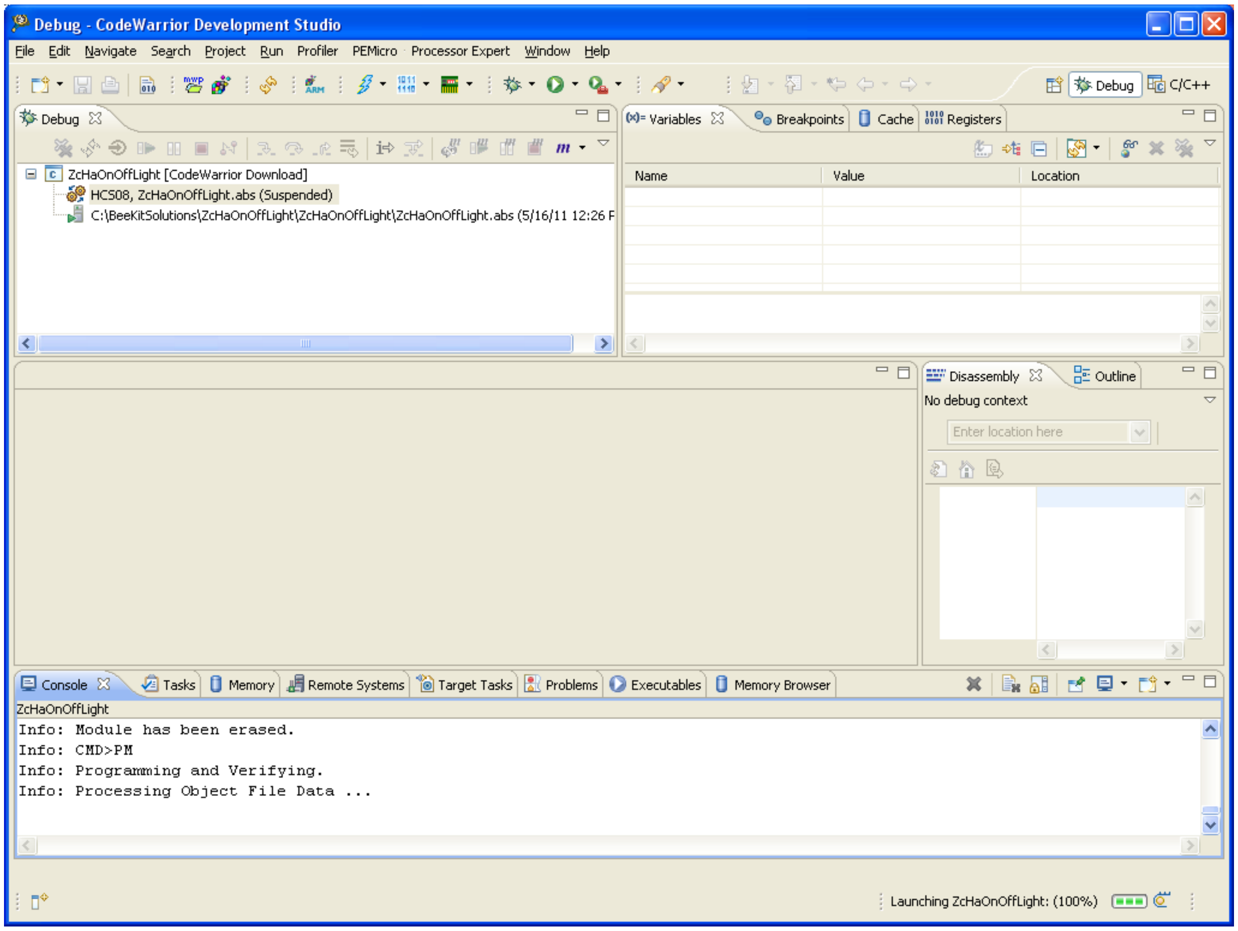
3.5.2 Loading the Code Image into a HCS08 Board via P&E BDM

To load the code image into a MC1323x/MC1321x/MC1320x HCS08 based evaluation board using the P&E BDM pod perform the following steps:

1. 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.
2. Connect the BDM pod to the device. Align pin 1 of the BDM port connector with the red wire of the flat cable connector
   — 1321x-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
   — 1323x-MRB: The connector is labeled J4 BDM and pin 1 is marked with ‘1’
3. Turn on the board.
4. 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.
5. Select the project for which you what to download the image.

6. Download the compiled image to the board in one of three ways:
   a) Press the Debug hot key F11
   b) Click the “Debug” icon (Looks like a bug.)
   c) From the menu select Run -> Debug

CodeWarrior switches to the Debug perspective as shown in Figure 3-26.


Debug - CodeWarrior Development Studio
File Edit Navigate Search Project Run Profiler PEMicro Processor Expert Window Help
Debug C/C++
Debug
ZcHaOnOffLight [CodeWarrior Download]
HCS08, ZcHaOnOffLight.abs (Suspended)
C:\BeeKitSolutions\ZcHaOnOffLight\ZcHaOnOffLight\ZcHaOnOffLight.abs (5/16/11 12:26 F
Variables Breakpoints Cache Registers
Name Value Location
Disassembly Outline
No debug context
Enter location here
Console Tasks Memory Remote Systems Target Tasks Problems Executables Memory Browser
ZcHaOnOffLight
Launching ZcHaOnOffLight: (100%)

```
Info: Module has been erased.
Info: CMD>PM
Info: Programming and Verifying.
Info: Processing Object File Data ...
```

Figure 3-26. CodeWarrior Debug Perspective

7. Close the debugger by selecting Run -> Terminate.

WARNING

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

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

Referring to switches SW1-SW4 in this chapter refers to the four push-buttons present on most of the Freescale development boards. For example, on the 1320x-QE128EVB, the four buttons are actually labeled SW2-SW5.

If using the MC1322x-LPN board, only two buttons and two leds are available. Restrictions and changes related to this board are mentioned in the corresponding steps if needed. See Section 2.1.1, “HCS08 MCU Board Details” for more information.

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 - not available if a MC1322x-LPN board is used for one of the nodes).

   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.

If using the MC1322x-LPN board, press SW2 instead of SW3 and monitor LED2 instead of LED3.

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.

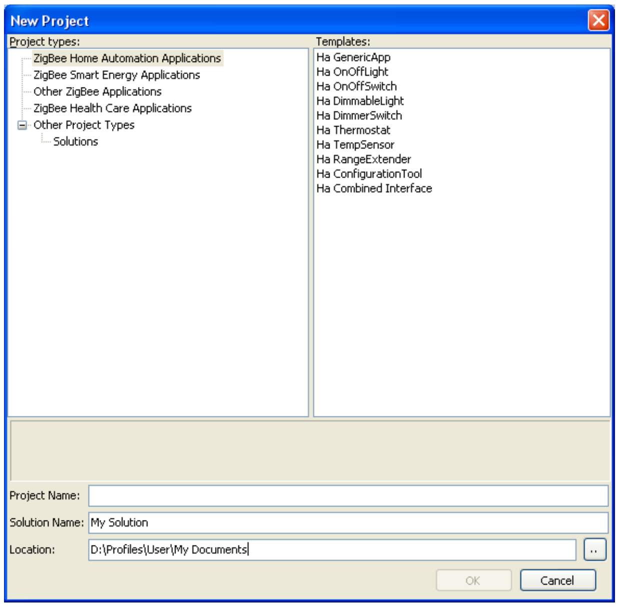
The Wireless UART runs in a fully buffered interrupt driven mode and is well suited for large file transfers. However, the example used in this chapter only sends a small amount of text to show Wireless UART basic functionality.

The following scenario is based on using a HCS08 BeeStack codebase. Exceptions are noted where there are differences when using an ARM7 BeeStack codebase.

5.1 Creating the Wireless UART BeeKit Project

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

1. Start the BeeKit Wireless Connectivity Toolkit.
2. If another Codebase (MAC, SMAC, BeeStack Consumer (RF4CE) or SynkroRF) 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, select a BeeStack Codebase version.
   c) Click the Set Active 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.



New Project

Project types:
- ZigBee Home Automation Applications
- ZigBee Smart Energy Applications
- Other ZigBee Applications
- ZigBee Health Care Applications
- Other Project Types
  - Solutions

Templates:
- Ha GenericApp
- Ha OnOffLight
- Ha OnOffSwitch
- Ha DimmableLight
- Ha DimmerSwitch
- Ha Thermostat
- Ha TempSensor
- Ha RangeExtender
- Ha ConfigurationTool
- Ha Combined Interface

Project Name:
Solution Name: My Solution
Location: D:\Profiles\User\My Documents

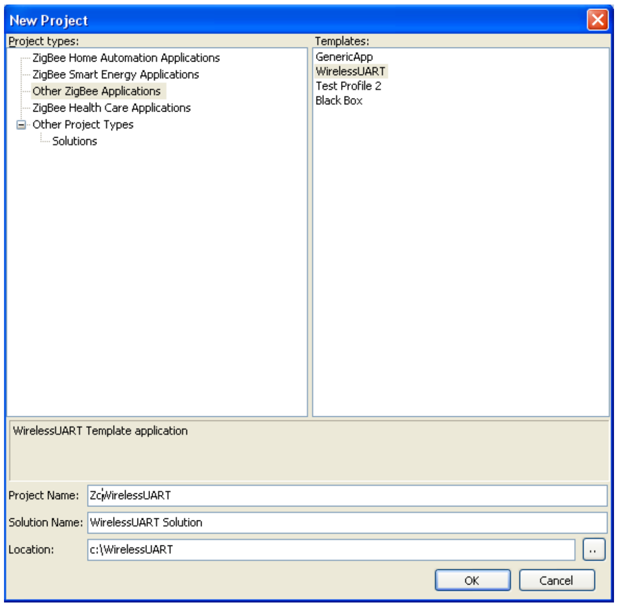
Figure 5-1. BeeKit New Project Window

4. If using the HCS08 codebase, select the Other ZigBee 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: ZcWirelessUART

   Solution Name: WirelessUART Solution

   Location: c:\WirelessUART (or other sub directory on host PC)



New Project

Project types:
- ZigBee Home Automation Applications
- ZigBee Smart Energy Applications
- Other ZigBee Applications
- ZigBee Health Care Applications
- Other Project Types
  - Solutions

Templates:
- GenericApp
- WirelessUART
- Test Profile 2
- Black Box

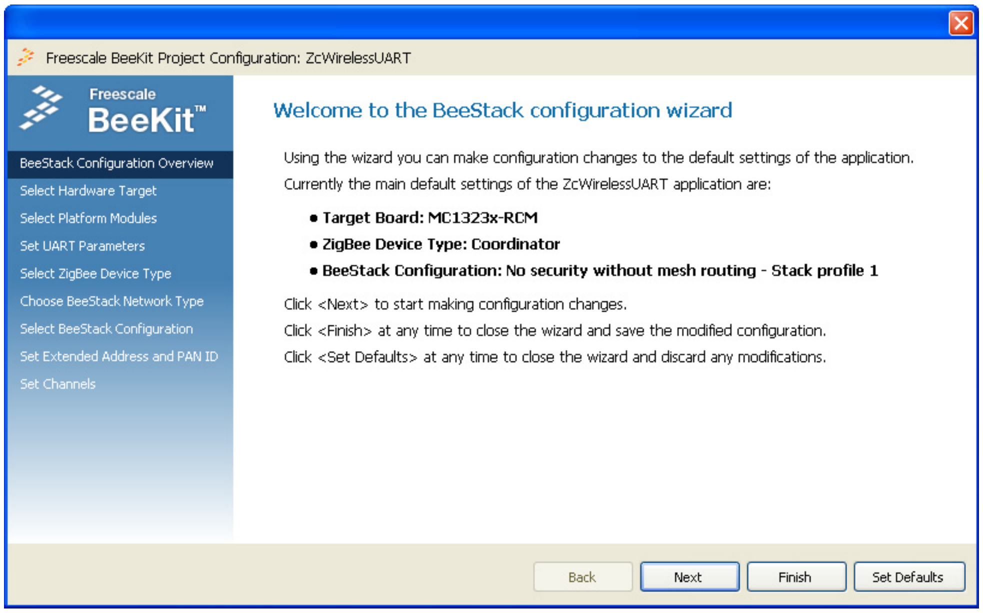
WirelessUART Template application

Project Name: ZcWirelessUART
Solution Name: WirelessUART Solution
Location: c:\WirelessUART

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.


Freescale BeeKit Project Configuration: ZcWirelessUART

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Welcome to the BeeStack configuration wizard

Using the wizard you can make configuration changes to the default settings of the application.
Currently the main default settings of the ZcWirelessUART application are:

- **Target Board: MC1323x-RCM**
- **ZigBee Device Type: Coordinator**
- **BeeStack Configuration: No security without mesh routing - Stack profile 1**

Click <Next> to start making configuration changes.
Click <Finish> at any time to close the wizard and save the modified configuration.
Click <Set Defaults> at any time to close the wizard and discard any modifications.

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 a MC1320x-S08QE128-EVB, 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”

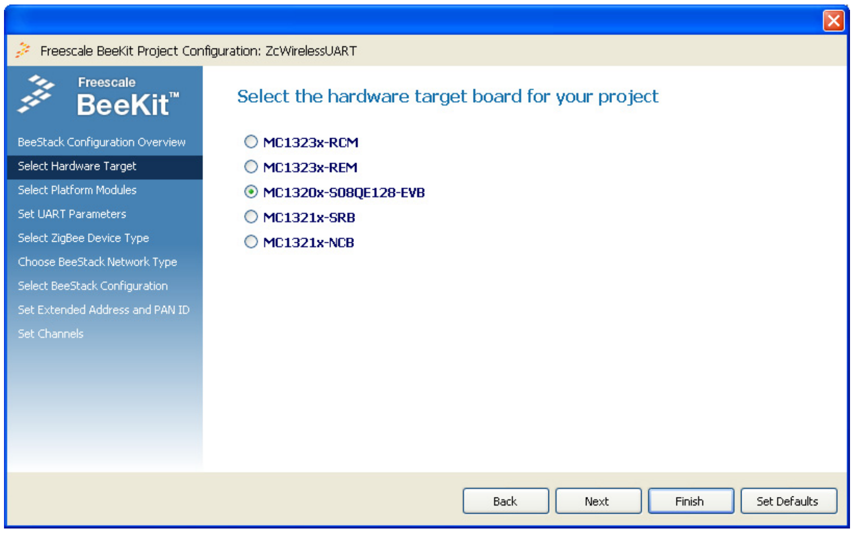
5.1.1 Custom Configuration Options

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.

NOTE

This example uses the MC1320x-S08QE128-EVB for the HCS08 codebase. If using an ARM7 codebase, the equivalent choice is the MC1322x Sensor Node.

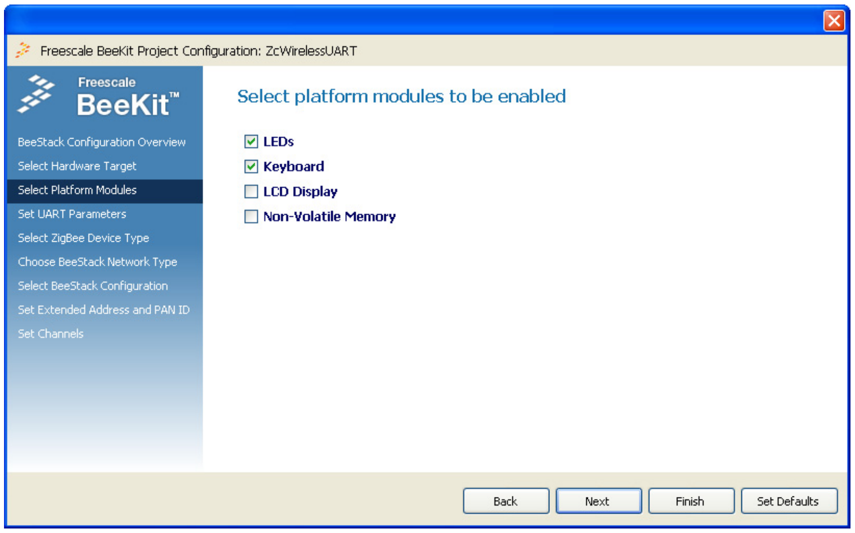

Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select the hardware target board for your project

- MC1323x-RCM
- MC1323x-REM
- [✓ MC1320x-S08QE128-EVB]
- MC1321x-SRB
- MC1321x-NCB

Figure 5-4. Hardware Target Page

2. Select the MC1320x-S08QE128-EVB option and click on the Next button. The Select Platform Modules window appears as shown in Figure 5-5.

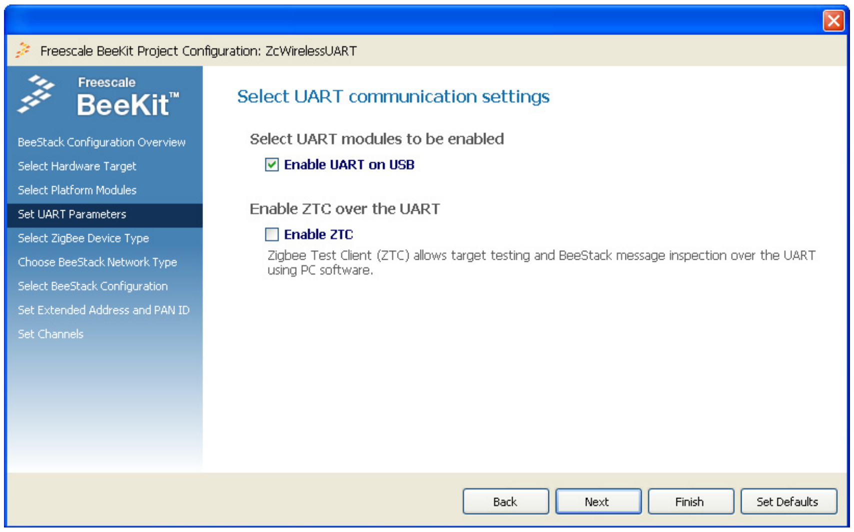

Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select platform modules to be enabled

- [✓ LEDs]
- [✓ Keyboard]
- LCD Display
- Non-Volatile Memory

Figure 5-5. Platform Modules Page

4. Leave the default platform modules settings unchanged. (LEDs and Keyboard modules on the MC1320x-S08QE128-EVB 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 MC1320x-S08QE128-EVB and the ZigBee Test Client (ZTC) disabled.)


Freescale BeeKit Project Configuration: ZcWirelessUART

Select UART communication settings

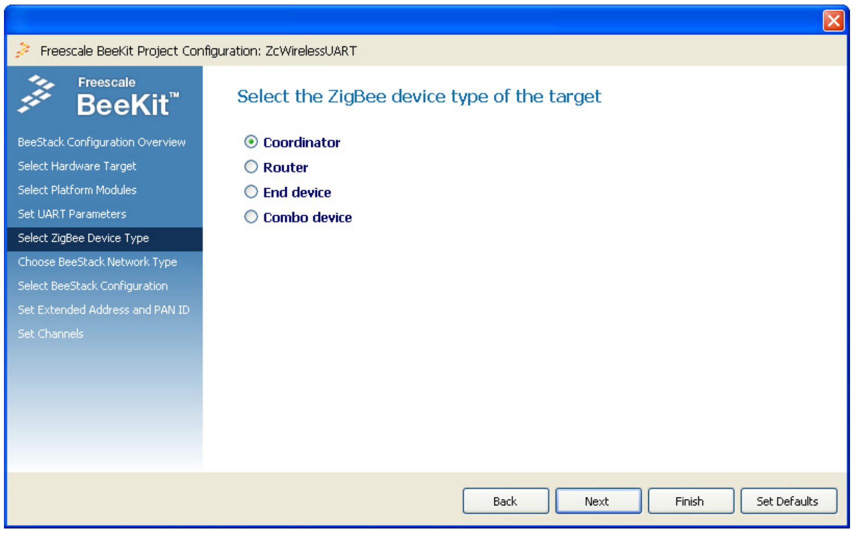
Select UART modules to be enabled
[✓] Enable UART on USB

Enable ZTC over the UART
[ ] Enable ZTC
Zigbee Test Client (ZTC) allows target testing and BeeStack message inspection over the UART using PC software.

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

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.


Freescale BeeKit Project Configuration: ZcWirelessUART

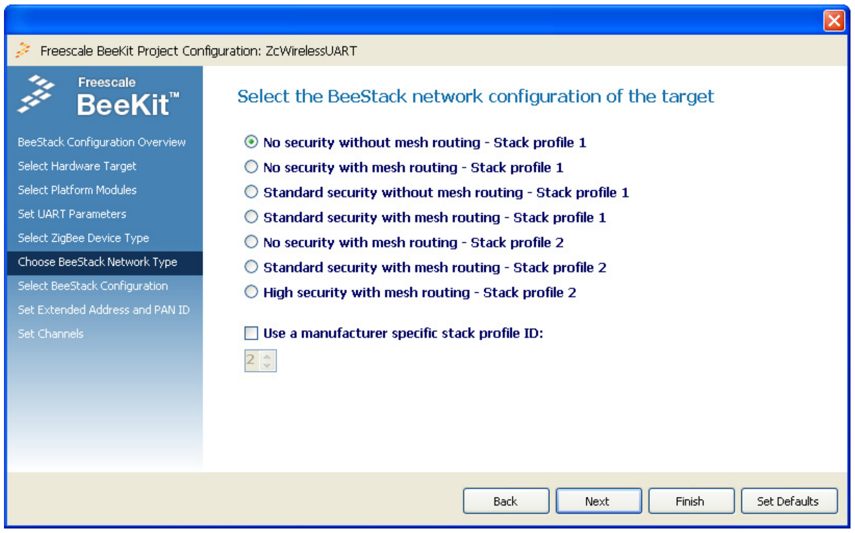
Select the ZigBee device type of the target

[✓ Coordinator]
Router
End device
Combo device

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

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.


Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

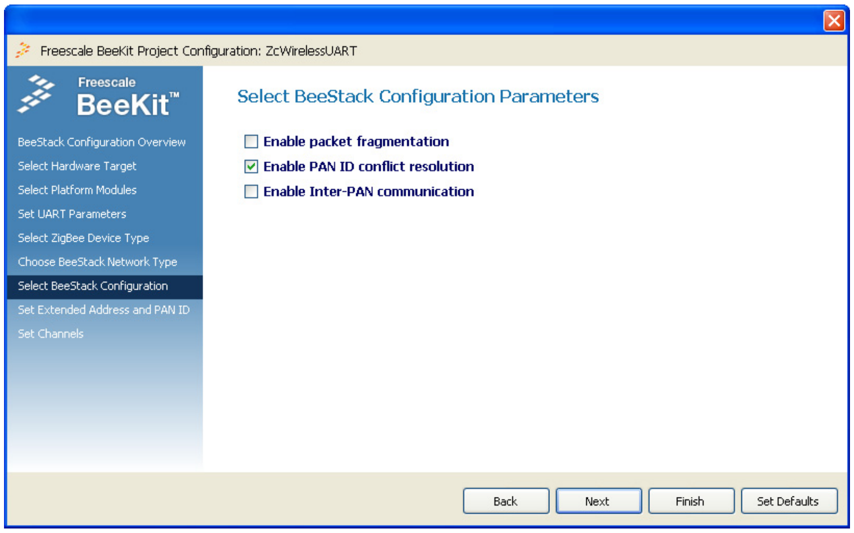
Select the BeeStack network configuration of the target

- [✓ No security without mesh routing - Stack profile 1]
- No security with mesh routing - Stack profile 1
- Standard security without mesh routing - Stack profile 1
- Standard security with mesh routing - Stack profile 1
- No security with mesh routing - Stack profile 2
- Standard security with mesh routing - Stack profile 2
- High security with mesh routing - Stack profile 2

☐ Use a manufacturer specific stack profile ID: 2

Figure 5-8. BeeStack Network Type Selection Page

8. Click the Next button and the Select BeeStack Configuration window appears as shown in Figure 5-9. Leave the default settings (PAN ID conflict enabled).

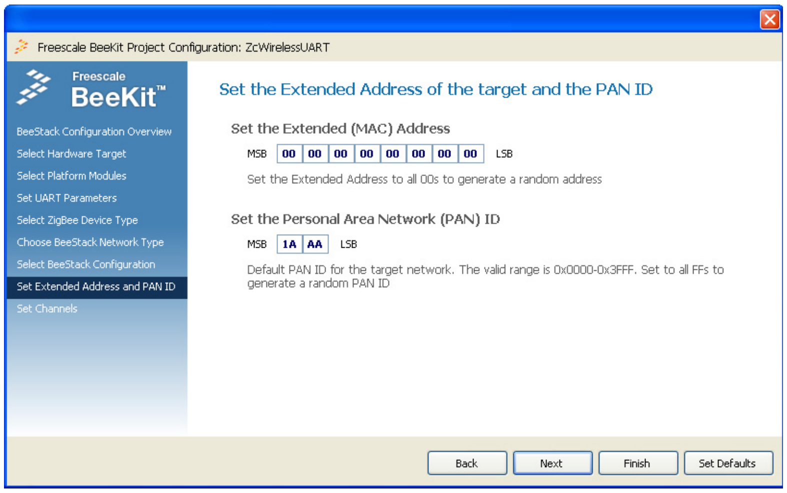

Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select BeeStack Configuration Parameters

☐ Enable packet fragmentation
☑ Enable PAN ID conflict resolution
☐ Enable Inter-PAN communication

Figure 5-9. BeeStack Configuration Page

9. Click the Next button and the Set Extended Address and PAN ID window appears as shown in Figure 5-10. For this example, leave the Extended Address option with all zeros. BeeStack 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.


Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Set the Extended Address of the target and the PAN ID

Set the Extended (MAC) Address

MSB 00 00 00 00 00 00 00 00 LSB

Set the Extended Address to all 00s to generate a random address

Set the Personal Area Network (PAN) ID

MSB 1A AA LSB

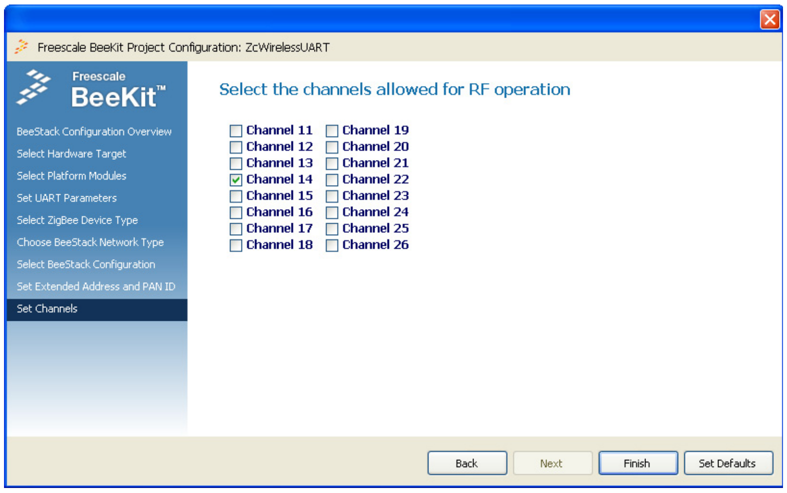
Default PAN ID for the target network. The valid range is 0x0000-0x3FFF. Set to all FFs to generate a random PAN ID

Figure 5-10. 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 BeeStack issue; this is a real-world ZigBee issue. User products must be able to cope with this.

10. Click the Next button and the Set Channels window appears as shown in Figure 5-11. Select ‘Channel 14’.


Freescale BeeKit Project Configuration: ZcWirelessUART
Freescale BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

Select the channels allowed for RF operation

☐ Channel 11	☐ Channel 19
☐ Channel 12	☐ Channel 20
☐ Channel 13	☐ Channel 21
✓ Channel 14	☐ Channel 22
☐ Channel 15	☐ Channel 23
☐ Channel 16	☐ Channel 24
☐ Channel 17	☐ Channel 25
☐ Channel 18	☐ Channel 26

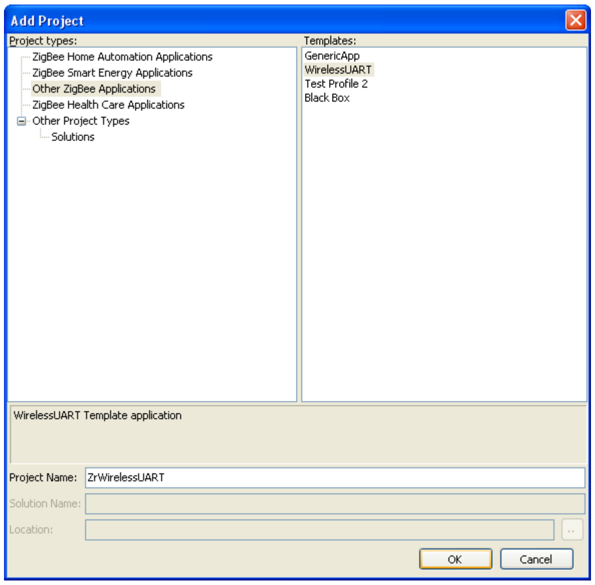
Figure 5-11. Channels Page

11. Click the Finish button.

5.1.2 Adding a Project for the Router

Now that the coordinator is added, users need to add a project for the Router. To add a project for the Router 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 5-12.

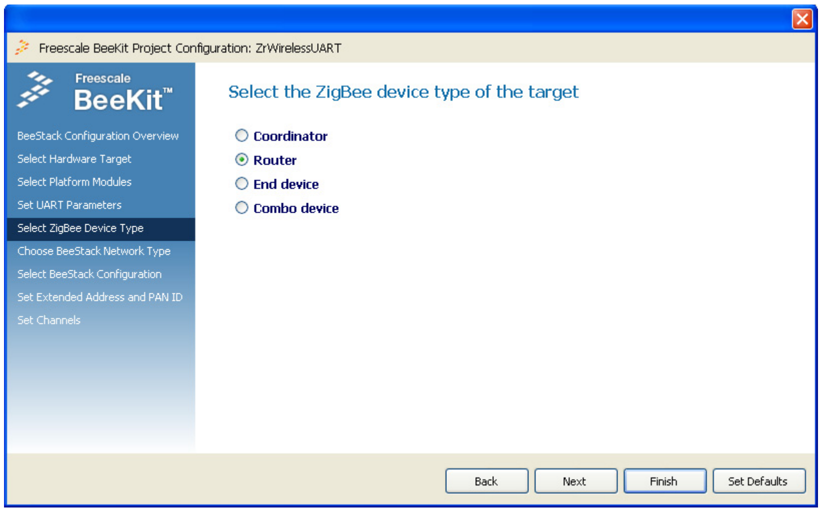


WirelessUART Template application

Project Name: ZrWirelessUART

Figure 5-12. Add New Project Page

2. Select “WirelessUART” and name the project ZrWirelessUART as shown in Figure 5-13.
3. Continue clicking the Next button until reaching the “Select ZigBee Device Type” window as shown in Figure 5-13. Select the “Router” option and click on the Next button.


Freescale BeeKit Project Configuration: ZrWirelessUART
Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Extended Address and PAN ID
Set Channels
Back
Next
Finish
Set Defaults

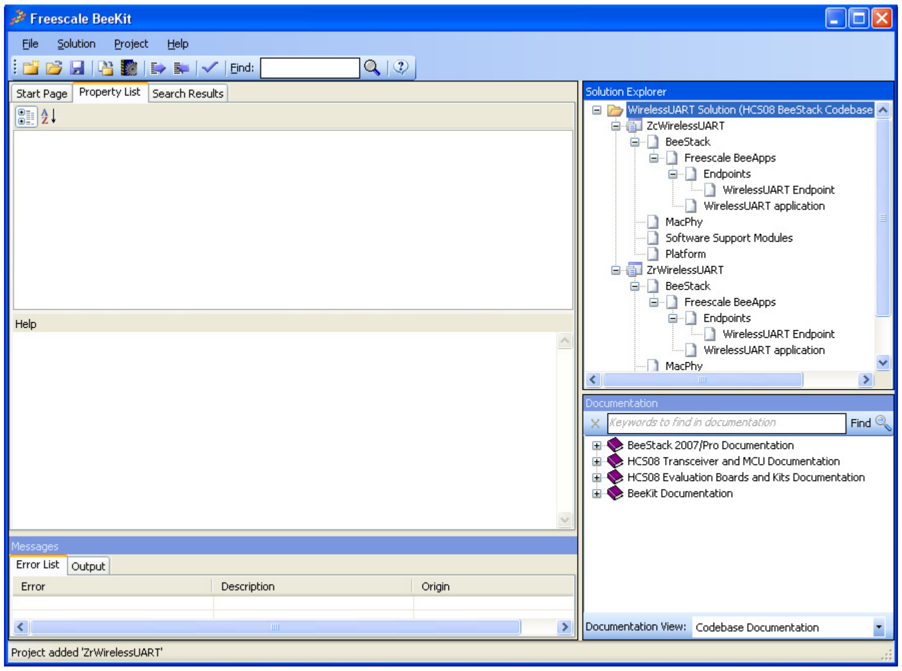
Select the ZigBee device type of the target

○ Coordinator
● Router
○ End device
○ Combo device

Figure 5-13. Selecting “Router” as the Device Type

4. 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).
5. Click the Next button and leave the default BeeStack Configuration as for the Coordinator as shown in Figure 5-9.
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 5-10.
7. Click the Next button and select the same channel as the Coordinator as shown in Figure 5-11.
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 a Router. The solution is now ready for export for use in CodeWarrior. The BeeKit main window should appear as shown in Figure 5-14.



Project added 'ZrWirelessUART'

Figure 5-14. Main Wireless UART Project Page

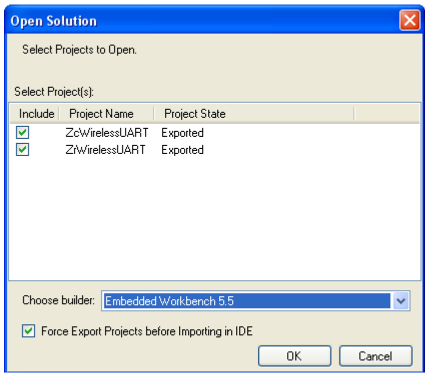
5.1.3 Exporting and Importing the BeeKit Projects to an IDE

After the solution is created, export it to an IDE (CodeWarrior or Embedded Workbench) to be built and loaded to the evaluation boards.

The export and import option is available only on Embedded Workbench.

To export the solution to Embedded Workbench, perform the following tasks:

1. From the BeeKit main window menu bar, choose Solution -> Export and Open Solution in Embedded Workbench. Before exporting, 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 solution and the IDE that will be used for building as shown in Figure 5-15.


Open Solution

Select Projects to Open.

Select Project(s):

Include	Project Name	Project State
✓	ZcWirelessUART	Exported
✓	ZrWirelessUART	Exported

Choose builder: Embedded Workbench 5.5

✓ Force Export Projects before Importing in IDE

Figure 5-15. Export BeeKit Project Solution Window

2. Click on the OK button to start the export process. BeeKit displays the steps it takes during the export process in the Messages display area of the BeeKit main window.
3. When BeeKit completes the export, it launches the IDE and imports the projects in a new workspace.

Figure 5-16 shows the projects imported in Embedded Workbench.

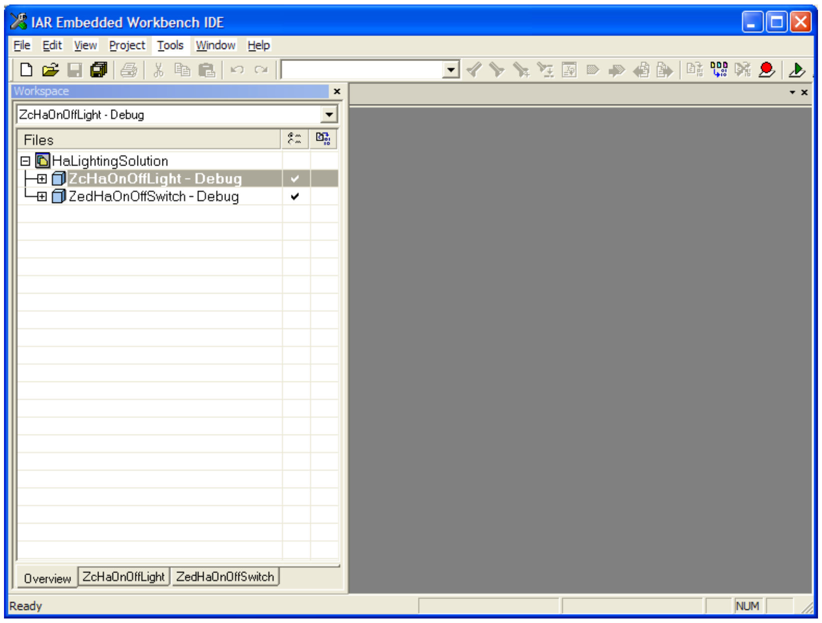


Figure 5-16. Projects Imported in Embedded Workbench

The export process is now complete.

4. Exit BeeKit by choosing File -> Exit from the tool bar.

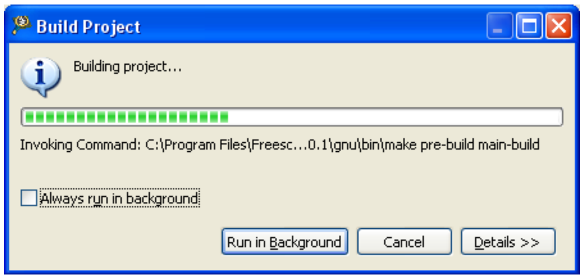
To export and open the projects manually in CodeWarrior, follow the steps presented in Section 3.3, "Manually Importing Projects into an IDE".

5.1.4 Building the Wireless UART Code Image

To build a project in the solution, select the tab corresponding to the project in the IDE workspace then proceed to build the application binary in one of three ways:

1. Click the "Build" icon (Looks like a hammer.)
2. From the menu select Project -> Build Project
3. Right click on the project -> Build Project

CodeWarrior reports the build progress in a window it opens and closes as shown in Figure 5-17. IAR Embedded Workbench displays the progress in the Build Messages panel.



Building project...

Invoking Command: C:\Program Files\Freesc...0.1\gnu\bin\make pre-build main-build

Figure 5-17. CodeWarrior Build Window

5.1.5 Loading the Wireless UART Code Image into a ZigBee Device

5.1.5.1 Loading the Wireless UART Code Image Using CodeWarrior

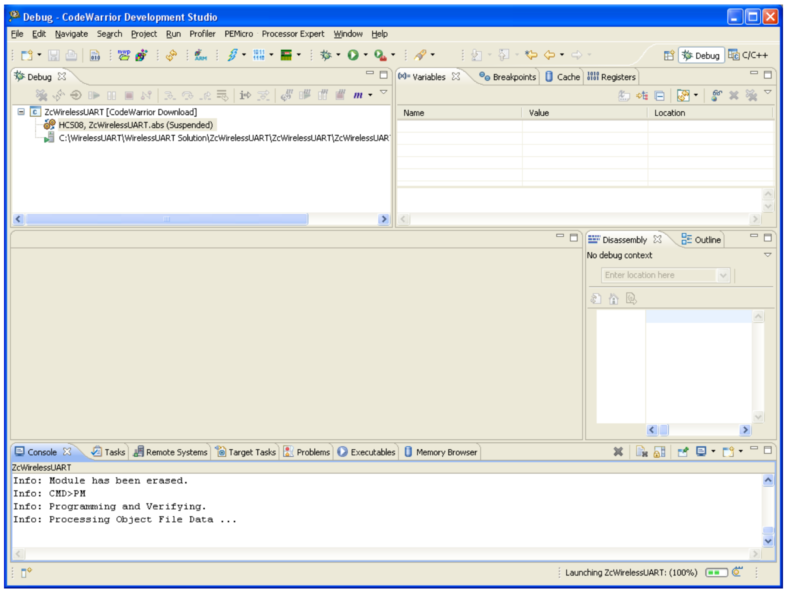
To use the Wireless UART application, the code images must be loaded into the Coordinator board and the Router board. This section describes how to load the Coordinator code image to the board, but the steps to load the code image to the Router board are the same.

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. Select the project where the image will be downloaded.

5. Download the compiled image to the board in one of three ways:
   a) Press the Debug hot key F11
   b) Click the “Debug” icon (Looks like a bug.)
   c) From the menu select Run -> Debug

CodeWarrior switches to the Debug perspective as shown in Figure 5-18.



ZcWirelessUART [CodeWarrior Download]
HCS08, ZcWirelessUART.abs (Suspended)
C:\WirelessUART\WirelessUART Solution\ZcWirelessUART\ZcWirelessUART\ZcWirelessUAR

ZcWirelessUART
```
Info: Module has been erased.
Info: CMD>PM
Info: Programming and Verifying.
Info: Processing Object File Data ...
```
Launching ZcWirelessUART: (100%)

Figure 5-18. CodeWarrior Debug Perspective

6. Close the debugger by selecting Run -> Terminate.

WARNING

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

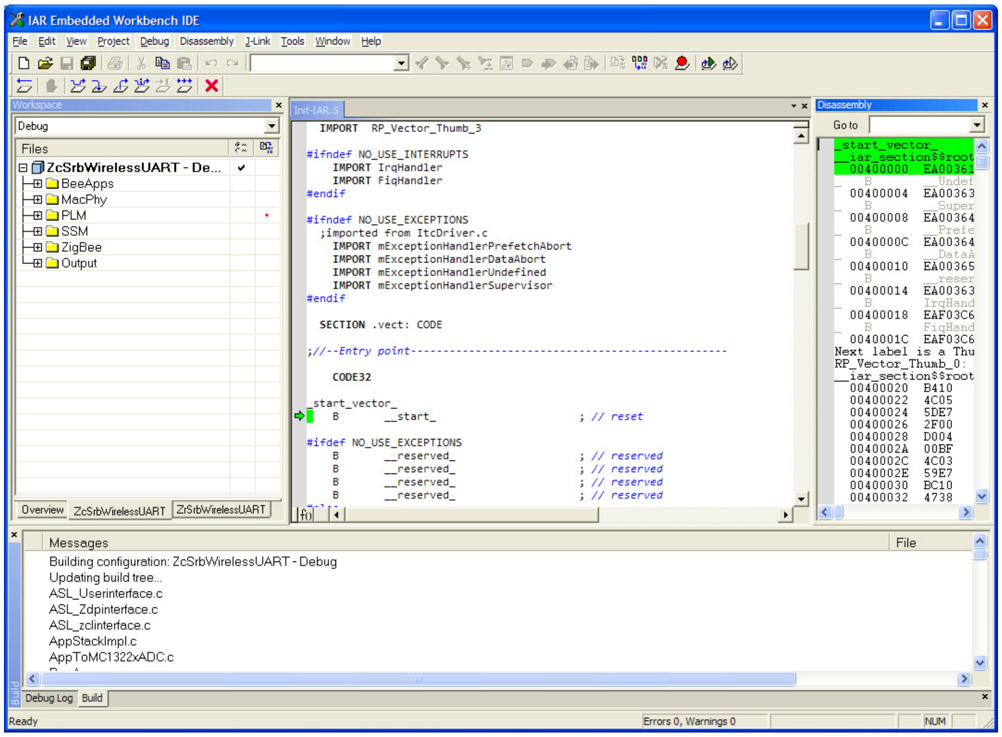
7. Disconnect the board and exit the CodeWarrior project, leaving CodeWarrior running.
8. Power cycle the board or press the reset button to get the board ready for use.
9. This concludes the steps needed to load the code image for the Coordinator.
10. Go to Step 1 and repeat the steps to load the code image to the Router board.

5.1.5.2 Loading the Wireless UART Code Image Using IAR EWB

To load the code image into a MC1322x evaluation board from within Embedded Workbench using the JLink JTAG debugger pod perform the following steps.

1. Select the Coordinator (ZcWirelessUART) project tab in the Workspace panel.
2. Connect the JLink pod to the computer using a USB cable.
3. Turn on the MC1322x Sensor Node board.
4. Connect the JLink ribbon cable to the JTAG pins on the evaluation board. Align pin 1 of the JTAG port which is marked with a white dot with the blue ribbon wire of the JLink.

5. Download the compiled image to the board by choosing one of three ways:
   a) Press the Debug hot key Ctrl+D.
   b) From the Project menu select Download and Debug (Embedded Workbench 5.50 or Debug (previous versions).
   c) Click the green triangle icon in the main toolbar.
6. Wait while the flash of the board is written. When this is complete, the debugging view of Embedded Workbench will launch as shown in Figure 5-19.


IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly J-Link Tools Window Help
Workspace
Debug
Files
ZcSrbWirelessUART - De...
BeeApps
MacPhy
PLM
SSM
ZigBee
Output
Overview ZcSrbWirelessUART ZrSrbWirelessUART
Init-IAR.S
Disassembly
Go to
Messages
File
Debug Log Build
Ready
Errors 0, Warnings 0
NUM

```
  IMPORT   RP_Vector_Thumb_3

#ifndef NO_USE_INTERRUPTS
    IMPORT IrqHandler
    IMPORT FiqHandler
#endif

#ifndef NO_USE_EXCEPTIONS
  ;imported from ItcDriver.c
    IMPORT mExceptionHandlerPrefetchAbort
    IMPORT mExceptionHandlerDataAbort
    IMPORT mExceptionHandlerUndefined
    IMPORT mExceptionHandlerSupervisor
#endif

  SECTION .vect: CODE

;//--Entry point----------------------------------------------

    CODE32

_start_vector_
    B       __start_                    ; // reset

#ifdef NO_USE_EXCEPTIONS
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
    B       __reserved_                 ; // reserved
```

```
_start_vector_:
__iar_section$$root
   00400000   EA00361
    B         __Undef
   00400004   EA00363
    B         __Super
   00400008   EA00364
    B         __Prefe
   0040000C   EA00364
    B         __DataA
   00400010   EA00365
    B         __reser
   00400014   EA00363
    B         IrqHand
   00400018   EAF03C6
    B         FiqHand
   0040001C   EAF03C6
Next label is a Thu
RP_Vector_Thumb_0:
__iar_section$$root
   00400020   B410
   00400022   4C05
   00400024   5DE7
   00400026   2F00
   00400028   D004
   0040002A   00BF
   0040002C   4C03
   0040002E   59E7
   00400030   BC10
   00400032   4738
```

```
Building configuration: ZcSrbWirelessUART - Debug
Updating build tree...
ASL_Userinterface.c
ASL_Zdpinterface.c
ASL_zclinterface.c
AppStackImpl.c
AppToMC1322xADC.c
```

Figure 5-19. IAR Embedded Workbench Debugging View

7. Stop the debugger by pressing Ctrl+Shift+D or by choosing “Stop Debugging” from the Debug menu.
8. Select the Router (ZrWirelessUART) project tab in the Workspace panel
9. Repeat the Steps 3-7 to put the code for the second application into another MC1322x Sensor Node board.

5.2 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.

5.2.1 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 power on the boards.
2. In the Windows Device Manager, under the Ports (COM & LPT) option, two devices labeled either "Freescale ZigBee/802.15.4 MAC COM Device" or "USB Serial Port" appear as shown in Figure 5-20. (The COM ports shown in Figure 5-20 will be different on every PC).

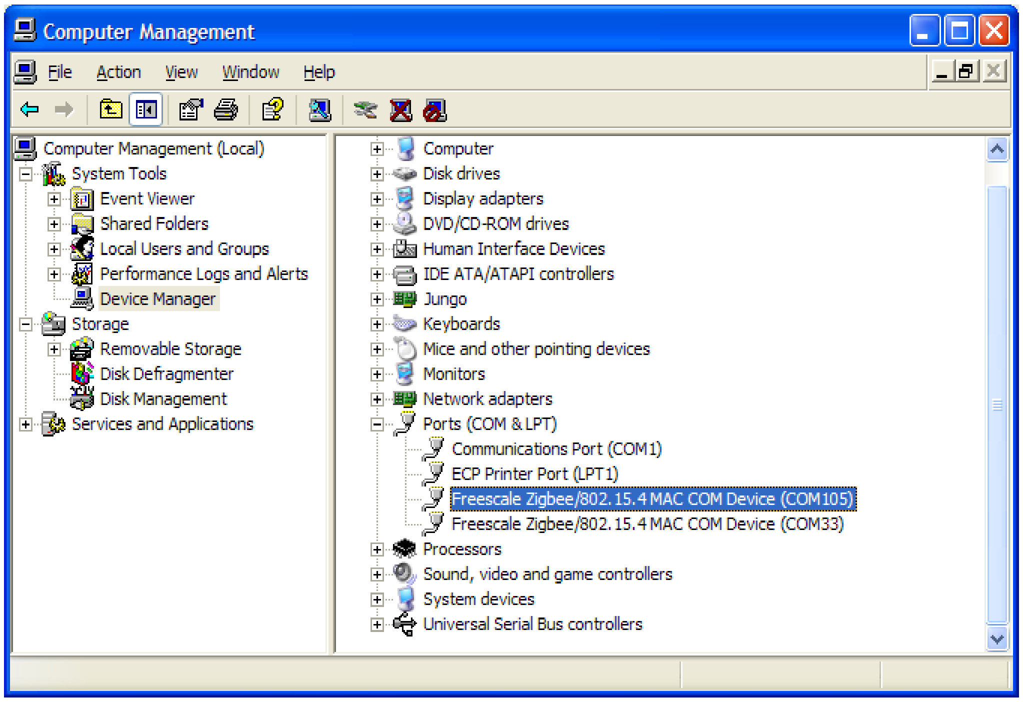
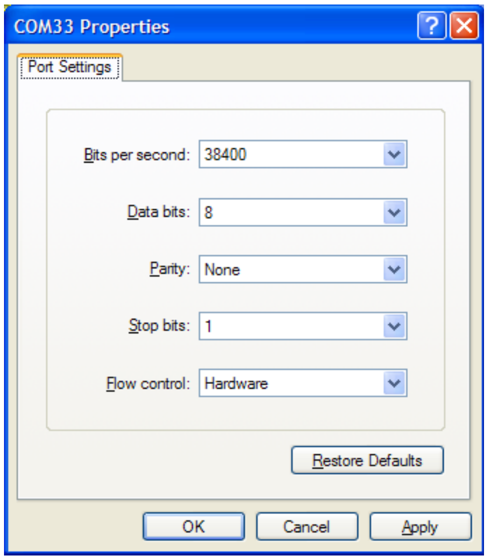


Figure 5-20. COM Ports in Device Manager

3. Using Hyperterminal, set up a Virtual COM Port for each of the two nodes. Figure 5-21 shows the correct COM Port settings to run the Wireless UART application.

NOTE

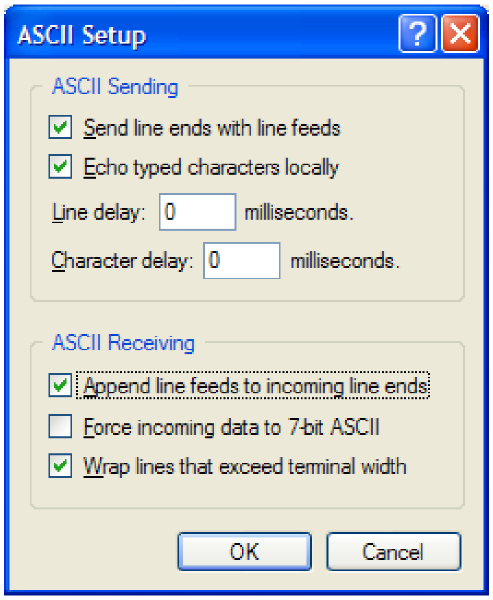
Flow control is enabled on both nodes.



Bits per second: 38400
Data bits: 8
Parity: None
Stop bits: 1
Flow control: Hardware

Figure 5-21. Default BeeStack RS-232 Settings

Figure 5-22 show correct ASCII Setup configuration used to run the Wireless UART application.



ASCII Sending
- [✓] Send line ends with line feeds
- [✓] Echo typed characters locally

Line delay: 0 milliseconds.
Character delay: 0 milliseconds.

ASCII Receiving
- [✓] Append line feeds to incoming line ends
- [ ] Force incoming data to 7-bit ASCII
- [✓] Wrap lines that exceed terminal width

Figure 5-22. Additional Terminal Program Settings

5.2.2 Starting the Wireless UART Application

1. Connect both the Coordinator and Router 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 5.2, "Wireless UART Setup and Operation".
3. Turn on the power for the Wireless UART Coordinator and Router.

   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.

5.2.3 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 Router.

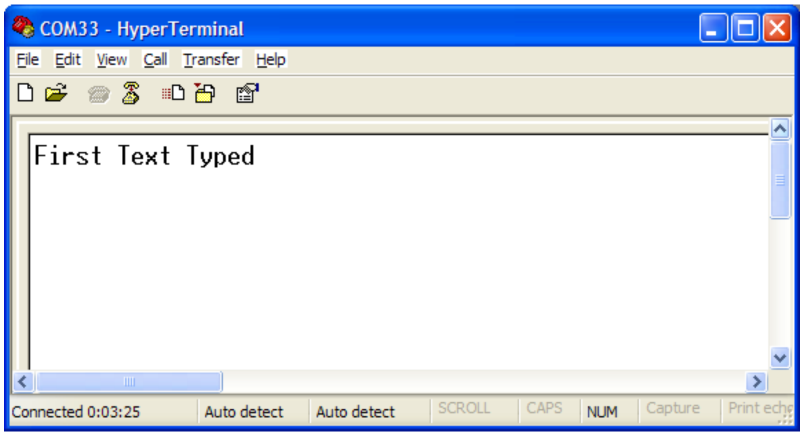
   All of the LEDs on this board will flash in a series, and finally only LED1 stays lit to indicate that the Router has joined the network that the Coordinator has formed.
3. To bind the devices, press SW3 first on one board, then within ten seconds, press SW3 on the other board.

   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.

5.2.4 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.

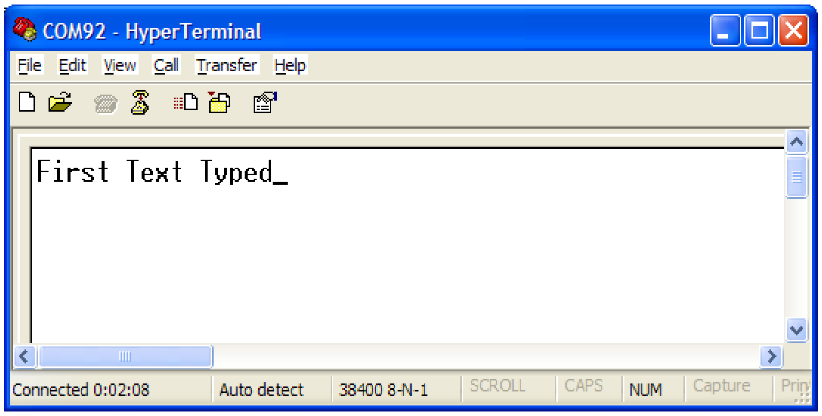
1. Type some text in one of the Hyperterminal windows as shown in Figure 5-23.



```
First Text Typed
```

Figure 5-23. Text Typed in First Hyperterminal Window

2. The typed text is displayed as output in the second Hyperterminal as shown in Figure 5-24



```
First Text Typed_
```

Figure 5-24. Text Echoed in Second Hyperterminal Window

Chapter 6
Creating a Smart Energy Network Application

This chapter shows how to create a Smart Energy Network consisting of three different applications using the Freescale BeeKit Wireless Connectivity Toolkit. The Freescale Smart Energy applications are part of the ZigBee Alliance Smart Energy (SE) profile. While this chapter describes how to use the Smart Energy application, it does not describe the projects in great detail.

6.1 Software and Hardware Requirements

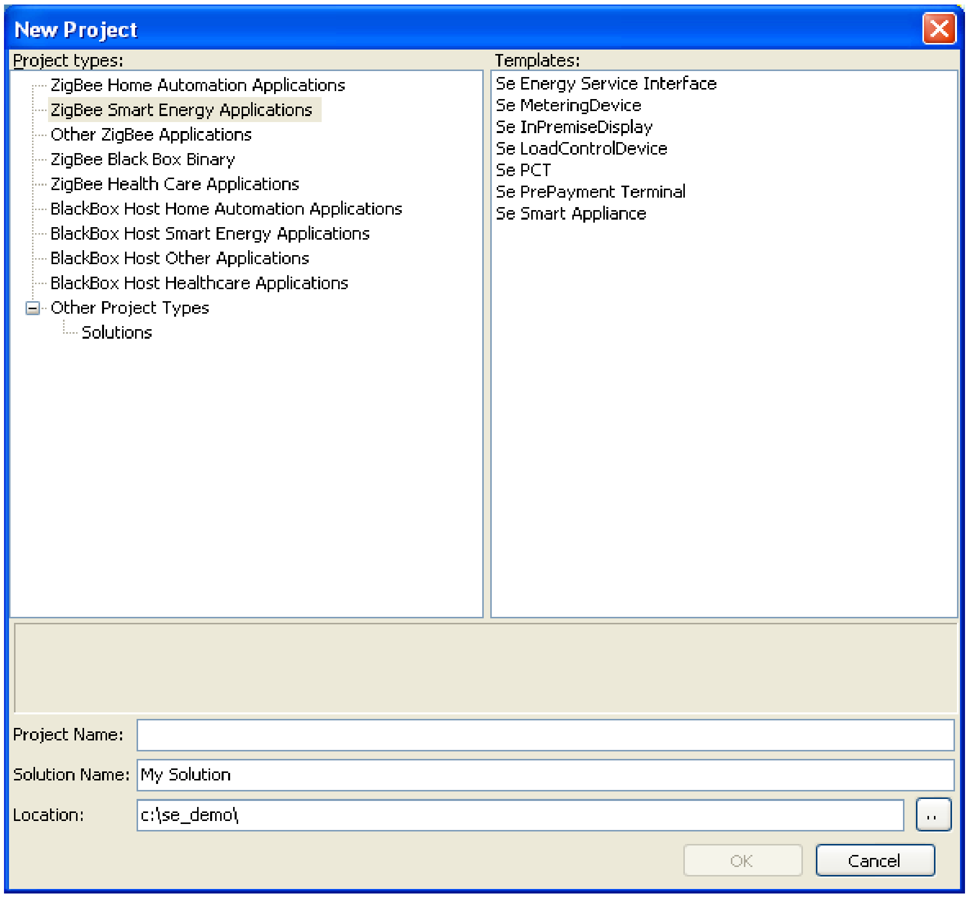
Before performing any of the steps in this chapter, users must ensure the following:

- Chapter 3 and Chapter 4 of this guide have been read, the steps performed and understood
- The Freescale Test Tool is installed and running on the PC
- The Freescale Test Tool User’s Guide, Section 2.5 has been read, the steps performed and understood
- One 1322x Sensor Node board is available to be used as the Energy Service Interface (ESI)
- One 1322x Sensor Node board is available to be used as the MeteringDevice
- One 1322x Network Node is available to be used as the InPremiseDisplay

6.2 Creating a BeeKit Project

Follow these steps to create a BeeKit project and configure the Coordinator, End Device and Router devices.

1. Start the BeeKit Wireless Connectivity Toolkit.
2. If another Codebase (BeeStack Consumer, MAC or SMAC) is selected, perform the following:
   a) From the tool bar, click on File -> Select Codebase… or click the “Select Other Codebase...” button.
   b) From the Codebase list, choose the appropriate 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 6-1.


New Project

Project types:
- ZigBee Home Automation Applications
- ZigBee Smart Energy Applications
- Other ZigBee Applications
- ZigBee Black Box Binary
- ZigBee Health Care Applications
- BlackBox Host Home Automation Applications
- BlackBox Host Smart Energy Applications
- BlackBox Host Other Applications
- BlackBox Host Healthcare Applications
- Other Project Types
  - Solutions

Templates:
- Se Energy Service Interface
- Se MeteringDevice
- Se InPremiseDisplay
- Se LoadControlDevice
- Se PCT
- Se PrePayment Terminal
- Se Smart Appliance

Project Name:

Solution Name: My Solution

Location: c:\se_demo\

Figure 6-1. New Project Window

4. Select the ZigBee Smart Energy Applications project type from the left side of the window.
5. Select the “Se Energy Service Interface” (ESI) template and create the project by entering the following information into the appropriate field:

   Project Name: ZcEnergyServiceInterface

   Solution Name: SE Demo Solution

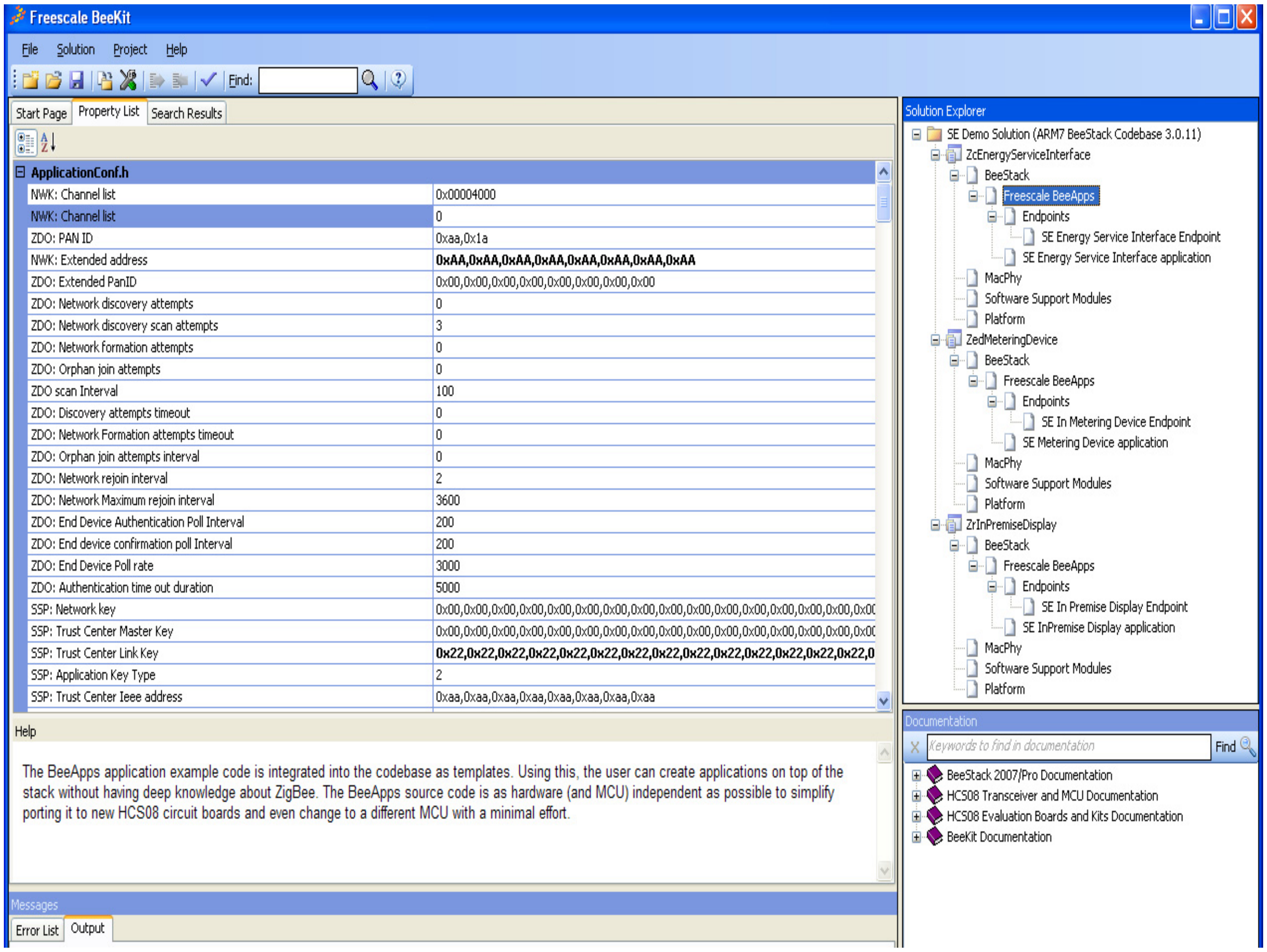
   Location: `c:\Se_demo` (or other sub directory on host PC)
6. Use the default settings and only change the hardware settings if a board other than a 1322x Sensor Node or 1322x Network Node are used.
7. Add an “SE MeteringDevice” to the Solution from the BeeKit main window by selecting Solution -> Add Project.
8. Select the ZigBee Smart Energy Applications project type from the left side of the window.
9. Select the “Se MeteringDevice” template and create the project by entering the following information into the appropriate field:

   Project Name: ZedMeteringDevice
10. Use the default settings and only change the hardware settings if a board other than a 1322x Sensor Node or 1322x Network Node are used.

11. Add an “Se InPremiseDisplay” to the Solution from the BeeKit main window by selecting Solution -> Add Project.
12. Select the ZigBee Smart Energy Applications project type from the left side of the window.
13. Select the “Se InPremiseDisplay” template and create the project by entering the following information into the appropriate field:

    Project Name: ZrInPremiseDisplay
14. Export the solution, compile the projects and program the boards using the IAR Embedded workbench for ARM as follows:
    — Program the Network Node as the InPremiseDisplay.
    — Program one Sensor Node as the MeteringDevice.
    — Program the other Sensor Node as the ESI.

Figure 6-2 shows the completed project.



ApplicationConf.h	
NWK: Channel list	0x00004000
NWK: Channel list	0
ZDO: PAN ID	0xaa,0x1a
NWK: Extended address	**0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA**
ZDO: Extended PanID	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
ZDO: Network discovery attempts	0
ZDO: Network discovery scan attempts	3
ZDO: Network formation attempts	0
ZDO: Orphan join attempts	0
ZDO scan Interval	100
ZDO: Discovery attempts timeout	0
ZDO: Network Formation attempts timeout	0
ZDO: Orphan join attempts interval	0
ZDO: Network rejoin interval	2
ZDO: Network Maximum rejoin interval	3600
ZDO: End Device Authentication Poll Interval	200
ZDO: End device confirmation poll Interval	200
ZDO: End Device Poll rate	3000
ZDO: Authentication time out duration	5000
SSP: Network key	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
SSP: Trust Center Master Key	0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
SSP: Trust Center Link Key	**0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0**
SSP: Application Key Type	2
SSP: Trust Center Ieee address	0xaa,0xaa,0xaa,0xaa,0xaa,0xaa,0xaa,0xaa

Help

The BeeApps application example code is integrated into the codebase as templates. Using this, the user can create applications on top of the stack without having deep knowledge about ZigBee. The BeeApps source code is as hardware (and MCU) independent as possible to simplify porting it to new HCS08 circuit boards and even change to a different MCU with a minimal effort.

- SE Demo Solution (ARM7 BeeStack Codebase 3.0.11)
  - ZcEnergyServiceInterface
    - BeeStack
      - Freescale BeeApps
        - Endpoints
          - SE Energy Service Interface Endpoint
        - SE Energy Service Interface application
      - MacPhy
      - Software Support Modules
      - Platform
  - ZedMeteringDevice
    - BeeStack
      - Freescale BeeApps
        - Endpoints
          - SE In Metering Device Endpoint
        - SE Metering Device application
      - MacPhy
      - Software Support Modules
      - Platform
  - ZrInPremiseDisplay
    - BeeStack
      - Freescale BeeApps
        - Endpoints
          - SE In Premise Display Endpoint
        - SE InPremise Display application
      - MacPhy
      - Software Support Modules
      - Platform

- BeeStack 2007/Pro Documentation
- HCS08 Transceiver and MCU Documentation
- HCS08 Evaluation Boards and Kits Documentation
- BeeKit Documentation

Figure 6-2. Complete Smart Energy Project

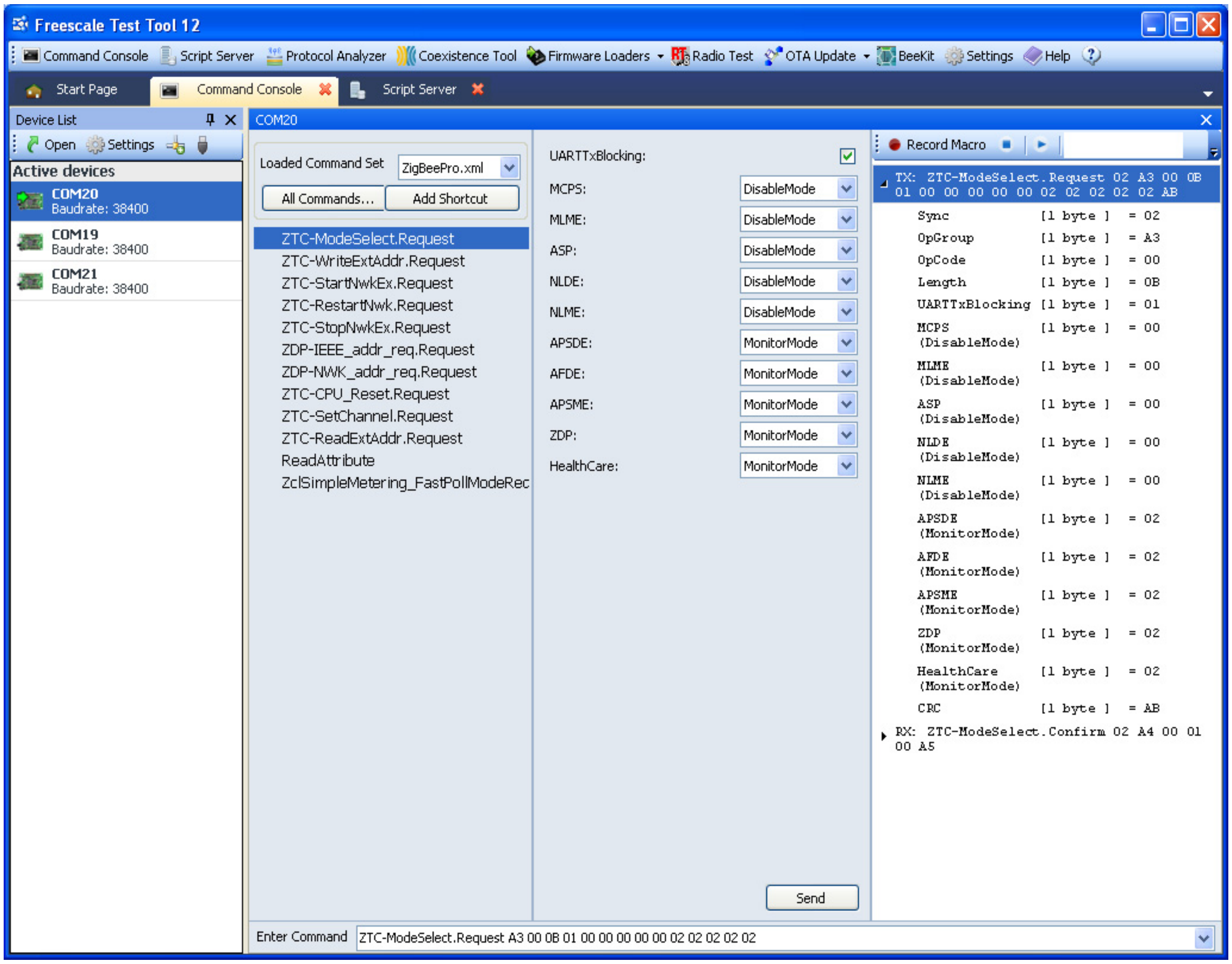
6.3 Controlling a Smart Energy Network

This section describes how to use the Energy Service Interface (ESI) to control a Smart Energy Network.

6.3.1 Board Connection and Network Startup

After creating and downloading the projects to the appropriate boards, perform the following tasks.

1. Connect the appropriate power source to all boards.
2. Connect the Sensor Node being used as the Energy Service Interface (ESI) to a PC using the USB port on the board. The Freescale Test Tool must already be installed and running on the PC.
3. Configure Test Tool so that it can communicate with the board. See the Freescale *Test Tool User's Guide* for more information on board communication.
4. From Test Tool, start the Command Console and ensure that the `ZigBeePro.xml` file is selected.
5. Configure the ZTC-ModeSelect.Request command as shown in Figure 6-3.



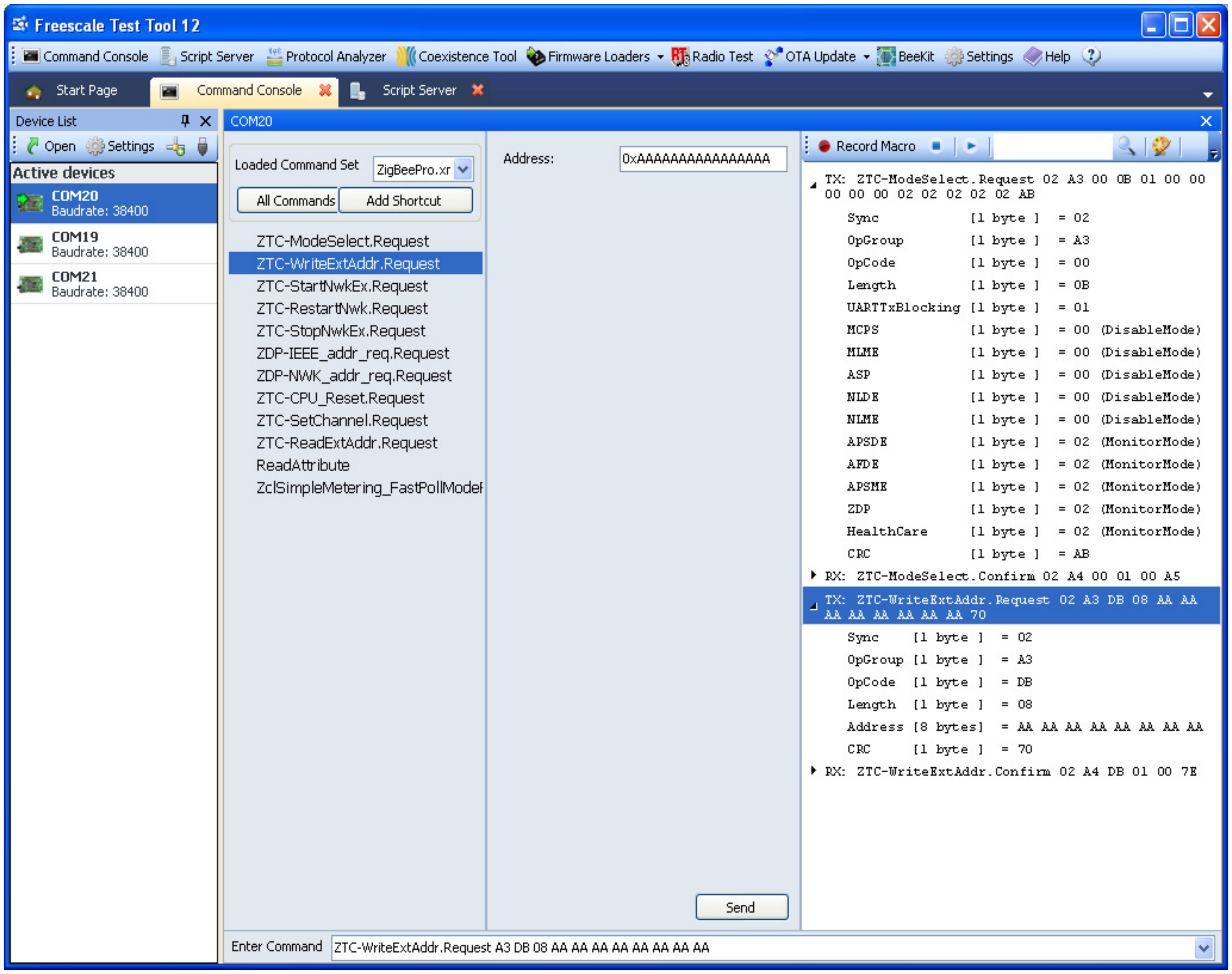
UARTTxBlocking:	✓
MCPS:	DisableMode
MLME:	DisableMode
ASP:	DisableMode
NLDE:	DisableMode
NLME:	DisableMode
APSDE:	MonitorMode
AFDE:	MonitorMode
APSME:	MonitorMode
ZDP:	MonitorMode
HealthCare:	MonitorMode

```
TX:  ZTC-ModeSelect.Request 02 A3 00 0B
01 00 00 00 00 00 02 02 02 02 02 AB
   Sync             [1 byte ]  = 02
   OpGroup          [1 byte ]  = A3
   OpCode           [1 byte ]  = 00
   Length           [1 byte ]  = 0B
   UARTTxBlocking   [1 byte ]  = 01
   MCPS             [1 byte ]  = 00
   (DisableMode)
   MLME             [1 byte ]  = 00
   (DisableMode)
   ASP              [1 byte ]  = 00
   (DisableMode)
   NLDE             [1 byte ]  = 00
   (DisableMode)
   NLME             [1 byte ]  = 00
   (DisableMode)
   APSDE            [1 byte ]  = 02
   (MonitorMode)
   AFDE             [1 byte ]  = 02
   (MonitorMode)
   APSME            [1 byte ]  = 02
   (MonitorMode)
   ZDP              [1 byte ]  = 02
   (MonitorMode)
   HealthCare       [1 byte ]  = 02
   (MonitorMode)
   CRC              [1 byte ]  = AB
RX:  ZTC-ModeSelect.Confirm 02 A4 00 01
00 A5
```

Enter Command: `ZTC-ModeSelect.Request A3 00 0B 01 00 00 00 00 00 02 02 02 02 02`

Figure 6-3. ZTC-ModeSelect.Request Command (Confirm Communication)

6. Select the Device/ComPort used by the ESI board and send the ZTC-ModeSelect.Request command to ensure that the board connection is working.
7. Send the ZTC-WriteExtAddr.Request command which sends the IEEE address of the ESI as shown in Figure 6-4.



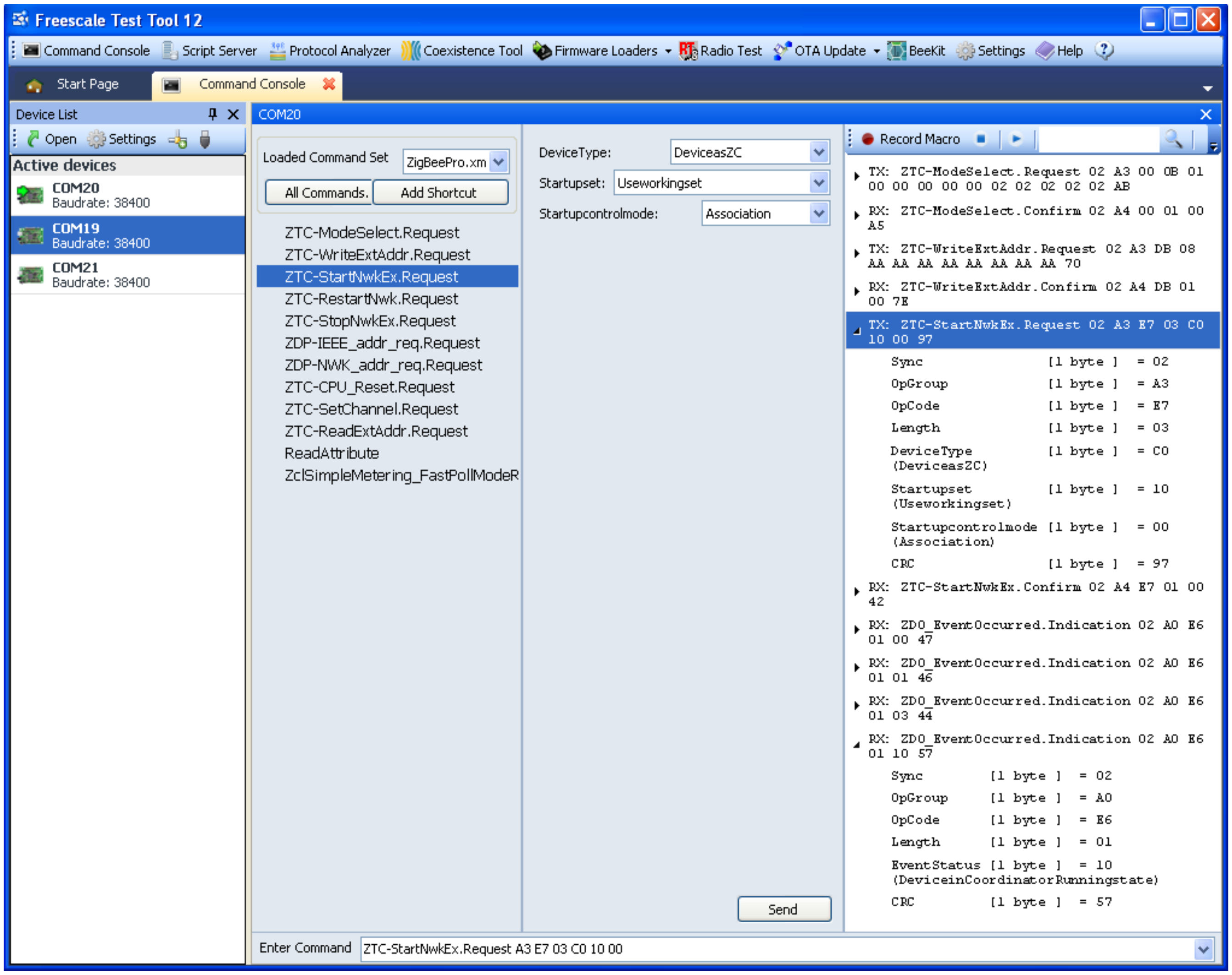
Address: 0xAAAAAAAAAAAAAAAA

```
TX: ZTC-ModeSelect.Request 02 A3 00 0B 01 00 00
00 00 00 02 02 02 02 AB
    Sync          [1 byte ]  = 02
    OpGroup       [1 byte ]  = A3
    OpCode        [1 byte ]  = 00
    Length        [1 byte ]  = 0B
    UARTTxBlocking [1 byte ]  = 01
    MCPS          [1 byte ]  = 00 (DisableMode)
    MLME          [1 byte ]  = 00 (DisableMode)
    ASP           [1 byte ]  = 00 (DisableMode)
    NLDE          [1 byte ]  = 00 (DisableMode)
    NLME          [1 byte ]  = 00 (DisableMode)
    APSDE         [1 byte ]  = 02 (MonitorMode)
    AFDE          [1 byte ]  = 02 (MonitorMode)
    APSME         [1 byte ]  = 02 (MonitorMode)
    ZDP           [1 byte ]  = 02 (MonitorMode)
    HealthCare    [1 byte ]  = 02 (MonitorMode)
    CRC           [1 byte ]  = AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00 A5
TX: ZTC-WriteExtAddr.Request 02 A3 DB 08 AA AA
AA AA AA AA AA AA 70
    Sync    [1 byte ]  = 02
    OpGroup [1 byte ]  = A3
    OpCode  [1 byte ]  = DB
    Length  [1 byte ]  = 08
    Address [8 bytes]  = AA AA AA AA AA AA AA AA
    CRC     [1 byte ]  = 70
RX: ZTC-WriteExtAddr.Confirm 02 A4 DB 01 00 7E
```

Enter Command ZTC-WriteExtAddr.Request A3 DB 08 AA AA AA AA AA AA AA AA

Figure 6-4. ZTC-WriteExtAddr.Request Command (Send IEEE Address)

8. Send the ZTC-StartNWK.Request to form a network as shown in Figure 6-5.



Active devices
COM20 Baudrate: 38400
COM19 Baudrate: 38400
COM21 Baudrate: 38400

COM20

Loaded Command Set: ZigBeePro.xm

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request
ReadAttribute
ZclSimpleMetering_FastPollModeR

DeviceType: DeviceasZC
Startupset: Useworkingset
Startupcontrolmode: Association

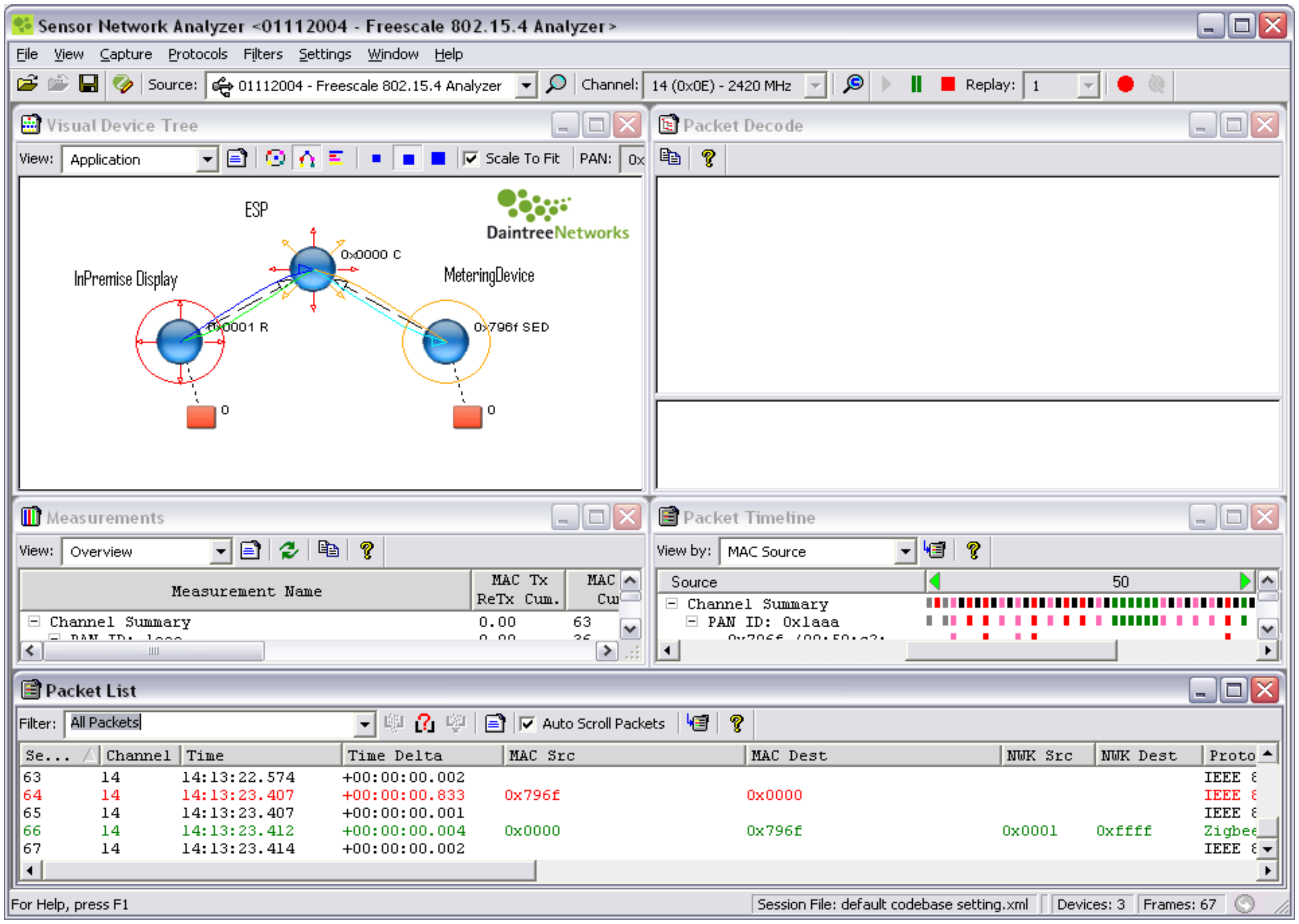
Enter Command: ZTC-StartNwkEx.Request A3 E7 03 C0 10 00

```
TX: ZTC-ModeSelect.Request 02 A3 00 0B 01
00 00 00 00 00 02 02 02 02 02 AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00
A5
TX: ZTC-WriteExtAddr.Request 02 A3 DB 08
AA AA AA AA AA AA AA AA 70
RX: ZTC-WriteExtAddr.Confirm 02 A4 DB 01
00 7E
TX: ZTC-StartNwkEx.Request 02 A3 E7 03 C0
10 00 97
   Sync               [1 byte ]  = 02
   OpGroup            [1 byte ]  = A3
   OpCode             [1 byte ]  = E7
   Length             [1 byte ]  = 03
   DeviceType         [1 byte ]  = C0
   (DeviceasZC)
   Startupset         [1 byte ]  = 10
   (Useworkingset)
   Startupcontrolmode [1 byte ]  = 00
   (Association)
   CRC                [1 byte ]  = 97
RX: ZTC-StartNwkEx.Confirm 02 A4 E7 01 00
42
RX: ZDO_EventOccurred.Indication 02 A0 E6
01 00 47
RX: ZDO_EventOccurred.Indication 02 A0 E6
01 01 46
RX: ZDO_EventOccurred.Indication 02 A0 E6
01 03 44
RX: ZDO_EventOccurred.Indication 02 A0 E6
01 10 57
   Sync        [1 byte ]  = 02
   OpGroup     [1 byte ]  = A0
   OpCode      [1 byte ]  = E6
   Length      [1 byte ]  = 01
   EventStatus [1 byte ]  = 10
   (DeviceinCoordinatorRunningstate)
   CRC         [1 byte ]  = 57
```

Figure 6-5. ZTC-StartNWK.Request Command (Forming the Network)

9. Press SW1 on the Sensor Board used as the InPremiseDisplay. This joins the board to the ESI.
10. Press SW1 on the Sensor Board used as the MeteringDevice. This joins the board to the ESI.
11. After the InPremiseDisplay and the MeteringDevice are joined, set the user interface to application mode by performing a long button press on SW1. The board serving as the InPremiseDisplay shows the time acquired from the ESI on its LCD display and the board serving as the MeteringDevice has its LED4 blink to indicate simulate that power consumption is ongoing.

Figure 6-6 shows a working Smart Energy Network containing the ESI, an InPremiseDisplay and a MeteringDevice.



Measurement Name	MAC Tx ReTx Cum.	MAC Cu
Channel Summary	0.00	63

Source
Channel Summary
PAN ID: 0x1aaa

Se...	Channel	Time	Time Delta	MAC Src	MAC Dest	NWK Src	NWK Dest	Proto
63	14	14:13:22.574	+00:00:00.002					IEEE 8
64	14	14:13:23.407	+00:00:00.833	0x796f	0x0000			IEEE 8
65	14	14:13:23.407	+00:00:00.001					IEEE 8
66	14	14:13:23.412	+00:00:00.004	0x0000	0x796f	0x0001	0xffff	Zigbee
67	14	14:13:23.414	+00:00:00.002					IEEE 8

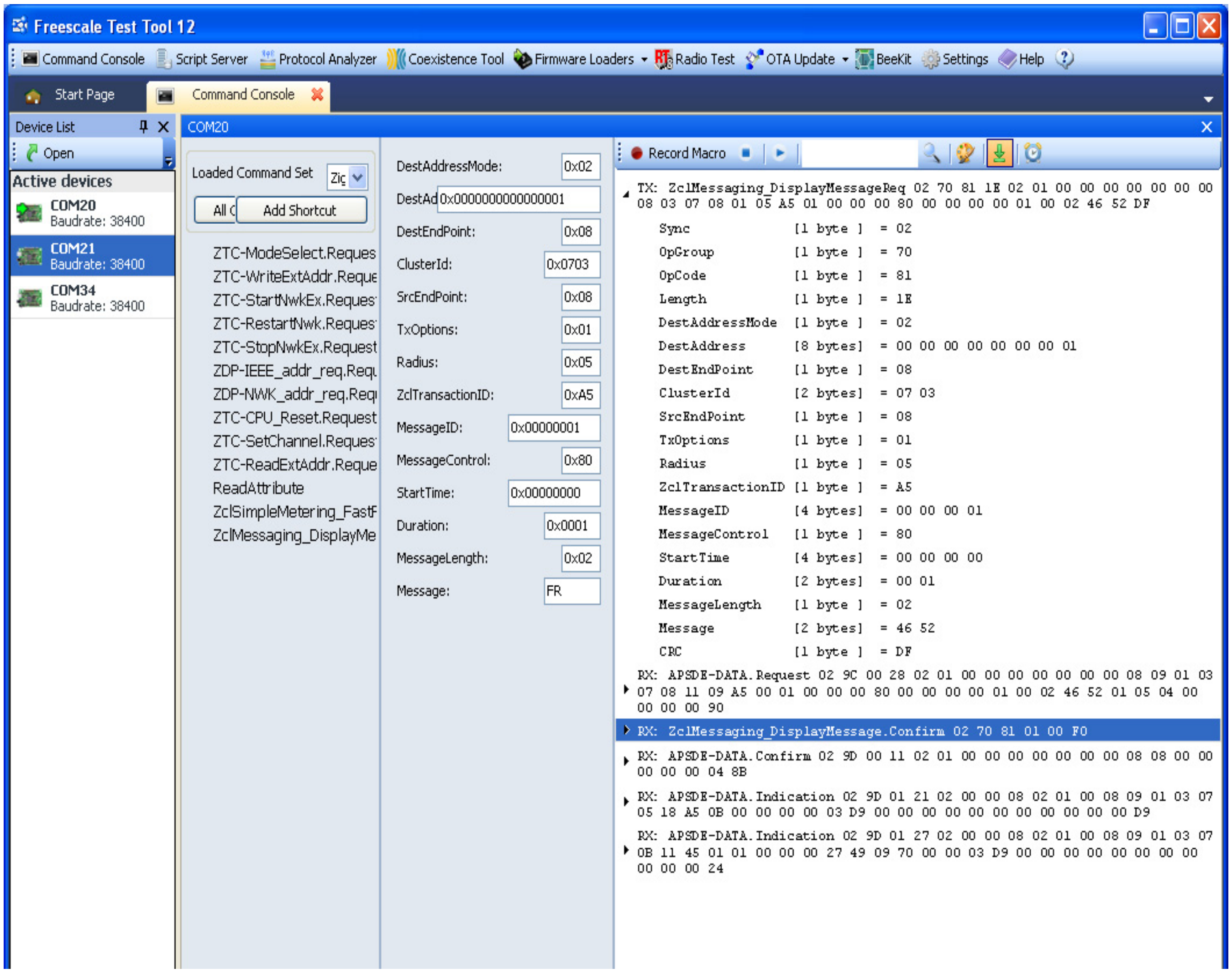
For Help, press F1 | Session File: default codebase setting.xml | Devices: 3 | Frames: 67

Figure 6-6. Working Smart Energy Network

6.3.2 Sending Commands

Use the Test Tool Command Console to send various text information and commands to the display with user confirmation required.

1. From Test Tool Command Console tool bar, click on All Commands -> ZigBee Cluster Library commands and select the ZclMessaging_DisplayMessageReq command.
2. Set the parameters in the Request as follows (also shown in Figure 6-7):
   — DestAddress — To the address of the InPremiseDisplay (0x0000000000000001)
   — ClusterID — As the message cluster (0x0703)
   — MessageControl — To 0x80 (To require a Confirmation)
   — Choose a desired message length and message text



DestAddressMode:	0x02
DestAd	0x0000000000000001
DestEndPoint:	0x08
ClusterId:	0x0703
SrcEndPoint:	0x08
TxOptions:	0x01
Radius:	0x05
ZclTransactionID:	0xA5
MessageID:	0x00000001
MessageControl:	0x80
StartTime:	0x00000000
Duration:	0x0001
MessageLength:	0x02
Message:	FR

```
TX: ZclMessaging_DisplayMessageReq 02 70 81 1E 02 01 00 00 00 00 00 00 00
08 03 07 08 01 05 A5 01 00 00 00 80 00 00 00 00 01 00 02 46 52 DF
   Sync              [1 byte ]  = 02
   OpGroup           [1 byte ]  = 70
   OpCode            [1 byte ]  = 81
   Length            [1 byte ]  = 1E
   DestAddressMode   [1 byte ]  = 02
   DestAddress       [8 bytes]  = 00 00 00 00 00 00 00 01
   DestEndPoint      [1 byte ]  = 08
   ClusterId         [2 bytes]  = 07 03
   SrcEndPoint       [1 byte ]  = 08
   TxOptions         [1 byte ]  = 01
   Radius            [1 byte ]  = 05
   ZclTransactionID  [1 byte ]  = A5
   MessageID         [4 bytes]  = 00 00 00 01
   MessageControl    [1 byte ]  = 80
   StartTime         [4 bytes]  = 00 00 00 00
   Duration          [2 bytes]  = 00 01
   MessageLength     [1 byte ]  = 02
   Message           [2 bytes]  = 46 52
   CRC               [1 byte ]  = DF
RX: APSDE-DATA.Request 02 9C 00 28 02 01 00 00 00 00 00 00 00 08 09 01 03
07 08 11 09 A5 00 01 00 00 00 80 00 00 00 00 01 00 02 46 52 01 05 04 00
00 00 00 90
RX: ZclMessaging_DisplayMessage.Confirm 02 70 81 01 00 F0
RX: APSDE-DATA.Confirm 02 9D 00 11 02 01 00 00 00 00 00 00 00 08 08 00 00
00 00 00 04 8B
RX: APSDE-DATA.Indication 02 9D 01 21 02 00 00 08 02 01 00 08 09 01 03 07
05 18 A5 0B 00 00 00 00 03 D9 00 00 00 00 00 00 00 00 00 00 00 D9
RX: APSDE-DATA.Indication 02 9D 01 27 02 00 00 08 02 01 00 08 09 01 03 07
0B 11 45 01 01 00 00 00 27 49 09 70 00 00 03 D9 00 00 00 00 00 00 00 00
00 00 00 24
```

Figure 6-7. ZclMessaging_DisplayMessageReq Command Configuration

The InPremiseDisplay now shows the message in its LCD display and LED2 blinks to indicate that a Confirmation is required.

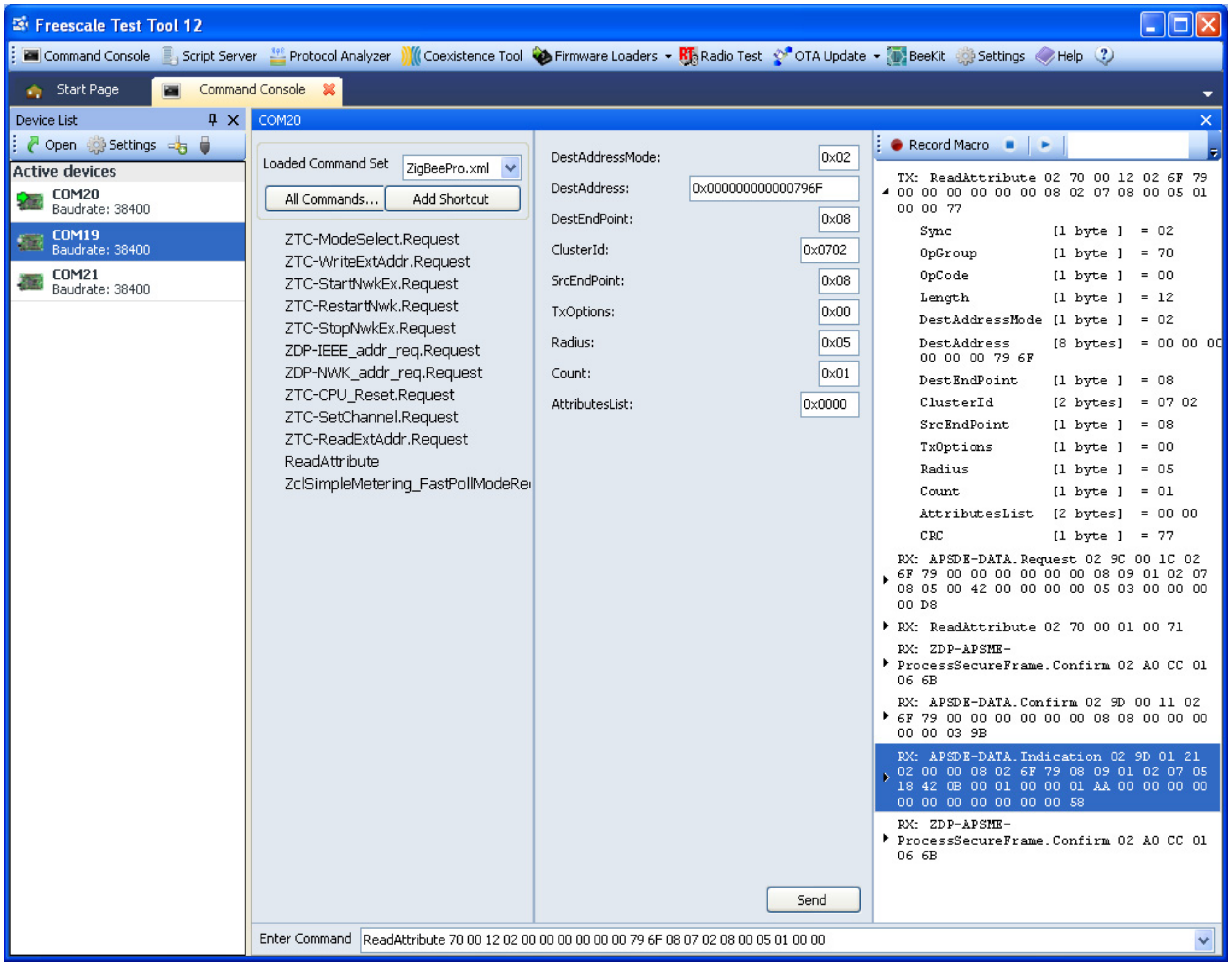
3. Press SW2 on the InPremiseDisplay board to send a confirmation back to the ESI.

NOTE

> The MeteringDevice does not have mandatory commands that can be used to request consumption information. However, it does use attribute commands. A read attribute command can be sent to the MeteringDevice to acquire its consumption data.

4. From Test Tool Command Console tool bar, click on All Commands -> ZigBee Cluster Library and select the ReadAttribute request command.
5. Set the parameters in the Request as follows (also shown in Figure 6-8):
   — DestAddress — To the address of the MeteringDevice (0x000000000000796F)
   — ClusterID — As the message cluster (0x0702)
   — Attribute list — To contain the ID of the “CurrentSummationDelivered” attribute (0x000)

   The response will be received as an APSDE-DATA.Indication.



DestAddressMode: 0x02
DestAddress: 0x000000000000796F
DestEndPoint: 0x08
ClusterId: 0x0702
SrcEndPoint: 0x08
TxOptions: 0x00
Radius: 0x05
Count: 0x01
AttributesList: 0x0000

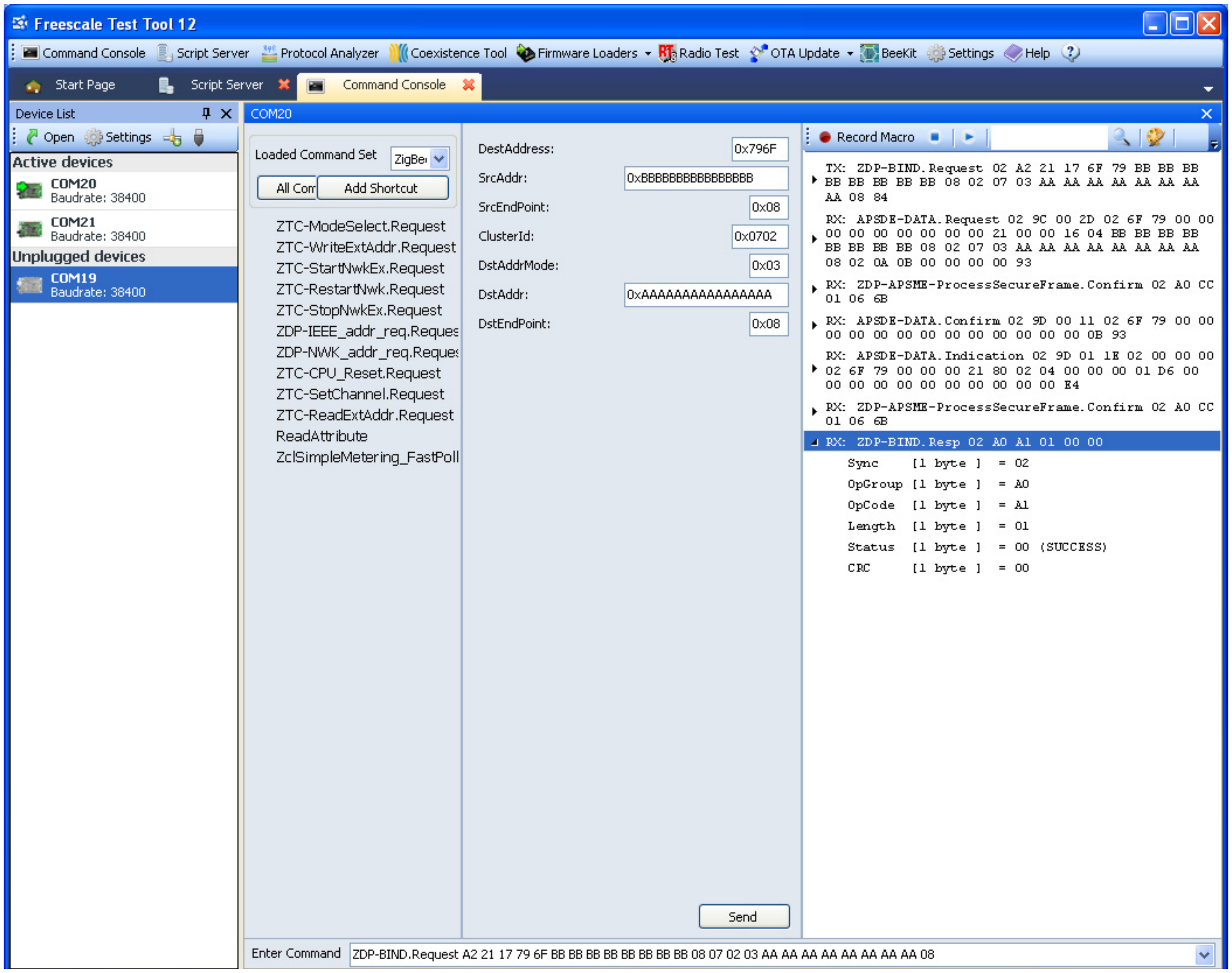
```
TX: ReadAttribute 02 70 00 12 02 6F 79
00 00 00 00 00 00 08 02 07 08 00 05 01
00 00 77
   Sync            [1 byte ]  = 02
   OpGroup         [1 byte ]  = 70
   OpCode          [1 byte ]  = 00
   Length          [1 byte ]  = 12
   DestAddressMode [1 byte ]  = 02
   DestAddress     [8 bytes]  = 00 00 00
   00 00 00 79 6F
   DestEndPoint    [1 byte ]  = 08
   ClusterId       [2 bytes]  = 07 02
   SrcEndPoint     [1 byte ]  = 08
   TxOptions       [1 byte ]  = 00
   Radius          [1 byte ]  = 05
   Count           [1 byte ]  = 01
   AttributesList  [2 bytes]  = 00 00
   CRC             [1 byte ]  = 77
RX: APSDE-DATA.Request 02 9C 00 1C 02
6F 79 00 00 00 00 00 00 08 09 01 02 07
08 05 00 42 00 00 00 00 05 03 00 00 00
00 D8
RX: ReadAttribute 02 70 00 01 00 71
RX: ZDP-APSME-
ProcessSecureFrame.Confirm 02 A0 CC 01
06 6B
RX: APSDE-DATA.Confirm 02 9D 00 11 02
6F 79 00 00 00 00 00 00 08 08 00 00 00
00 00 03 9B
RX: APSDE-DATA.Indication 02 9D 01 21
02 00 00 08 02 6F 79 08 09 01 02 07 05
18 42 0B 00 01 00 00 01 AA 00 00 00 00
00 00 00 00 00 00 00 58
RX: ZDP-APSME-
ProcessSecureFrame.Confirm 02 A0 CC 01
06 6B
```

Enter Command: ReadAttribute 70 00 12 02 00 00 00 00 00 00 79 6F 08 07 02 08 00 05 01 00 00

Figure 6-8. ReadAttribute Request Command Configuration

As an option, the MeteringDevice consumption data can be set to automatically report to the ESI based on a fixed time interval using the Configure reporting command. To obtain an Attribute report, set up a binding on the originator as follows.

1. From Test Tool Command Console tool bar click on All commands -> ZDO Layer commands and choose the ZDP-Bind.Request command.
2. Set the parameters in the Request as follows (also shown in Figure 6-9):
   - DestAddress — To the short address of the MeteringDevice (0x796F)
   - SrcAddr — To the IEEE address of the MeteringDevice
   - ClusterId — As the Simple Meter Cluster ID (0x0702)
   - DestAddrMode — To 0x03 (IEEE Address)
   - Destination address — To the IEEE of the ESI (0xAAAAAAAAAAAAAAAA)
   - Both SrcEndpoint and DstEndpoint to 0x08 (default endpoint of the demo applications)



DestAddress:	0x796F
SrcAddr:	0xBBBBBBBBBBBBBBBB
SrcEndPoint:	0x08
ClusterId:	0x0702
DstAddrMode:	0x03
DstAddr:	0xAAAAAAAAAAAAAAAA
DstEndPoint:	0x08

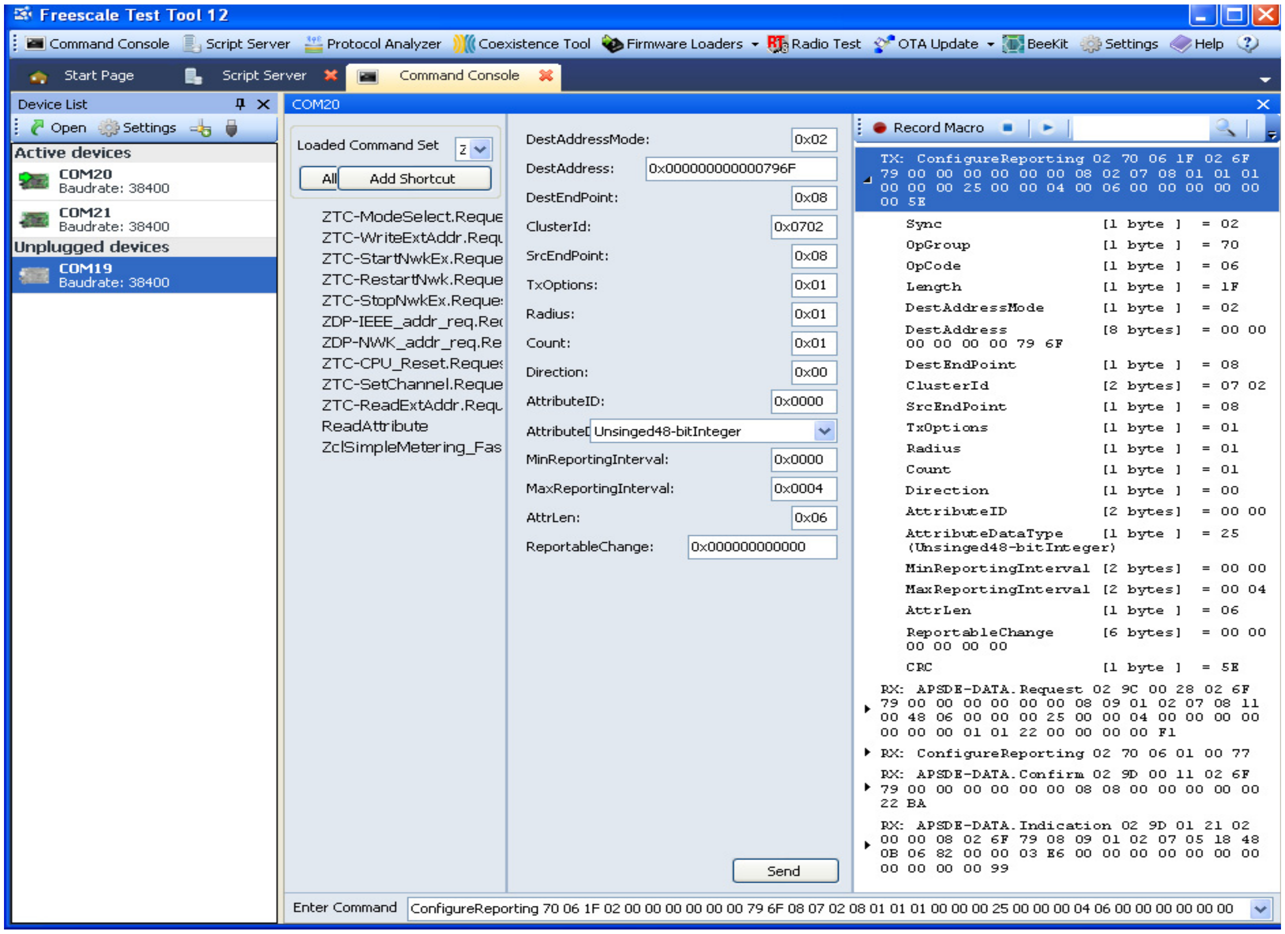
```
TX: ZDP-BIND.Request 02 A2 21 17 6F 79 BB BB BB
BB BB BB BB BB 08 02 07 03 AA AA AA AA AA AA AA
AA 08 84

RX: APSDE-DATA.Request 02 9C 00 2D 02 6F 79 00 00
00 00 00 00 00 00 00 21 00 00 16 04 BB BB BB BB
BB BB BB BB 08 02 07 03 AA AA AA AA AA AA AA AA
08 02 0A 0B 00 00 00 00 93

RX: ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC
01 06 6B

RX: APSDE-DATA.Confirm 02 9D 00 11 02 6F 79 00 00
00 00 00 00 00 00 00 00 00 00 00 0B 93

RX: APSDE-DATA.Indication 02 9D 01 1E 02 00 00 00
02 6F 79 00 00 00 21 80 02 04 00 00 00 01 D6 00
00 00 00 00 00 00 00 00 00 00 E4

RX: ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC
01 06 6B

RX: ZDP-BIND.Resp 02 A0 A1 01 00 00
   Sync     [1 byte ]  = 02
   OpGroup  [1 byte ]  = A0
   OpCode   [1 byte ]  = A1
   Length   [1 byte ]  = 01
   Status   [1 byte ]  = 00 (SUCCESS)
   CRC      [1 byte ]  = 00
```

```
ZDP-BIND.Request A2 21 17 79 6F BB BB BB BB BB BB BB BB 08 07 02 03 AA AA AA AA AA AA AA AA 08
```

Figure 6-9. ZDP-Bind.Request Command Configuration

Use the Configure Reporting command to configure a report after setting up the binding as follows:

1. From Test Tool Command Console tool bar click on All commands -> ZigBee Cluster Library Commands and select the ConfigureReporting command.
2. Set the parameters in the Request as follows (also shown in Figure 6-10):
   - — DestAddress — To the short address of the MeteringDevice (0x796F)
   - — ClusterId — As the Simple Meter Cluster ID (0x0702)
   - — DestAddrMode — To 0x03 (IEEE Address)
   - — Both SrcEndpoint and DstEndpoint — To 0x08 (default endpoint of the demo applications)
   - — Attribute Id — To contain the ID of the "CurrentSummationDelivered" attribute (0x000)
   - — Attribute data type — To "48 bit unsigned integer"
   - — Direction to 0x00 — To indicate that the report is from the client side to the server side
   - — MinReportingInterval — To 0x000 and the MaxReportingInterval — To 0x0004 (This forces the Metering Device to send out the attribute every 4 seconds.)
   - — Attribute Lenght — Ro 6 (length of a 48 bit integer)
   - — Reportable Change — To 0x00,0x00,0x00,0x00,0x00,0x00



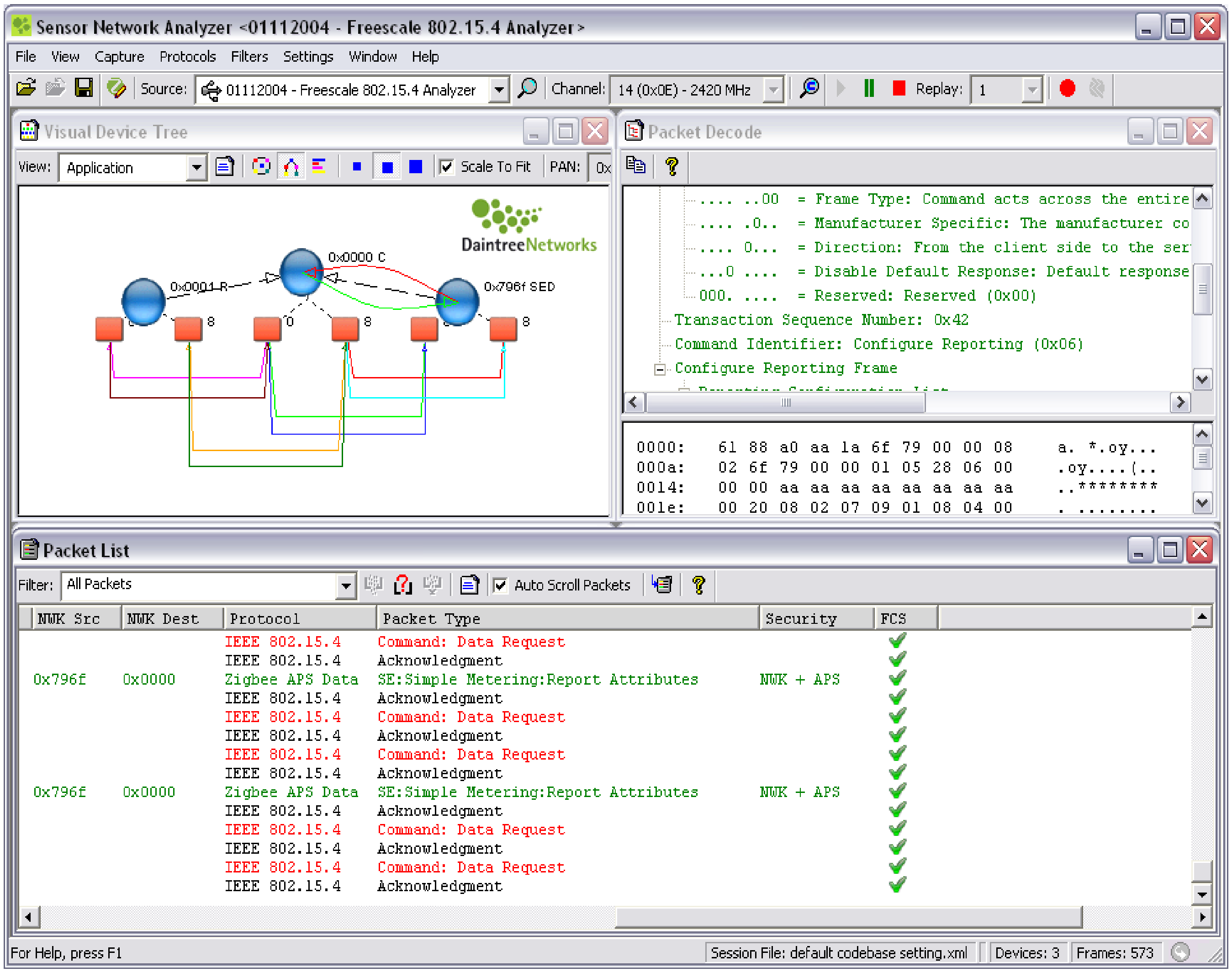
DestAddressMode:	0x02
DestAddress:	0x000000000000796F
DestEndPoint:	0x08
ClusterId:	0x0702
SrcEndPoint:	0x08
TxOptions:	0x01
Radius:	0x01
Count:	0x01
Direction:	0x00
AttributeID:	0x0000
Attributel	Unsinged48-bitInteger
MinReportingInterval:	0x0000
MaxReportingInterval:	0x0004
AttrLen:	0x06
ReportableChange:	0x000000000000

```
TX:  ConfigureReporting 02 70 06 1F 02 6F
79 00 00 00 00 00 00 08 02 07 08 01 01 01
00 00 00 25 00 00 04 00 06 00 00 00 00 00
00 5E
     Sync                  [1 byte ]  = 02
     OpGroup               [1 byte ]  = 70
     OpCode                [1 byte ]  = 06
     Length                [1 byte ]  = 1F
     DestAddressMode       [1 byte ]  = 02
     DestAddress           [8 bytes]  = 00 00
     00 00 00 00 79 6F
     DestEndPoint          [1 byte ]  = 08
     ClusterId             [2 bytes]  = 07 02
     SrcEndPoint           [1 byte ]  = 08
     TxOptions             [1 byte ]  = 01
     Radius                [1 byte ]  = 01
     Count                 [1 byte ]  = 01
     Direction             [1 byte ]  = 00
     AttributeID           [2 bytes]  = 00 00
     AttributeDataType     [1 byte ]  = 25
     (Unsinged48-bitInteger)
     MinReportingInterval  [2 bytes]  = 00 00
     MaxReportingInterval  [2 bytes]  = 00 04
     AttrLen               [1 byte ]  = 06
     ReportableChange      [6 bytes]  = 00 00
     00 00 00 00
     CRC                   [1 byte ]  = 5E
RX:  APSDE-DATA.Request 02 9C 00 28 02 6F
79 00 00 00 00 00 00 08 09 01 02 07 08 11
00 48 06 00 00 00 25 00 00 04 00 00 00 00
00 00 00 01 01 22 00 00 00 00 F1
RX:  ConfigureReporting 02 70 06 01 00 77
RX:  APSDE-DATA.Confirm 02 9D 00 11 02 6F
79 00 00 00 00 00 00 08 08 00 00 00 00 00
22 BA
RX:  APSDE-DATA.Indication 02 9D 01 21 02
00 00 08 02 6F 79 08 09 01 02 07 05 18 48
0B 06 82 00 00 03 E6 00 00 00 00 00 00 00
00 00 00 00 99
```

Enter Command: ConfigureReporting 70 06 1F 02 00 00 00 00 00 00 79 6F 08 07 02 08 01 01 01 00 00 00 25 00 00 00 04 06 00 00 00 00 00 00

Figure 6-10. ConfigureReporting Command Configuration

The MeteringDevice now sends out an attribute report every four seconds that simulates meter consumption as shown in Figure 6-11.


Sensor Network Analyzer <01112004 - Freescale 802.15.4 Analyzer>
File View Capture Protocols Filters Settings Window Help
Source: 01112004 - Freescale 802.15.4 Analyzer
Channel: 14 (0x0E) - 2420 MHz
Replay: 1
Visual Device Tree
View: Application
Scale To Fit
PAN: 0x
DaintreeNetworks
0x0000 C
0x0001 R
0x796f SED
8
0
8
8
Packet Decode
Packet List
Filter: All Packets
Auto Scroll Packets
For Help, press F1
Session File: default codebase setting.xml
Devices: 3
Frames: 573

```
.... ..00   = Frame Type: Command acts across the entire
.... .0..   = Manufacturer Specific: The manufacturer co
.... 0...   = Direction: From the client side to the ser
...0 ....   = Disable Default Response: Default response
000. ....   = Reserved: Reserved (0x00)
Transaction Sequence Number: 0x42
Command Identifier: Configure Reporting (0x06)
Configure Reporting Frame
```

```
0000:   61 88 a0 aa 1a 6f 79 00 00 08    a. *.oy...
000a:   02 6f 79 00 00 01 05 28 06 00    .oy....(..
0014:   00 00 aa aa aa aa aa aa aa aa    ..********
001e:   00 20 08 02 07 09 01 08 04 00    . ........
```

```
 NWK Src   NWK Dest   Protocol        Packet Type                          Security   FCS
                      IEEE 802.15.4   Command: Data Request                           ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
 0x796f    0x0000     Zigbee APS Data SE:Simple Metering:Report Attributes NWK + APS  ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
                      IEEE 802.15.4   Command: Data Request                           ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
                      IEEE 802.15.4   Command: Data Request                           ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
 0x796f    0x0000     Zigbee APS Data SE:Simple Metering:Report Attributes NWK + APS  ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
                      IEEE 802.15.4   Command: Data Request                           ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
                      IEEE 802.15.4   Command: Data Request                           ✓
                      IEEE 802.15.4   Acknowledgment                                  ✓
```

Figure 6-11. Metering Device Attribute Reporting

NOTE

Certain Smart Energy clusters require that APS/Link key security is applied. This is accomplished by setting the TxOptions field to 0x01.

6.4 Demand Response and Load Control (DRLC) and Price Cluster Demonstration

To demonstrate the Demand Response and Load Control (DRLC) and Price Cluster capabilities requires three sample applications. These applications must be generated using the BeeKit Wireless Connectivity ToolKit software.

NOTE

Before performing any of the steps described in this section, users must understand DRLC and Price Cluster commands. Refer to Chapter 7, "Example Applications" for a better understanding the sample applications used in this section and the Smart Energy Specification. The commands sent from Test Tool in the following steps use the "DestAddress" set to the short address of the device that will receive the request.

6.4.1 Creating the Projects and Configuring the Network

Perform the following tasks to create the projects and configure the network for this demonstration:

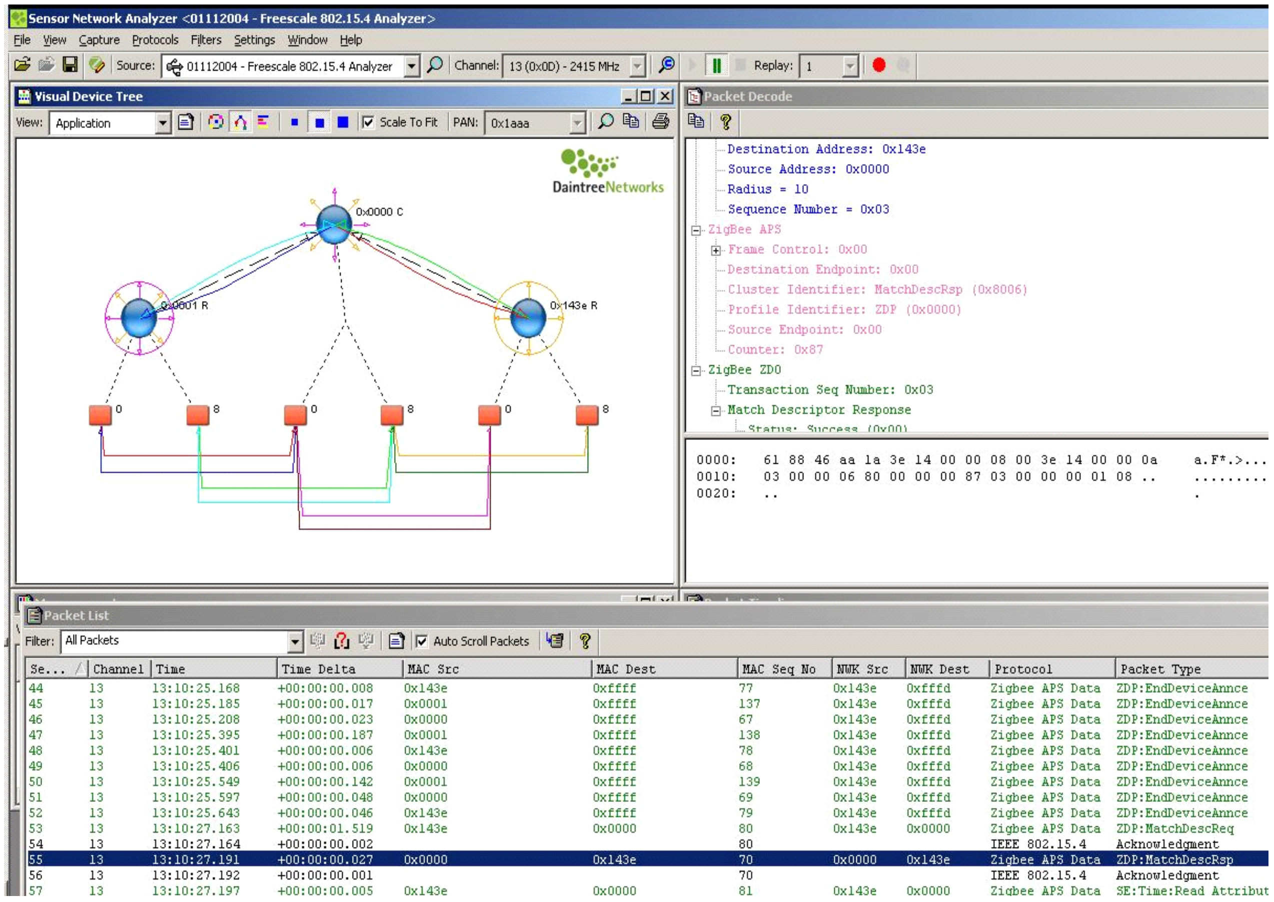
1. Create and generate the following projects using the BeeKit Wireless Connectivity ToolKit software:
   - Se Energy Service Interface (ESI)
   - Se PCT (Ensure that the Circulation Fan device is selected on the application properties)
   - Se LoadControlDevice

   Ensure that the hardware settings are configured according to the selected target boards. (1322x Sensor Node or 1322x Network Node)
2. Compile the projects and program the boards using the IAR Embedded Workbench for ARM.
3. Connect the Energy Service Interface (ESI) to a PC using the USB port on the board and configure the Freescale Test Tool to communicate with the ESI as described in Section 6.3.1, "Board Connection and Network Startup". This ensures that the ESI is started (network formed) and can communicate properly with Test Tool.
4. Press SW1 on the boards serving as the "Se PCT" and "Se LoadControlDevice" to join the two nodes to the ESI. In this example, the "Se PCT" node has the short address 0x143E and the "Se LoadControlDevice"node has the short address 0x0001.

NOTE

These short addresses may vary depending on stack profile and how many nodes are in the network

A working Smart Energy Network is shown in Figure 6-12.


Sensor Network Analyzer <01112004 - Freescale 802.15.4 Analyzer>
File View Capture Protocols Filters Settings Window Help
Source: 01112004 - Freescale 802.15.4 Analyzer
Channel: 13 (0x0D) - 2415 MHz
Replay: 1
Visual Device Tree
View: Application
Scale To Fit
PAN: 0x1aaa
DaintreeNetworks
0x0000 C
0x0001 R
0x143e R
0
8
0
8
0
8
Packet Decode
Packet List
Filter: All Packets
Auto Scroll Packets

```
Destination Address: 0x143e
Source Address: 0x0000
Radius = 10
Sequence Number = 0x03
ZigBee APS
   Frame Control: 0x00
   Destination Endpoint: 0x00
   Cluster Identifier: MatchDescRsp (0x8006)
   Profile Identifier: ZDP (0x0000)
   Source Endpoint: 0x00
   Counter: 0x87
ZigBee ZDO
   Transaction Seq Number: 0x03
   Match Descriptor Response
      Status: Success (0x00)
```

```
0000:   61 88 46 aa 1a 3e 14 00 00 08 00 3e 14 00 00 0a    a.F*.>...
0010:   03 00 00 06 80 00 00 00 87 03 00 00 00 01 08 ..    .........
0020:   ..                                                 .
```

Se...	Channel	Time	Time Delta	MAC Src	MAC Dest	MAC Seq No	NWK Src	NWK Dest	Protocol	Packet Type
44	13	13:10:25.168	+00:00:00.008	0x143e	0xffff	77	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
45	13	13:10:25.185	+00:00:00.017	0x0001	0xffff	137	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
46	13	13:10:25.208	+00:00:00.023	0x0000	0xffff	67	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
47	13	13:10:25.395	+00:00:00.187	0x0001	0xffff	138	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
48	13	13:10:25.401	+00:00:00.006	0x143e	0xffff	78	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
49	13	13:10:25.406	+00:00:00.006	0x0000	0xffff	68	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
50	13	13:10:25.549	+00:00:00.142	0x0001	0xffff	139	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
51	13	13:10:25.597	+00:00:00.048	0x0000	0xffff	69	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
52	13	13:10:25.643	+00:00:00.046	0x143e	0xffff	79	0x143e	0xfffd	Zigbee APS Data	ZDP:EndDeviceAnnce
53	13	13:10:27.163	+00:00:01.519	0x143e	0x0000	80	0x143e	0x0000	Zigbee APS Data	ZDP:MatchDescReq
54	13	13:10:27.164	+00:00:00.002			80			IEEE 802.15.4	Acknowledgment
55	13	13:10:27.191	+00:00:00.027	0x0000	0x143e	70	0x0000	0x143e	Zigbee APS Data	ZDP:MatchDescRsp
56	13	13:10:27.192	+00:00:00.001			70			IEEE 802.15.4	Acknowledgment
57	13	13:10:27.197	+00:00:00.005	0x143e	0x0000	81	0x143e	0x0000	Zigbee APS Data	SE:Time:Read Attribut

Figure 6-12. Smart Energy Network

5. After the “Se PCT” and “Se LoadControlDevice” nodes have joined (serial flash stops, LED1 on), perform a long press on SW1 to enter application mode.

   On the “Se LoadControlDevice” node, by default, the duty cycle is set to 90%. This means that LED2 is Off for 0.5 seconds and then On for 4.5 seconds which is a duty cycle period of 5 seconds. If a display is connected to the board, the following message appears:

   DutyCycle=090%ON

   On the “Se PCT” node that acts as a Circulation Fan device, the default fan speed is set to 2 (50%). This means that LED1 flashes at a (800/2) millisecond interval. If a display is connected to the board, the following message appears:

   Speed= 2 (050%)

6. On both the “Se PCT” and “Se LoadControlDevice” nodes, perform a long press on SW3. LED3 turns “On”. This means that voluntary load control events will be accepted. By default, “Se LoadControlDevice” node is in Load Control Mode.

7. On the “Se LoadControlDevice” node, users can change the duty cycle using SW1 and SW2. Set the duty cycle to 100% (LED2 On).

   On the “Se PCT” node, users can change the fan speed using SW1 and SW2. Set the fan speed to four (100% duty cycle and LED1 flashes at a (800/4) millisecond interval).

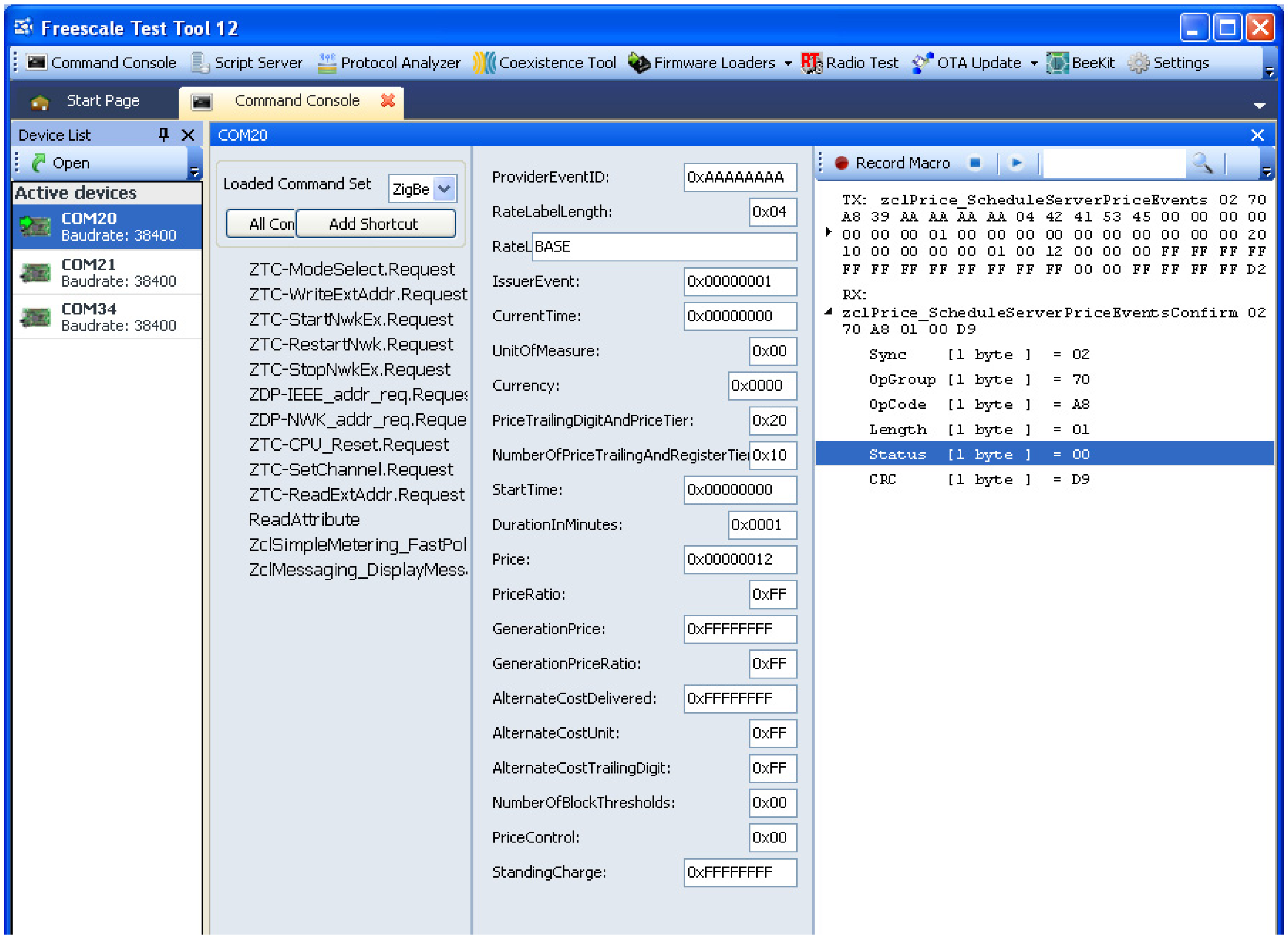
6.4.2 Sending DRLC and Price Commands

All the Load Control Event requests sent to the “Se LoadControlDevice” node have the Utility Group field set to 0x00 and the Device Group Class field set to 0x0380. For the “Se PCT” node, the ESI should send the Load Control Event requests with Utility Group field set to 0x00 and the Device Group Class field set to 0x0001. The Utility Group attribute can be changed using Test Tool by making a ZCL Write Attribute command. All of the Smart Energy commands described and used in this section are found by clicking Test Tool -> All Commands -> ZigBee Cluster Library Commands.

After performing the steps in Section 6.4.1, “Creating the Projects and Configuring the Network”, perform the following steps to use the Smart Energy Price and Load Control commands:

1. Configure the ESI with initial pricing information. To do this, from Test Tool, send a “Schedule Server Price Events command” with the field values as shown in Figure 6-13.

   This command loads a price event with a duration of one minute and the price value at 0.18. (See the Smart Energy Profile Specification for detailed field descriptions.) This does not send the Publish Price response over the air. After sending this command, ensure that the confirmation shows the “Success” (0x00) status.



ProviderEventID:	0xAAAAAAAA
RateLabelLength:	0x04
RateL	BASE
IssuerEvent:	0x00000001
CurrentTime:	0x00000000
UnitOfMeasure:	0x00
Currency:	0x0000
PriceTrailingDigitAndPriceTier:	0x20
NumberOfPriceTrailingAndRegisterTie	0x10
StartTime:	0x00000000
DurationInMinutes:	0x0001
Price:	0x00000012
PriceRatio:	0xFF
GenerationPrice:	0xFFFFFFFF
GenerationPriceRatio:	0xFF
AlternateCostDelivered:	0xFFFFFFFF
AlternateCostUnit:	0xFF
AlternateCostTrailingDigit:	0xFF
NumberOfBlockThresholds:	0x00
PriceControl:	0x00
StandingCharge:	0xFFFFFFFF

```
TX: zclPrice_ScheduleServerPriceEvents 02 70
A8 39 AA AA AA AA 04 42 41 53 45 00 00 00 00
00 00 00 01 00 00 00 00 00 00 00 00 00 00 20
10 00 00 00 00 01 00 12 00 00 00 FF FF FF FF
FF FF FF FF FF FF FF FF 00 00 FF FF FF FF D2

RX:
zclPrice_ScheduleServerPriceEventsConfirm 02
70 A8 01 00 D9
    Sync     [1 byte ]  = 02
    OpGroup  [1 byte ]  = 70
    OpCode   [1 byte ]  = A8
    Length   [1 byte ]  = 01
    Status   [1 byte ]  = 00
    CRC      [1 byte ]  = D9
```

Figure 6-13. Schedule Server Price Events Command

2. From Test Tool, send a “Load Control Event request” with the field values as shown in Figure 6-14.

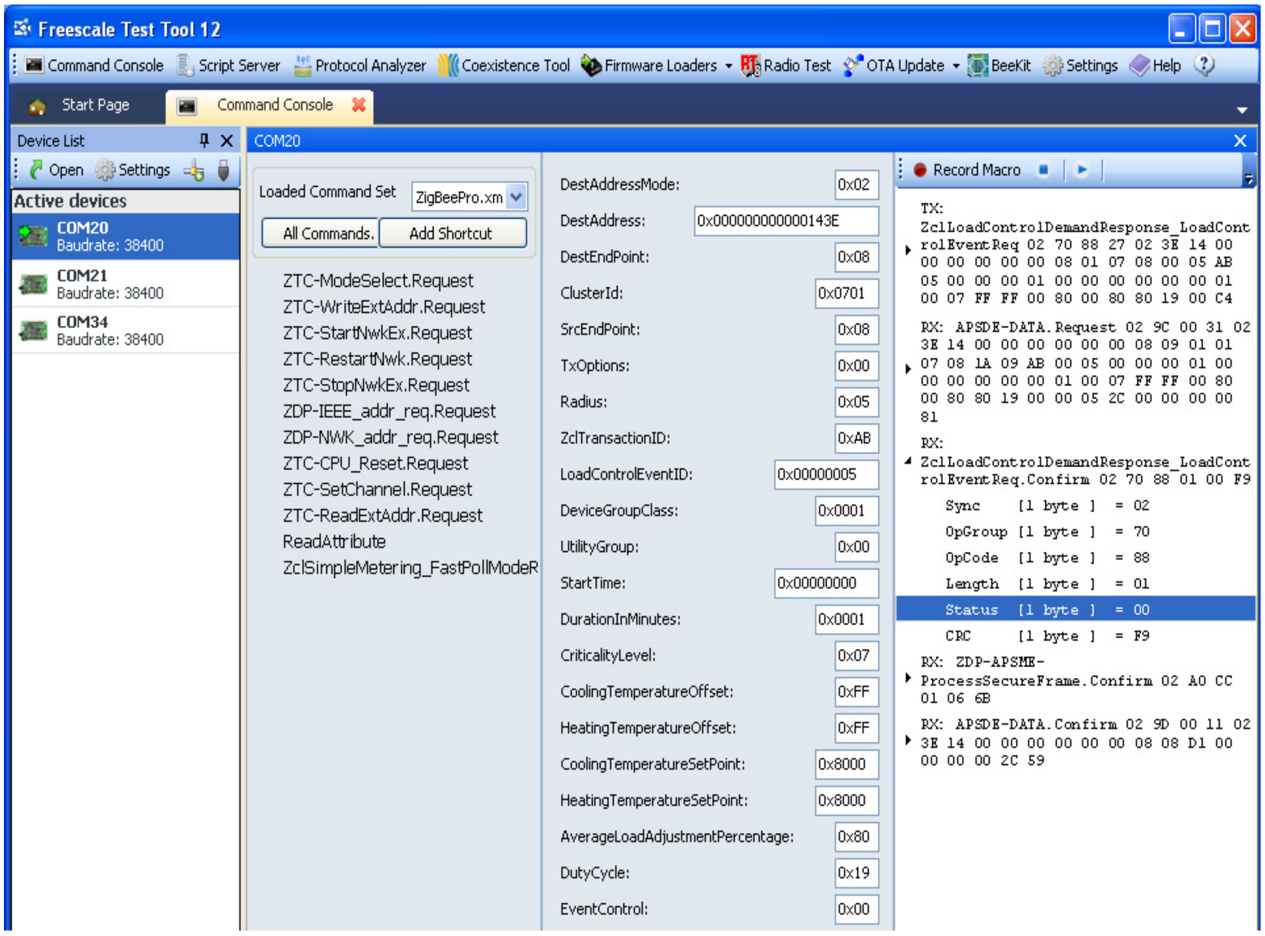
   This command specifies a voluntary load control event (critical level is 0x04) that starts immediately, has a duration of one minute and a duty cycle of 50% (0x32). The request is sent over the air from the ESI to the “Se LoadControlDevice” node (short address is 0x0001), to set the duty cycle to 50%.

   On the “Se LoadControlDevice” node, the execution of the received load control event is signified as follows:

   — LED1 flashes during event duration

   — LED2 has a duty cycle of 2.5 seconds Off and 2.5 seconds On

   After one minute, the event completes and the duty cycle is set back to the duty cycle value that was configured by users before receiving the event. (100% after Step 7 in Section 6.4.1, “Creating the Projects and Configuring the Network”.)



DestAddressMode:	0x02
DestAddress:	0x000000000000143E
DestEndPoint:	0x08
ClusterId:	0x0701
SrcEndPoint:	0x08
TxOptions:	0x00
Radius:	0x05
ZclTransactionID:	0xAB
LoadControlEventID:	0x00000005
DeviceGroupClass:	0x0001
UtilityGroup:	0x00
StartTime:	0x00000000
DurationInMinutes:	0x0001
CriticalityLevel:	0x07
CoolingTemperatureOffset:	0xFF
HeatingTemperatureOffset:	0xFF
CoolingTemperatureSetPoint:	0x8000
HeatingTemperatureSetPoint:	0x8000
AverageLoadAdjustmentPercentage:	0x80
DutyCycle:	0x19
EventControl:	0x00

```
TX:
ZclLoadControlDemandResponse_LoadCont
rolEventReq 02 70 88 27 02 3E 14 00
00 00 00 00 00 08 01 07 08 00 05 AB
05 00 00 00 01 00 00 00 00 00 00 01
00 07 FF FF 00 80 00 80 80 19 00 C4

RX: APSDE-DATA.Request 02 9C 00 31 02
3E 14 00 00 00 00 00 00 08 09 01 01
07 08 1A 09 AB 00 05 00 00 00 01 00
00 00 00 00 00 01 00 07 FF FF 00 80
00 80 80 19 00 00 05 2C 00 00 00 00
81

RX:
ZclLoadControlDemandResponse_LoadCont
rolEventReq.Confirm 02 70 88 01 00 F9
   Sync     [1 byte ]  = 02
   OpGroup  [1 byte ]  = 70
   OpCode   [1 byte ]  = 88
   Length   [1 byte ]  = 01
   Status   [1 byte ]  = 00
   CRC      [1 byte ]  = F9

RX: ZDP-APSME-
ProcessSecureFrame.Confirm 02 A0 CC
01 06 6B

RX: APSDE-DATA.Confirm 02 9D 00 11 02
3E 14 00 00 00 00 00 00 08 08 D1 00
00 00 00 2C 59
```

Figure 6-14. Load Control Event Request

3. On the “Se LoadControlDevice” node, perform a long press on SW4 to place the device into Price Mode (LED4 is On). This means that the adjustment of the duty cycle is based on price instead of DRLC events, except when a critical/mandatory load control event is received.

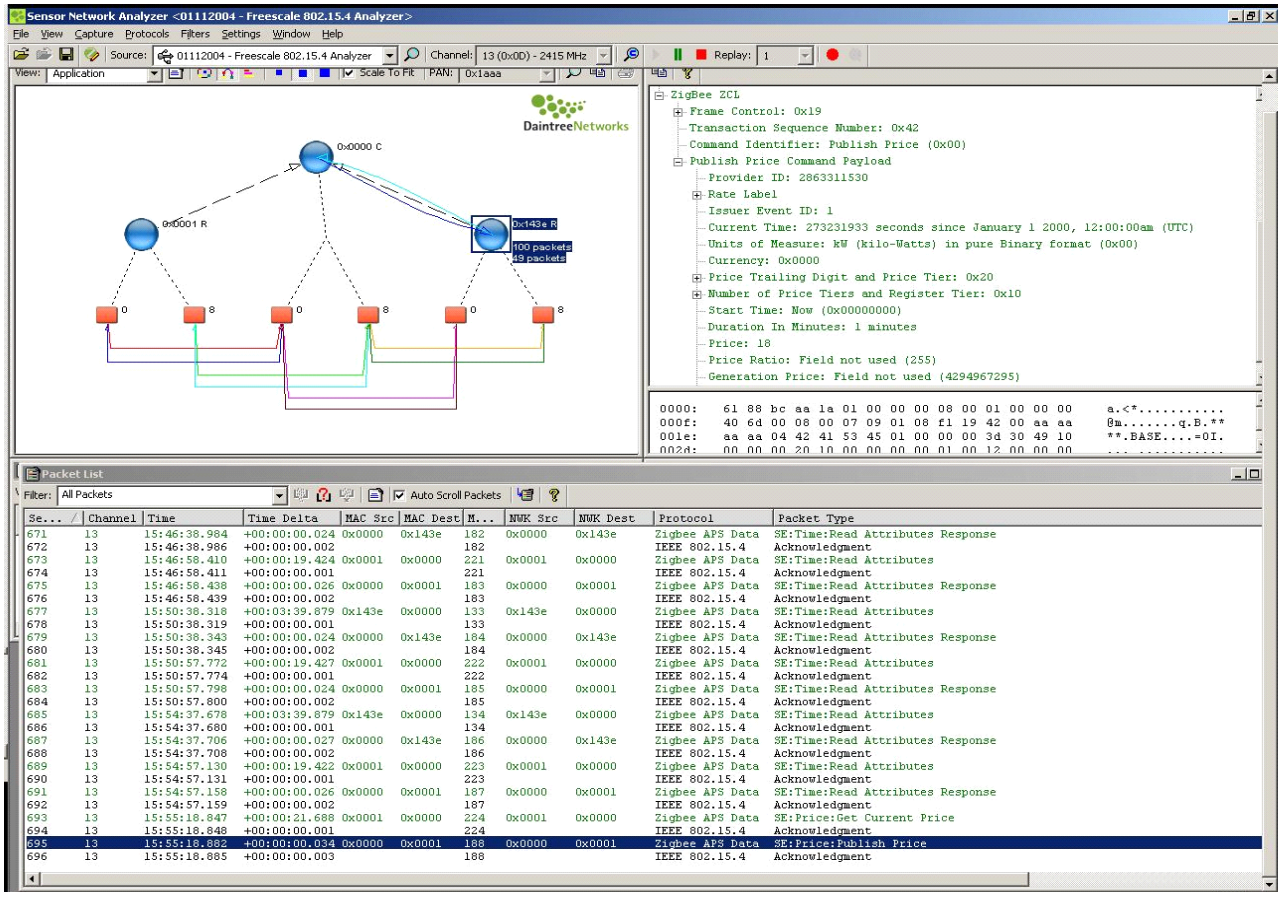
   If the price was loaded in the ESI (as shown in Step 1), a long press of SW4 will send a “Get Current Price command” to the ESI with the Requester Rx On When Idle sub-field value of one. (See the Smart Energy Specification for more price field details.) This means the ESI always sends price updates to the “Se LoadControlDevice” node without receiving a “Get Current Price command”.

   The “Se LoadControlDevice” node receives a “Publish Price Response” with the price value of 0.18. This means the device sets the duty cycle to 70% On for the event duration. (The device is in the “On state” for 70% of the 5 second duty cycle).

   LED1 flashes during the event duration and LED2 shows the duty cycle. (See Chapter 7, “Example Applications” for more details about the “Se LoadControlDevice” application.)

   When a price event expires, the duty cycle is set back to the duty cycle value that was configured by users before receiving the event. (To 100% after Step 7 in Section 6.4.1, “Creating the Projects and Configuring the Network”).

Figure 6-15 shows a Get Current Price command with the Requester Rx On When Idle sub-field value of one as well as the Publish Price.


Sensor Network Analyzer <01112004 - Freescale 802.15.4 Analyzer>
File View Capture Protocols Filters Settings Window Help
Source: 01112004 - Freescale 802.15.4 Analyzer
Channel: 13 (0x0D) - 2415 MHz
Replay: 1
View: Application
Scale To Fit
PAN: 0x1aaa
DaintreeNetworks
0x0000 C
0x0001 R
0x143e R
100 packets
49 packets
Packet List
Filter: All Packets
Auto Scroll Packets

```
ZigBee ZCL
  Frame Control: 0x19
  Transaction Sequence Number: 0x42
  Command Identifier: Publish Price (0x00)
  Publish Price Command Payload
    Provider ID: 2863311530
    Rate Label
    Issuer Event ID: 1
    Current Time: 273231933 seconds since January 1 2000, 12:00:00am (UTC)
    Units of Measure: kW (kilo-Watts) in pure Binary format (0x00)
    Currency: 0x0000
    Price Trailing Digit and Price Tier: 0x20
    Number of Price Tiers and Register Tier: 0x10
    Start Time: Now (0x00000000)
    Duration In Minutes: 1 minutes
    Price: 18
    Price Ratio: Field not used (255)
    Generation Price: Field not used (4294967295)
```

```
0000:   61 88 bc aa 1a 01 00 00 00 08 00 01 00 00 00    a.<*...........
000f:   40 6d 00 08 00 07 09 01 08 f1 19 42 00 aa aa    @m.......q.B.**
001e:   aa aa 04 42 41 53 45 01 00 00 00 3d 30 49 10    **.BASE....=0I.
002d:   00 00 00 20 10 00 00 00 00 01 00 12 00 00 00    ... ...........
```

```
Se...  Channel Time          Time Delta        MAC Src  MAC Dest M...  NWK Src  NWK Dest  Protocol          Packet Type
671    13      15:46:38.984  +00:00:00.024 0x0000   0x143e   182   0x0000   0x143e    Zigbee APS Data   SE:Time:Read Attributes Response
672    13      15:46:38.986  +00:00:00.002                   182                      IEEE 802.15.4     Acknowledgment
673    13      15:46:58.410  +00:00:19.424 0x0001   0x0000   221   0x0001   0x0000    Zigbee APS Data   SE:Time:Read Attributes
674    13      15:46:58.411  +00:00:00.001                   221                      IEEE 802.15.4     Acknowledgment
675    13      15:46:58.438  +00:00:00.026 0x0000   0x0001   183   0x0000   0x0001    Zigbee APS Data   SE:Time:Read Attributes Response
676    13      15:46:58.439  +00:00:00.002                   183                      IEEE 802.15.4     Acknowledgment
677    13      15:50:38.318  +00:03:39.879 0x143e   0x0000   133   0x143e   0x0000    Zigbee APS Data   SE:Time:Read Attributes
678    13      15:50:38.319  +00:00:00.001                   133                      IEEE 802.15.4     Acknowledgment
679    13      15:50:38.343  +00:00:00.024 0x0000   0x143e   184   0x0000   0x143e    Zigbee APS Data   SE:Time:Read Attributes Response
680    13      15:50:38.345  +00:00:00.002                   184                      IEEE 802.15.4     Acknowledgment
681    13      15:50:57.772  +00:00:19.427 0x0001   0x0000   222   0x0001   0x0000    Zigbee APS Data   SE:Time:Read Attributes
682    13      15:50:57.774  +00:00:00.001                   222                      IEEE 802.15.4     Acknowledgment
683    13      15:50:57.798  +00:00:00.024 0x0000   0x0001   185   0x0000   0x0001    Zigbee APS Data   SE:Time:Read Attributes Response
684    13      15:50:57.800  +00:00:00.002                   185                      IEEE 802.15.4     Acknowledgment
685    13      15:54:37.678  +00:03:39.879 0x143e   0x0000   134   0x143e   0x0000    Zigbee APS Data   SE:Time:Read Attributes
686    13      15:54:37.680  +00:00:00.001                   134                      IEEE 802.15.4     Acknowledgment
687    13      15:54:37.706  +00:00:00.027 0x0000   0x143e   186   0x0000   0x143e    Zigbee APS Data   SE:Time:Read Attributes Response
688    13      15:54:37.708  +00:00:00.002                   186                      IEEE 802.15.4     Acknowledgment
689    13      15:54:57.130  +00:00:19.422 0x0001   0x0000   223   0x0001   0x0000    Zigbee APS Data   SE:Time:Read Attributes
690    13      15:54:57.131  +00:00:00.001                   223                      IEEE 802.15.4     Acknowledgment
691    13      15:54:57.158  +00:00:00.026 0x0000   0x0001   187   0x0000   0x0001    Zigbee APS Data   SE:Time:Read Attributes Response
692    13      15:54:57.159  +00:00:00.002                   187                      IEEE 802.15.4     Acknowledgment
693    13      15:55:18.847  +00:00:21.688 0x0001   0x0000   224   0x0001   0x0000    Zigbee APS Data   SE:Price:Get Current Price
694    13      15:55:18.848  +00:00:00.001                   224                      IEEE 802.15.4     Acknowledgment
695    13      15:55:18.882  +00:00:00.034 0x0000   0x0001   188   0x0000   0x0001    Zigbee APS Data   SE:Price:Publish Price
696    13      15:55:18.885  +00:00:00.003                   188                      IEEE 802.15.4     Acknowledgment
```

Figure 6-15. Get Current Price Command Values

4. After the received event as described in Step 3 completes, from Test Tool, select the “Update Server Price command” and enter the field values as shown in Figure 6-16.

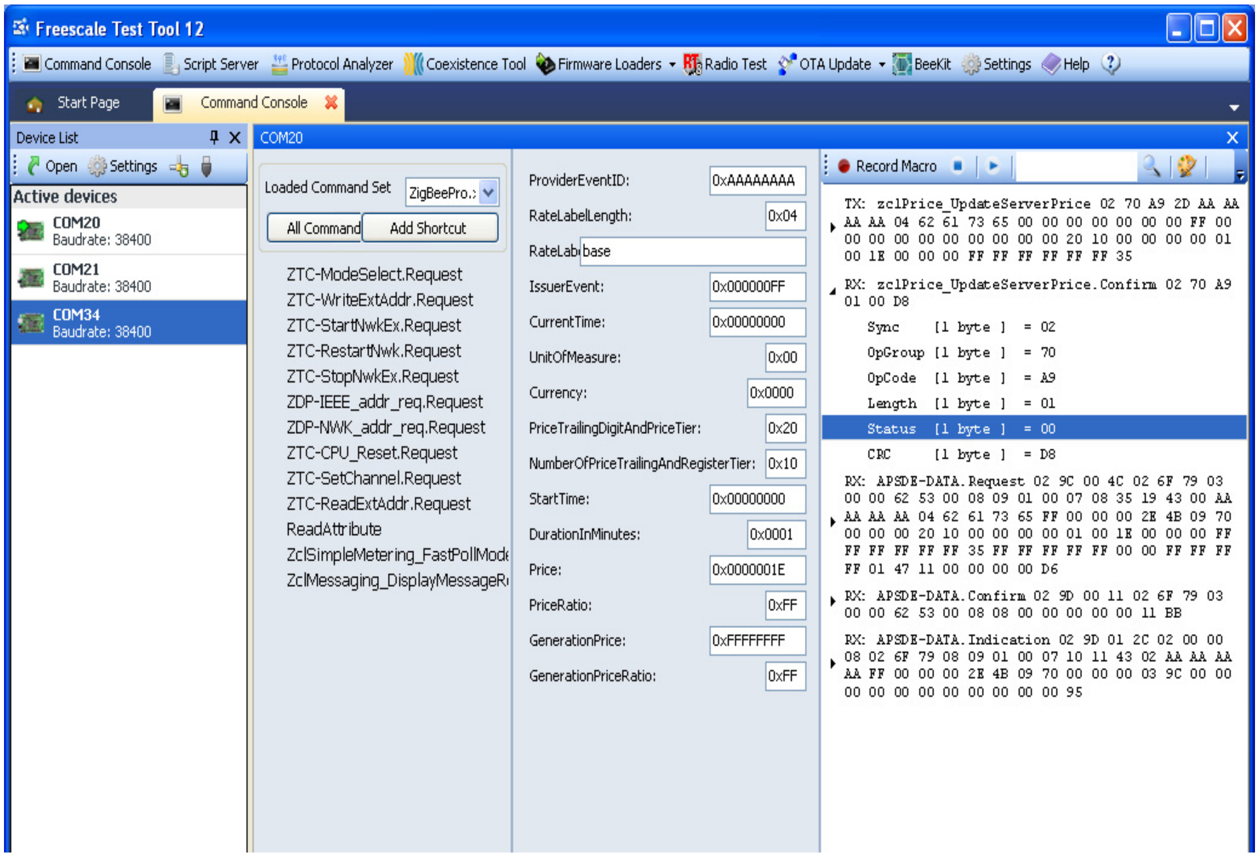
NOTE

The “Update Server Price command” updates a price event in the ESI that previously was loaded using the “Schedule Server Price Events command” from Test Tool. An “Update Server Price command” updates a loaded event if it has the same “Start Time”, “DurationInMinutes” and “ProviderEventId” as the loaded event and the “IssuerEvent” field has a value higher than the loaded event.

5. Send the command to the ESI and check if the confirmation is received with “Success” (0x00) status. If any other status is received, ensure to comply with the notes previously described in this section.

   If the status is “Success”, the ESI updates the price event fields and sends a “Publish Price Response” with the updated price fields over the air. The price is updated with a value of 0.30 (the price is 0x000000001E and Price Trailing Digit is 2). This means that the device sets the duty cycle to 50% for the event duration. (The device is in an “On” state for 50% of the 5 second duty cycle.) LED1 flashes during the event duration and LED2 shows the duty cycle.

   When the price event expires, the duty cycle is set back to the value that was configured by users before receiving the event.



ProviderEventID:	0xAAAAAAAA
RateLabelLength:	0x04
RateLab	base
IssuerEvent:	0x000000FF
CurrentTime:	0x00000000
UnitOfMeasure:	0x00
Currency:	0x0000
PriceTrailingDigitAndPriceTier:	0x20
NumberOfPriceTrailingAndRegisterTier:	0x10
StartTime:	0x00000000
DurationInMinutes:	0x0001
Price:	0x0000001E
PriceRatio:	0xFF
GenerationPrice:	0xFFFFFFFF
GenerationPriceRatio:	0xFF

```
TX: zclPrice_UpdateServerPrice 02 70 A9 2D AA AA
AA AA 04 62 61 73 65 00 00 00 00 00 00 00 FF 00
00 00 00 00 00 00 00 00 00 20 10 00 00 00 00 01
00 1E 00 00 00 FF FF FF FF FF FF 35

RX: zclPrice_UpdateServerPrice.Confirm 02 70 A9
01 00 D8
   Sync    [1 byte ]  = 02
   OpGroup [1 byte ]  = 70
   OpCode  [1 byte ]  = A9
   Length  [1 byte ]  = 01
   Status  [1 byte ]  = 00
   CRC     [1 byte ]  = D8

RX: APSDE-DATA.Request 02 9C 00 4C 02 6F 79 03
00 00 62 53 00 08 09 01 00 07 08 35 19 43 00 AA
AA AA AA 04 62 61 73 65 FF 00 00 00 2E 4B 09 70
00 00 00 20 10 00 00 00 00 01 00 1E 00 00 00 FF
FF FF FF FF FF 35 FF FF FF FF FF 00 00 FF FF FF
FF 01 47 11 00 00 00 00 D6

RX: APSDE-DATA.Confirm 02 9D 00 11 02 6F 79 03
00 00 62 53 00 08 08 00 00 00 00 00 11 BB

RX: APSDE-DATA.Indication 02 9D 01 2C 02 00 00
08 02 6F 79 08 09 01 00 07 10 11 43 02 AA AA AA
AA FF 00 00 00 2E 4B 09 70 00 00 00 03 9C 00 00
00 00 00 00 00 00 00 00 00 95
```

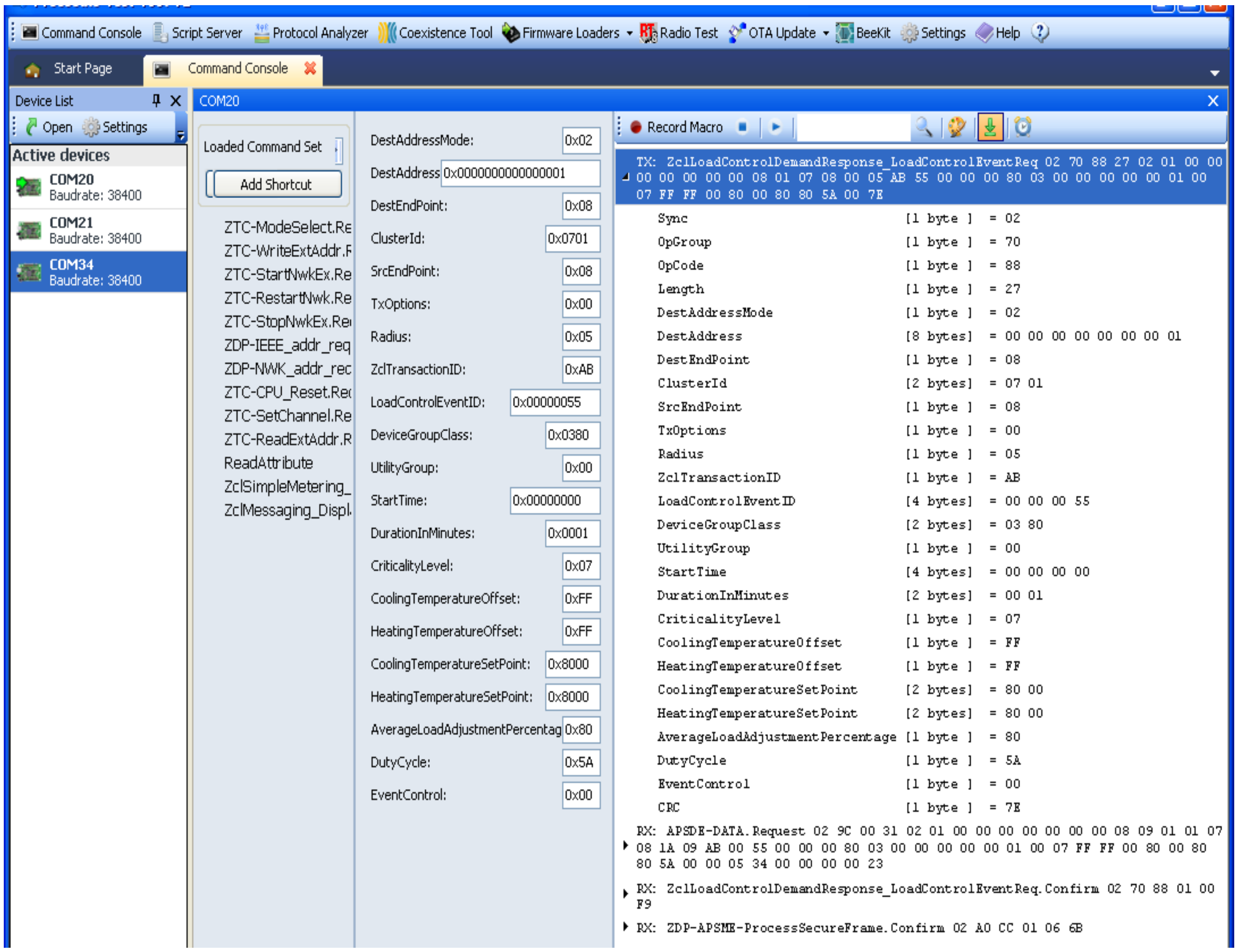
Figure 6-16. Update Server Price Command (Step 4)

6. On the “Se LoadControlDevice” node, perform a long press on SW4 to place the device into Load Control Mode (LED4 is Off). This means that the duty cycle adjustment is based on DRLC events instead of price events.
7. Perform Step 2 again. During load control event execution, send the other “Load Control Event request” with the field values as shown in Figure 6-17.

   The “Load Control Event request” has a LoadControlEventID higher than the executing event, CriticalLevel is 0x07(event is critical), the same “Start Time” and “Duration” and a Duty Cycle of 90% (0x5A).

   The previously loaded load control event is superseded with the new critical event. LED1 flashes during critical event duration, LED2 shows the duty cycle (the device is in an On state for 90% of the 5 second duty cycle).

   When the load control event expires, the duty cycle is set back to the duty cycle value that was configured by users before receiving the event.



DestAddressMode:	0x02
DestAddress	0x0000000000000001
DestEndPoint:	0x08
ClusterId:	0x0701
SrcEndPoint:	0x08
TxOptions:	0x00
Radius:	0x05
ZclTransactionID:	0xAB
LoadControlEventID:	0x00000055
DeviceGroupClass:	0x0380
UtilityGroup:	0x00
StartTime:	0x00000000
DurationInMinutes:	0x0001
CriticalityLevel:	0x07
CoolingTemperatureOffset:	0xFF
HeatingTemperatureOffset:	0xFF
CoolingTemperatureSetPoint:	0x8000
HeatingTemperatureSetPoint:	0x8000
AverageLoadAdjustmentPercentag	0x80
DutyCycle:	0x5A
EventControl:	0x00

```
TX: ZclLoadControlDemandResponse_LoadControlEventReq 02 70 88 27 02 01 00 00
00 00 00 00 00 08 01 07 08 00 05 AB 55 00 00 00 80 03 00 00 00 00 00 01 00
07 FF FF 00 80 00 80 80 5A 00 7E
    Sync                              [1 byte ]  = 02
    OpGroup                           [1 byte ]  = 70
    OpCode                            [1 byte ]  = 88
    Length                            [1 byte ]  = 27
    DestAddressMode                   [1 byte ]  = 02
    DestAddress                       [8 bytes]  = 00 00 00 00 00 00 00 01
    DestEndPoint                      [1 byte ]  = 08
    ClusterId                         [2 bytes]  = 07 01
    SrcEndPoint                       [1 byte ]  = 08
    TxOptions                         [1 byte ]  = 00
    Radius                            [1 byte ]  = 05
    ZclTransactionID                  [1 byte ]  = AB
    LoadControlEventID                [4 bytes]  = 00 00 00 55
    DeviceGroupClass                  [2 bytes]  = 03 80
    UtilityGroup                      [1 byte ]  = 00
    StartTime                         [4 bytes]  = 00 00 00 00
    DurationInMinutes                 [2 bytes]  = 00 01
    CriticalityLevel                  [1 byte ]  = 07
    CoolingTemperatureOffset          [1 byte ]  = FF
    HeatingTemperatureOffset          [1 byte ]  = FF
    CoolingTemperatureSetPoint        [2 bytes]  = 80 00
    HeatingTemperatureSetPoint        [2 bytes]  = 80 00
    AverageLoadAdjustmentPercentage [1 byte ]  = 80
    DutyCycle                         [1 byte ]  = 5A
    EventControl                      [1 byte ]  = 00
    CRC                               [1 byte ]  = 7E
RX: APSDE-DATA.Request 02 9C 00 31 02 01 00 00 00 00 00 00 00 08 09 01 01 07
08 1A 09 AB 00 55 00 00 00 80 03 00 00 00 00 00 01 00 07 FF FF 00 80 00 80
80 5A 00 00 05 34 00 00 00 00 23
RX: ZclLoadControlDemandResponse_LoadControlEventReq.Confirm 02 70 88 01 00
F9
RX: ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC 01 06 6B
```

Figure 6-17. Load Control Event (Step 6)

NOTE

If the device is in Price Mode and a critical load control event is received during price event execution, the critical event is accepted and the price event is overridden.

8. From Test Tool, send a "Load Control Event request" with the field values as shown in Figure 6-18.

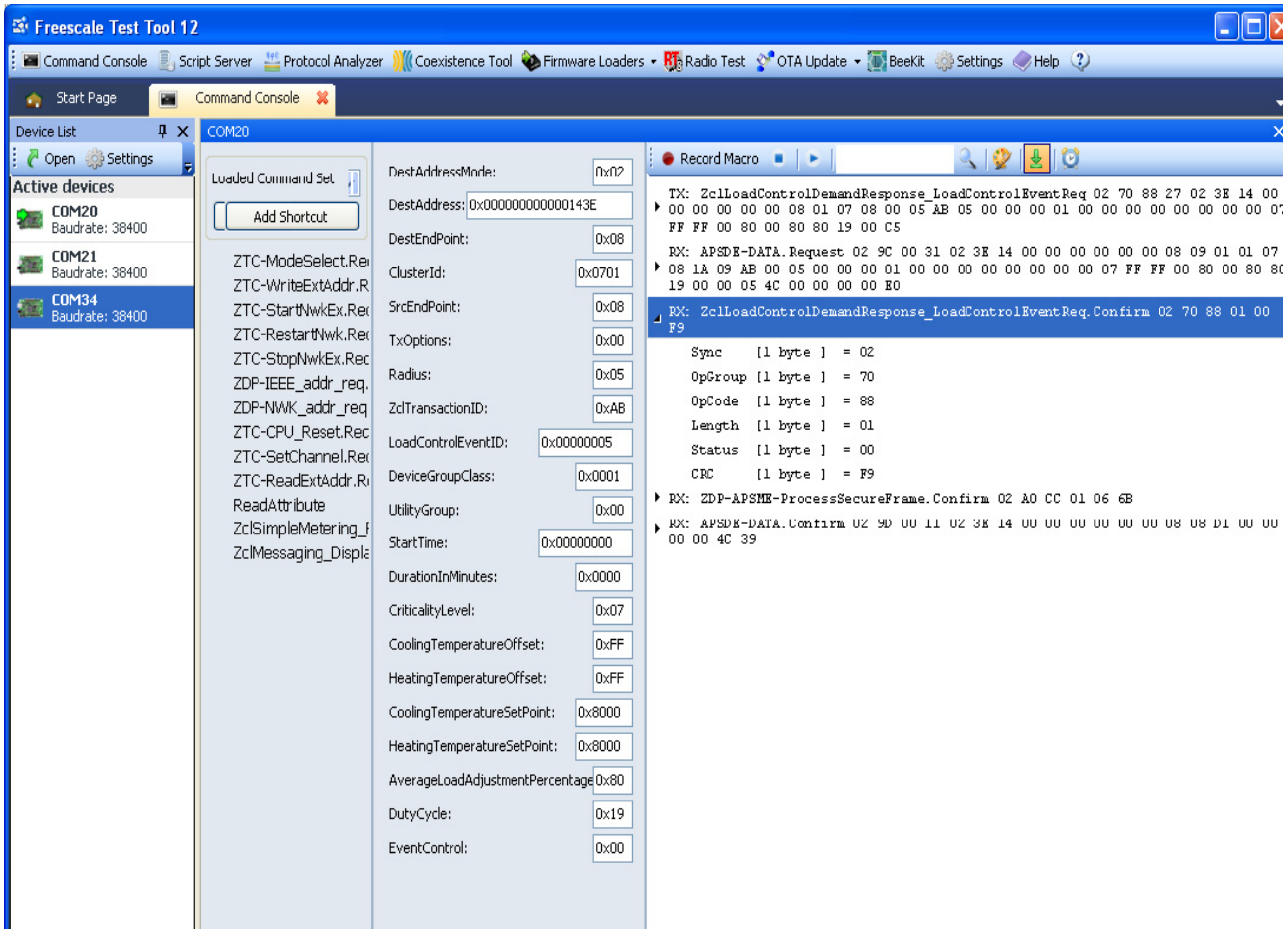
   This command specifies a load control event that starts immediately, has a duration of one minute and a duty cycle of 25% (0x19). The request is sent over the air from the ESI to the "Se PCT" node (short address is 0x143E) that acts as a Circulation device which sets the fan speed to one (25%).

   On the "Se PCT" node, the execution of the received load control event is indicated by the LEDs as follows:

   — LED1 flashes with a period of 800 milliseconds during event duration

   — LED2, LED3, and LED4 are Off

   After one minute, the event completes and the fan speed is set back to the fan speed value that was configured by users before receiving the event. (Speed four (100%) after Step 7 in Section 6.4.1, "Creating the Projects and Configuring the Network".



DestAddressMode:	0x02
DestAddress:	0x000000000000143E
DestEndPoint:	0x08
ClusterId:	0x0701
SrcEndPoint:	0x08
TxOptions:	0x00
Radius:	0x05
ZclTransactionID:	0xAB
LoadControlEventID:	0x00000005
DeviceGroupClass:	0x0001
UtilityGroup:	0x00
StartTime:	0x00000000
DurationInMinutes:	0x0000
CriticalityLevel:	0x07
CoolingTemperatureOffset:	0xFF
HeatingTemperatureOffset:	0xFF
CoolingTemperatureSetPoint:	0x8000
HeatingTemperatureSetPoint:	0x8000
AverageLoadAdjustmentPercentage	0x80
DutyCycle:	0x19
EventControl:	0x00

```
TX: ZclLoadControlDemandResponse_LoadControlEventReq 02 70 88 27 02 3E 14 00
00 00 00 00 00 08 01 07 08 00 05 AB 05 00 00 00 01 00 00 00 00 00 00 00 00 07
FF FF 00 80 00 80 80 19 00 C5
RX: APSDE-DATA.Request 02 9C 00 31 02 3E 14 00 00 00 00 00 00 08 09 01 01 07
08 1A 09 AB 00 05 00 00 00 01 00 00 00 00 00 00 00 00 07 FF FF 00 80 00 80 80
19 00 00 05 4C 00 00 00 00 E0
RX: ZclLoadControlDemandResponse_LoadControlEventReq.Confirm 02 70 88 01 00
F9
   Sync    [1 byte ]  = 02
   OpGroup [1 byte ]  = 70
   OpCode  [1 byte ]  = 88
   Length  [1 byte ]  = 01
   Status  [1 byte ]  = 00
   CRC     [1 byte ]  = F9
RX: ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC 01 06 6B
RX: APSDE-DATA.Confirm 02 9D 00 11 02 3E 14 00 00 00 00 00 00 08 08 D1 00 00
00 00 4C 39
```

Figure 6-18. Load Control Event (Step 8)

9. Wait until the received event at the end of Step 8 completes. Perform Step 8 again from the beginning.
10. During event execution, press SW1 or SW2 to set the required speed.

    This sends the ESI a “Report Status Event” over the air with the “Opt Out” Status and the fan speed is decreased/increased starting from the current speed. After one minute, the event completes with the “Event partially completed with User "Opt-Out” status. (See the Smart Energy Specification for more details).

NOTE

> If using the buttons to decrease or increase the fan speed during a critical or voluntary event duration, a Report Status Event with the "Opt Out" status is sent for the active event. The fan speed is decreased or increased starting from the current speed. Only load control events with known duty cycles are accepted. (0%, 25%, 50%, 75%, 100%).

11. Using the IAR Embedded Workbench, select the “Se PCT” project.
12. Open the `BeeApp.h` file and set the `gPCTHeatCoolDevice_d` option to TRUE. This allows the PCT device to act as a Cooling/Heating device. Compile the project and program the board. Press SW1 to join the network and enter Application Mode (long press on SW1).
13. Perform a long press on SW3 to accept voluntary events. By default, the temperature is 24 °C.
14. From Test Tool, send a “Load Control Event request” with the field values as shown in Figure 6-19.

    This command specifies a load control event that starts immediately, has a duration of one minute and specifies the thermostat fields. (By default, the Device Group Class is 0x0001.)

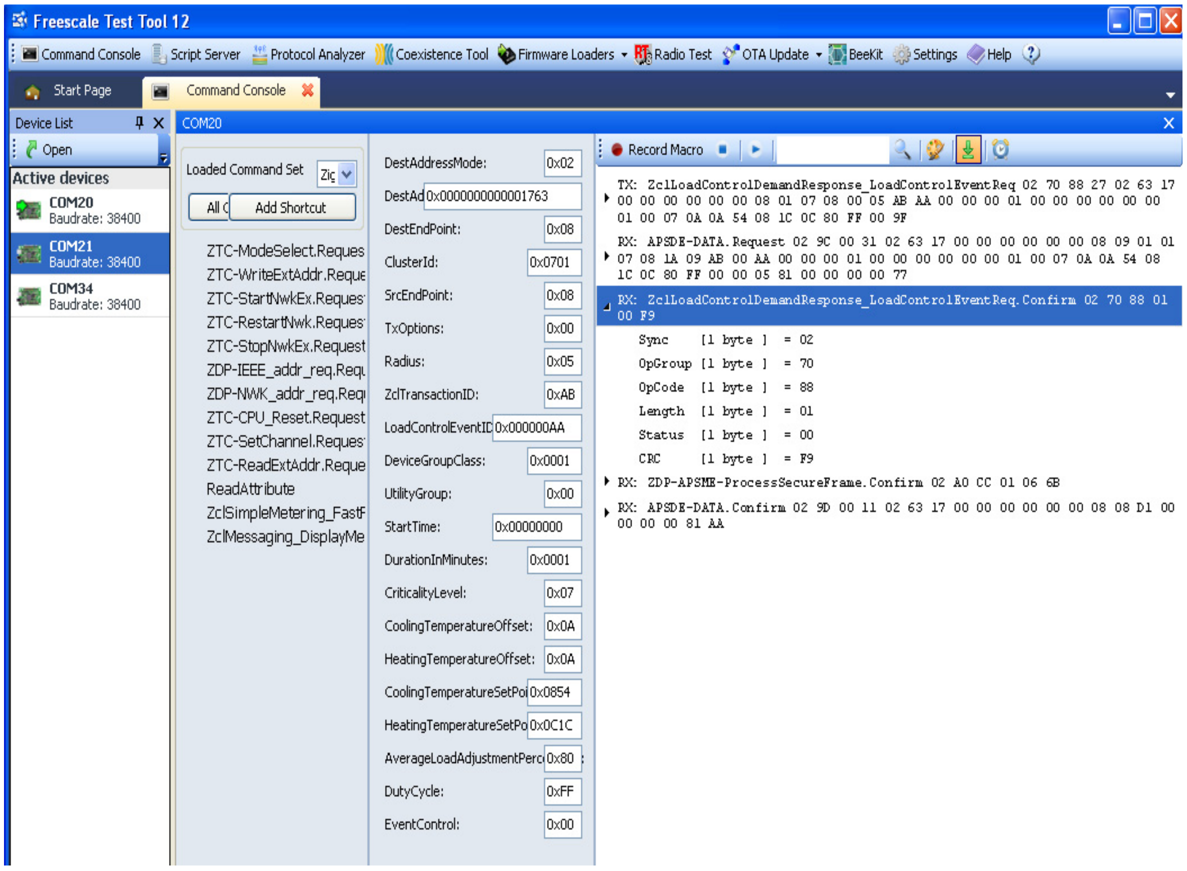
    The request is sent over the air from the ESI to the “Se PCT” node (short address is 0x1763) that acts a Cooling/Heating device, to set the desired temperature to 30 °C.

    On the “Se PCT” node, the execution of the received load control event is indicated by the LEDs as follows:

    — LED1 flashes during event duration

    — LED2, LED3 and LED4 are On

    After one minute, the event completes and the temperature is set back to the value that was configured by users before receiving the event (to 24 °C).



DestAddressMode:	0x02
DestAd	0x0000000000001763
DestEndPoint:	0x08
ClusterId:	0x0701
SrcEndPoint:	0x08
TxOptions:	0x00
Radius:	0x05
ZclTransactionID:	0xAB
LoadControlEventID	0x000000AA
DeviceGroupClass:	0x0001
UtilityGroup:	0x00
StartTime:	0x00000000
DurationInMinutes:	0x0001
CriticalityLevel:	0x07
CoolingTemperatureOffset:	0x0A
HeatingTemperatureOffset:	0x0A
CoolingTemperatureSetPoi	0x0854
HeatingTemperatureSetPo	0x0C1C
AverageLoadAdjustmentPerc	0x80
DutyCycle:	0xFF
EventControl:	0x00

```
TX:  ZclLoadControlDemandResponse_LoadControlEventReq 02 70 88 27 02 63 17
00 00 00 00 00 00 08 01 07 08 00 05 AB AA 00 00 00 01 00 00 00 00 00 00
01 00 07 0A 0A 54 08 1C 0C 80 FF 00 9F

RX:  APSDE-DATA.Request 02 9C 00 31 02 63 17 00 00 00 00 00 00 08 09 01 01
07 08 1A 09 AB 00 AA 00 00 00 01 00 00 00 00 00 00 01 00 07 0A 0A 54 08
1C 0C 80 FF 00 00 05 81 00 00 00 00 77

RX:  ZclLoadControlDemandResponse_LoadControlEventReq.Confirm 02 70 88 01
00 F9
   Sync     [1 byte ]  = 02
   OpGroup  [1 byte ]  = 70
   OpCode   [1 byte ]  = 88
   Length   [1 byte ]  = 01
   Status   [1 byte ]  = 00
   CRC      [1 byte ]  = F9

RX:  ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC 01 06 6B

RX:  APSDE-DATA.Confirm 02 9D 00 11 02 63 17 00 00 00 00 00 00 08 08 D1 00
00 00 00 81 AA
```

Figure 6-19. Load Control Event Request (Step 9)

NOTE

For more information about these applications, see Chapter 7, “Example Applications”.

Chapter 7 Example Applications

BeeStack includes example applications to:

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

7.1 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.

7.2 Example Application User Interfaces

Almost 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 MC1322x Network Node, MC1321x-NCB, MC1323x-RCM, and 1320x-QE128EVB also have an 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. On every board, the push-buttons and LEDs are in single rows. LED1 is above SW1, LED2 is above SW2, and so on.

NOTE

Referring to switches SW1-SW4 in this chapter refers to the four push-buttons present on most of the Freescale development boards. For example, on the 1320x-QE128EVB, the four buttons are actually labeled SW2-SW5 and on the SARD they are labeled S101-S104.

If using the MC1322x-LPN board, only two buttons and two leds are available. Restrictions and changes related to this board are mentioned in the corresponding steps if needed. See Section 2.1.1, “HCS08 MCU Board Details” for more information.

7.3 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 the following two modes:

- Configuration
- 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, such as turn a light on or off remotely, report data, etc.

7.3.1 ASL 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

7.3.2 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

7.4 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.

7.4.1 Configuration Mode

Table 7-1. 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		
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 7-2) for one second before returning to the previous display pattern.

Table 7-2. 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

7.4.2 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 7-3.

Table 7-3. 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	

7.5 Application Support Library (ASL) Specific Configuration for MC1322x-LPN

The MC1322x Low Power Node (MC1322x-LPN) only has two switches and two LEDs. This imposes a series of changes and restrictions to the ASL functionality for switches and LEDs as already described. This section describes the switch and LED functionality for the Low Power Node.

7.5.1 Switches in 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 – End device bind/match (not available during Low Power Mode)

Long push button presses do the following:

- Long SW1 – Go to run mode
- Long SW2 – Leave the network (not available during Low Power Mode)

Permit join toggle, clear bindings, change radio channel, form/join with new configuration are not available using the switches for the MC1322x Low Power Node.

7.5.2 Switches in Run Mode

Short push button presses do the following:

- SW1 – This does the main action of the application: toggle a light, etc.
- SW2 – Application Specific (not available during Low Power Mode)

Long push button presses do the following:

- Long SW1 – Go to configuration mode
- Long SW2 – Application Specific (not available during Low Power mode)

Identify, groups and scenes functionality is not available using switches and does not provide LED indication for the MC1322x Low Power Node.

7.5.3 LEDs in Configuration Mode

Table 7-4. Configuration Mode LED Status for MC1322x Low Power Node

Internal State	LED1	LED2
Idle (not in a network, not joining)	Flashing	Off
Forming (ZC) or joining a network	The 2 LEDs in a flash sequence	
Network formed/joined	On	Off
End device bind request: find a bind partner		Flash
End device bind success		On
End device bind failure		Off

Applications that use more than 2 LEDs in Run Mode such as the Dimmable Light should not be deployed on a MC1322x Low Power Node.

7.6 OnOffLight

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

7.6.1 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)

7.6.2 Display

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

7.7 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.

7.7.1 Keyboard

In run mode, key presses do the following:

- SW1 - Toggle remote light
- SW2 - Based on the SwitchActions attribute, moves the switch from state 1 to state 2
- Long SW2 - Based on the SwitchActions attribute, moves the switch from state 2 to state 1

7.7.2 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

7.8 Dimmable Light

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

7.8.1 Keyboard

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

7.8.2 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

The LCD display on the MC1322x Network Node shows the light intensity as a percentage and a histogram.

7.9 Dimmer Control Switch

This controls the dimmable light.

7.9.1 Keyboard

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

7.9.2 Display

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

7.10 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.

7.10.1 Keyboard

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

7.10.2 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 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
- The LCD display on the MC1322x Network Node displays various icons depending on the action being performed. For example:
  - A fan icon appears for the thermostat
  - A snow flake icon appears when the cooling mode is on
  - A sun icon appears when the heating mode is on
  - A fan intensity icon appears for the fan cluster values

7.11 Temperature Sensor

The temperature sensor application uses the hardware temperature sensor found on the MC1321x-SRB and MC1322x Sensor Node 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.

7.11.1 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).

7.11.2 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

7.12 HA Generic Application

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

7.13 HA Range Extender

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

7.14 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 7.3.1, "ASL Configuration Mode" and Section 7.4.1, "Configuration Mode".

7.15 Energy Service Portal (ESP)

The Energy Service Portal (ESP) acts as a gateway to a ZigBee network and optionally acts as a Metering Device. It connects the energy supply company network to the metering and energy management devices within the home. It is assumed that a HOST (usually a PC) will relay the messages from the utility company to the ZigBee network. The ESP only routes the messages to the devices from the network. That is, the HOST must keep track of what messages are sent and must ensure that the messages have the correct parameters. The ESP is a ZigBee Coordinator that maintains a list of registered Smart Energy devices (to have secure communications), and also can send inter-pan messages to non-Smart Energy devices.

For the ESP demonstration application, users should employ the Test Tool software as a HOST to relay commands or messages to and from the ZigBee devices. The device does not include any functionality for application run mode.

The configuration running mode has all the functionality as shown in Section 7.3.1, "ASL Configuration Mode" and Section 7.4.1, "Configuration Mode" without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

7.16 Metering Device

The Metering Device demonstration application uses a timer based scheme to demonstrate continuous energy consumption so that the metering data read through the read attribute or through reporting, shows an increasing consumption. For example, the Current Summation Delivered attribute will increment with a random value (between 0 and 32) every 10 seconds. The Attribute value can be reported, read to/from the ESP, which provides information to the utility company. The metering device is typically a “sleepy” end device, configured by the ESP to report the metering data. The Metering Device demonstration application shows how an attribute can be “decoupled” and updated asynchronously. It decouples the Current Summation Delivered attribute so that read attribute responses or attribute reports can be sent later. For example, a real meter is read through a slow serial interface.

The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

7.16.1 Keyboard

In application run mode, key presses do the following:

- SW2 - Toggle the status attribute bit “Check Meter”
- SW4 - Toggle on and off energy consumption

7.16.2 Display

The LEDs in run application mode do this:

- LED2 - Toggles with each toggle command on SW2
- LED4 – Flashing when consumption is On

7.17 In-Premise Display

The In-Premise Display demonstration application relays the data to users using a text display. The display shows simple text messages or pricing information to inform users about the utility company actions or current pricing information. The time is updated in the Smart Energy network by the utility company through the ESP and is displayed for all received messages. The In-Premise Display device is typically a ZigBee router that can receive information delivered through the ESP and make it available to users.

The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

7.17.1 Keyboard

In application run mode, key presses do the following

- SW2 - Confirm a received messages (if a message needs to be confirmed)
- Long SW2 - Send a "Get Last Message Request"

- Long SW3 - Send a "Get Current Price Request"
- SW4 - Run through information fields

7.17.2 Display

The LEDs in run application mode do the following:

- LED2 - Flashing when a received message need to be confirmed

  Depending on the target, the LCD can display information in different ways.

The MC1321xNCB board LCD displays information as follows:

- LCD line1 - "InPremiseDisplay" - display application name
- LCD lines 2-3 - display messages received from the ESP
- LCD line 4 - " PriceTier 1" - display some information fields of the current price; pressing SW4 user can run trough other fields.
- LCD line 5 - "PriceStarted" - display price status (received, started, updated, completed, no current price available)
- LCD line 6 - "Time= 07:55:20" - display the current time

The other boards with a 16 character LCD display information as follows:

- LCD line1 - The last received event is displayed (e.g. Msg Rcvd, Price Rcvd, etc)
- LCD line2 - Displays the current information field. The available fields are: last message received, current price, duration, price tier, label, unit of measure and current time. The user can switch the current information field by pressing SW4

7.18 Programmable Communicating Thermostat (PCT)

The Programmable Communicating Thermostat (PCT) demonstration application shows how received Demand Response and Load Control (DRLC) and Price events are managed by a device that supports load control and prices. The PCT can act as a thermostat (Heating/Cooling device) or as a circulation fan. Use the `BeeApp.h` file (`gPCTHeatCoolDevice_d` and `gPCTCirculationFanDevice_d`) to select between the two devices.

The prices and load control commands are sent to the PCT device from a host CPU or the Test Tool software through the ESP device using serial communication. This allows users to experiment with the Smart Energy Price and Load Control commands.

The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

NOTE

By default, the Utility Group is 0x00 and Device Class is 0x0001 for both devices.

7.18.1 Heat/Cool Device (Selected at Compile Time in BeeApp.h)

If the PCT is a Heating/Cooling device, it has two different modes, selected by an application mode button (long press of SW4):

- Demand Response and Load Control (DRLC) Mode — In DRLC Mode, the PCT adjusts the desired temperature as defined by the DRLC event and display. (If voluntary events are accepted.) Critical events must always be executed. In DRLC Mode, the adjustment is made using the fields of the load control event specific for the thermostat.
- Price Mode — In Price Mode, temperature adjustment is based on Price instead of DRLC events, except when a critical/mandatory load control event is received.

Only one event (price or load control) will be active on the same duration. The last received event will override the executing event, unless the executing event is critical. Critical events are executed until they end.

All the adjustments are made on the last desired temperature value as changed locally by the user. For example, if the active event completes execution, the temperature is set back to the value set locally before receiving the event.

If a price event is received and the device is in Price Mode, the temperature is adjusted to match the following values:

- For low price (between [0.8 - 0.12))
  - Temperature Heating/Cooling Set Points are raised +1 °C
  - Temperature Heating/Cooling Offsets are set to 1 °C
- For medium price (between [0.12 – 0.18))
  - Temperature Heating/Cooling Set Points are not raised or lowered (+0 °C)
  - Temperature Heating/Cooling Offsets are set to 1 °C
- For peak price (between [0.18 – 0.30))
  - Temperature Heating/Cooling Set Points are lowered -2 °C
  - Temperature Heating/Cooling Offsets are set to 2 °C
- For critical price (between [0.30 – 0.40))
  - Temperature Heating/Cooling Set Points are lowered -4 °C
  - Temperature Heating/Cooling Offsets are set to 3 °C

NOTE

If a display is connected to the board, information such as event duration, event duty cycle, temperature Heating/Cooling Set Points, temperature Heating/Cooling Offsets, event status (started, superseded, canceled, completed etc.) are displayed for load control events. For price events, fields such as tier number, label, price, duration, unite of measure are displayed.

7.18.1.1 Keyboard

In application run mode, key presses do the following:

- SW1 - Decreases the desired temperature. If performed during the event duration, first choose SW3 (not to participate on the active event), to activate SW1. During a load control critical event, it is NOT possible to decrease the desired temperature until the event completes execution.
- SW2 - Increases the desired temperature. If performed during the event duration, first choose SW3 (not to participate on the active event), to activate SW1. During a load control critical event, it is NOT possible to increase the desired temperature until the event completes execution.
- SW3 - Select Opt Out, (not to participate on the last received event) If the event is critical, it is not possible to choose Opt Out (the event will be executed until it ends).
- SW4 - Run through the last received event fields (it can be a load control or price event)
- Long Press SW1 - Toggle display/keyboard mode (Configuration and Application). When entering application mode, a “Get Current Price command” is sent over the air.
- Long Press SW2 - Toggle the LEDs. Displays the desired temperature on the LEDs or Application Settings LEDs (accept or not accept voluntary events, the device is in Load Control or Price Mode).
- Long Press SW3 - Toggle on/off accepting voluntary load control events (by default voluntary events are not accepted)
- Long Press SW4 - Toggle between Load Control and Price Mode. In Price Mode, only critical load control and price events are accepted. In Load Control Mode, the price events are rejected. That is, the non-critical active event is overridden with the last received event (price or load control event). By default, the device is in Load Control Mode. If no current price is active when users enter the Price Mode (LED4 On), a “Get Current Price command” is sent over the air. The mode of operation can not be changed during critical load control event execution.

7.18.1.2 Display

Depending on the target, the LCD displays information in different ways.

The LEDs in run application mode do the following:

- LED2 - LED4 - When the desired temperature is changed, it shows the level of the desired temperature as follows:
  - — LED2 is On and LED3 and LED4 are Off, the local temperature is between 10 °C and 20 °C.
  - — LED2 and LED3 are On and LED4 is Off, the local temperature is between 20 °C and 30 °C.
  - — LED2, LED3 and LED4 are On, the local temperature is between 30 °C and 40 °C
  - — LED2, LED3 On and LED4 is flashing, the local temperature is above 40 °C
- LED1 - When an event starts executing, the LED flashes during the entire event duration.
- LED3 - When a long press of SW3 occurs, it shows that voluntary load control events are accepted or not.
- LED4 - When a long press of SW4 occurs, it shows if the device is in Load Control or Price Mode. By default, LED4 is Off and Load Control Mode selected.

Depending on the target, the LCD displays information in different ways.

The MC1321xNCB board LCD displays information as follows:

- LCD Line 1 - "DT= 24C" - Displays the temperature
- LCD Line 2 - "App 1AAA 13 0001" - Displays the PAN ID, channel and short address in application mode
- LCD Line 5 - "Ev 010Min 80%ON" - Displays the last received event fields (Can be a load control event or price event). Use SW4 to run through the event fields
- LCD line 6 - "Started" - Displays event status.

The other boards with a 16 character LCD display information as follows:

- LCD Line 1 - Displays the last received event
- LCD Line 2 - Displays the current information field. The available fields are : desired temperature, price or load control event, heating setpoint, cooling setpoint, heating offset, cooling offset. Switching the current field is done by pressing on SW4

7.18.2 Circulation Fan Device (Selected at Compile Time in BeeApp.h)

The circulation fan device only reacts to DRLC events. The fan has four speeds (not including the “Off” duty cycle), as designated by the duty cycle field of the load control event as follow:

Speed 1	25% Duty Cycle — LED1flashes at a 800 milliseconds period.
Speed 2	50% Duty Cycle — LED2 is ON, LED1 flashes at a (800/2) milliseconds period.
Speed 3	75% Duty Cycle — LED2 and LED3 are ON, LED1 flashes at a (800/3) milliseconds period.
Speed 4	100% Duty Cycle — LED2, LED3 and LED4 are ON, LED1 flashes at a (800/4) milliseconds period.
Fan is Off	0% Duty Cycle — All LEDs are Off (LED1 is not flashing).

NOTE

If a user locally decreases or increases the fan speed during critical or voluntary event duration, a Report Status Event with the "Opt Out" status is sent for the active event and the fan speed is decreased or increased starting from current speed. (This is different behavior than the Heating/Cooling device). Only load control events with known duty cycles are accepted (0%, 25%, 50%, 75% and 100%.)

7.18.2.1 Keyboard

In application run mode, key presses do the following:

- SW1 - Locally decreases the fan speed. If this done during the event duration, a Report Status Event with the "Opt Out" status is sent for the active event and the fan speed is decreased starting from the current speed
- SW2 - Locally increases the fan speed. If this is done during the event duration, a Report Status Event with the "Opt Out" status is sent for active event and the fan speed is increased starting from current speed

- SW4 - Runs through the last received event fields (load control or price event)
- Long Press SW1 - Toggles display/keyboard mode (Configuration and Application)
- Long Press SW3 - Toggles on/off the ability to accept voluntary load control events. (By default, voluntary events are not accepted.)

7.18.2.2 Display

During the application mode, the LEDs display as follows:

- LED1 - Flashes with a period specified by speed (and the fan speed is specified by duty cycle).
- LED1- LED4 - Specifies the fan speed. (For example, speed three (75%) - LED1 flashes, LED2 and LED3 are On)
- LED3- When a long press of SW3 occurs, shows if voluntary load control events are accepted. (By default, LED3 is Off and voluntary events are not accepted.)

Depending on the target, the LCD displays information in different ways.

The MC1321xNCB board LCD displays information as follows:

- LCD Line1 - "Speed = 1 (025%)" - Displays the speed.
- LCD Line 2 - "App 1AAA 13 0001" - Display PAN ID, channel and short address in application mode.
- LCD Line 5 - "Ev 010Min 25%ON" - Displays the last received event fields. Use SW4 to run through event fields.
- LCD Line 6 - "Started" - Displays the event status

The other boards with a 16 character LCD display information as follows:

- LCD Line 1 - Displays the last received event
- LCD Line 2 - Displays the current information field. The available fields are : speed, price or load control event, heating setpoint, cooling setpoint, heating offset, cooling offset. Switching the current field is done by pressing on SW4

7.19 Load Control (LC)

The Load Control (LC) demonstration application looks similar to the PCT Heating/Cooling device regarding the load control management and price events. The only difference is that the Load Control application uses the duty cycle field of the load control event to simulate the On/Off state of the device in a defined period of time (5 seconds).

In Price Mode, adjustment of the duty cycle is based on Price instead of DRLC events, except when a critical/mandatory load control event is received.

If an event is completed, the duty cycle is set back to the value that was set locally by the user (not set by other event).

If a price event is received and the device is in Price Mode, the desired duty cycle is adjusted to match the following values:

- For low price (between [0.8 - 0.12))

  - Duty cycle is set to 100% (Device is in the “On” state.)
- For medium price (between [0.12 – 0.18))
  - Duty cycle is set to 90% (Device is in the “On” state for 90% of the 5 second duty cycle.)
- For peak price (between [0.18 – 0.30))
  - Duty cycle is set to 70% (Device is in the “On” state for 70% of the 5 second duty cycle.)
- For critical price (between [0.30 – 0.40))
  - Duty cycle is set to 50% (Device is in the “On” state for 50% of the 5 second duty cycle.)

The prices and load control commands are sent to the LC device from a host CPU or the Test Tool software through the ESP device using serial communication. This allows users to experiment with the Smart Energy Price and Load Control commands. The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

NOTE

By default, the Utility Group is 0x00 and Device Class is 0x0380.

7.19.1 Keyboard

In application run mode, key presses do the following:

- SW1 - Decreases the duty cycle 10%. If an event is executing, first choose SW3 (not to participate on the last received event), to locally set the duty cycle. During a load control critical event, it is NOT possible to decrease the duty cycle until the event completes execution.
- SW2 - Increases the duty cycle 10%. If an event is executing, first choose SW3 (not to participate on the last received event), to locally set the duty cycle. During a load control critical event, it is NOT possible to increase the duty cycle until the event completes execution.
- SW3 - Select Opt Out (not to participate on the last received event). If the event is critical, it is not possible to choose Opt Out. (The event will be executed until it ends.)
- SW4 - Run through the last received event fields (load control or price event).
- Long Press SW1 - Toggles display/keyboard mode (Configuration and Application). When entering the application mode, a “Get Current Price command” is sent over the air.
- Long Press SW3 - Toggles on/off the ability to accept voluntary load control events. (By default, voluntary events are not accepted.)
- Long Press SW4 - Toggles between Load Control and Price Mode. In Price Mode, only critical load control and price events are accepted. In Load Control Mode, the price events are rejected. That is, the non-critical active events are overridden with the last received event (price or load control event). By default, the device is in Load Control Mode. If no current price is active when users enter the Price Mode (LED4-On), a “Get Current Price command” is sent over the air. The mode of operation can not be changed during critical load control event execution.

7.19.2 Display

The LEDs in run application mode do the following:

- LED1 - Flashes during the entire event duration.
- LED2 - Shows the current "On" state of the 5 second duty cycle. For example, a duty cycle of 90% signifies that the device is the "On" state for 4.5 seconds and in the "Off" state for 0.5 seconds. This device behavior is repeated every 5 seconds ("Off" state precedes the "On" state).
- LED3 - When a long press of SW3 occurs, it shows that voluntary load control events are accepted or not. By default, LED3 is Off and voluntary events are not accepted.
- LED4 - When a long press of SW4 occurs, it shows the running mode of the device (Load Control or Price Mode). By default, LED4 is Off and Load Control Mode selected.

Depending on the target, the LCD displays information in different ways.

The MC1321xNCB board LCD displays information as follows:

- LCD Line 1 - "DutyCycle= 90%" - Displays the duty cycle.
- LCD Line 2 - "App 1AAA 13 0001" - Displays the PAN ID, channel and short address in application mode.
- LCD Line 5- "Ev 010Min 25%ON" - Displays the last received event fields (load control event or price event). Use SW4 to run through event fields.
- LCD Line 6 - "Started" - Display event status.

The other boards with a 16 character LCD display information as follows:

- LCD Line1 - displays the last received event status (price or load control)
- LCD Line2 - displays the current information field. The fields can be the duty cycle and the last received event. Switching between the fields is done by pressing SW4.

7.20 Range Extender

The Range Extender is a simple device that acts as a router for other devices from the network. It has no functionality in the application mode The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

7.21 Smart Appliance, Prepayment Terminal

The Load Control, Smart Appliance, Prepayment Terminal are sample applications that have no functionality in the run mode of the application. The configuration running mode has all the functionality as shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode” without end device bind/match (SW3 short press) and remove all bindings (SW3 – long press).

7.22 Manufacturer Specific Applications

The Wireless UART and Accelerometer applications are example applications that are not part of the Home Automation Application Profile. They are also not part of any other ZigBee public profile and are considered to be custom application profiles. The ZigBee Alliance calls these “Manufacturer Specific” applications. These applications are only intended to operate among devices from a single primary manufacturer. Other manufacturers can use these applications, if they have permission to do so, from the primary manufacturer.

7.22.1 Manufacturer Specific Configuration Mode User Interface

The following Configuration Mode User Interface is slightly different from 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

7.23 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.

7.23.1 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

7.23.2 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.

7.23.3 Baud Rate Display User Interface

7.23.3.1 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 7-5. Baud Rate LED Display

Baud Rate	LED1	LED2	LED3
1200	On	Off	Off
2400	Off	On	Off
2400	On	On	Off

Table 7-5. Baud Rate LED Display (continued)

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

7.24 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.

7.24.1 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.

7.24.2 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.

7.25 Health Care Data Collection Unit

The Health Care Data Collection Unit (Hc DataCollectionUnit) application implements a demonstration application for a ZigBee Health Care Profile Data Collection Unit device. The application is a prototype for the IEEE 11073-20601 Application Profile - Optimized Exchange Protocol (OEP) Manager device specification. The OEP layer of the application uses a ZigBee tunnel cluster to exchange health care specific device association, deassociation and data messages as shown in ZigBee Alliance document 075360.

The Data Collection Unit application acts as an OEP Manager by replying to the association request from an OEP Agent and receiving health care data indications from it as described in Section 7.26, “Health Care Thermometer”.

NOTE

Perform the End Device Bind procedure between the DataCollectionUnit device and the Agent before sending the initial ZigBee Health care tunnel RECONNECT status notification request from the Agent device.

The application only implements the Configuration mode and does not implement the Run Mode. The switch and LED behavior in Configuration mode for the Health Care Data Collection application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.26 Health Care Thermometer

The Health Care Thermometer (Hc Thermometer) application implements a demonstration application for a ZigBee Health care Profile Thermometer device. The application is a prototype for the IEEE 11073-20601 Application Profile - Optimized Exchange Protocol (OEP) Agent device specification. The OEP layer of the application uses a ZigBee tunnel cluster to exchange health care specific device association, deassociation and data messages as shown in ZigBee Alliance document 075360.

The Thermometer application acts as an OEP Agent by sending an Association Request to the Manager to which the Manager replies with an Association Response. Users of the Hc Thermometer application can initiate a ZigBee tunnel reconnect status notification request to trigger the OEP layer association and send simulated temperature data indications to the Manager.

NOTE

Perform the End Device Bind procedure between the Thermometer device and the Manager device before sending the initial ZigBee Health care tunnel RECONNECT status notification request to the Manager device.

The switch and LED behavior in Configuration mode for the Thermometer application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.26.1 Keyboard

In run mode, key presses do the following:

- SW1 - Increases the local temperature.
- SW2 - Decreases the local temperature.
- SW3 - Sends a body temperature Event Report to the Manager device.
- SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster RECONNECT status notification to the Manager device. The Manager replies with a Tunnel Connect Request and the tunnel is fully established once the CONNECTED status notification is sent back from the Agent to the Manager. Once the tunnel is established, the Thermometer continues to send a 11073 OEP Association Request to the Manager, followed by an Association Response from the Manager. This completes the tunnel and the IEEE 11073 OEP association process
- Long SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster DISCONNECT status notification to the Manager device. This triggers a tunnel disconnect and releases the OEP association.

7.27 Health Care Weight Scale

The Health Care Weight Scale (Hc WeightScale) application implements a demonstration application for a ZigBee Health care Profile WeightScale device. The application is a prototype for the IEEE 11073-20601 Application Profile - Optimized Exchange Protocol (OEP) Agent device specification. The OEP layer of the application uses a ZigBee tunnel cluster to exchange health care specific device association, deassociation and data messages as shown in ZigBee Alliance document 075360.

The WeightScale application acts as an OEP Agent by sending an Association Request to the Manager to which the Manager replies with an Association Response. Users of the Hc WeightScale application can initiate a ZigBee tunnel reconnect status notification request to trigger the OEP layer association and send simulated values of height and weight data indications to the Manager.

NOTE

Perform the End Device Bind procedure between the WeightScale device and the Manager device before sending the initial ZigBee Health care tunnel RECONNECT status notification request to the Manager device.

The switch and LED behavior in Configuration mode for the WeightScale application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.27.1 Keyboard

In run mode, key presses do the following:

- SW1 - Increases the current displayed value.
- SW2 - Decreases the current displayed value.
- SW3 - Sends an Event Report to the Manager device.
- Long SW3 - Switch display mode (height or weight).
- SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster RECONNECT status notification to the Manager device. The Manager replies with a Tunnel Connect Request and the tunnel is fully established once the CONNECTED status notification is sent back from the Agent to the Manager. Once the tunnel is established, the WeightScale continues to send a 11073 OEP Association Request to the Manager, followed by an Association Response from the Manager. This completes the tunnel and the IEEE 11073 OEP association process
- Long SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster DISCONNECT status notification to the Manager device. This triggers a tunnel disconnect and releases the OEP association.

7.28 Health Care Glucose Meter

The Health Care Glucose Meter (Hc GlucoseMeter) application implements a demonstration application for a ZigBee Health care Profile Glucose Meter device. The application is a prototype for the IEEE 11073-20601 Application Profile - Optimized Exchange Protocol (OEP) Agent device specification. The OEP layer of the application uses a ZigBee tunnel cluster to exchange health care specific device association, deassociation and data messages as shown in ZigBee Alliance document 075360.

The Glucose Meter application acts as an OEP Agent by sending an Association Request to the Manager to which the Manager replies with an Association Response. Users of the Hc Glucose Meter application can initiate a ZigBee tunnel reconnect status notification request to trigger the OEP layer association and send simulated values of blood glucose and meal context data indications to the Manager.

NOTE

Perform the End Device Bind procedure between the WeightScale device and the Manager device before sending the initial ZigBee Health care tunnel RECONNECT status notification request to the Manager device.

The switch and LED behavior in Configuration mode for the GlucoseMeter application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.28.1 Keyboard

In run mode, key presses do the following:

- SW1 - Increases the current displayed value.
- SW2 - Decreases the current displayed value.
- SW3 - Sends an Event Report to the Manager device.
- Long SW3 - Switch display mode (blood glucose and meal context).
- SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster RECONNECT status notification to the Manager device. The Manager replies with a Tunnel Connect Request and the tunnel is fully established once the CONNECTED status notification is sent back from the Agent to the Manager. Once the tunnel is established, the GlucoseMeter continues to send a 11073 OEP Association Request to the Manager, followed by an Association Response from the Manager. This completes the tunnel and the IEEE 11073 OEP association process
- Long SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster DISCONNECT status notification to the Manager device. This triggers a tunnel disconnect and releases the OEP association.

7.29 Health Care Blood Pressure Monitor

The Health Care Blood Pressure Monitor (Hc BpMonitor) application implements a demonstration application for a ZigBee Health care Profile Blood Pressure Monitor device. The application is a prototype for the IEEE 11073-20601 Application Profile - Optimized Exchange Protocol (OEP) Agent device specification. The OEP layer of the application uses a ZigBee tunnel cluster to exchange health care specific device association, deassociation and data messages as shown in ZigBee Alliance document 075360.

The BpMonitor application acts as an OEP Agent by sending an Association Request to the Manager to which the Manager replies with an Association Response. Users of the Hc BpMonitor application can initiate a ZigBee tunnel reconnect status notification request to trigger the OEP layer association and send simulated values of systolic, diastolic and pulse rate data indications to the Manager.

NOTE

Perform the End Device Bind procedure between the BpMonitor device and the Manager device before sending the initial ZigBee Health care tunnel RECONNECT status notification request to the Manager device.

The switch and LED behavior in Configuration mode for the BpMonitor application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.29.1 Keyboard

In run mode, key presses do the following:

- SW1 - Increases the current displayed value.
- SW2 - Decreases the current displayed value.
- SW3 - Sends an Event Report to the Manager device.
- Long SW3 - Switch display mode (systolic, diastolic and pulse rate).
- SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster RECONNECT status notification to the Manager device. The Manager replies with a Tunnel Connect Request and the tunnel is fully established once the CONNECTED status notification is sent back from the Agent to the Manager. Once the tunnel is established, the BpMonitor continues to send a 11073 OEP Association Request to the Manager, followed by an Association Response from the Manager. This completes the tunnel and the IEEE 11073 OEP association process

Long SW4 - Sends a ZigBee Health care IEEE 11073 Tunnel cluster DISCONNECT status notification to the Manager device. This triggers a tunnel disconnect and releases the OEP association.

7.30 Health Care Data Collection Unit ZTC Host

The Health Care Data Collection Unit ZTC Host (Hc DataCollectionUnit ZTC Host) application implements only the level of functionality defined by the ZHC specification, with the IEEE 11073-20601 manager message processing stack disabled. This can be useful if the IEEE 11073 message interpretation functionality is transferred to a different MCU (or another entity such as a PC application) communicating via the ZTC UART protocol to the BeeStack node. Once the ZHC tunnel is established, the entity connected to the ZTC Host will be able to send and receive ASPDE-DATA packets containing BeeStack ZCL Health Care TransferAPDU frames which encapsulate IEEE 11073 frames. See the Freescale BlackBox and ZTC Reference Manual as well as the ZigBee Health Care Specification for reference information on the format of the frames transmitted using ZTC.

NOTE

Perform the End Device Bind procedure between the DataCollectionUnit device and the Agent before sending the initial ZigBee Health care tunnel RECONNECT status notification request from the Agent device.

The application only implements the Configuration mode and does not implement the Run Mode. The switch and LED behavior in Configuration mode for the Health Care Data Collection application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.31 Health Care Data Collection Unit UART Tunnel

The Health Care Data Collection Unit UART Tunnel Host (Hc DataCollectionUnit UART Tunnel) application implements only the level of functionality defined by the ZHC specification, with the IEEE 11073-20601 manager message processing stack disabled. This can be useful if the IEEE 11073 message interpretation functionality is transferred to a different entity designed to use generic IEEE 11073 frames as the communication protocol. The BeeStack node will output on the UART only the standard IEEE 11073 frames without any ZigBee specific data field. Similarly, the frames received on the port must be standard (must begin with the InvokeId field). Frames received on the UART are encapsulated into TransferAPDU ZCL Health Care frames which are then transmitted over the air to the latest agent node that has communicated with the Data Collection Unit on the ZHC cluster.

NOTES

ZTC must be disabled when using this application

Perform the End Device Bind procedure between the DataCollectionUnit device and the Agent before sending the initial ZigBee Health care tunnel RECONNECT status notification request from the Agent device.

The application only implements the Configuration mode and does not implement the Run Mode. The switch and LED behavior in Configuration mode for the Health Care Data Collection application is the same as the behavior of the applications shown in Section 7.3.1, “ASL Configuration Mode” and Section 7.4.1, “Configuration Mode”.

7.32 Health Care Generic Agent and Generic Manager

The Health Care Generic Agent and Generic Manager application serve as empty placeholders that can be used as starting points for the user to begin adding custom IEEE 11073 attributes and functionality for a specific health care application. Users should edit the BeeApp.c/.h and Oep11073Config.c/.h to define and implement a useful application.

Chapter 8 Creating a BeeStack BlackBox Binary

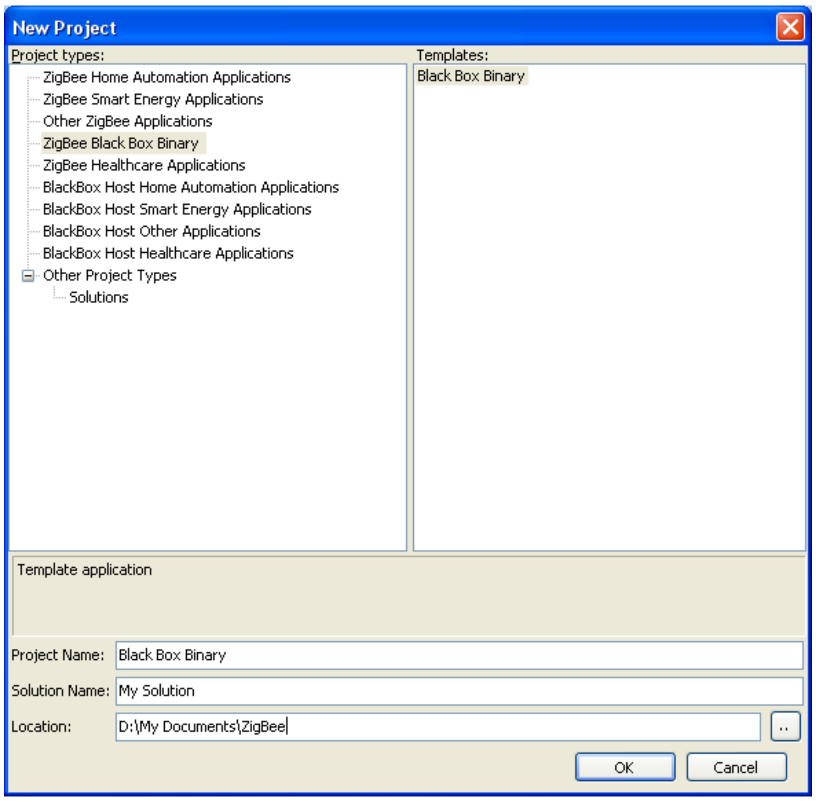
This chapter shows how to create a BeeStack BlackBox binary using the Freescale BeeKit Wireless Connectivity Toolkit, how to download the image to the target board and how to use the BeeStack BlackBox to create a small network formed by an HA OnOffLight device running as the PAN coordinator and a HA OnOffSwitch running as an end device.

The Freescale Test Tool is used to send and receive data. The BeeStack Black box Reference Manual Provides further details about host interface, protocol framing and command formatting.

8.1 BeeStack BlackBox Binary Generation

Following these steps on both the coordinator and the end device to generate the BeeStack BlackBox image.

1. Start the BeeKit Wireless Connectivity Toolkit.
2. Select the appropriate BeeStack codebase.
3. From the menu, select File->New Project… The New Project window appears as shown in Figure 8-1.
4. From the left side of the New Project, select ZigBee Black Box Binary, as shown in Figure 8-1.
5. Select the Black Box Binary template from the right side of the window.


New Project

Project types:
- ZigBee Home Automation Applications
- ZigBee Smart Energy Applications
- Other ZigBee Applications
- ZigBee Black Box Binary
- ZigBee Healthcare Applications
- BlackBox Host Home Automation Applications
- BlackBox Host Smart Energy Applications
- BlackBox Host Other Applications
- BlackBox Host Healthcare Applications
- Other Project Types
  - Solutions

Templates: Black Box Binary

Template application

Project Name: Black Box Binary

Solution Name: My Solution

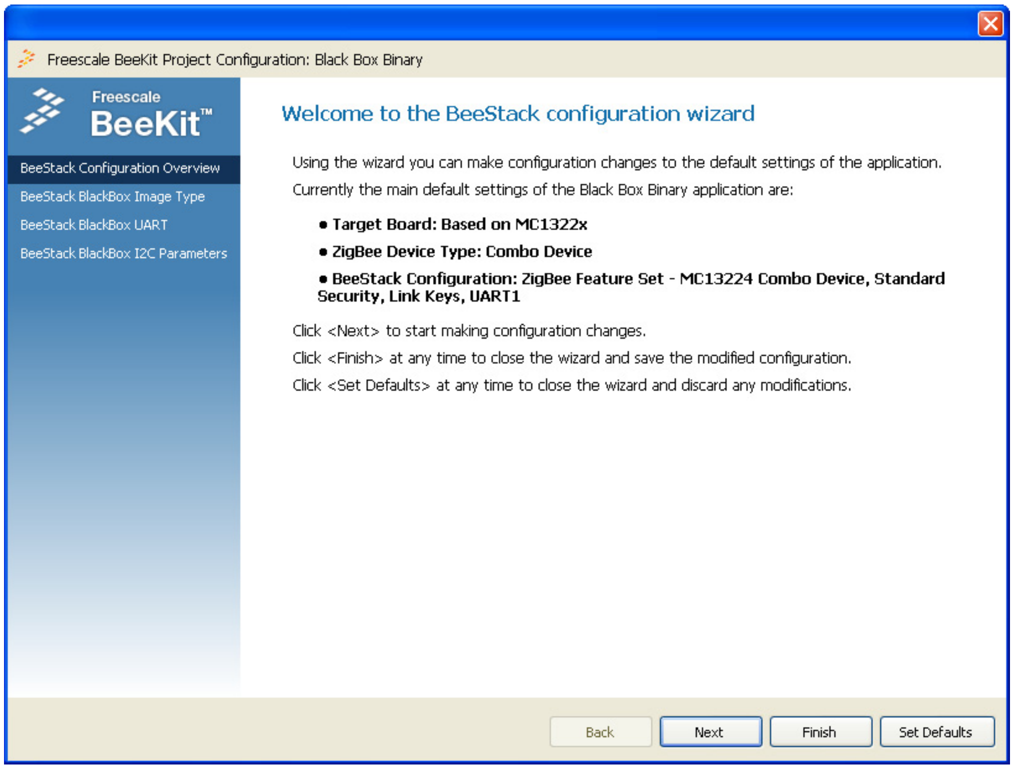
Location: D:\My Documents\ZigBee

OK Cancel

Figure 8-1. BeeKit New Project Window

6. Enter a name, solution location and project name at the bottom of the New Project window.
7. Click OK to create the project.

   The New Project window closes and the BeeKit Project Configuration window appears as shown in Figure 8-2.


Freescale BeeKit Project Configuration: Black Box Binary
Freescale BeeKit™
BeeStack Configuration Overview
BeeStack BlackBox Image Type
BeeStack BlackBox UART
BeeStack BlackBox I2C Parameters
Back
Next
Finish
Set Defaults

Welcome to the BeeStack configuration wizard

Using the wizard you can make configuration changes to the default settings of the application.

Currently the main default settings of the Black Box Binary application are:

- **Target Board: Based on MC1322x**
- **ZigBee Device Type: Combo Device**
- **BeeStack Configuration: ZigBee Feature Set - MC13224 Combo Device, Standard Security, Link Keys, UART1**

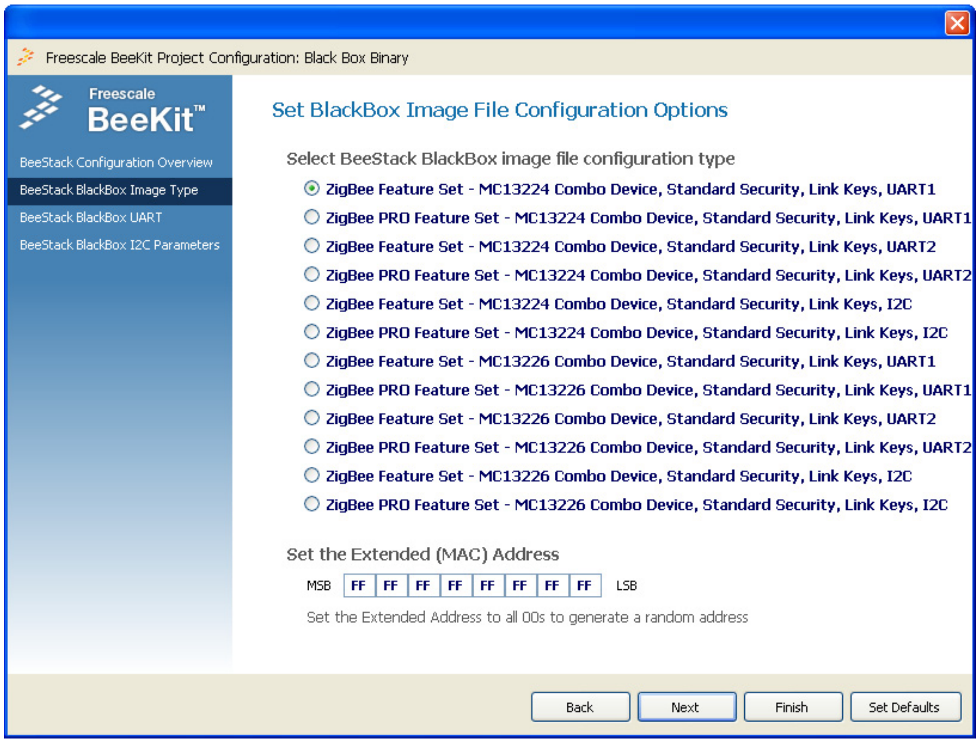
Click <Next> to start making configuration changes.

Click <Finish> at any time to close the wizard and save the modified configuration.

Click <Set Defaults> at any time to close the wizard and discard any modifications.

Figure 8-2. BeeKit Project Configuration window

8. Click on the Next button. The BeeStack BlackBox Image Type window appears as show in Figure 8-3.


Freescale BeeKit Project Configuration: Black Box Binary
Freescale
BeeKit™
BeeStack Configuration Overview
BeeStack BlackBox Image Type
BeeStack BlackBox UART
BeeStack BlackBox I2C Parameters
Back
Next
Finish
Set Defaults

Set BlackBox Image File Configuration Options

Select BeeStack BlackBox image file configuration type

- [✓ ZigBee Feature Set - MC13224 Combo Device, Standard Security, Link Keys, UART1]
- ZigBee PRO Feature Set - MC13224 Combo Device, Standard Security, Link Keys, UART1
- ZigBee Feature Set - MC13224 Combo Device, Standard Security, Link Keys, UART2
- ZigBee PRO Feature Set - MC13224 Combo Device, Standard Security, Link Keys, UART2
- ZigBee Feature Set - MC13224 Combo Device, Standard Security, Link Keys, I2C
- ZigBee PRO Feature Set - MC13224 Combo Device, Standard Security, Link Keys, I2C
- ZigBee Feature Set - MC13226 Combo Device, Standard Security, Link Keys, UART1
- ZigBee PRO Feature Set - MC13226 Combo Device, Standard Security, Link Keys, UART1
- ZigBee Feature Set - MC13226 Combo Device, Standard Security, Link Keys, UART2
- ZigBee PRO Feature Set - MC13226 Combo Device, Standard Security, Link Keys, UART2
- ZigBee Feature Set - MC13226 Combo Device, Standard Security, Link Keys, I2C
- ZigBee PRO Feature Set - MC13226 Combo Device, Standard Security, Link Keys, I2C

Set the Extended (MAC) Address

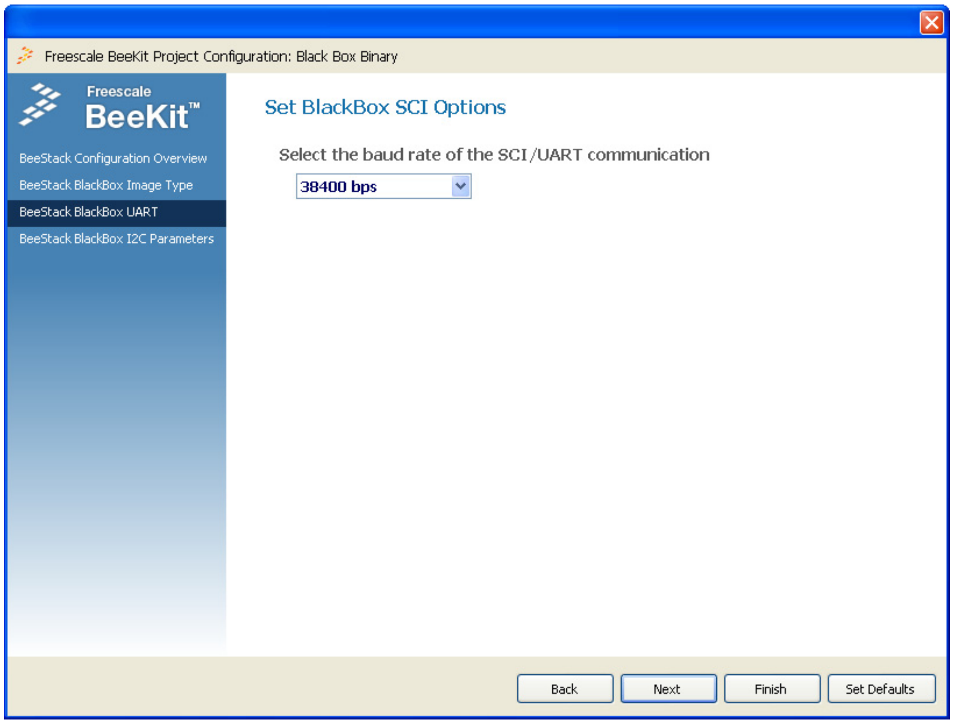
MSB FF FF FF FF FF FF FF FF LSB

Set the Extended Address to all 00s to generate a random address

Figure 8-3. BeeStack Black Box image type window

9. Select the BeeStack BlackBox image file from the available options.Set the Extended (MAC) Address of the device.
10. Click on the Next Button. The BlackBox connection settings window appears. This window shows the connection parameters specific to the selected communication interface (UART or I2C).

    If a UART based image was selected, the connection settings window allow users to set the UART baud rate, as shown in Figure 8-4.

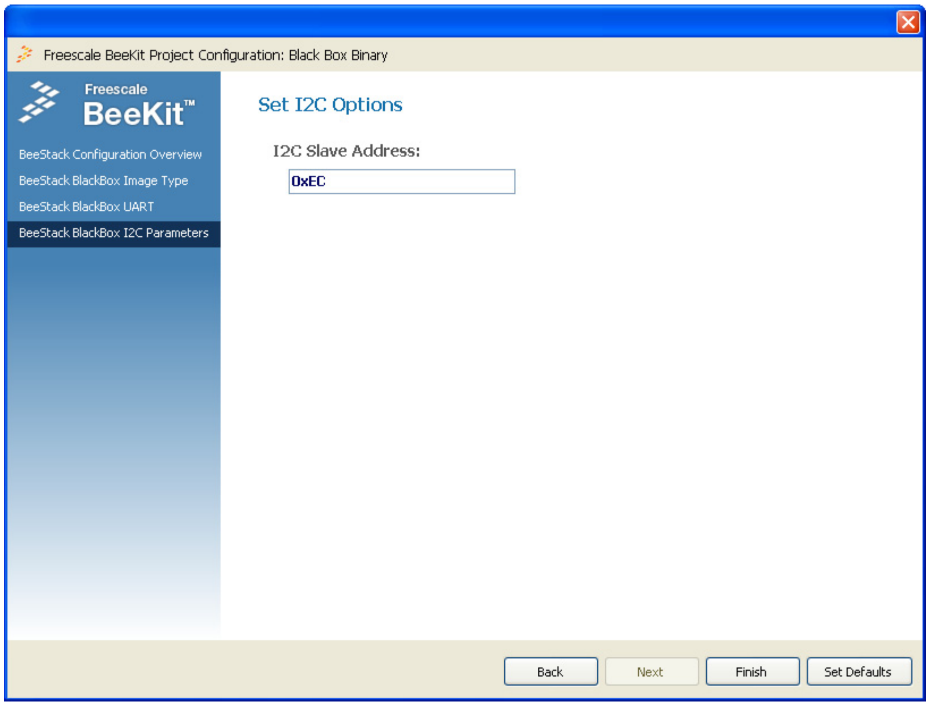

Freescale BeeKit Project Configuration: Black Box Binary
Freescale
BeeKit™
BeeStack Configuration Overview
BeeStack BlackBox Image Type
BeeStack BlackBox UART
BeeStack BlackBox I2C Parameters
Back
Next
Finish
Set Defaults

Set BlackBox SCI Options

Select the baud rate of the SCI/UART communication: 38400 bps

Figure 8-4. UART Settings Window

If an I2C interface has been selected, the connection settings window will allow setting the I2C slave address, as shown in Figure 8-5.


Freescale BeeKit Project Configuration: Black Box Binary
Freescale
BeeKit™
BeeStack Configuration Overview
BeeStack BlackBox Image Type
BeeStack BlackBox UART
BeeStack BlackBox I2C Parameters
Back
Next
Finish
Set Defaults

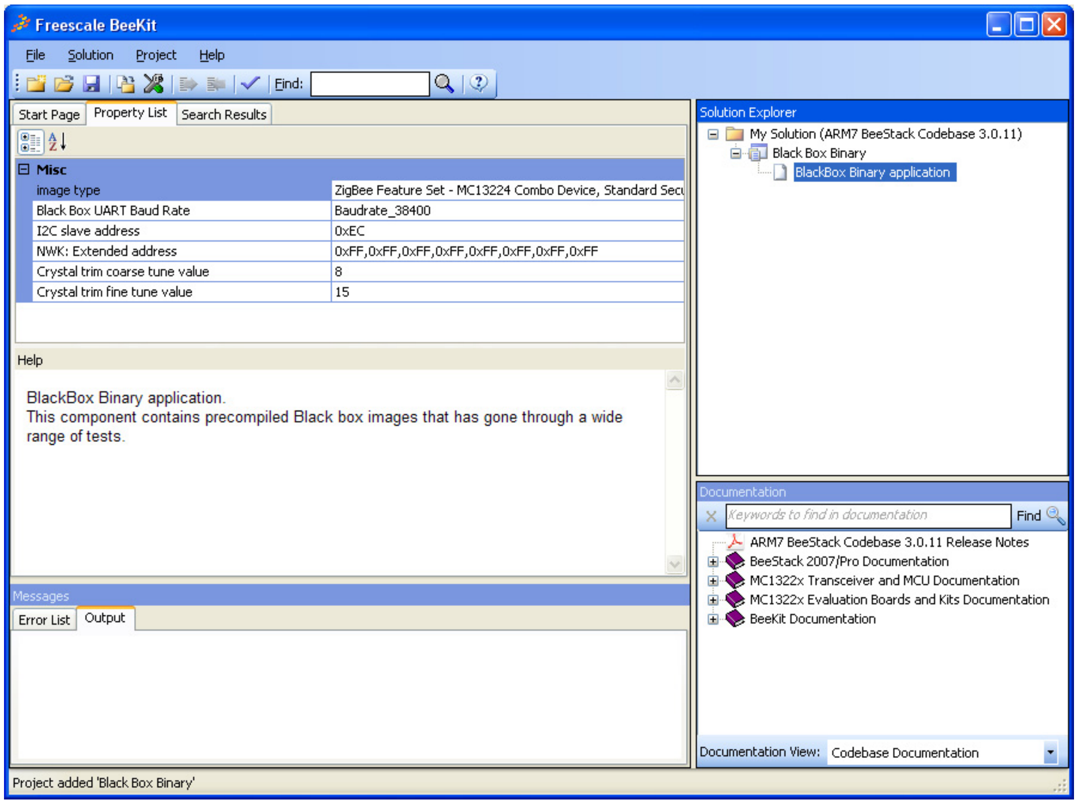
Set I2C Options

I2C Slave Address: 0xEC

Figure 8-5. I2C Settings Window

11. Click on the Finish button.

    The BeeKit main window appears as shown in Figure 8-6, and the created solution is now ready for export to the development tool.



Misc

image type	ZigBee Feature Set - MC13224 Combo Device, Standard Secu
Black Box UART Baud Rate	Baudrate_38400
I2C slave address	0xEC
NWK: Extended address	0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
Crystal trim coarse tune value	8
Crystal trim fine tune value	15

Help

BlackBox Binary application.
This component contains precompiled Black box images that has gone through a wide range of tests.

My Solution (ARM7 BeeStack Codebase 3.0.11)
Black Box Binary
BlackBox Binary application

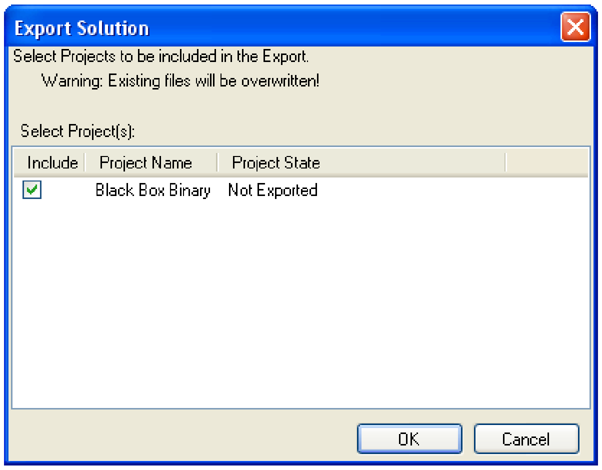
ARM7 BeeStack Codebase 3.0.11 Release Notes
BeeStack 2007/Pro Documentation
MC1322x Transceiver and MCU Documentation
MC1322x Evaluation Boards and Kits Documentation
BeeKit Documentation

Documentation View: Codebase Documentation

Project added 'Black Box Binary'

Figure 8-6. BeeKit Main Window (BlackBox Project)

12. From the menu select Solution\Export Solution. The Export Solution window appears, as shown in Figure 8-7.



Select Projects to be included in the Export.
Warning: Existing files will be overwritten!

Select Project(s):

Include	Project Name	Project State
✓	Black Box Binary	Not Exported

Figure 8-7. Export Solution Window

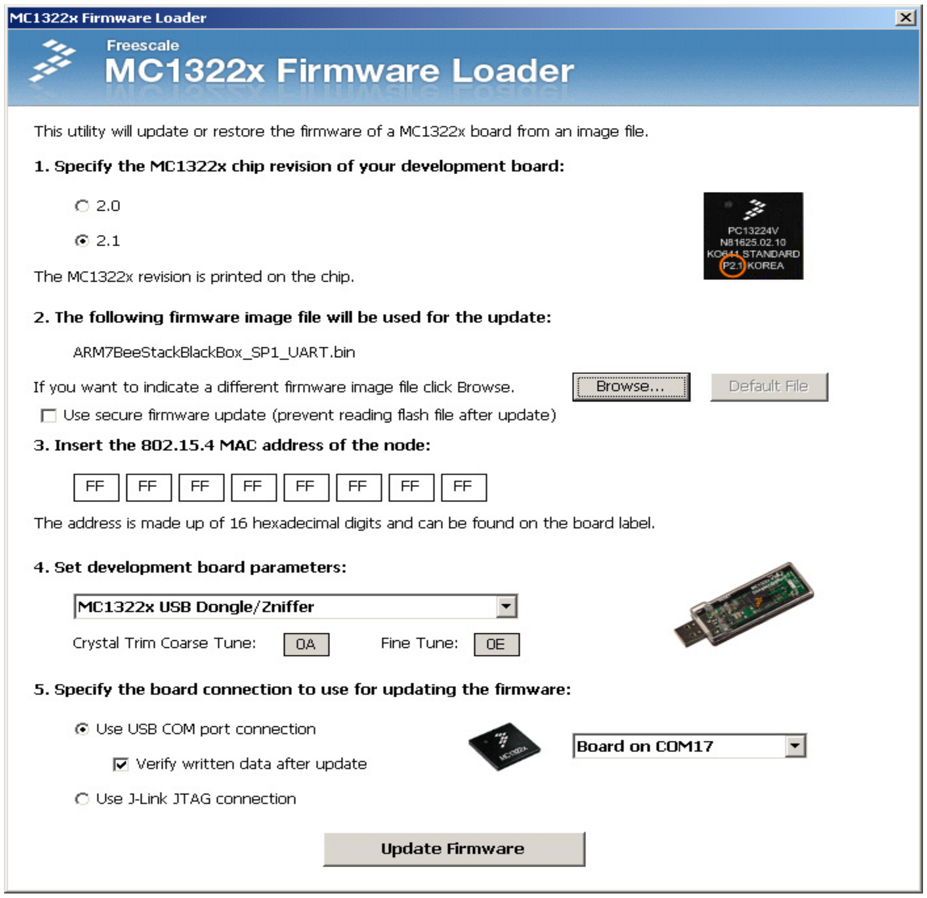
13. Click the OK button. A window appears where the available BlackBox image can be selected.

8.2 Uploading the BlackBox Image to the Target Board

If the target board is based on the MC1322x, then load the BlackBox image using the Freescale TestTool MC1322x Firmware Loader. If the target board is HCS08 based, use Freescale TestTool HCS08 Firmware Loader.

8.2.1 Uploading the Image on a MC1322x Target Board

1. From the Test Tool start page, click on Launch ARM7 Firmware Loader. The MC1322x Firmware Loader appears as shown in Figure 8-8.



This utility will update or restore the firmware of a MC1322x board from an image file.

1. Specify the MC1322x chip revision of your development board:

( ) 2.0
[✓ 2.1]

The MC1322x revision is printed on the chip.

2. The following firmware image file will be used for the update:

ARM7BeeStackBlackBox_SP1_UART.bin

If you want to indicate a different firmware image file click Browse.

[ ] Use secure firmware update (prevent reading flash file after update)

3. Insert the 802.15.4 MAC address of the node:

FF FF FF FF FF FF FF FF

The address is made up of 16 hexadecimal digits and can be found on the board label.

4. Set development board parameters:

MC1322x USB Dongle/Zniffer

Crystal Trim Coarse Tune: 0A Fine Tune: 0E

5. Specify the board connection to use for updating the firmware:

[✓ Use USB COM port connection]

[✓ Verify written data after update]

( ) Use J-Link JTAG connection

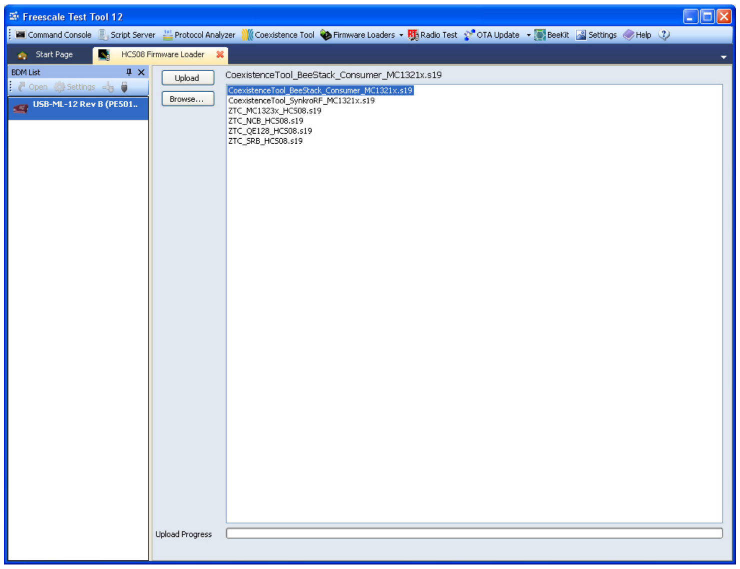
Board on COM17

Figure 8-8. MC1322x Firmware Loader Window

2. To select the BlackBox image click on the Browse button.
3. Select the appropriate target board type.
4. To upload the image, select which connection to use (JTAG or USB COM).
5. Click the Update Firmware button.
6. Follow the upload instructions as they appear.

8.2.2 Uploading the Image on a HCS08 Target Board

1. Connect a BDM to the target board.
2. From the Test Tool (version 12 or later) start page, click on Launch HCS08 Firmware Loader. The HCS08 Firmware Loader appears as shown in Figure 8-9.



CoexistenceTool_BeeStack_Consumer_MC1321x.s19

CoexistenceTool_BeeStack_Consumer_MC1321x.s19
CoexistenceTool_SynkroRF_MC1321x.s19
ZTC_MC1323x_HCS08.s19
ZTC_NCB_HCS08.s19
ZTC_QE128_HCS08.s19
ZTC_SRB_HCS08.s19

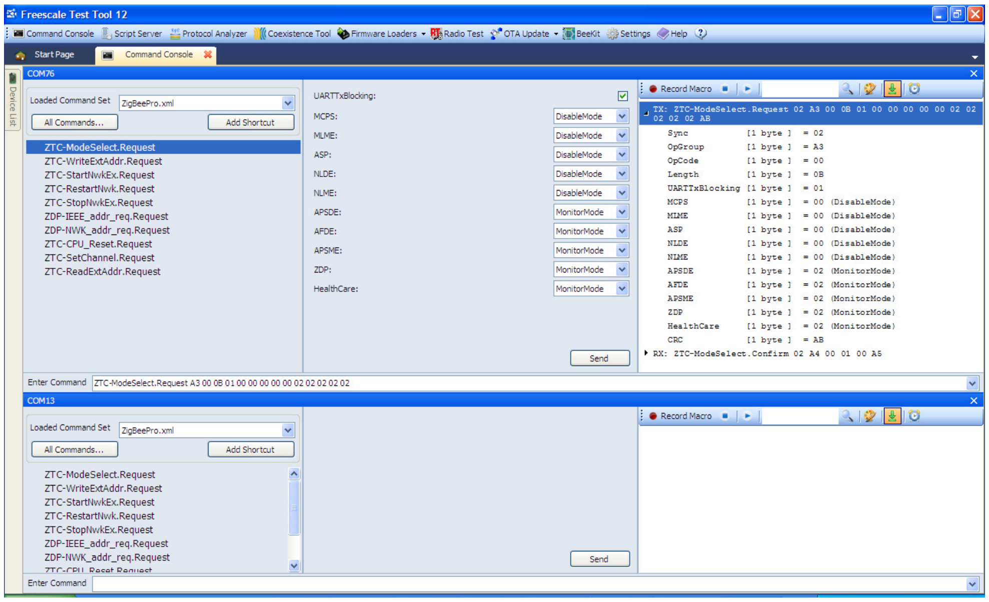
Figure 8-9. HCS08 Firmware Loader Window

3. To select the BlackBox image click on the Browse button, navigate to the location of the BlackBox *.s19 file and select it.
4. To upload the image, click the Upload button and wait until the upload is complete.

8.3 Connecting the Freescale TestTool to the Boards

After the BeeStack BlackBox image has been downloaded on the two target boards, the Freescale Test Tool can be used to send commands to the devices. The following steps must be performed on both the coordinator and the end device to connect to the Freescale Test Tool.

1. Connect the BlackBox device to the USB port of the PC.
2. Configure Test Tool so that it can communicate with the board. See the *Freescale Test Tool User's Guide* for more information.
3. From Test Tool, start the Command Console and ensure that the `ZigBeePro.xml` file is selected.
4. Send a ZTC-ModeSelect.Request command with the parameters shown in Figure 8-10.



COM76 — Loaded Command Set: ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

UARTTxBlocking:	✓
MCPS:	DisableMode
MLME:	DisableMode
ASP:	DisableMode
NLDE:	DisableMode
NLME:	DisableMode
APSDE:	MonitorMode
AFDE:	MonitorMode
APSME:	MonitorMode
ZDP:	MonitorMode
HealthCare:	MonitorMode

```
TX: ZTC-ModeSelect.Request 02 A3 00 0B 01 00 00 00 00 00 02 02
02 02 02 AB
    Sync          [1 byte ]  = 02
    OpGroup       [1 byte ]  = A3
    OpCode        [1 byte ]  = 00
    Length        [1 byte ]  = 0B
    UARTTxBlocking [1 byte ] = 01
    MCPS          [1 byte ]  = 00 (DisableMode)
    MLME          [1 byte ]  = 00 (DisableMode)
    ASP           [1 byte ]  = 00 (DisableMode)
    NLDE          [1 byte ]  = 00 (DisableMode)
    NLME          [1 byte ]  = 00 (DisableMode)
    APSDE         [1 byte ]  = 02 (MonitorMode)
    AFDE          [1 byte ]  = 02 (MonitorMode)
    APSME         [1 byte ]  = 02 (MonitorMode)
    ZDP           [1 byte ]  = 02 (MonitorMode)
    HealthCare    [1 byte ]  = 02 (MonitorMode)
    CRC           [1 byte ]  = AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00 A5
```

Enter Command: ZTC-ModeSelect.Request A3 00 0B 01 00 00 00 00 00 02 02 02 02 02

COM13 — Loaded Command Set: ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request

Enter Command

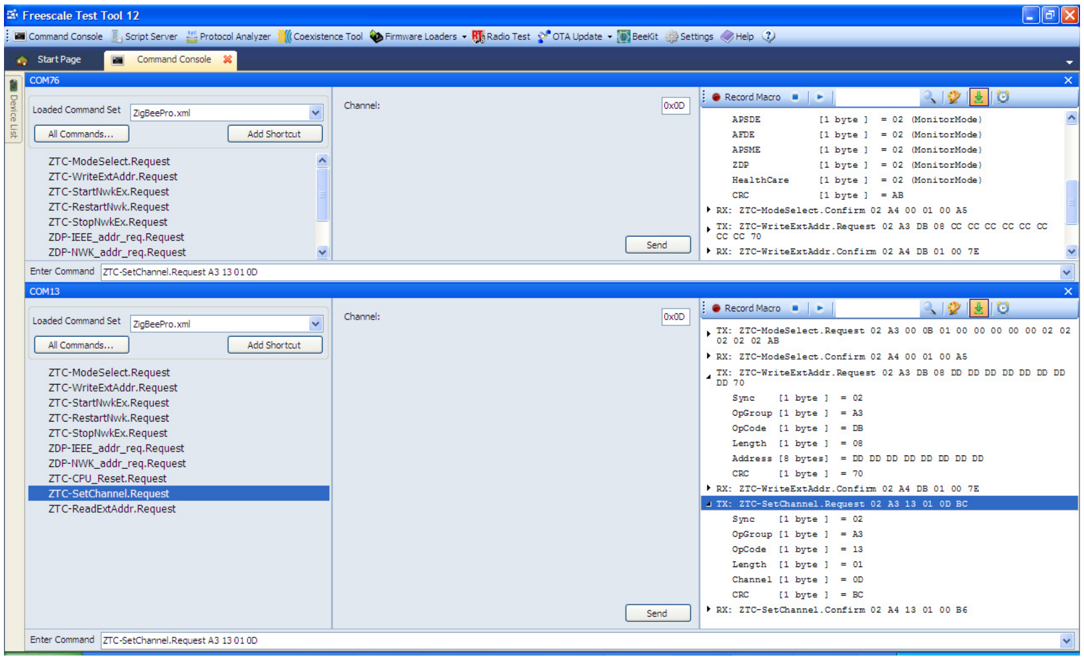
Figure 8-10. ZTC-ModeSelectRequest Command

8.4 Forming the ZigBee Network

1. First set the extended address on both devices by sending a ZCL-WriteExtAddr.Request command to each of the devices.

   For this example, use the addresses 0xCCCCCCCCCCCCCCCC for the PAN coordinator and 0xDDDDDDDDDDDDDDDD for the end device.

   Also, set the operating channel on the PAN coordinator by sending a ZTC-SetChannel command to it. In this example, channel 0x0D is used.



COM76 — Loaded Command Set: ZigBeePro.xml — Channel: 0x0D

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request

```
    APSDE         [1 byte ]  = 02 (MonitorMode)
    AFDE          [1 byte ]  = 02 (MonitorMode)
    APSME         [1 byte ]  = 02 (MonitorMode)
    ZDP           [1 byte ]  = 02 (MonitorMode)
    HealthCare    [1 byte ]  = 02 (MonitorMode)
    CRC           [1 byte ]  = AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00 A5
TX: ZTC-WriteExtAddr.Request 02 A3 DB 08 CC CC CC CC CC CC
CC CC 70
RX: ZTC-WriteExtAddr.Confirm 02 A4 DB 01 00 7E
```

Enter Command: ZTC-SetChannel.Request A3 13 01 0D

COM13 — Loaded Command Set: ZigBeePro.xml — Channel: 0x0D

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

```
TX: ZTC-ModeSelect.Request 02 A3 00 0B 01 00 00 00 00 00 02 02
02 02 02 AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00 A5
TX: ZTC-WriteExtAddr.Request 02 A3 DB 08 DD DD DD DD DD DD DD
DD 70
    Sync    [1 byte ]  = 02
    OpGroup [1 byte ]  = A3
    OpCode  [1 byte ]  = DB
    Length  [1 byte ]  = 08
    Address [8 bytes]  = DD DD DD DD DD DD DD DD
    CRC     [1 byte ]  = 70
RX: ZTC-WriteExtAddr.Confirm 02 A4 DB 01 00 7E
TX: ZTC-SetChannel.Request 02 A3 13 01 0D BC
    Sync    [1 byte ]  = 02
    OpGroup [1 byte ]  = A3
    OpCode  [1 byte ]  = 13
    Length  [1 byte ]  = 01
    Channel [1 byte ]  = 0D
    CRC     [1 byte ]  = BC
RX: ZTC-SetChannel.Confirm 02 A4 13 01 00 B6
```

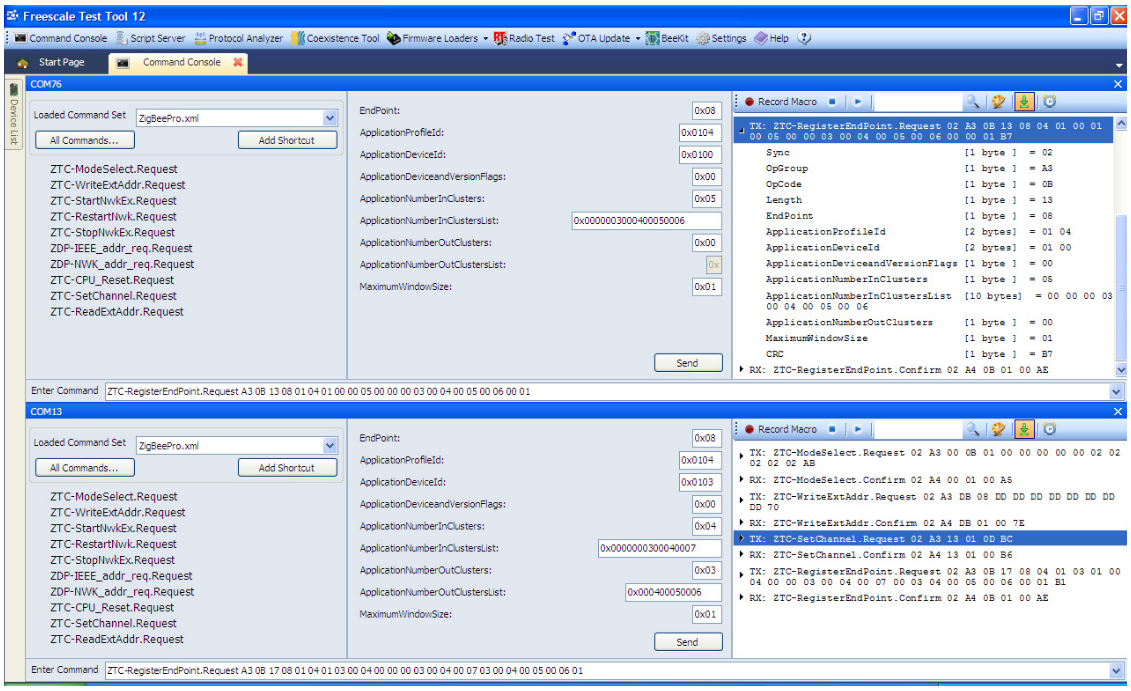
Enter Command: ZTC-SetChannel.Request A3 13 01 0D

Figure 8-11. Setting the Extended Address and Channel

8.4.1 Registering End Points

Follow these steps to register the endpoints on the PAN coordinator.

1. Choose the ZTC-RegisterEndPoint.Request command from the All Commands-> ZTC Commands menu.
2. Enter the following values as shown in Figure 8-12.
   — EndPoint: 8
   — ApplicationProfileId: 0x0104 (HA)
   — ApplicationDeviceId: 0x0100 (OnOffLight)
   — ApplicationDeviceAndVersionFlags: 0x00
   — ApplicationNumberInClusters: 0x05
   — ApplicatioonNumberInClustersList: 0x0000, 0x0003, 0x0004, 0x0005, 0x0006
   — ApplicationNumberOutClusters: 0



COM76

Loaded Command Set: ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

EndPoint: 0x08
ApplicationProfileId: 0x0104
ApplicationDeviceId: 0x0100
ApplicationDeviceandVersionFlags: 0x00
ApplicationNumberInClusters: 0x05
ApplicationNumberInClustersList: 0x00000003000400050006
ApplicationNumberOutClusters: 0x00
ApplicationNumberOutClustersList: 0x
MaximumWindowSize: 0x01

```
TX: ZTC-RegisterEndPoint.Request 02 A3 0B 13 08 04 01 00 01
00 05 00 00 03 00 04 00 05 00 06 00 00 01 B7
    Sync                              [1 byte ]  = 02
    OpGroup                           [1 byte ]  = A3
    OpCode                            [1 byte ]  = 0B
    Length                            [1 byte ]  = 13
    EndPoint                          [1 byte ]  = 08
    ApplicationProfileId              [2 bytes]  = 01 04
    ApplicationDeviceId               [2 bytes]  = 01 00
    ApplicationDeviceandVersionFlags  [1 byte ]  = 00
    ApplicationNumberInClusters       [1 byte ]  = 05
    ApplicationNumberInClustersList   [10 bytes]  = 00 00 00 03
    00 04 00 05 00 06
    ApplicationNumberOutClusters      [1 byte ]  = 00
    MaximumWindowSize                 [1 byte ]  = 01
    CRC                               [1 byte ]  = B7
RX: ZTC-RegisterEndPoint.Confirm 02 A4 0B 01 00 AE
```

Enter Command: ZTC-RegisterEndPoint.Request A3 0B 13 08 01 04 01 00 00 05 00 00 00 03 00 04 00 05 00 06 00 01

COM13

Loaded Command Set: ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

EndPoint: 0x08
ApplicationProfileId: 0x0104
ApplicationDeviceId: 0x0103
ApplicationDeviceandVersionFlags: 0x00
ApplicationNumberInClusters: 0x04
ApplicationNumberInClustersList: 0x0000000300040007
ApplicationNumberOutClusters: 0x03
ApplicationNumberOutClustersList: 0x000400050006
MaximumWindowSize: 0x01

```
TX: ZTC-ModeSelect.Request 02 A3 00 0B 01 00 00 00 00 00 02 02
02 02 02 AB
RX: ZTC-ModeSelect.Confirm 02 A4 00 01 00 A5
TX: ZTC-WriteExtAddr.Request 02 A3 DB 08 DD DD DD DD DD DD DD
DD 70
RX: ZTC-WriteExtAddr.Confirm 02 A4 DB 01 00 7E
TX: ZTC-SetChannel.Request 02 A3 13 01 0D BC
RX: ZTC-SetChannel.Confirm 02 A4 13 01 00 B6
TX: ZTC-RegisterEndPoint.Request 02 A3 0B 17 08 04 01 03 01 00
04 00 00 03 00 04 00 07 00 03 04 00 05 00 06 00 01 B1
RX: ZTC-RegisterEndPoint.Confirm 02 A4 0B 01 00 AE
```

Enter Command: ZTC-RegisterEndPoint.Request A3 0B 17 08 01 04 01 03 00 04 00 00 00 03 00 04 00 07 03 00 04 00 05 00 06 01

Figure 8-12. EndPointDescription Window.

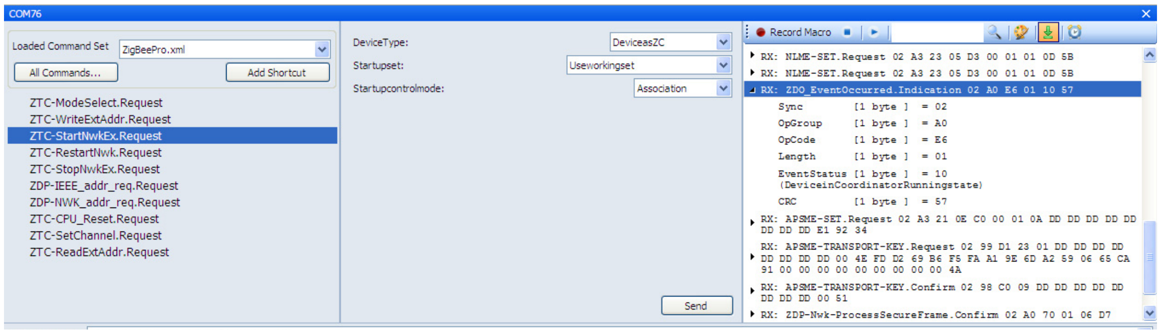
3. Send the command to the coordinator.

   Follow the same steps to register the endpoint on the end device. Use the the following settings for the end point description parameters:
   — EndPoint: 8
   — ApplicationProfileId: 0x0104 (HA)
   — ApplicationDeviceId: 0x0103 (OnOffSwitch)
   — ApplicationDeviceAndVersionFlags: 0x00
   — ApplicationNumberInClusters: 0x04

— ApplicatioonNumberInClustersList: 0x0000, 0x0003, 0x0004, 0x0007
— ApplicationNumberOutClusters: 0x03
— ApplicatioonNumberInClustersList: 0x0004, 0x0005, 0x0006

8.4.2 Starting The PAN

As shown in Figure 8-13, on the PAN coordinator, send a ZTC-StartNwkEx.Request to start a PAN. This example uses the default startup set of attribute values. In addition, the values of the startup set can be modified using the WriteSAS command.



ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

DeviceType: DeviceasZC
Startupset: Useworkingset
Startupcontrolmode: Association

```
RX: NLME-SET.Request 02 A3 23 05 D3 00 01 01 0D 5B
RX: NLME-SET.Request 02 A3 23 05 D3 00 01 01 0D 5B
RX: ZDO_EventOccurred.Indication 02 A0 E6 01 10 57
    Sync        [1 byte ]  = 02
    OpGroup     [1 byte ]  = A0
    OpCode      [1 byte ]  = E6
    Length      [1 byte ]  = 01
    EventStatus [1 byte ]  = 10
    (DeviceinCoordinatorRunningstate)
    CRC         [1 byte ]  = 57
RX: APSME-SET.Request 02 A3 21 0E C0 00 01 0A DD DD DD DD DD
DD DD DD E1 92 34
RX: APSME-TRANSPORT-KEY.Request 02 99 D1 23 01 DD DD DD DD
DD DD DD DD 00 4E FD D2 69 B6 F5 FA A1 9E 6D A2 59 06 65 CA
91 00 00 00 00 00 00 00 00 00 4A
RX: APSME-TRANSPORT-KEY.Confirm 02 98 C0 09 DD DD DD DD DD
DD DD DD 00 51
RX: ZDP-Nwk-ProcessSecureFrame.Confirm 02 A0 70 01 06 D7
```

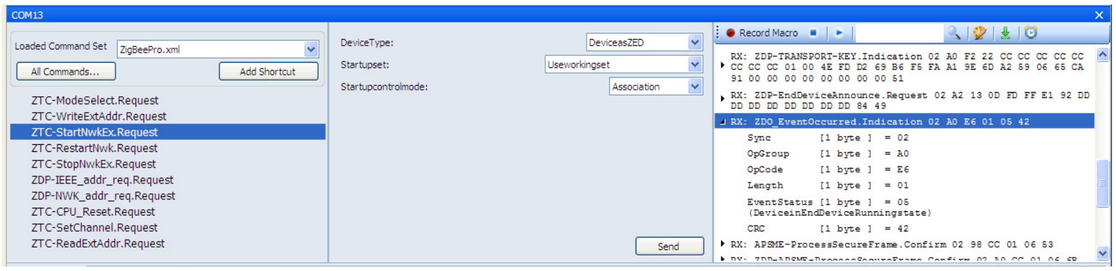
Figure 8-13. ZTC-StartNwkEx-Request command

When the ZTC-StartNwkEx-Request is set the ZigBee ZDO state machine reports what is happening. The NLME Service access point is monitored automatically to allow the host CPU to capture, join and leave events. In this case, the system shows the Network discovery request and network formation request made by the ZDO state machine. The ZDO state machine also reports its state change through the ZDO_EventOccurred.Indication. The last event shows that the coordinator is now running and ready.

To allow devices to join the newly formed PAN, send a ZDP-Mgmt_Permit_Joining.Request to the coordinator. This request is sent locally on the coordinator, which has a network address of 0x0000, so this address should be used for the destination address. Also, use 0xFF and 0x00 for the PermitDuration and TCSignificance parameters.

8.4.3 Joining the PAN

Send the ZTC-StartNwkEx-Request command to the end device to join the PAN. See Figure 8-14. Set the DeviceType to “Device as ZED”.



ZigBeePro.xml

ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request

DeviceType: DeviceasZED
Startupset: Useworkingset
Startupcontrolmode: Association

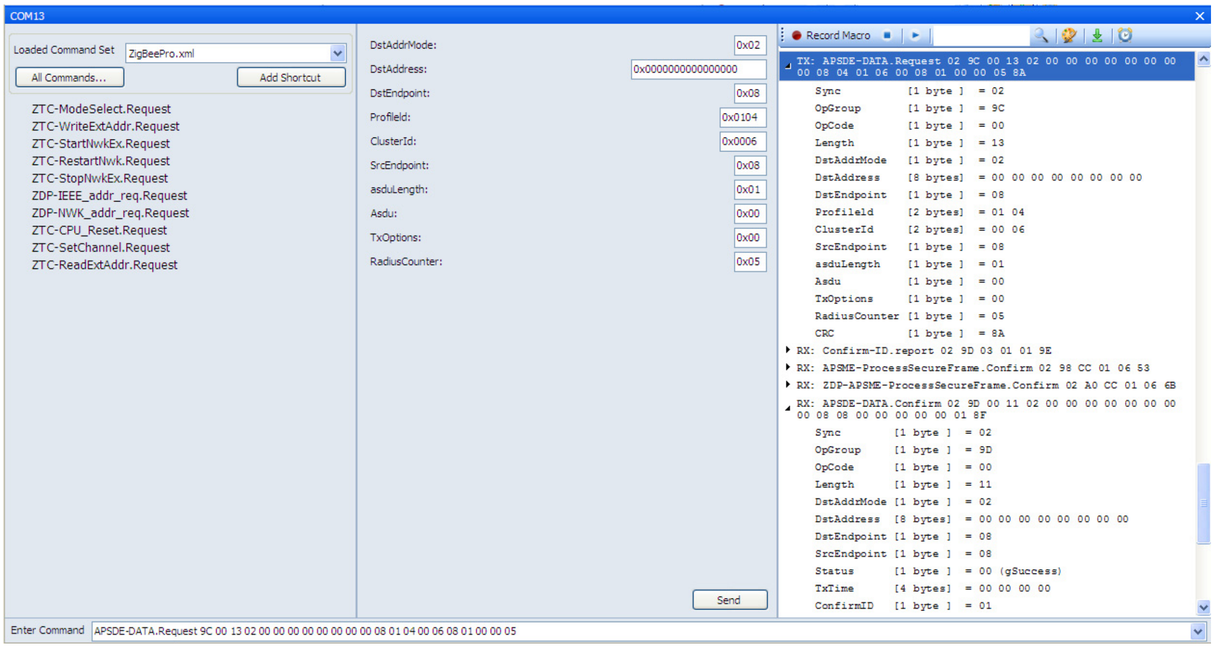
```
RX: ZDP-TRANSPORT-KEY.Indication 02 A0 F2 22 CC CC CC CC CC
CC CC CC 01 00 4E FD D2 69 B6 F5 FA A1 9E 6D A2 59 06 65 CA
91 00 00 00 00 00 00 00 00 51
RX: ZDP-EndDeviceAnnounce.Request 02 A2 13 0D FD FF E1 92 DD
DD DD DD DD DD DD DD 84 49
RX: ZDO_EventOccurred.Indication 02 A0 E6 01 05 42
    Sync        [1 byte ]  = 02
    OpGroup     [1 byte ]  = A0
    OpCode      [1 byte ]  = E6
    Length      [1 byte ]  = 01
    EventStatus [1 byte ]  = 05
    (DeviceinEndDeviceRunningstate)
    CRC         [1 byte ]  = 42
RX: APSME-ProcessSecureFrame.Confirm 02 98 CC 01 06 53
```

Figure 8-14. ZED Joined

The APSME service access point is also monitored and in this case it shows the Security indications received during the join.

8.4.4 Sending Data

Once joined, the end device can now use the APSDE-Data.Request command to send data to the coordinator. Use the settings in as shown in Figure 8-15.



```
ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request
```

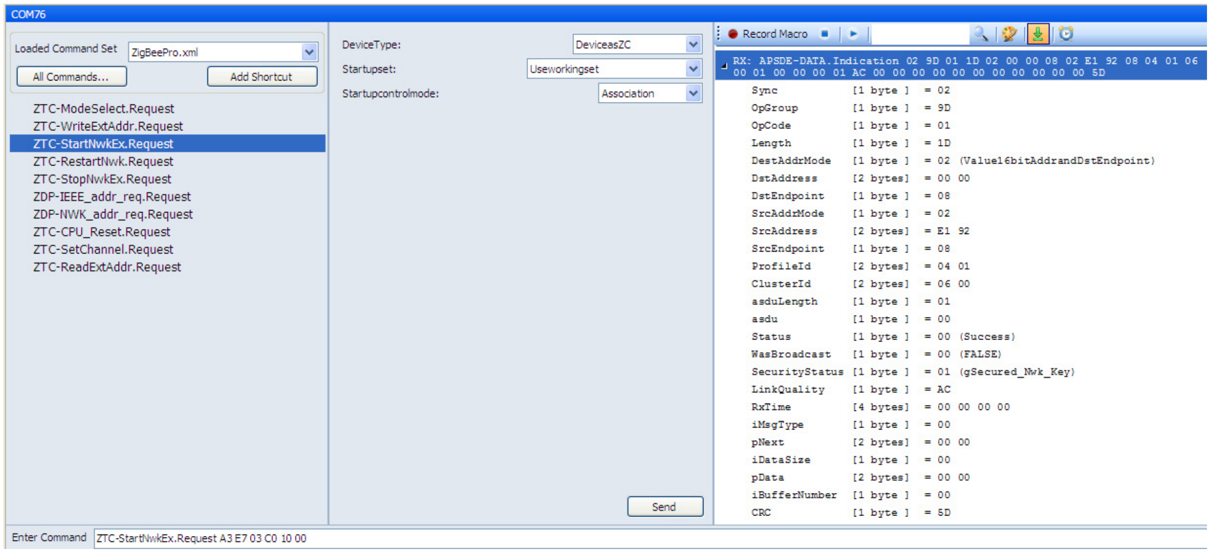
```
DstAddrMode:     0x02
DstAddress:      0x0000000000000000
DstEndpoint:     0x08
ProfileId:       0x0104
ClusterId:       0x0006
SrcEndpoint:     0x08
asduLength:      0x01
Asdu:            0x00
TxOptions:       0x00
RadiusCounter:   0x05
```

```
TX: APSDE-DATA.Request 02 9C 00 13 02 00 00 00 00 00 00 00
00 08 04 01 06 00 08 01 00 00 05 8A
    Sync          [1 byte ]  = 02
    OpGroup       [1 byte ]  = 9C
    OpCode        [1 byte ]  = 00
    Length        [1 byte ]  = 13
    DstAddrMode   [1 byte ]  = 02
    DstAddress    [8 bytes]  = 00 00 00 00 00 00 00 00
    DstEndpoint   [1 byte ]  = 08
    Profileld     [2 bytes]  = 01 04
    ClusterId     [2 bytes]  = 00 06
    SrcEndpoint   [1 byte ]  = 08
    asduLength    [1 byte ]  = 01
    Asdu          [1 byte ]  = 00
    TxOptions     [1 byte ]  = 00
    RadiusCounter [1 byte ]  = 05
    CRC           [1 byte ]  = 8A
RX: Confirm-ID.report 02 9D 03 01 01 9E
RX: APSME-ProcessSecureFrame.Confirm 02 98 CC 01 06 53
RX: ZDP-APSME-ProcessSecureFrame.Confirm 02 A0 CC 01 06 6B
RX: APSDE-DATA.Confirm 02 9D 00 11 02 00 00 00 00 00 00 00
00 08 08 00 00 00 00 00 01 8F
    Sync        [1 byte ]  = 02
    OpGroup     [1 byte ]  = 9D
    OpCode      [1 byte ]  = 00
    Length      [1 byte ]  = 11
    DstAddrMode [1 byte ]  = 02
    DstAddress  [8 bytes]  = 00 00 00 00 00 00 00 00
    DstEndpoint [1 byte ]  = 08
    SrcEndpoint [1 byte ]  = 08
    Status      [1 byte ]  = 00 (gSuccess)
    TxTime      [4 bytes]  = 00 00 00 00
    ConfirmID   [1 byte ]  = 01
```

```
Enter Command  APSDE-DATA.Request 9C 00 13 02 00 00 00 00 00 00 00 00 08 01 04 00 06 08 01 00 00 05
```

Figure 8-15. APSDE-Data.Request Command

On the coordinator side, an APSDE-DATA.Indication is received as shown in Figure 8-16.



```
ZTC-ModeSelect.Request
ZTC-WriteExtAddr.Request
ZTC-StartNwkEx.Request
ZTC-RestartNwk.Request
ZTC-StopNwkEx.Request
ZDP-IEEE_addr_req.Request
ZDP-NWK_addr_req.Request
ZTC-CPU_Reset.Request
ZTC-SetChannel.Request
ZTC-ReadExtAddr.Request
```

```
DeviceType:            DeviceasZC
Startupset:            Useworkingset
Startupcontrolmode:    Association
```

```
RX: APSDE-DATA.Indication 02 9D 01 1D 02 00 00 08 02 E1 92 08 04 01 06
00 01 00 00 00 01 AC 00 00 00 00 00 00 00 00 00 00 00 5D
    Sync           [1 byte ]  = 02
    OpGroup        [1 byte ]  = 9D
    OpCode         [1 byte ]  = 01
    Length         [1 byte ]  = 1D
    DestAddrMode   [1 byte ]  = 02 (Value16bitAddrandDstEndpoint)
    DstAddress     [2 bytes]  = 00 00
    DstEndpoint    [1 byte ]  = 08
    SrcAddrMode    [1 byte ]  = 02
    SrcAddress     [2 bytes]  = E1 92
    SrcEndpoint    [1 byte ]  = 08
    ProfileId      [2 bytes]  = 04 01
    ClusterId      [2 bytes]  = 06 00
    asduLength     [1 byte ]  = 01
    asdu           [1 byte ]  = 00
    Status         [1 byte ]  = 00 (Success)
    WasBroadcast   [1 byte ]  = 00 (FALSE)
    SecurityStatus [1 byte ]  = 01 (gSecured_Nwk_Key)
    LinkQuality    [1 byte ]  = AC
    RxTime         [4 bytes]  = 00 00 00 00
    iMsgType       [1 byte ]  = 00
    pNext          [2 bytes]  = 00 00
    iDataSize      [1 byte ]  = 00
    pData          [2 bytes]  = 00 00
    iBufferNumber  [1 byte ]  = 00
    CRC            [1 byte ]  = 5D
```

```
Enter Command  ZTC-StartNwkEx.Request A3 E7 03 C0 10 00
```

Figure 8-16. APSDE-Data.Indication on Coordinator

Chapter 9
Creating a ZigBee BlackBox Host Application

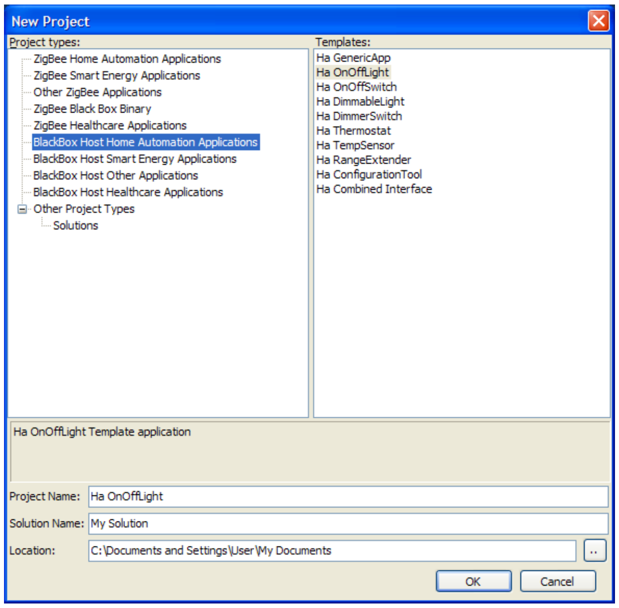
This chapter shows how to create the ZigBee BlackBox Host Application using the Freescale BeeKit Wireless Connectivity Toolkit.

9.1 Introduction

ZigBee BlackBox Host applications are alternate versions of the standard Freescale BeeStack demonstration applications but the ZigBee BlackBox Host applications do not contain ZigBee APS, ZigBee NWK or IEEE 802.15.4 MAC layers. Instead, they delegate these tasks to a ZigBee BlackBox application created as shown in Chapter 8, “Creating a BeeStack BlackBox Binary”. This approach allows for a two MCU based solution where application functionality is performed on the Host MCU and ZigBee functionality is performed on the ZigBee BlackBox MCU. The two MCUs must communicate on the UART interface as shown in the *BeeStack Application Development Guide for ZigBee 2007* (BSADGZB2007). The host applications contain wrapper functions that encapsulate network layer calls and indications to and from the UART ZTC packet data sent to the ZigBee BlackBox.

9.2 Creating the Application

To create a ZigBee BlackBox Host application using BeeKit, choose a starting project template from the BlackBox Host project type sections as shown in Figure 9-1. Proceed with configuration as shown in Chapter 3, “Using BeeKit to Create BeeStack Applications”.


New Project
Project types:
ZigBee Home Automation Applications
ZigBee Smart Energy Applications
Other ZigBee Applications
ZigBee Black Box Binary
ZigBee Healthcare Applications
BlackBox Host Home Automation Applications
BlackBox Host Smart Energy Applications
BlackBox Host Other Applications
BlackBox Host Healthcare Applications
Other Project Types
Solutions
Templates:
Ha GenericApp
Ha OnOffLight
Ha OnOffSwitch
Ha DimmableLight
Ha DimmerSwitch
Ha Thermostat
Ha TempSensor
Ha RangeExtender
Ha ConfigurationTool
Ha Combined Interface
OK
Cancel

Ha OnOffLight Template application

Project Name: Ha OnOffLight

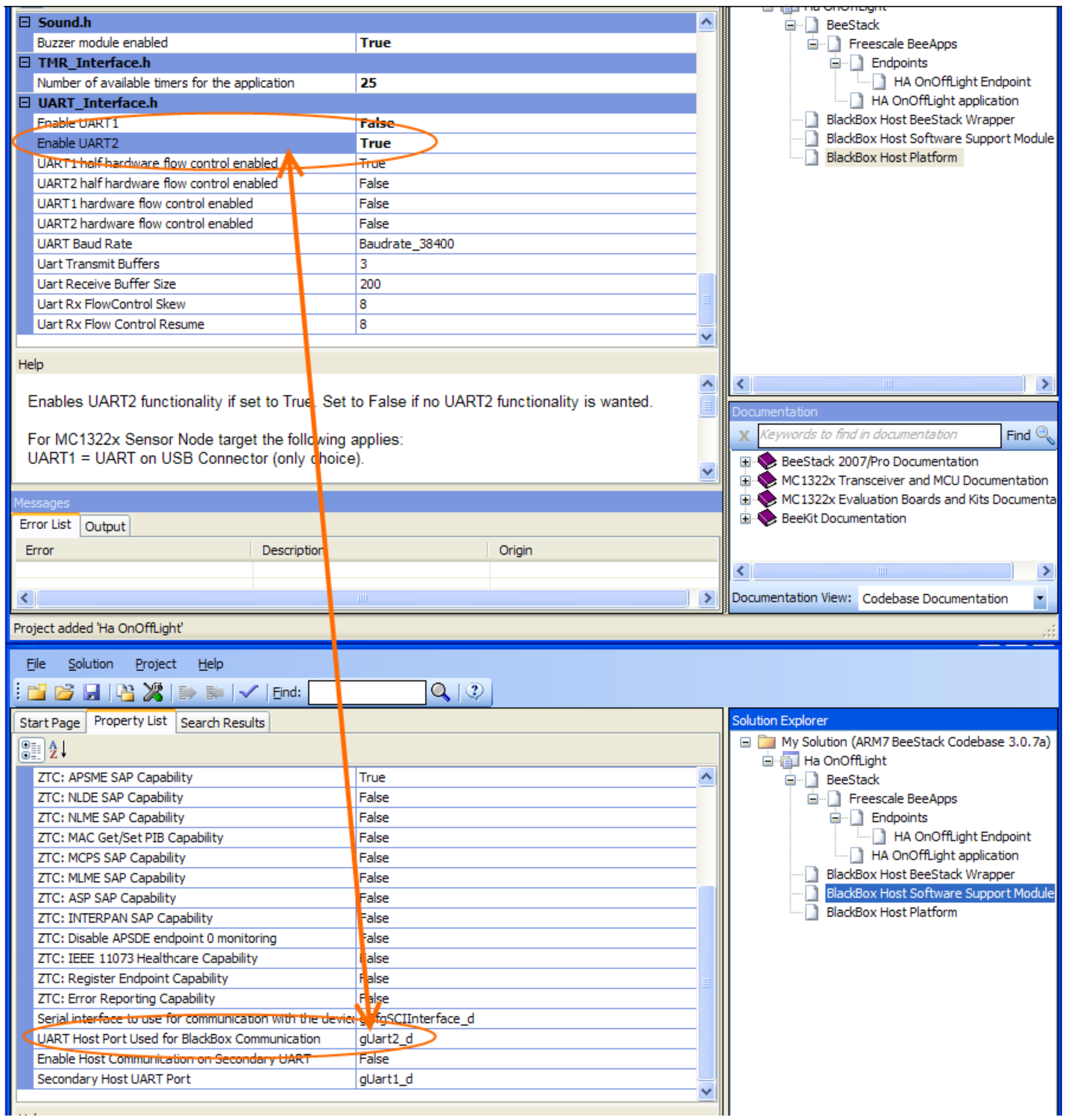
Solution Name: My Solution

Location: C:\Documents and Settings\User\My Documents

Figure 9-1. Creating a BlackBox Host Application

9.3 UART and ZTC Configuration

ZigBee BlackBox Host application users must configure the UART_Interface section of the ZigBee BlackBox Host Platform and the ZTCInterface section of the ZigBee BlackBox Host Software Support Modules sections of the BeeKit projects to correspond to the number of UART ports used to connect to the BlackBox. The UART port number must be enabled in the UART_Interface and selected as the UART Host Port Used for ZigBee BlackBox Communication in the ZTCInterface section as shown in Figure 9-2.



Sound.h	
Buzzer module enabled	True
TMR_Interface.h	
Number of available timers for the application	25
UART_Interface.h	
Enable UART1	False
Enable UART2	True
UART1 half hardware flow control enabled	True
UART2 half hardware flow control enabled	False
UART1 hardware flow control enabled	False
UART2 hardware flow control enabled	False
UART Baud Rate	Baudrate_38400
Uart Transmit Buffers	3
Uart Receive Buffer Size	200
Uart Rx FlowControl Skew	8
Uart Rx Flow Control Resume	8

Help

Enables UART2 functionality if set to True. Set to False if no UART2 functionality is wanted.

For MC1322x Sensor Node target the following applies:
UART1 = UART on USB Connector (only choice).

Messages

Error List | Output

Error	Description	Origin

Project added 'Ha OnOffLight'

BeeStack
Freescale BeeApps
Endpoints
HA OnOffLight Endpoint
HA OnOffLight application
BlackBox Host BeeStack Wrapper
BlackBox Host Software Support Module
BlackBox Host Platform
Documentation
Keywords to find in documentation
Find
BeeStack 2007/Pro Documentation
MC1322x Transceiver and MCU Documentation
MC1322x Evaluation Boards and Kits Documenta
BeeKit Documentation
Documentation View: Codebase Documentation
File
Solution
Project
Help
Find:
Start Page
Property List
Search Results
Solution Explorer
My Solution (ARM7 BeeStack Codebase 3.0.7a)
Ha OnOffLight
BeeStack
Freescale BeeApps
Endpoints
HA OnOffLight Endpoint
HA OnOffLight application
BlackBox Host BeeStack Wrapper
BlackBox Host Software Support Module
BlackBox Host Platform

ZTC: APSME SAP Capability	True
ZTC: NLDE SAP Capability	False
ZTC: NLME SAP Capability	False
ZTC: MAC Get/Set PIB Capability	False
ZTC: MCPS SAP Capability	False
ZTC: MLME SAP Capability	False
ZTC: ASP SAP Capability	False
ZTC: INTERPAN SAP Capability	False
ZTC: Disable APSDE endpoint 0 monitoring	False
ZTC: IEEE 11073 Healthcare Capability	False
ZTC: Register Endpoint Capability	False
ZTC: Error Reporting Capability	False
Serial interface to use for communication with the device	gFgSCIInterface_d
UART Host Port Used for BlackBox Communication	gUart2_d
Enable Host Communication on Secondary UART	False
Secondary Host UART Port	gUart1_d

Figure 9-2. UART and ZTC Configuration on Host for Communication with ZigBee BlackBox

NOTE

Even though the ZigBee BlackBox Host demonstration applications use a two MCU approach, they essentially have the same functionality as single MCU applications shown in Chapters 3 - 8 of this guide.

Chapter 10
Using the Over the Air (OTA) Upgrade Cluster

BeeStack Codebases 3.0.10 and later include ZigBee Over the Air (OTA) Image Upgrade Cluster sample functionality. This chapter shows how to use the OTA programming cluster to update a firmware image over the air on an OTA cluster client node from a ZTC-based OTA server node. To use the server node, the OTA BeeStack Update View in Test Tool 11.3.0 or later must be used.

10.1 Introduction

The ZCL OTA Programming Cluster allows the firmware image of a remote node to be updated from a server node. BeeStack supports the OTA Programming Cluster with a low level platform component and external storage bootloader for both MC1320x-QE128 and MC1322x development boards and a high level OTA Cluster implementation for the over the air data exchange. For more details about the OTA cluster see ZigBee Alliance document 095264.

10.1.1 OTA Support in the Platform Component

10.1.1.1 External Storage Support

To store the firmware image files received over the air, the OTA clients should have an external non-volatile memory storage device installed. For MC1320x-QE128 development boards, an AT24C1024BW EEPROM 1MBit chip is provided. MC1322x development boards do not have an external storage chip attached but an extension board can be attached on the general I/O pins. The OTA sample implementation uses a SST25WF010 1MBit external serial FLASH controlled via the SPI interface. The external FLASH module is similar to the one embedded in the MC1322x MCU.

10.1.1.2 External Storage Bootloader

At the platform level, BeeStack provides reference implementations for a bootloader component which can be included by setting the “Enable external storage bootloader” option to True in the PLM component of the BeeKit project. This triggers the following effects:

- On both S08QE128 and ARM7 (MC1322x) platforms, a sector in the internal FLASH memory of the MCU is allocated for the bootloader code at a fixed location
  - On MC1322x, the first 4K sector of the FLASH are used for the bootloader code
  -)On QE128, the bootloader is placed at memory location 0xFC00
- When the node is reset, the bootloader takes control and checks flags that indicate whether the external FLASH contains a new firmware image that is available for update in the external storage
- If a new image is available, the bootloader updates the content of the MCU internal FLASH from external storage, invalidates the new image flags and then reboots the MCU for it to run the new

firmware; when the bootloader runs again at reboot, it jumps to the application as the new image flags are no longer set

- The bootloader sector allocated in internal FLASH is not updated during the firmware upgrade; to ensure update images are booting correctly, they must also be compiled with the “Enable external storage bootloader” FLASH configuration active so that the memory maps match

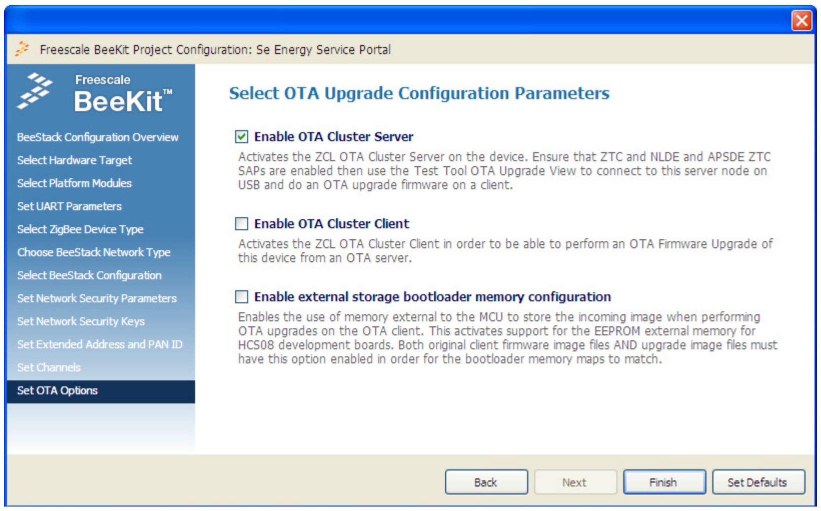
10.1.1.3 OTA Support Component

The BeeStack PLM component includes an application interface in the OtaSupport header and module files for using the external storage modules while receiving the firmware image blocks over the air. The application can start a new image to be written to the external storage, push one or more blocks to a storage location, or commit an image to indicate that the bootloader should use it for updating the internal FLASH at the next reboot. The component also allows for a Freescale specific sub element to an OTA cluster image file which allows the user to indicate a bitmap containing the sectors that should or should not be erased and reprogrammed in the internal FLASH. The bitmap for MC1322x must be 4 bytes long, with each bit representing a 4K FLASH sector, whereas the bitmap for QE128 must be 32 byte long, each bit representing a 512B FLASH sector. If a bit is set to 1, this allows for the erasure of the corresponding FLASH sector. If a bit is set to 0, the FLASH sector is protected.

Running an OTA Cluster Scenario

To create OTA enabled applications and perform an OTA image upgrade from a ZTC server to a client to switch between two applications, perform the following steps:

1. In BeeKit using Codebase version 3.0.10 or newer add three new Smart Energy projects. This example uses the HCS08 CodeBase and a SE EnergyServicePortal Coordinator, a SE InPremiseDisplay Router and a SE PCT all running on MC1320xQE128. In the last page of the configuration wizard choose:
   — Enable OTA Server Cluster on the SE Energy Service Portal project as shown in Figure 10-1.


Freescale BeeKit Project Configuration: Se Energy Service Portal

Select OTA Upgrade Configuration Parameters

[✓] Enable OTA Cluster Server
Activates the ZCL OTA Cluster Server on the device. Ensure that ZTC and NLDE and APSDE ZTC SAPs are enabled then use the Test Tool OTA Upgrade View to connect to this server node on USB and do an OTA upgrade firmware on a client.

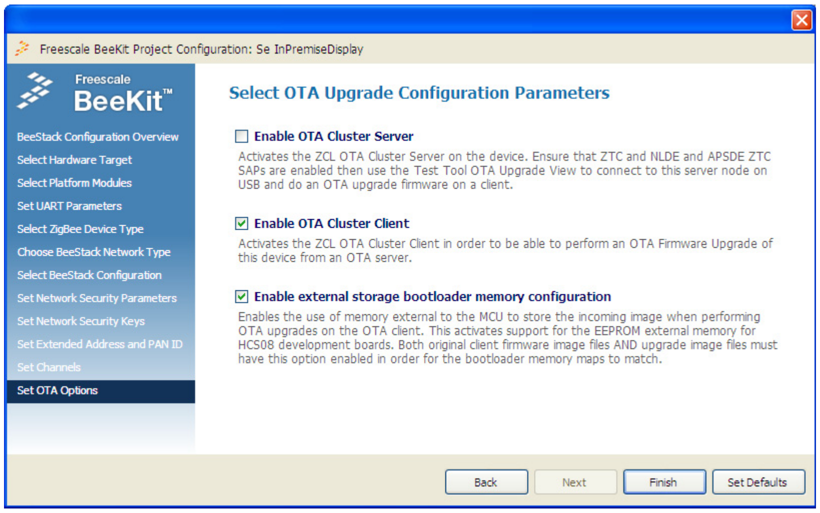
[ ] Enable OTA Cluster Client
Activates the ZCL OTA Cluster Client in order to be able to perform an OTA Firmware Upgrade of this device from an OTA server.

[ ] Enable external storage bootloader memory configuration
Enables the use of memory external to the MCU to store the incoming image when performing OTA upgrades on the OTA client. This activates support for the EEPROM external memory for HCS08 development boards. Both original client firmware image files AND upgrade image files must have this option enabled in order for the bootloader memory maps to match.

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Network Security Parameters
Set Network Security Keys
Set Extended Address and PAN ID
Set Channels
Set OTA Options
Back
Next
Finish
Set Defaults

Figure 10-1. Enabling OTA Cluster Server on SE Energy Service Portal Coordinator

— Enable OTA Server Client and Enable external bootloader memory configuration options on the SE InPremiseDisplay and SE PCT as shown in Figure 10-2.


Freescale BeeKit Project Configuration: Se InPremiseDisplay

Freescale
BeeKit™
BeeStack Configuration Overview
Select Hardware Target
Select Platform Modules
Set UART Parameters
Select ZigBee Device Type
Choose BeeStack Network Type
Select BeeStack Configuration
Set Network Security Parameters
Set Network Security Keys
Set Extended Address and PAN ID
Set Channels
Set OTA Options
Back
Next
Finish
Set Defaults

Select OTA Upgrade Configuration Parameters

☐ Enable OTA Cluster Server
Activates the ZCL OTA Cluster Server on the device. Ensure that ZTC and NLDE and APSDE ZTC SAPs are enabled then use the Test Tool OTA Upgrade View to connect to this server node on USB and do an OTA upgrade firmware on a client.

✓ Enable OTA Cluster Client
Activates the ZCL OTA Cluster Client in order to be able to perform an OTA Firmware Upgrade of this device from an OTA server.

✓ Enable external storage bootloader memory configuration
Enables the use of memory external to the MCU to store the incoming image when performing OTA upgrades on the OTA client. This activates support for the EEPROM external memory for HCS08 development boards. Both original client firmware image files AND upgrade image files must have this option enabled in order for the bootloader memory maps to match.

Figure 10-2. Enabling OTA Cluster Client on SE InPremiseDisplay and PCT

2. Export to CodeWarrior and load the SE Energy Service Portal and InPremise Display projects to the evaluation boards.
3. Start Test Tool and choose View -> OTA Update in the menu bar in Test Tool 11.3.0 or OTA Update -> OTA BeeStack in Test Tool 12.0.0 to launch the OTA Update View as shown in Figure 10-3.

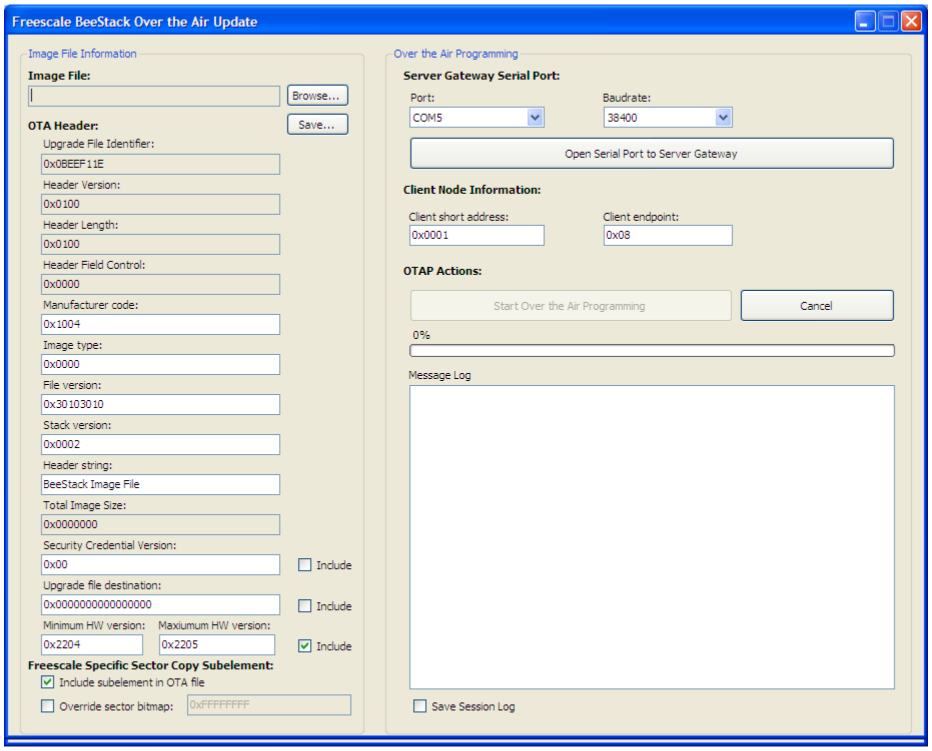

Freescale BeeStack Over the Air Update

Image File Information

Image File:

OTA Header:
Upgrade File Identifier: 0x0BEEF11E
Header Version: 0x0100
Header Length: 0x0100
Header Field Control: 0x0000
Manufacturer code: 0x1004
Image type: 0x0000
File version: 0x30103010
Stack version: 0x0002
Header string: BeeStack Image File
Total Image Size: 0x0000000
Security Credential Version: 0x00 ☐ Include
Upgrade file destination: 0x0000000000000000 ☐ Include
Minimum HW version: 0x2204 Maxiumum HW version: 0x2205 ✓ Include
Freescale Specific Sector Copy Subelement:
✓ Include subelement in OTA file
☐ Override sector bitmap: 0xFFFFFFFF

Over the Air Programming

Server Gateway Serial Port:
Port: COM5 Baudrate: 38400

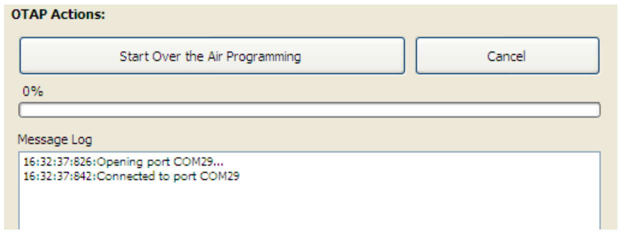
Client Node Information:
Client short address: 0x0001 Client endpoint: 0x08

OTAP Actions:
0%
Message Log
☐ Save Session Log

Browse...
Save...
Open Serial Port to Server Gateway
Start Over the Air Programming
Cancel

Figure 10-3. BeeStack OTA Update View

4. Connect the Energy Service Portal board to the PC and load drivers if necessary.
5. In the OTA Update View in the Server Gateway serial port, select the Port of the connected Energy Service Portal board.
6. Click “Open Serial Port to Server Gateway” to establish the serial USB connection to the Energy Service Portal board. The status of the connection is displayed in the Message Log area as shown in Figure 10-4.



```
16:32:37:826:Opening port COM29...
16:32:37:842:Connected to port COM29
```

Figure 10-4. Connection to board established

7. Form a ZigBee network on the Energy Service Portal node and join the In Premise Display node to the network.
8. In the OTA Update View in the Image File Information area click Browse... and navigate to the Se PCT project “bin” folder.
9. Select HCS08 S19 Files in the “Files of type” drop down in the Open window as shown in Figure 10-5.

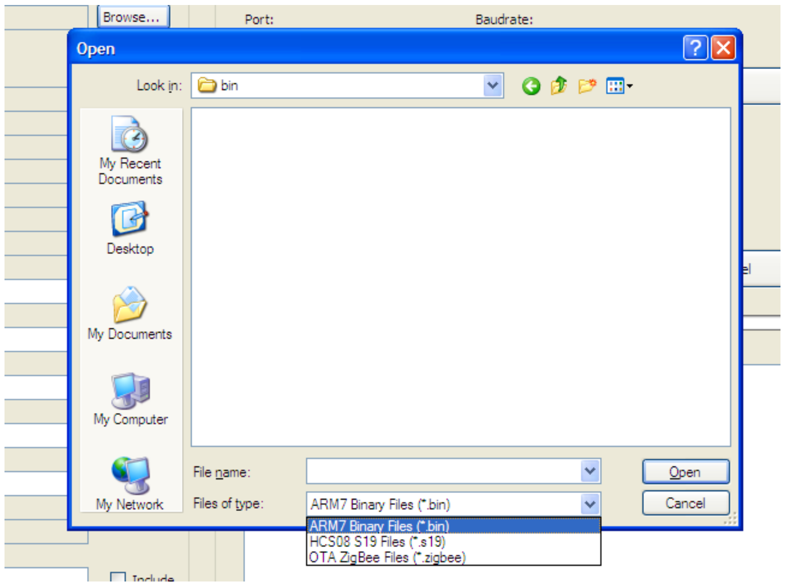


Figure 10-5. Selecting S19 File Type

10. Select the PCT S19 file compiled previously. The OTA headers in the OTA Update view for the file is filled in automatically.
11. In the OTA Update View in the Client short address text box enter the address of the In Premise Display board.

12. Click Start Over the Air Programming. This uses the SE Energy Service Portal OTA ZTC server to start pushing the PCT image to the current In Premise Display board and displaying the progress as shown in Figure 10-6.

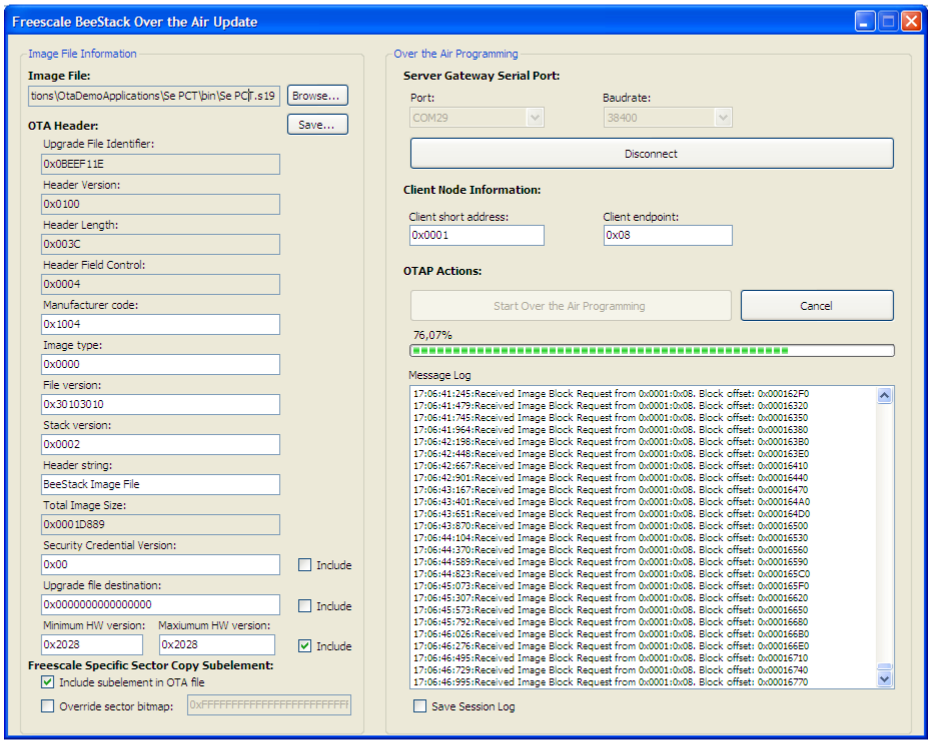


Image File: tions\OtaDemoApplications\Se PCT\bin\Se PCT.s19

OTA Header:
Upgrade File Identifier: 0x0BEEF11E
Header Version: 0x0100
Header Length: 0x003C
Header Field Control: 0x0004
Manufacturer code: 0x1004
Image type: 0x0000
File version: 0x30103010
Stack version: 0x0002
Header string: BeeStack Image File
Total Image Size: 0x0001D889
Security Credential Version: 0x00 ☐ Include
Upgrade file destination: 0x0000000000000000 ☐ Include
Minimum HW version: 0x2028 Maxiumum HW version: 0x2028 ✓ Include

Freescale Specific Sector Copy Subelement:
✓ Include subelement in OTA file
☐ Override sector bitmap: 0xFFFFFFFFFFFFFFFFFFFFFFFF

Server Gateway Serial Port:
Port: COM29 Baudrate: 38400

Client Node Information:
Client short address: 0x0001 Client endpoint: 0x08

OTAP Actions:
76,07%

Message Log

```
17:06:41:245:Received Image Block Request from 0x0001:0x08. Block offset: 0x000162F0
17:06:41:479:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016320
17:06:41:745:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016350
17:06:41:964:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016380
17:06:42:198:Received Image Block Request from 0x0001:0x08. Block offset: 0x000163B0
17:06:42:448:Received Image Block Request from 0x0001:0x08. Block offset: 0x000163E0
17:06:42:667:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016410
17:06:42:901:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016440
17:06:43:167:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016470
17:06:43:401:Received Image Block Request from 0x0001:0x08. Block offset: 0x000164A0
17:06:43:651:Received Image Block Request from 0x0001:0x08. Block offset: 0x000164D0
17:06:43:870:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016500
17:06:44:104:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016530
17:06:44:370:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016560
17:06:44:589:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016590
17:06:44:823:Received Image Block Request from 0x0001:0x08. Block offset: 0x000165C0
17:06:45:073:Received Image Block Request from 0x0001:0x08. Block offset: 0x000165F0
17:06:45:307:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016620
17:06:45:573:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016650
17:06:45:792:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016680
17:06:46:026:Received Image Block Request from 0x0001:0x08. Block offset: 0x000166B0
17:06:46:276:Received Image Block Request from 0x0001:0x08. Block offset: 0x000166E0
17:06:46:495:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016710
17:06:46:729:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016740
17:06:46:995:Received Image Block Request from 0x0001:0x08. Block offset: 0x00016770
```

☐ Save Session Log

Figure 10-6. Checking OTA Update Progress

13. Wait a few minutes until the process reaches 100%. The board resets itself when programming completes.
14. Wait a few minutes until the bootloader on the board reprograms the internal FLASH of the In PremiseDisplay node. The node reboots as a PCT node.