Freescale BeeStack Consumer™ Network Demonstration Application

User's Guide

Document Number: BSCONNAUG
Rev. 1.2
2/2012

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

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

Contents

Chapter 5
BeeStack Consumer Sample Applications

Chapter 6
BeeStack Consumer/SynkroRF Hybrid Sample Applications

About This Book

The *BeeStack Consumer Network Application User's Guide* explains how to install and run the BeeStack Consumer Network and BeeStack Consumer/SynkroRF Hybrid applications included in Freescale's BeeKit Wireless Connectivity Toolkit for RF4CE (BeeStack Consumer) applications.

Audience

This guide is intended for use with the BeeStack Consumer Network example applications in BeeKit. Anyone needing to demonstrate the applications or to become familiar with their behavior should read this guide. BeeStack Consumer application developers should read this guide along with all other BeeStack Consumer documentation to understand what the applications are and what they do.

Organization

This document is organized into the following chapters.

Chapter 1 Introduction – introduces the Freescale implementation of RF4CE (BeeStack Consumer) network.

Chapter 2 Creating Applications with BeeKit – describes how to build a BeeStack Consumer project project file from BeeKit.

Chapter 3 Starting and Running a Simple RF4CE Network - describes the steps to start and run a simple RF4CE network using the Controller and Target Node Applications.

Chapter 4 Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software - details how to use the Test Tool software with the ZTC nodes to start a RF4CE network.

Chapter 5 BeeStack Consumer Sample Applications - describes the BeeStack Consumer sample applications provided with the BeeStack Consumer Codebase.

Chapter 6 BeeStack Consumer/SynkroRF Hybrid Sample Applications - describes the BeeStack Consumer/SynkroRF Hybrid sample applications provided with the BeeStack Consumer Codebase.

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

API	Application Programming Interface
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
Pairing	Associating two nodes in a network for specific functions (e.g., a controller and target)
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 RF4CE 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
Stack	RF4CE 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 is 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
802.15.4	An IEEE standard radio specification that underlies the RF4CE specification
BeeStack Consumer	Freescale’s implementation of the ZigBee RF4CE Standard
CE	Consumer Electronics
LQI	Link Quality Indication
NIB	Network Information Base
NLDE	Network Layer Data Entity
NLME	Network Layer Management Entity
NVM	Non volatile memory
RC	Remote Control
RF	Radio Frequency
SAP	Service Access Point
SynkroRF	An entertainment control network
OTAP	Over the air programming

References

The following documents were referenced to build this document.

1. Freescale *BeeStack Consumer Reference Manual*, Document BSCONRM, March 2011.
2. Freescale *BeeStack Consumer User’s Guide*, Document BSCONUG, November 2010
3. Freescale *ZigBee Remote Control (ZRC) Application Profile Reference Manual*, Document ZRCAPRM, March 20011.
4. Freescale *ZigBee Remote Control (ZRC) Application Profile User’s Guide*, Document ZRCAPUG, February 20011
5. Freescale *BeeStack Consumer Private Profile Reference Manual*, Document BSCONPPRM, September 2009.
6. Freescale *BeeStack Consumer Private Profile User’s Guide*, Document BSCONPPRM, July 2009
7. Freescale *BeeStack Consumer Blackbox Interface User’s Guide*, Document BSCONBBIUG
8. Document 094945r00, *ZigBee RF4CE Specification version 1.01*, ZigBee Alliance, January 2010
9. Document 094946r00, *ZigBee RF4CE: CERC Profile Specification*, ZigBee Alliance, March 2009
10. Document 095275r10, *RF4CE ZRC Profile Specification Errata*, ZigBee Alliance, October 2010

11. Document 095264r00, *ZigBee Over-the-Air Upgrading Cluster*, ZigBee Alliance, January 2010
12. The data sheets for the MC13193, MC13203, MC1321x, MC1323x radios
13. Freescale *MC9S08GB/GT Data Sheet*, Document MC9S08GB60, December 2004

Revision History

The following table summarizes revisions to this guide since the previous release (Rev. 1.1).

Revision History

Location	Revisions
Dev Team, Feb 2012	Minor changes for March software release.

Chapter 1
Introduction

The Freescale BeeKit Wireless Connectivity Toolkit includes a set of example applications for the RF4CE (BeeStack Consumer) using the ZigBee Remote Control (ZRC) and Freescale Private profiles, supplemented by Generic Node Framework and Repeater Node (RF4CE and SynkroRF) example applications. This guide also describes the BeeStack Consumer SynkroRF Hybrid applications. This user's guide describes how to:

- Configure the applications in BeeKit for any of the Freescale development boards that BeeKit supports
- Export the configured applications from BeeKit
- Import the configured applications into 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/MC13226 based development or version 5.40 or 5.50 for codebases supporting both ARM7 MC13226 and MC13224/MC13226. 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 RF4CE 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 RF4CE/802.15.4 specifications.

1.1 What This Document Describes

This document explains how to:

- Get the example RF4CE (BeeStack Consumer) 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 Consumer to a platform that BeeKit does not already support
- Learn about RF4CE (BeeStack Consumer). For a tutorial on RF4CE, go to www.freescale.com/zigbee and click on Online Training

Chapter 2
Creating Applications with BeeKit

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.

2.1 Creating a BeeKit Project

Perform the following tasks to create a ZigBee Remote Control network project and configure the individual devices.

NOTE

The tasks outlined in this chapter are valid for both the HCS08 and ARM7 based BeeStack Consumer Codebases. However, because the ARM7 based BeeStack Consumer Codebase also supports more features, configuration options as shown in some of the figures, may be different than those of the HCS08 based Codebase.

1. Start BeeKit.
2. If another Codebase (MAC, SMAC or ZigBee Pro) is selected, perform the following:
   a) Select File -> Select Codebase… or click the "Select Other Codebase..." link.
   b) Select the BeeStack Consumer 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 2-1.

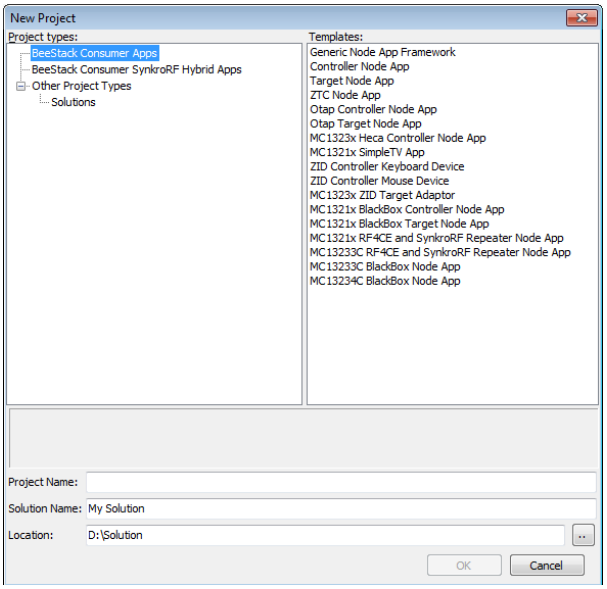

Figure 2-1. BeeKit New Project Window

4. Select the BeeStack Consumer Applications project type from the left side of the window.
5. As shown in Figure 2-2, select the Controller Node App template.

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

   Project Name: Controller Node App
   Solution Name: Solutions
   Location: BeeKitSolutions (sub directory on host PC)

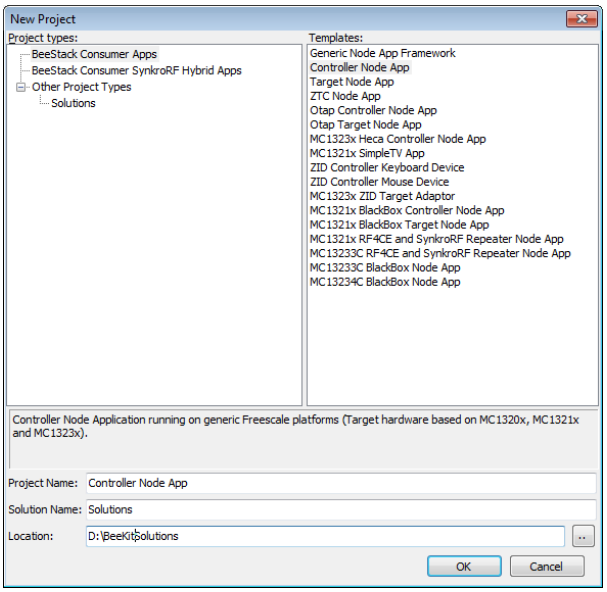

Figure 2-2. Project/Template Select

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

2.1.1 Basic Options

After the New Project window closes, the BeeKit Project Wizard Welcome window opens as shown in Figure 2-3.

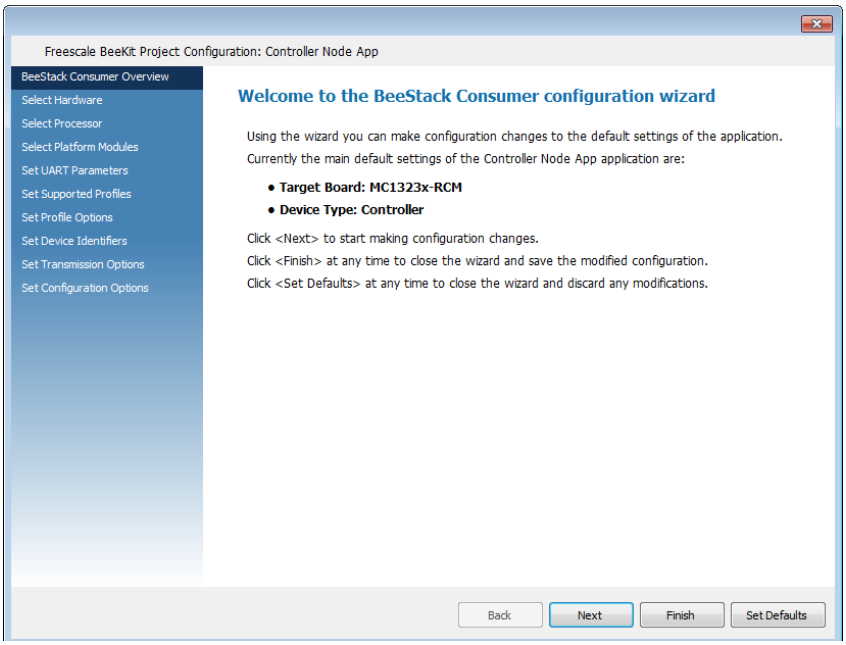

Figure 2-3. BeeKit Project Wizard Welcome Page

Review the current project settings. In this example, the settings are as follows:

- The board is a 1323x-RCM
- The device type is controller ZigBee Remote Control (ZRC)
- BeeStack Consumer Private profiles are enabled as default

The selected profiles include more options and features enabled, which are the default settings for HCS08 BeeStack Consumer 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 2.1.2, “Custom Configuration Options”.
- If users wish to discard the modifications they have made to the default configuration and close the wizard, they can click the Set Defaults button.

2.1.2 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 2-4 (ARM7 codebase version) and Figure 2-5 (HCS08 codebase version).

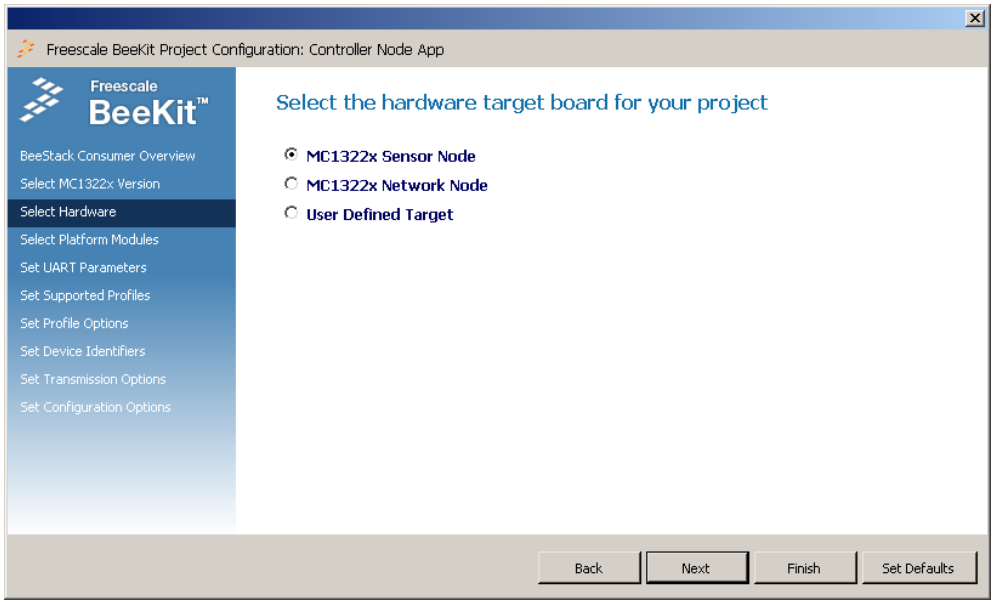

Figure 2-4. Hardware Target Page for ARM7 BeeStack Consumer Codebase

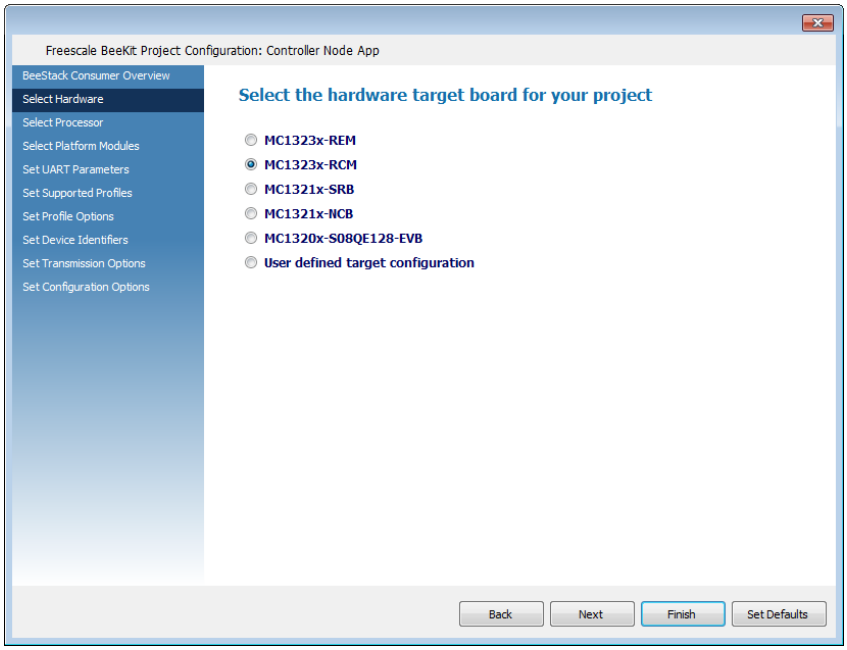

Figure 2-5. Hardware Target Page for HCS08 BeeStack Consumer Codebase

1. Change the device to a specific platform. This example uses the MC1322x Sensor Node for ARM7 codebases and the MC1323x-RCM for the HCS08 codebases.
2. Select the MC1323x-RCM radio button for hardware target and MC13233C for processor target; click on the Next button. The Platform Modules page appears as shown in Figure 2-6.

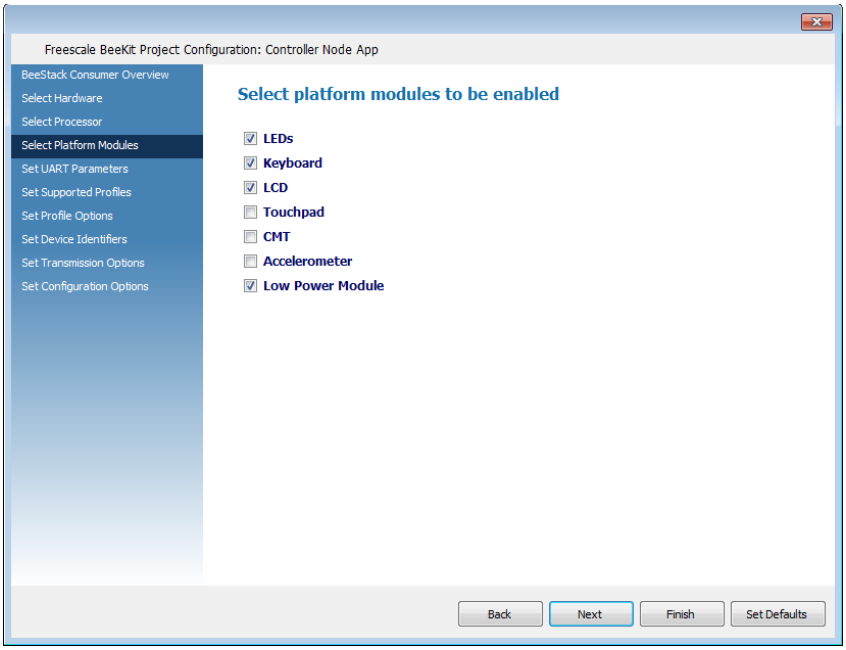

Figure 2-6. Platform Modules Page

3. Leave the default platform modules settings unchanged. LEDs and Keyboard modules on the board are enabled. Click Next to go to the UART Parameters page shown in Figure 2-7.

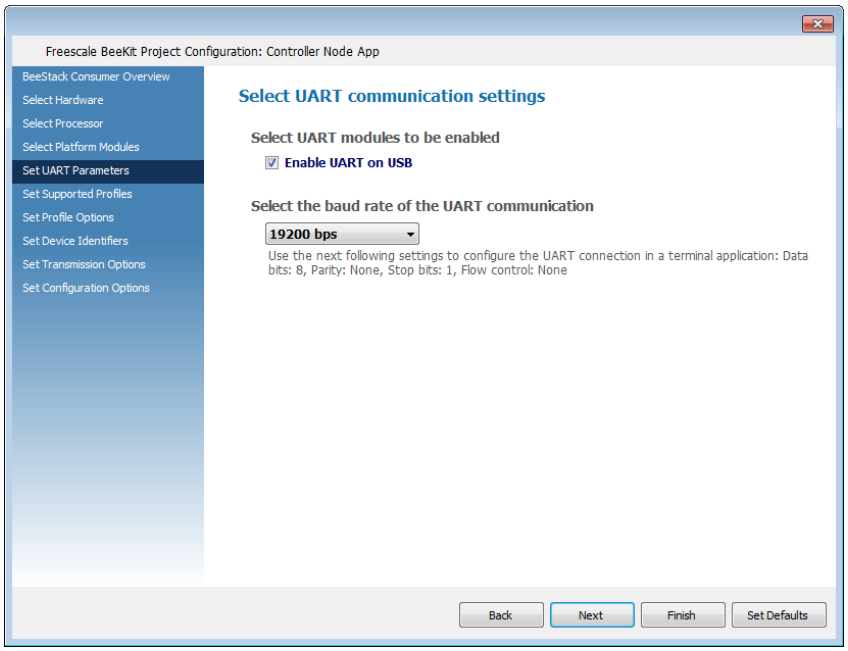

Figure 2-7. UART Parameters Page

4. Leave the default UART settings as is. The UART module are enabled on the USB port of the board. Click Next to go to the Supported RF4CE Profiles Selection wizard page shown in Figure 2-8.

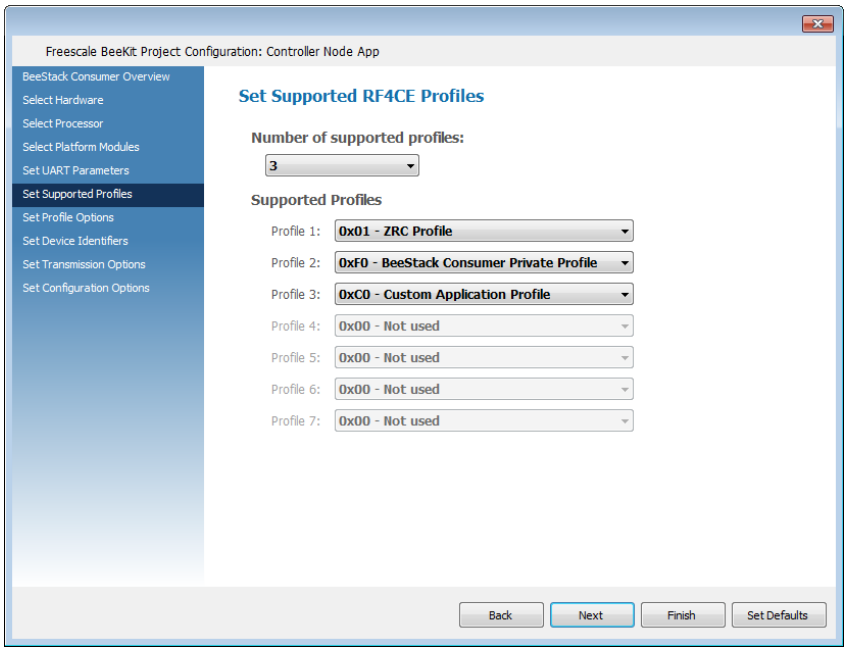

Figure 2-8. Set Supported RF4CE Profiles Page

5. Leave the default Supported RF4CE Profiles settings as it is. Click the Next button. The RF4CE Profiles Options page appears as shown in Figure 2-9 (HCS08 codebase version).

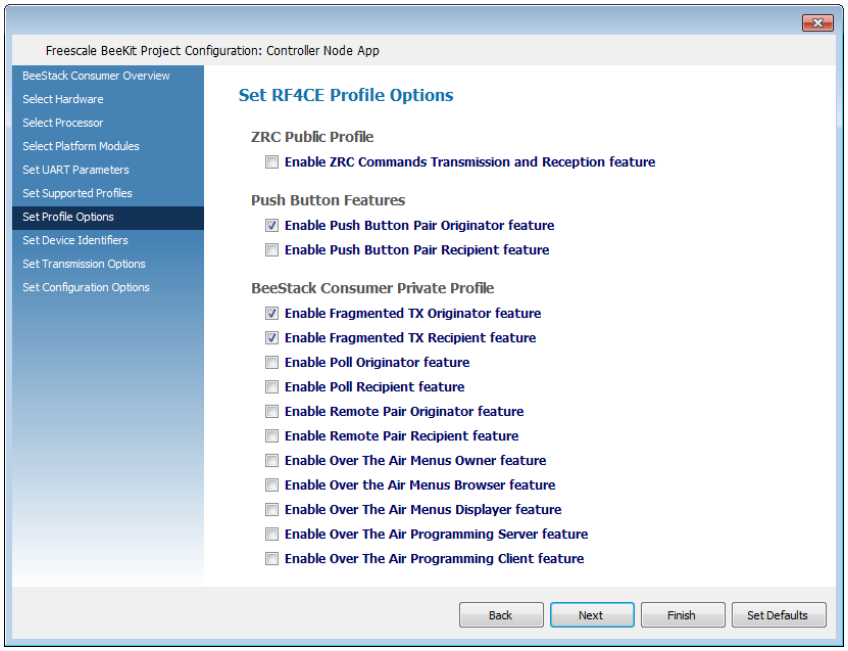

Figure 2-9. Set RF4CE Profile Options Page

6. Select the “Enable ZRC Commands Transmission and Reception feature” ZRC Profile option. Leave the other options unchanged.
7. Click Next.
8. The window as shown in Figure 2-10 displays additional RF4CE device configuration options: identification string, RF4CE device type, extended address, vendor identifier, vendor name.
9. Click Next to keep these default options and move to the next window.

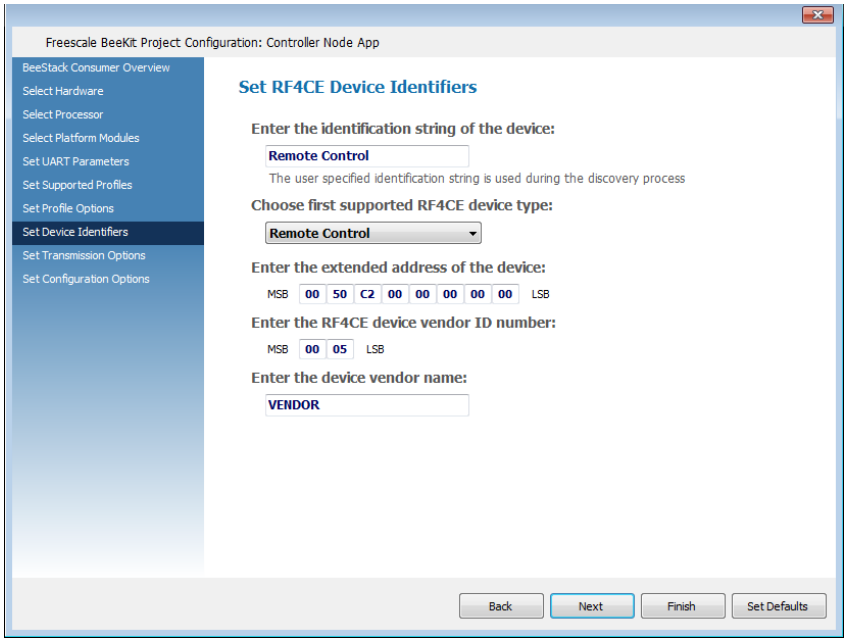

Figure 2-10. Set RF4CE Device Identifiers Page

10. The Set RF4CE Data Request Options page appears as shown in Figure 2-11.

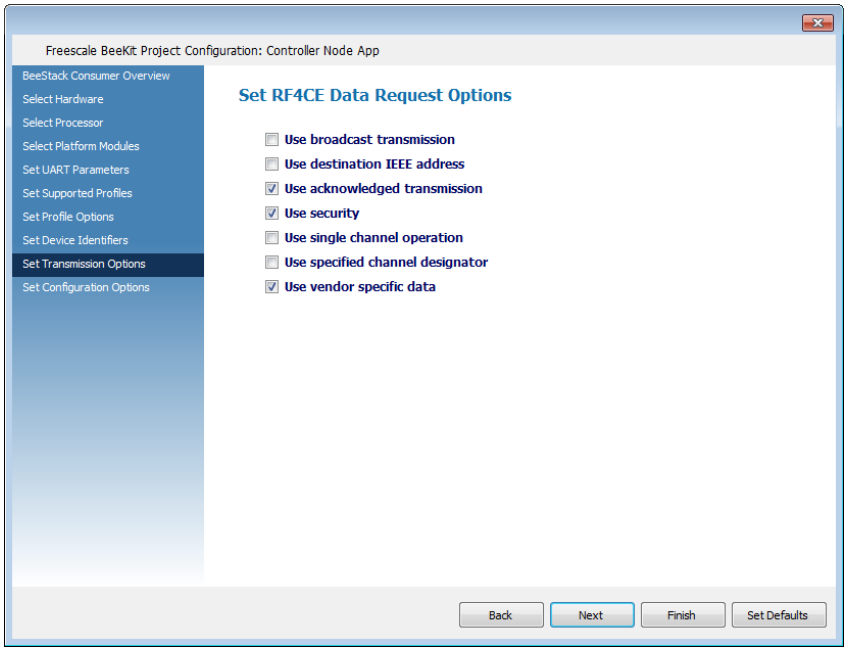

Figure 2-11. Set RF4CE Data Request Options Page

11. For this example, leave the these options unchanged.
12. Click Next. The Set Configuration Options page appears as shown in Figure 2-12.

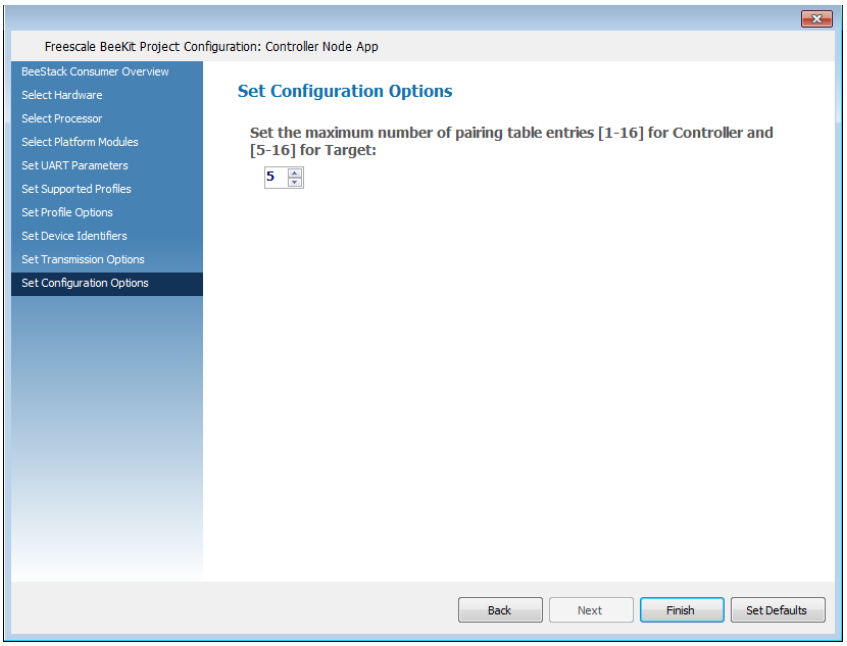

Figure 2-12. Set Configuration Options Page

13. Click on the Finish button.

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

2.2 Creating Additional Devices

To set up a RF4CE wireless network, the controller 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. (Figure 2-1)
3. Select BeeStack Consumer Applications as was done for the controller. However, this time, select the Target Node App.
4. Repeat the steps as described in Section 2.1.1, “Basic Options” and Section 2.1.2, “Custom Configuration Options”. Adjust the settings to suit the application. Be careful to notice that the current settings for the target node are different than the controller node.
5. After selecting Finish for the target node application, the BeeKit main window appears as shown in Figure 2-13.

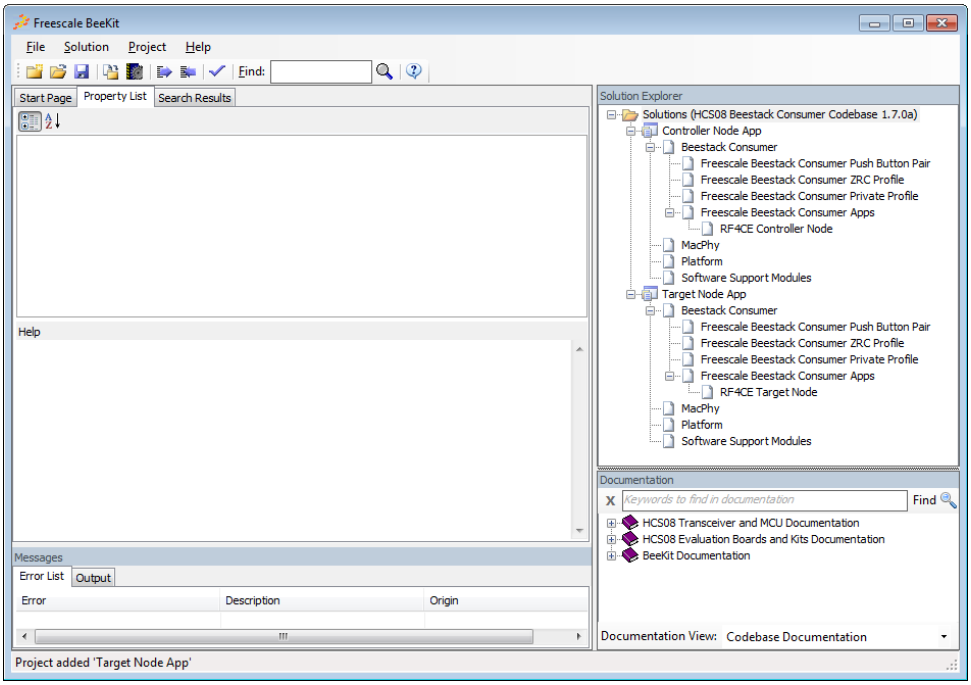

Figure 2-13. BeeKit Main Window

When clicking on a component in the Solution Explorer area, information about that item appears under the Property List tab where users can change numerous settings. For example;

- To change the development board:
  a) Select the Platform component in the Explorer window.
  b) Select the Hardware Target property in the Property List.
  c) Use the pull-down menu to select the board
- To change the controller configuration options:
  a) Select the RF4CEController component in the Explorer window.
  b) Select the 802.15.4 extended address property in the Property List.

c) Change the extended address of the device

- To change the BeeStack Consumer ZigBee Remote Control (ZRC) default configuration features:
  a) Select the Freescale BeeStack Consumer ZRC Profile component in the Explorer window.
  b) Select the ZRC commands transmission and reception feature in the Property List.
  c) Use the pull-down menu to select the configuration

2.2.1 Exporting Created BeeKit Projects

Once all the applications for 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 for compiling and debugging.

To export the saved solution:

1. From the Solution menu, select Export Solution. BeeKit starts to verify the internal consistency of the configuration. If the verification succeeds, the window that opens will display all the created devices, each with a checked box as shown in Figure 2-14.

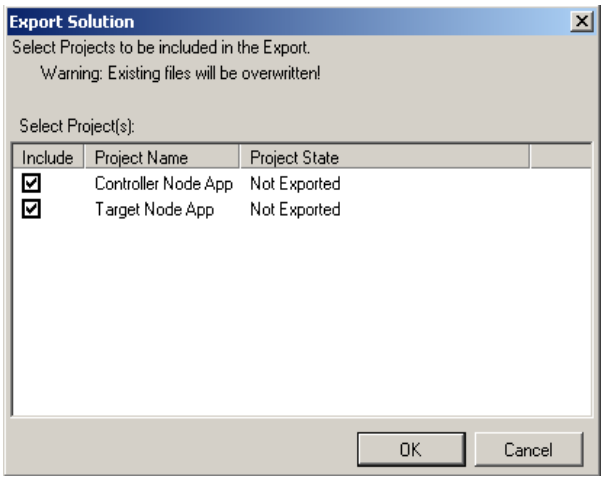

Figure 2-14. Export BeeKit Project Solution Window

2. Click on the OK button to start the export process. While BeeKit executes this, it displays the window shown in Figure 2-15, and the steps it takes scroll by in the Messages window pane.

Figure 2-15. 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.

2.3 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.

To export and import the solution into the appropriate IDE automatically perform the appropriate task depending on the IDE being used:

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

   Or

   From the BeeKit tool bar, click on Solution -> Export -> Open Solution in CodeWarrior.

   Or

   Click the IDE icon in the toolbar as shown in Figure 2-16 and export the solution in IAR EWB.

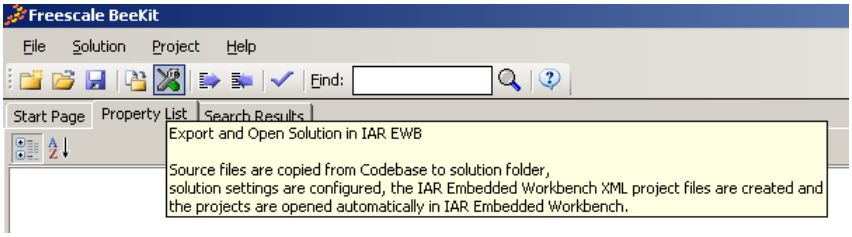

Figure 2-16. Exporting and Opening the Solution in Embedded Workbench

BeeKit exports the solution as shown in Section 2.1, "Creating a BeeKit Project", automatically opens the IDE and loads the selected projects that were exported. Figure 2-17 shows the projects of the solution previously created and imported into the IAR EWB.

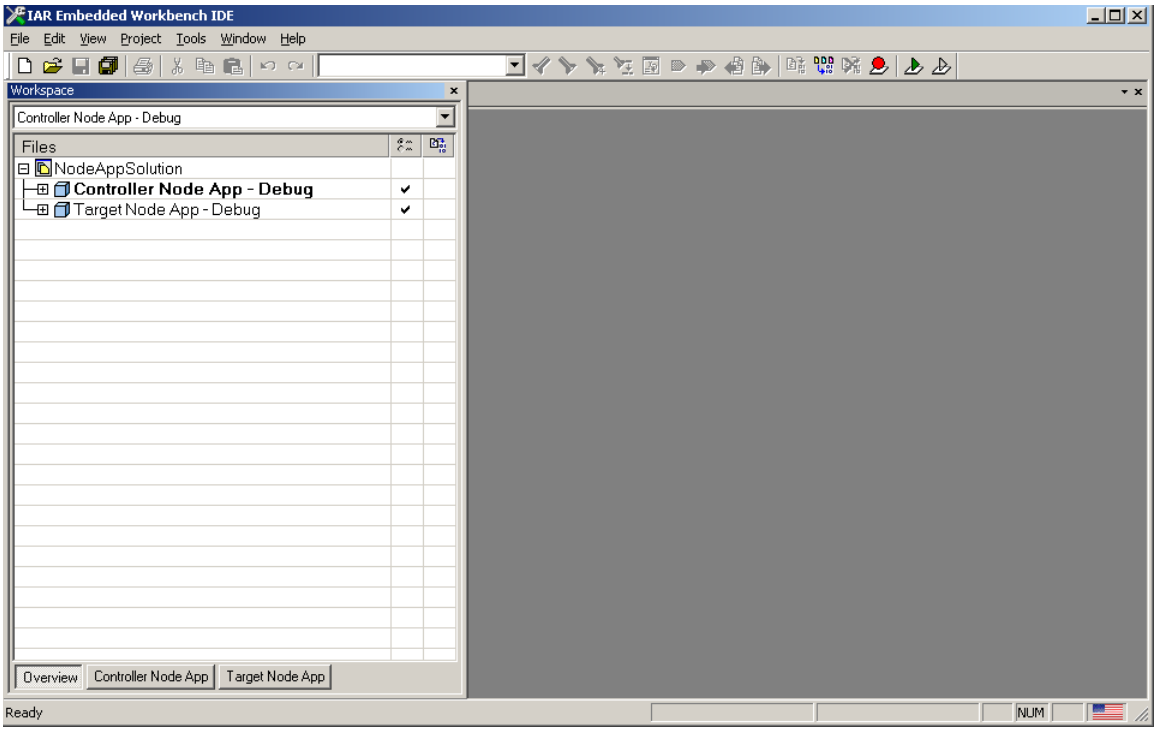

Figure 2-17. Solution Projects Imported in IAR Embedded Workbench

2.4 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 with the menu at the top.
2. Select File -> Import Project… as shown in Figure 2-18.

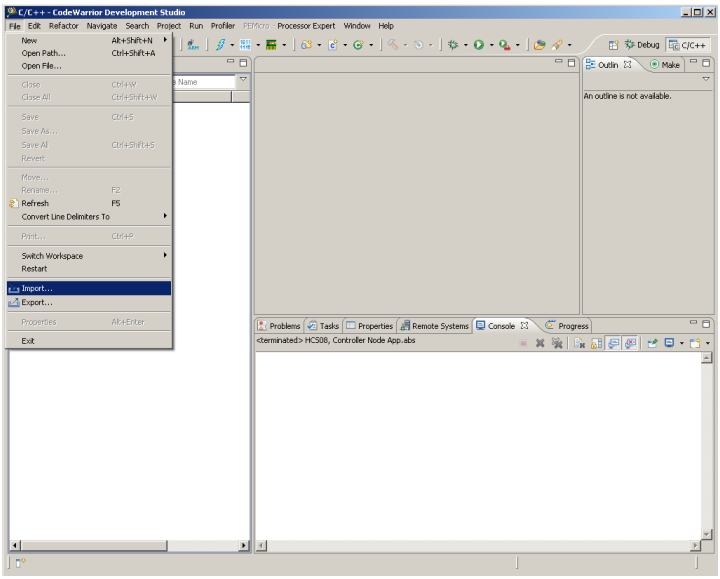

Figure 2-18. CodeWarrior Import Existing Project

3. Select Existing Projects into Workspace as shown inFigure 2-19.

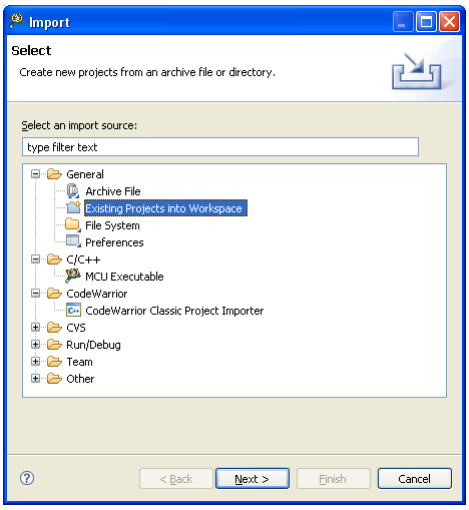

Figure 2-19. CodeWarrior Import Existing Project into Workspace

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 Controller Node App and Target Node App.Users can import both projects simultaneously by selecting the parent directory of Controller Node App and Target Node App directories.
6. In this example as shown in Figure 2-20, it is: `D:\BeeKitSolutions\Solutions`.

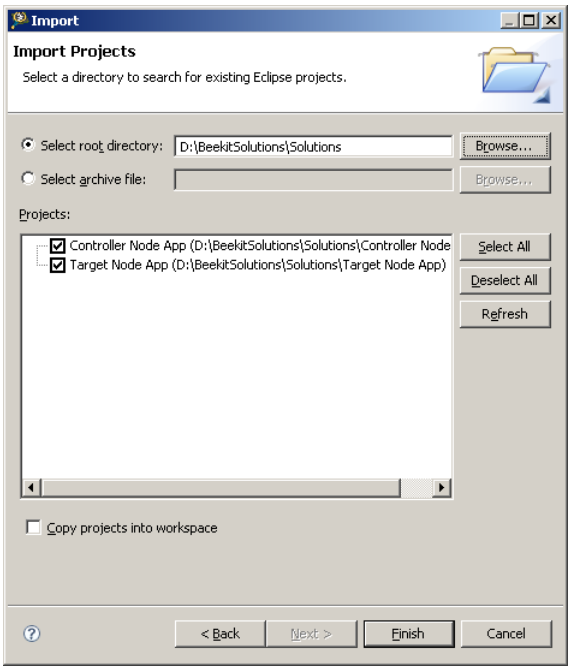

Figure 2-20. CodeWarrior Name New Project Window

7. Click Finish.
8. The first time users import an 1323x-RCM project into the current workspace, the Remote System System Missing window opens and displays a warning as shown in Figure 2-21.

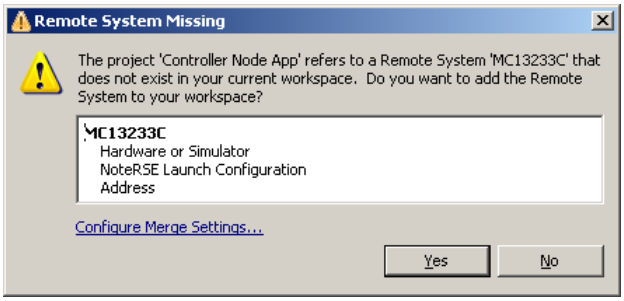

Figure 2-21. CodeWarrior Controller Project Window

9. Click Yes to add the Remote System from the current project to the workspace.

10. CodeWarrior loads the project file as shown in Figure 2-22.

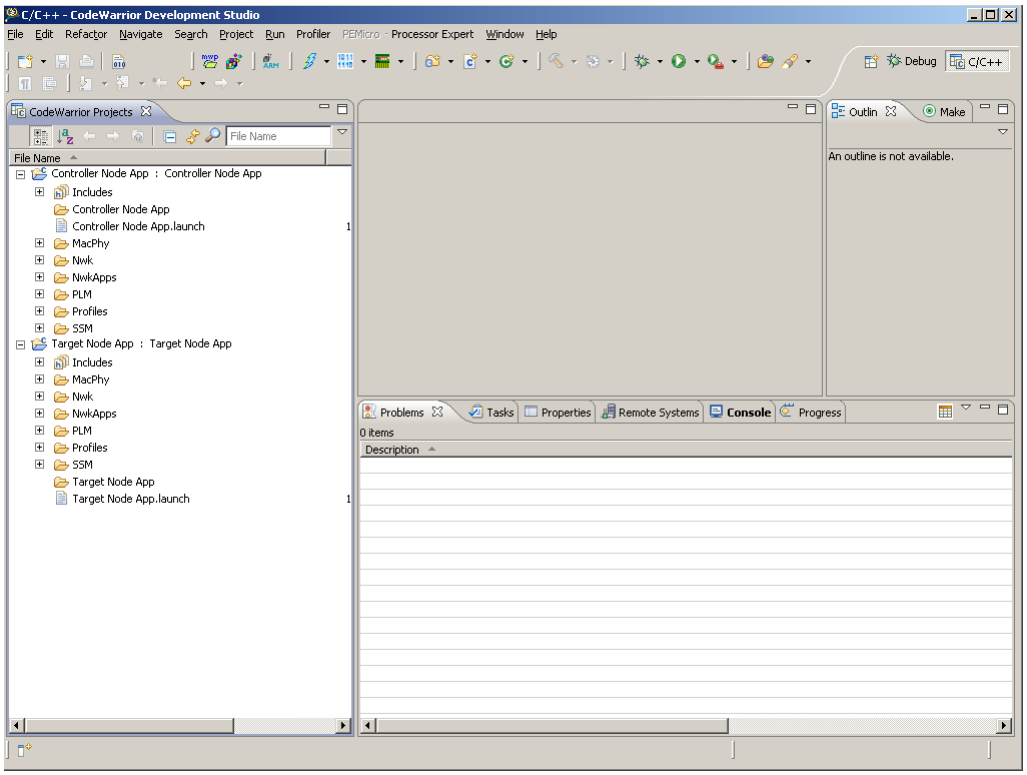

Figure 2-22. CodeWarrior Controller Project Window

2.5 Building a Code Image Using the IDEs

2.5.1 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 2-23.

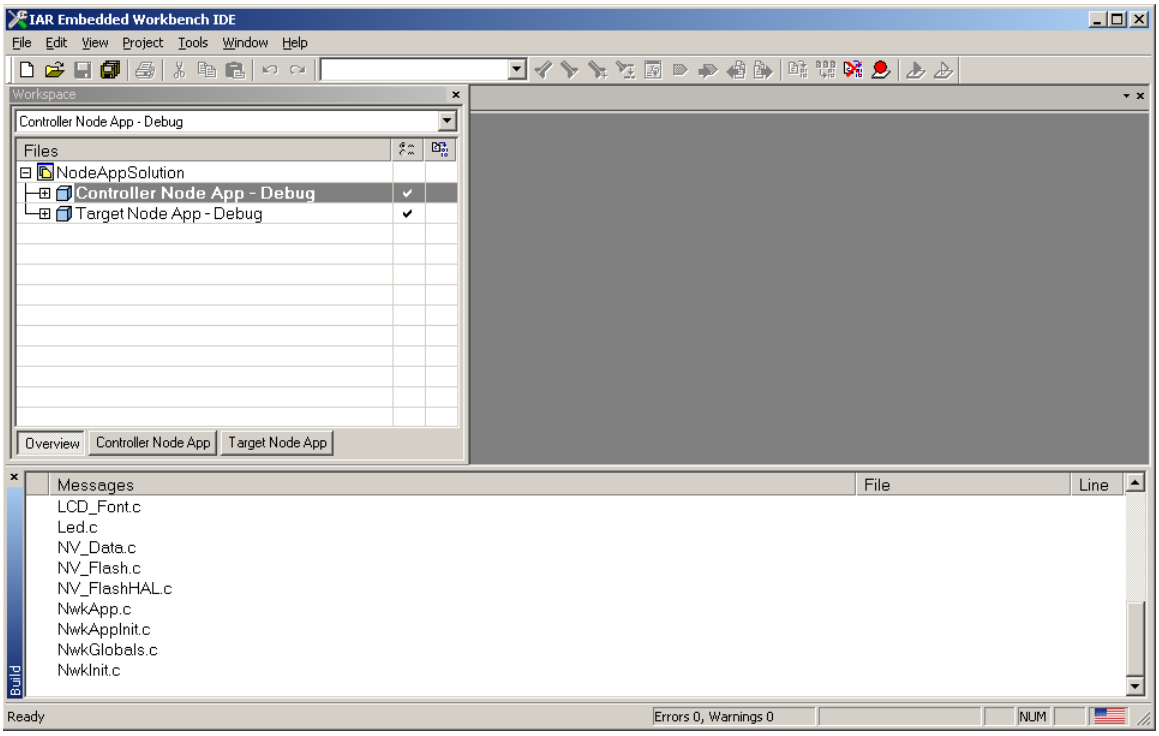

Figure 2-23. IAR EWB Showing the Build Messages Panel

2.5.2 Building a Code Image Using CodeWarrior

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

1. Click the “Build” icon.
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 2-24

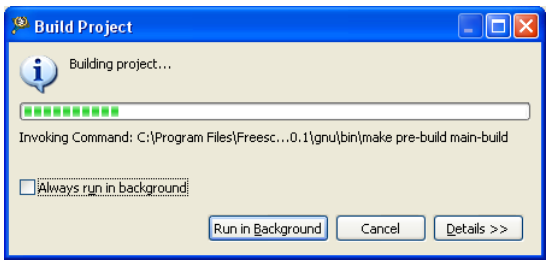

Figure 2-24. CodeWarrior Build Window

2.6 Loading the Code Image into a RF4CE Device

2.6.1 Loading the Code Image into a MC1322x Board via JLink

Perform the following tasks 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 2-25.

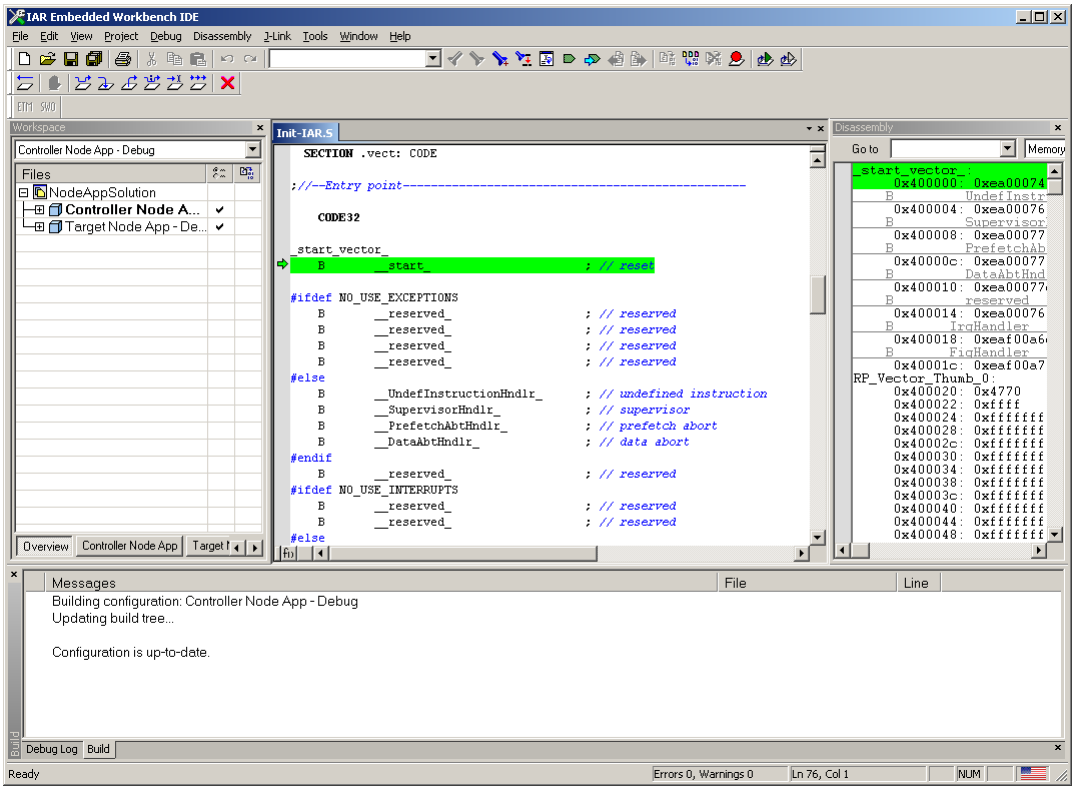

Figure 2-25. IAR EWB Debugging Window

7. Stop the debugger by pressing Ctrl+Shift+D or by choosing “Stop Debugging” from the Debug menu.
8. Right click on the Target Node application in the Workspace Overview panel and select “Set as Active” from the context menu.
9. Repeat the steps as shown in Section 2.6.1, “Loading the Code Image into a MC1322x Board via JLink” to load the code for the other application into another board.

2.6.2 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 tasks:

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 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.
5. Select the project that contains the image to download.
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.
   c) From the menu select Run -> Debug

   CodeWarrior switches to the Debug perspective as shown in Figure 2-26.

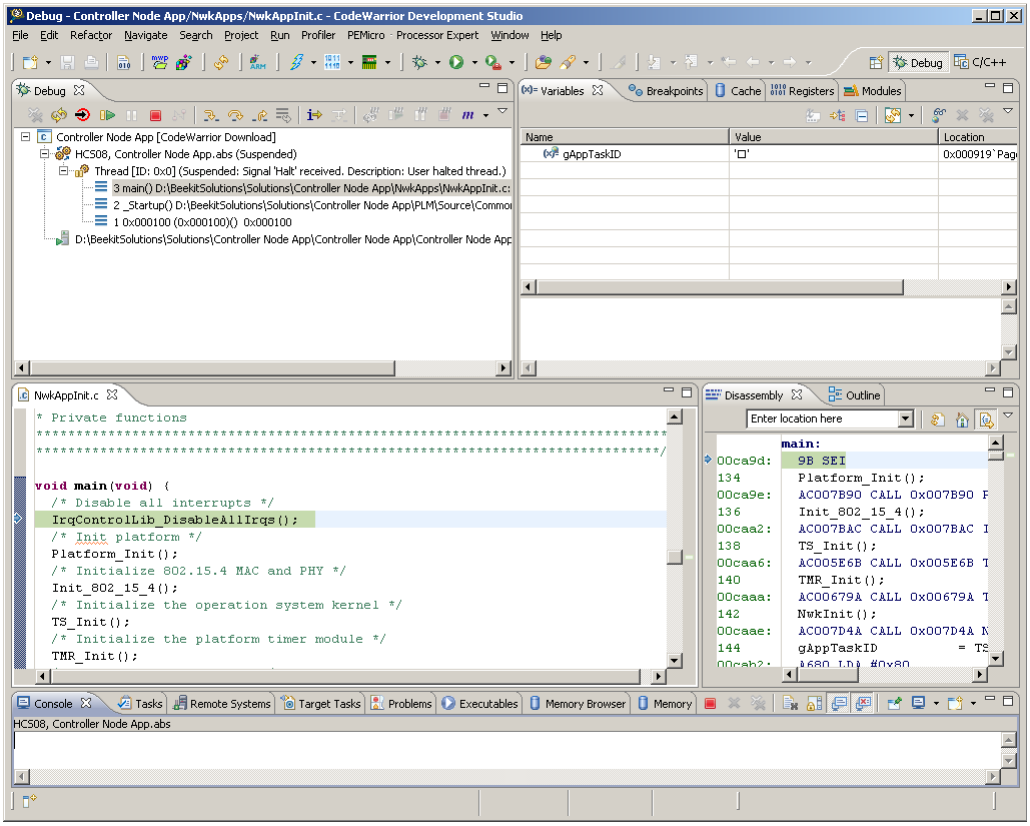

Figure 2-26. CodeWarrior Debug Perspective

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 prepare the board for use.
10. Repeat these steps to place the Target Node code into another board.

Chapter 3 Starting and Running a Simple RF4CE Network

This chapter provides the steps required to establish a small RF4CE network using the two nodes programmed in Chapter 2, “Creating Applications with BeeKit”. The network consists of the Controller node and the Target node.

3.1 UART Setup and Operation

The following sections show how to identify and setup the UART/USB virtual COM ports, set up HyperTerminal, start the controller and target applications, pair the nodes, and use the UART terminals.

3.1.1 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 3-1. (The COM ports shown in Figure 3-1 may be different on other PCs.)

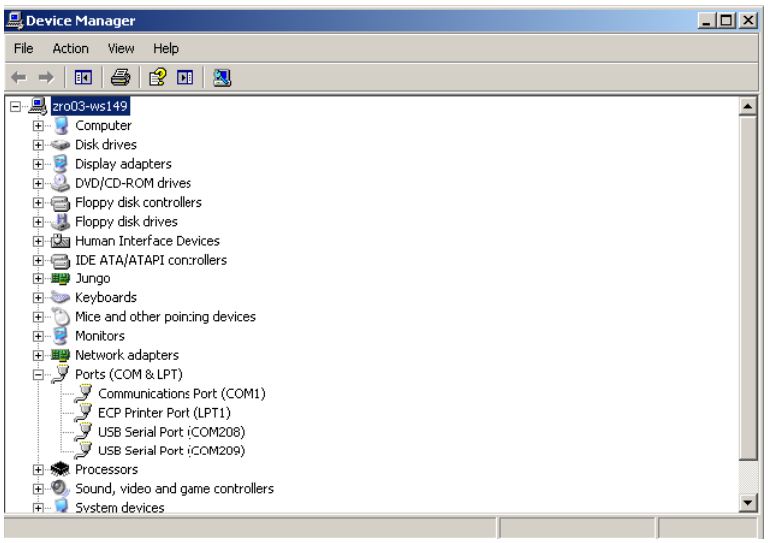

Figure 3-1. COM Ports in Device Manager

3. Using HyperTerminal, set up a Virtual COM Port for each of the two nodes. Figure 3-2 shows the correct COM Port settings to run the applications.

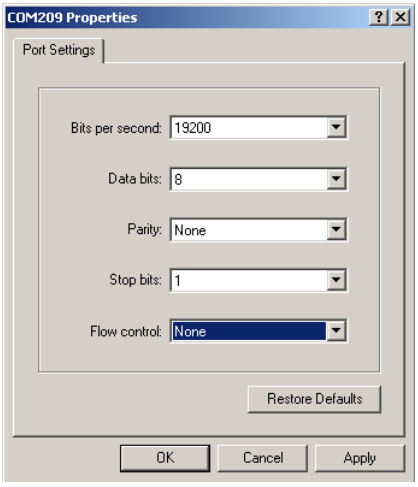

Figure 3-2. Default BeeStack RS-232 Settings

NOTE

If the Flow control setting is set to “Hardware”, users must ensure that the application has the hardware flow control enabled. For this, select the “Platform” component and set the “UART 1 Hardware flow control enable” property to TRUE.

3.1.2 Starting the Controller and Target Node Applications

1. Connect both the Controller and Target 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 3.1, “UART Setup and Operation”.
3. Turn on the power for the Controller and Target boards. Each application will start the node and will print a menu on the assigned Hyperterminal program as it shows the Figure 3-3 and Figure 3-4

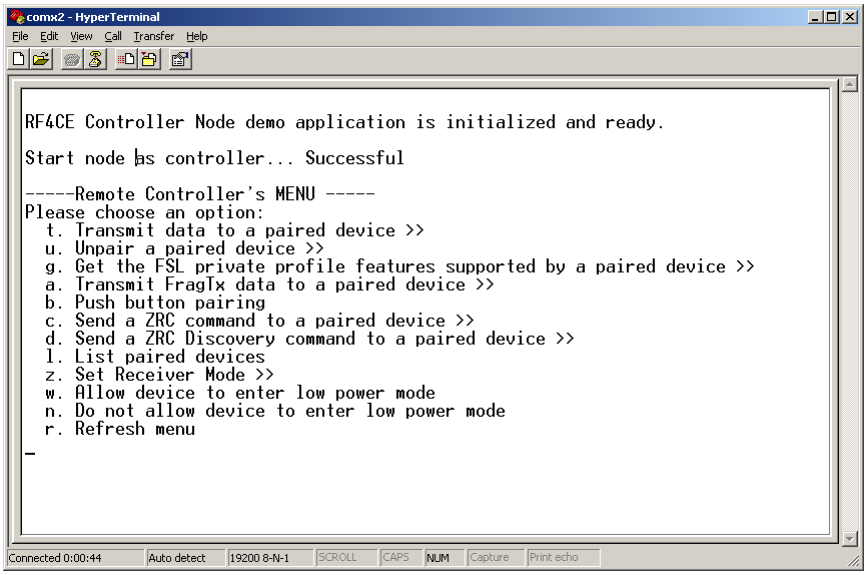

Figure 3-3. Controller Node UART Terminal Menu

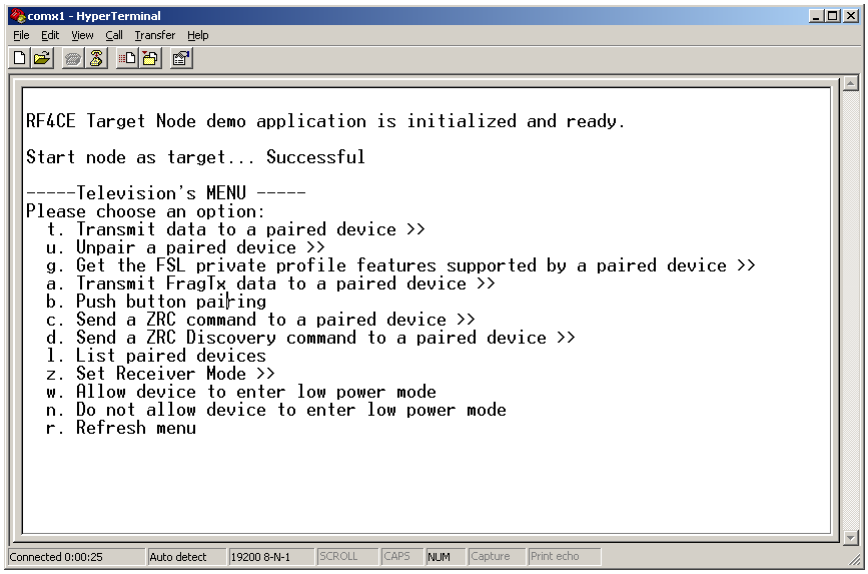

Figure 3-4. Target Node UART Terminal Menu

NOTE

Depending on configuration settings (i.e. the selected profiles, features, device type), the printed options of the menu may be different. For example, when the project is created with the BeeKit, if the Freescale (FSL) Private Profile is disabled, all the menu options related to the FSL Private Profile will not be printed and will not be accessible in the HyperTerminal program.

3.1.3 Using the Controller and Target Node Applications

Once the applications are started, and the devices are running, the Controller and Target applications are ready to be used:

1. In both HyperTerminal programs press the ‘b’ key to start the push button pair process. Both HyperTerminal programs display the process status when it ends.
2. If the devices are paired successful, in the controller HyperTerminal press ‘c’ key to send a ZigBee Remote Control (ZRC) command to the paired target. A text message appears on the HyperTerminal that allows to select a device for transmitting the ZRC command. Select the “TV device” pressing the ‘1’ key. Once the key is pressed the controller sends the command and displays the transmission status. In the target HyperTerminal, the received ZRC command is displayed in details. To refresh the menu press the ‘r’ key.
3. In the target HyperTerminal, press the ‘t’ key to send data to the pair device. A text message appears on the HyperTerminal that allows to select the destination device. Select the “Remote Control” pressing the ‘1’ key. Once the key is pressed the target sends the data and displays the transmission status. In the controller HyperTerminal, the received data request is displayed in details. To refresh the menu press the ‘r’ key.

   These steps are shown in Figure 3-5 and Figure 3-6.

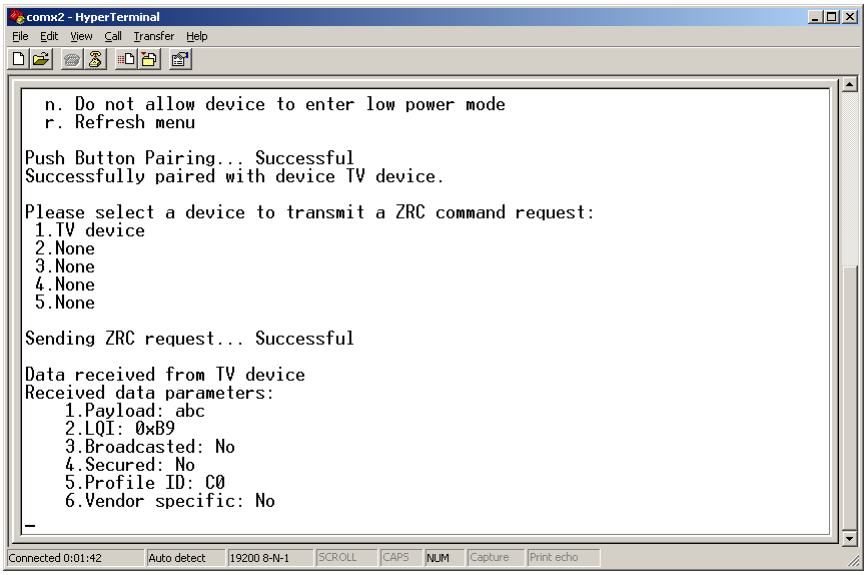

Figure 3-5. Controller HyperTerminal

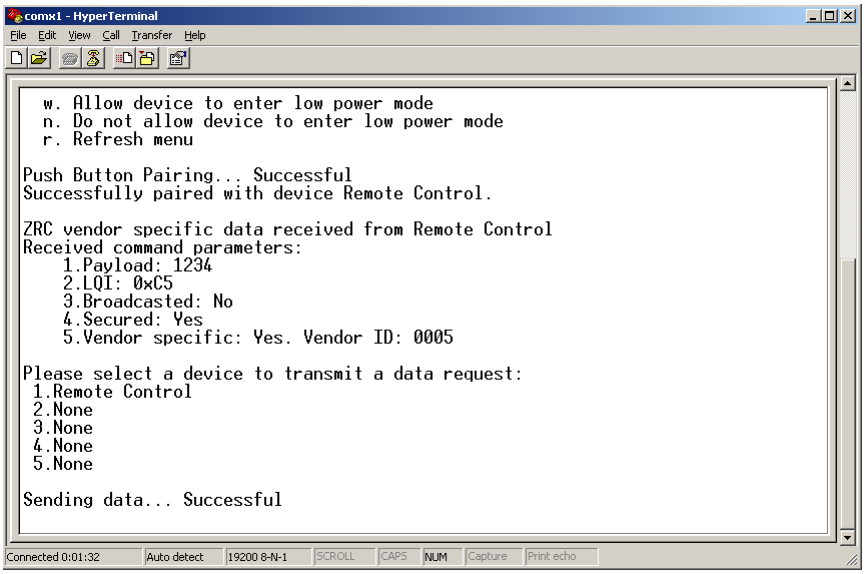

Figure 3-6. Target HyperTerminal

4. Using both HyperTerminal programs, access the rest of the menu options to exercise different features: set the receiver mode, allow or disallow a device to enter in low power mode, list the paired devices, send a ZRC discovery command, unpair a paired device etc.

NOTE

If the device can enter in low power mode (pressing the ‘w’ key) and the receiver is set to be “Off” (pressing the ‘z’ key followed by ‘2’ key), the device will enter in low power mode and the UART connection with the HyperTerminal will be lost. To wake up the device, press any switch on the board. In the HyperTerminal, disallow the device to enter in low power mode by pressing the ‘n’ key and set the receiver “On” (pressing the ‘z’ key followed by ‘1’ key).

Similarly, the receiver can be set to a “Intermittent Rx mode” (pressing the ‘z’ key follow by ‘3’ key). In this mode, the receiver is enabled for an application defined duration (known as the active period) and then disabled. This mechanism is then repeated at an application defined interval (known as the duty cycle). The node periodically enables its receiver for only a short time allowing it to conserve power but still be able to be active on the network. If the device enters in low power (low power mode is allowed), the UART connection with the HyperTerminal will be lost during the inactive period and will be reestablished in the active period. In this case, the pressed key in the HyperTerminal is handled only during the active period. To access the menu options in the HyperTerminal, users should keep the key pressed until the selected option is executed.

Chapter 4
Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software

This chapter shows how to create a RF4CE Network consisting ZigBee Test Client (ZTC) Node application and using the Freescale BeeKit Wireless Connectivity Toolkit. While this chapter describes how to use the ZTC Node application, it does not describe the projects in great detail.

4.1 Software and Hardware Requirements

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

- Chapter 2, "Creating Applications with BeeKit" and Chapter 3, "Starting and Running a Simple RF4CE Network" 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, has been read, the steps performed and understood
- Two 1323x -RCM boards are available for use.

4.2 Creating a ZTC Node App Project

Follow these steps to create a BeeKit project and configure a ZTC (ZigBee Test Client) Node application.

1. Start the BeeKit Wireless Connectivity Toolkit.
2. If another Codebase (MAC, SMAC or ZigbeePro) 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…
4. Select the BeeStack Consumer Apps project type from the left side of the window.
5. Select the "ZTC Node App" template and create the project by entering the following information into the appropriate field:

   Project Name: ZTC Node App

   Solution Name: ZTC Demo Solution

   Location: `c:\Ztc_demo` (or other sub directory on host PC)
6. Access the "Target hardware" property from "Platform" component to change the hardware settings. Set the 1323x-REM board. If a board other than a 1323x –REM is used, choose the correct platform.

7. Select the “Freescale BeeStack Consumer ZRC Profile” component and enable the ZRC commands transmission and reception features as shown in Figure 4-1.

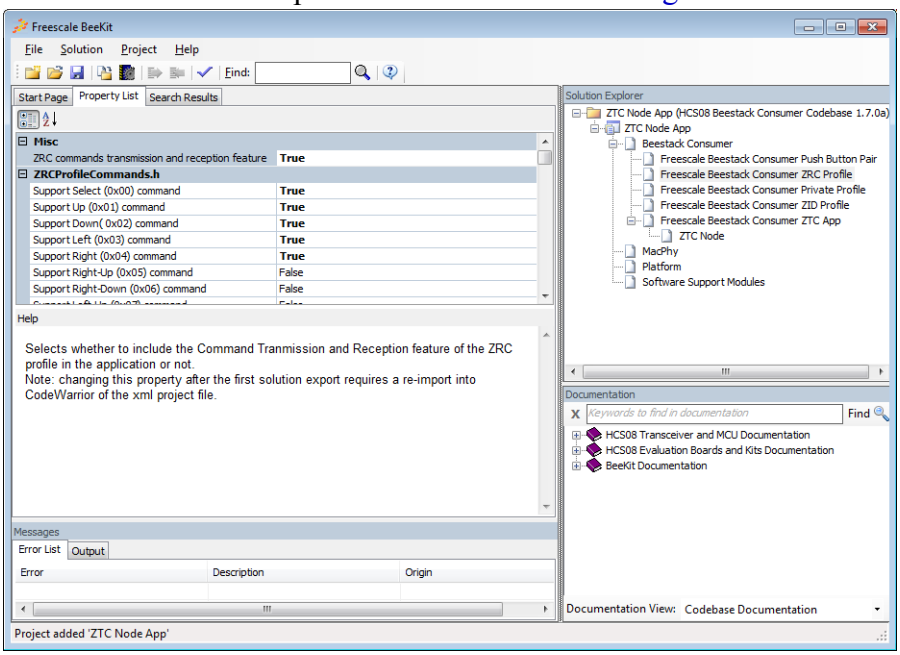

Figure 4-1. Enable the ZRC features

8. Select the “Freescale BeeStack Consumer Push Button Pair” component and enable the Push Button pair Originator and Recipient as shown in Figure 4-2.

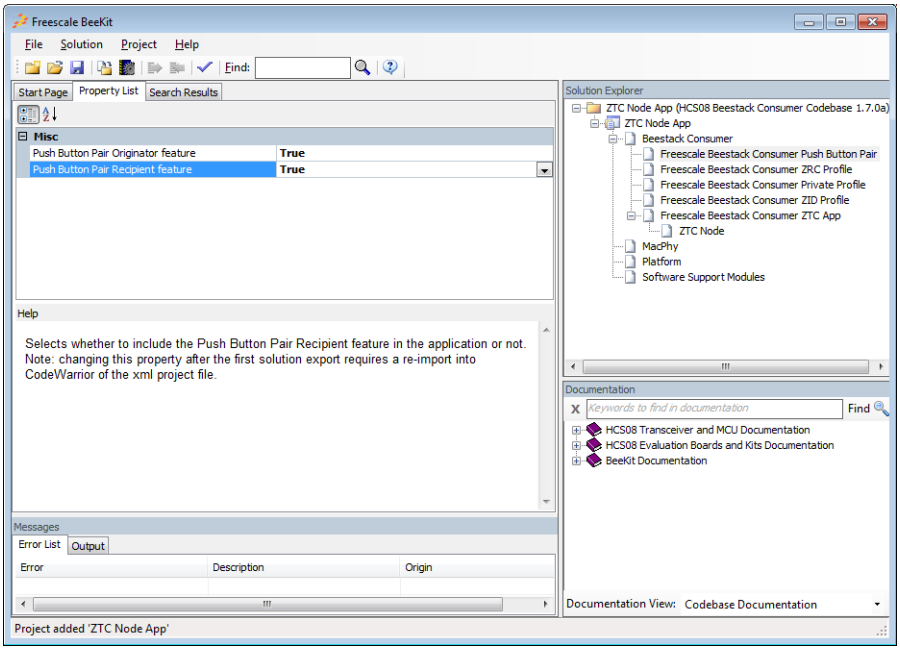

Figure 4-2. Enable the Push Button Pair Features

9. Export the solution, compile the project and program both boards with the ZTC Node application using CodeWarrior.

4.3 Starting a RF4CE Network

This section describes how to use the Test Tool software to access the BeeStack Consumer stack features through a UART interface and also monitors the communication between MAC, network, profile layers and application.

4.3.1 Board Connection and Network Startup

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

1. Connect the appropriate power source to all boards.
2. Connect the boards being used 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 boards: baud rate-38400, ZTC Length-Auto, Use Flow Control-enabled. 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 `ZigBeeRF4CE.xml` file is selected.
5. From Test Tool Command Console tool bar, click on All Commands ->ZTC Commands->ZTC-ModeSelect.Request command and select the parameters in the request as shown in Figure 4-3 to ensure that the board connection are working. Perform this step for both devices.

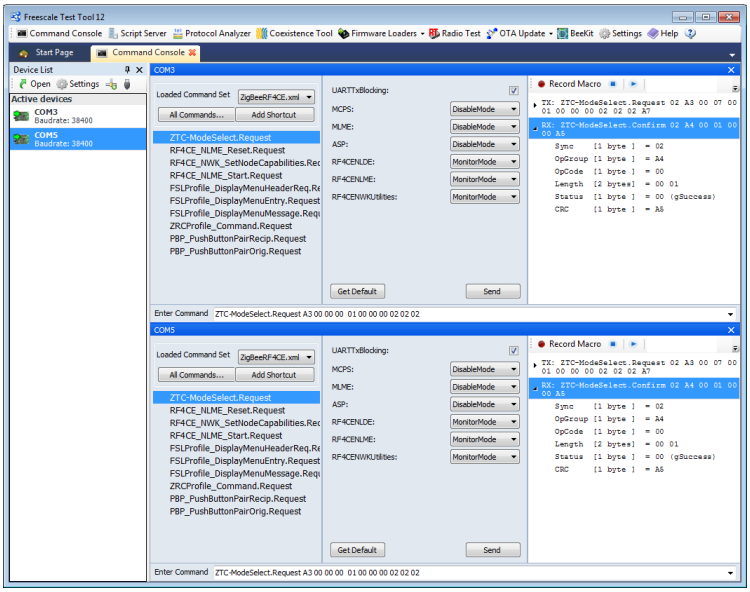

Figure 4-3. Send ZTC-ModeSelect.Request command

7. From Test Tool Command Console tool bar, click on All Commands ->ZTC Commands->ZTC-WriteExtAddr.Request command and set on both devices the IEEE address of the nodes as shown in Figure 4-4. For this example, use the address 0xAAAAAAAAAAAAAAAA for first device from the list of active devices and 0xBBBBBBBBBBBBBBBB for the second device.

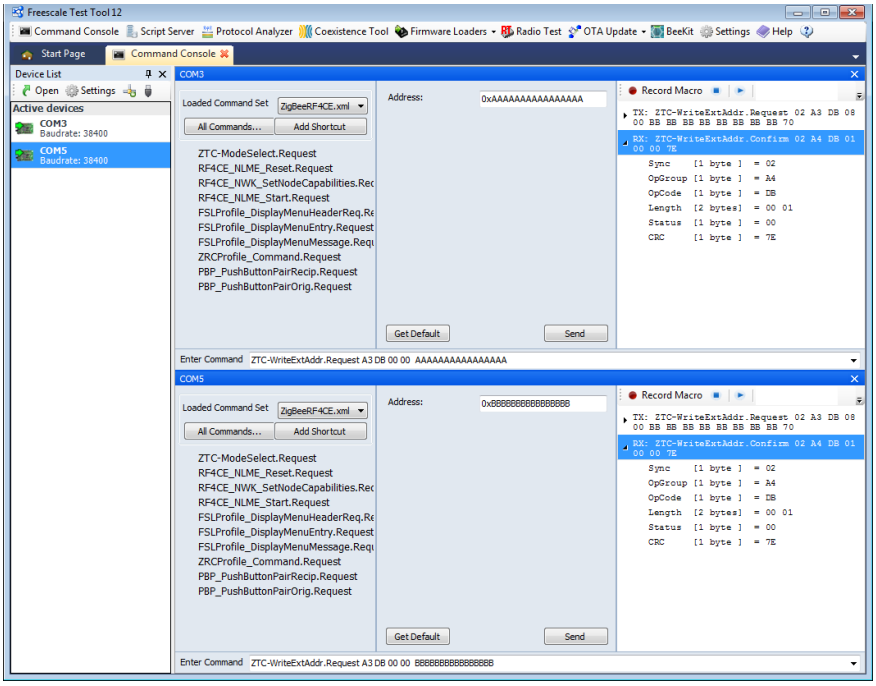

Figure 4-4. Send ZTC-WriteExtAddr.Request command

8. Select the first device from the list of active devices.Configure and start the node to act as a controller. For this, click on All Commands ->RF4CE layer command and send the following Test Tool RF4CE commands in the order shown in Figure 4-5:
    - RF4CE_NWK_Reset.Request
    - RF4CE_NWK_SetNodeCapabilities.Request command with the nwkcNodeCapabilities parameter value 0x0C which means that the node is a controller and supports security.
    - RF4CE_NLME_Start.Request command as shown in Figure 4-5.

Check if the status of the RF4CE_NLME_Start.Confirm message is successful (0x00) as shown in Figure 4-5.

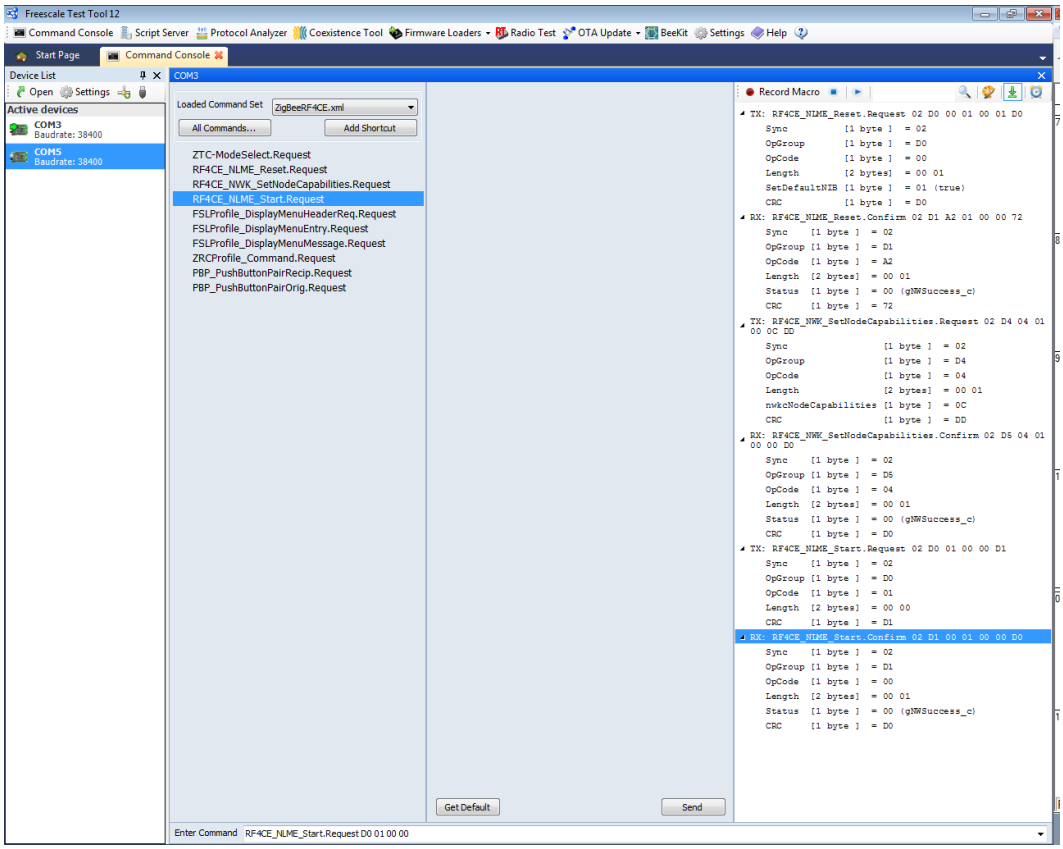

Figure 4-5. Configure and start the controller node

9. Select the second device from the list of active devices.Configure and start the node to act as a target. For this, click on All Commands ->RF4CE layer command and send the following Test Tool RF4CE commands respecting the order from Figure 4-6:
    - RF4CE_NWK_Reset.Request
    - RF4CE_NWK_SetNodeCapabilities.Request command with the nwkcNodeCapabilities parameter value 0x0F which means that the node is a target and supports security.
    - RF4CE_NLME_Start.Request command as shown in Figure 4-6.

    Check if the status of the RF4CE_NLME_Start.Confirm message is successful (0x00) as shown in Figure 4-6.

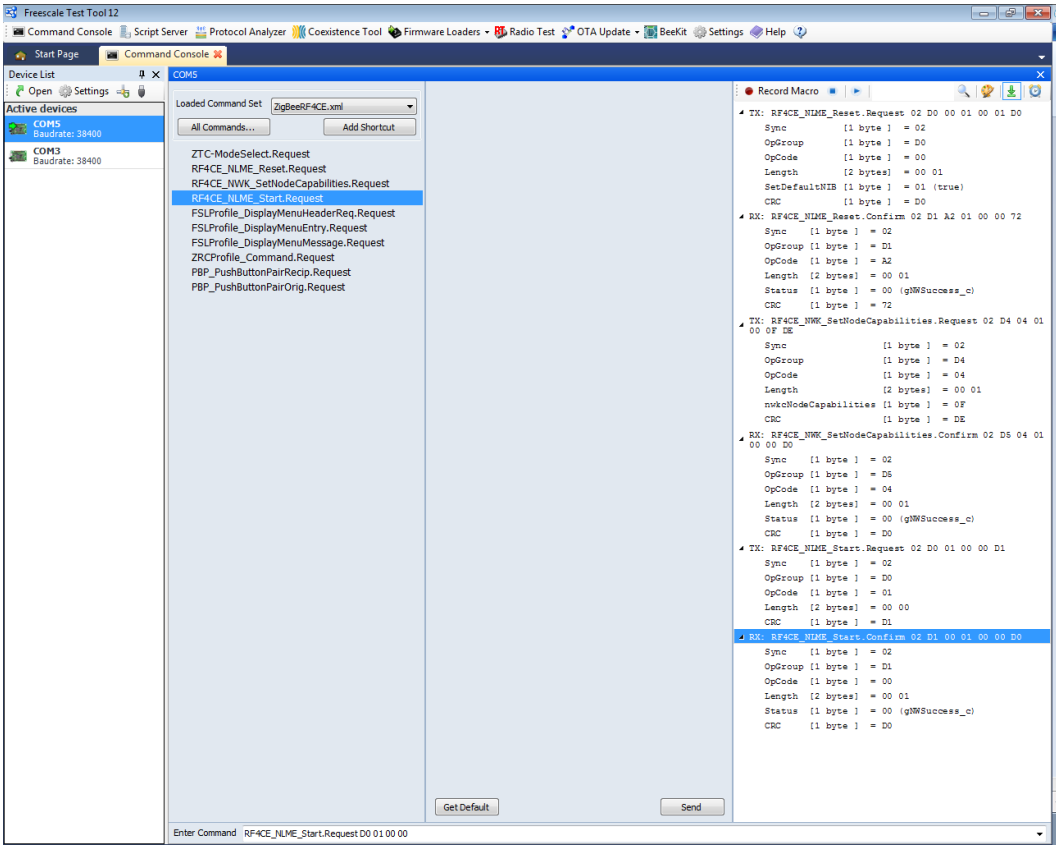

Figure 4-6. Configure and start the target node

10. On the controller device, click on All Commands ->RF4CE layer command->PBP_PushButtonPairOrig.Request command and set the parameters in the Request as shown in Figure 4-7. Do not send the command yet.
11. On the target device, click on All Commands ->RF4CE layer command->PBP_PushButtonPairRecip.Request command and set the parameters in the Request as shown in Figure 4-8. Do not send the command yet.
12. Start the paring process sending the commands from Step 10 and Step 11, so that the interval between the transmission of these requests is less than 30 seconds. The order of the transmission of the commands does not matter. The pairing process ends successfully if the status field of the confirmation messages on both devices is “Success”.

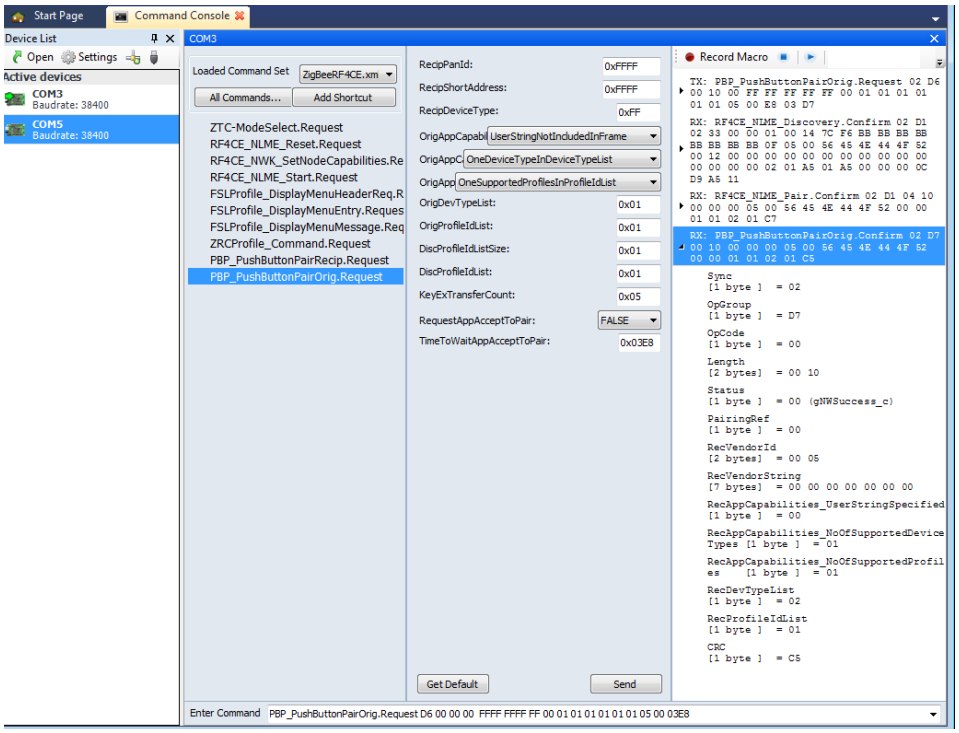

Figure 4-7. Push Button Pair Originator parameters

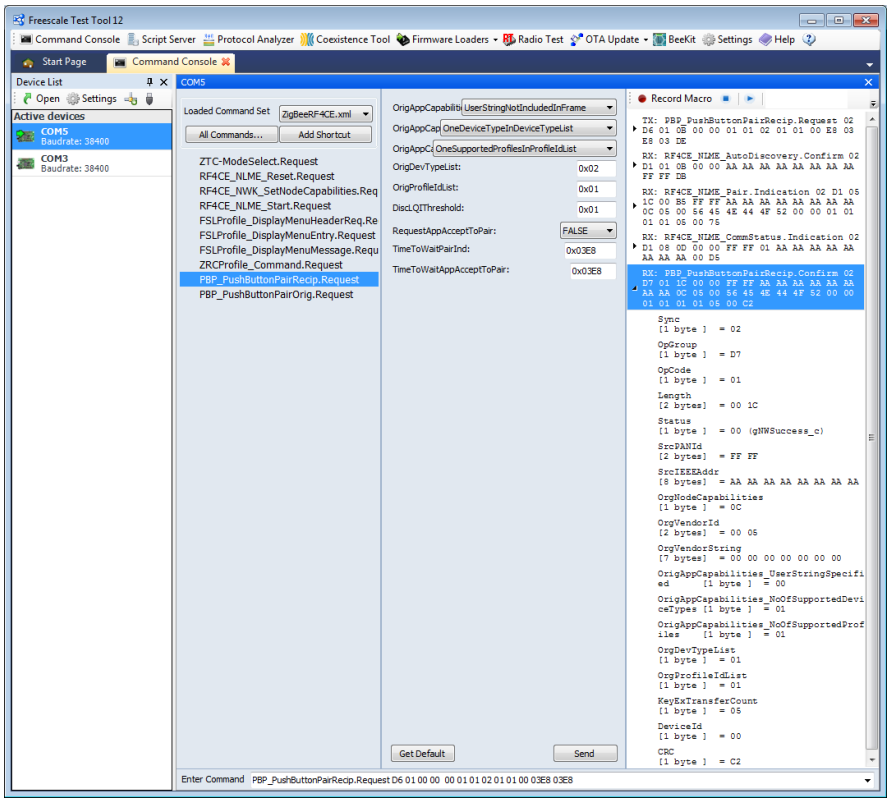

Figure 4-8. Push Button Pair Recipient parameters

13. On the controller device, click on All Commands ->RF4CE layer command->ZRCProfile_Command.Request and set the parameters in the Request as shown in Figure 4-9. Check the confirmation message of the request and if a ZRCProfile_Command.Indication message was received on the second device as shown in Figure 4-9.

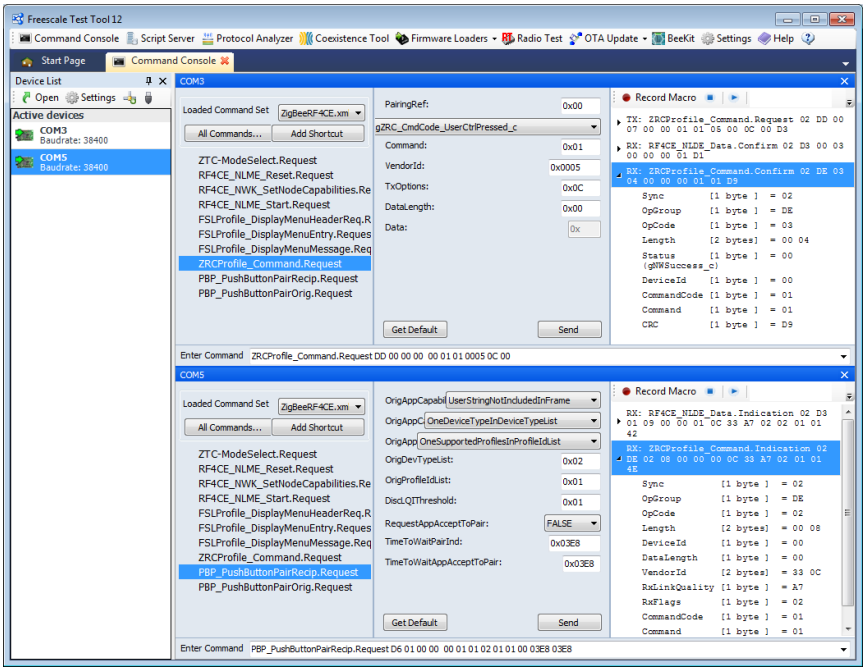

Figure 4-9. Send ZRC Command

14. Repeat Step 13 for the target device.
15. Exercise other RF4CE, ZRC profile and FSL Private Profile commands following the BeeStack Consumer Blackbox Interface document (BSCONBBIUG.pdf).

NOTE

In the same manner, users can use the Test Tool software with the BlackBox Node applications to access the BeeStack Consumer stack features through a serial interface (SCI, SPI or I2C) and also to monitor the communication between network, profile layers and application.

To create a BlackBox application from Beekit, use the ZTC Node application project, select the “Freescale BeeStack Consumer ZTC App” component and enable the “BlackBox enabled” property. Enabling this property will allow to inject and monitor messages (requests and responses) only to/from the stack layer, disabling the Mac layer .

To access the desired features using the commands from Test Tool software, users must first enable and configure these features (RF4CE, ZRC profile or FSL Private profile features) in the BeeKit project.

4.4 Using a BeeStack Consumer BlackBox Binary

The BeeStack Consumer Codebase comes with a series of BlackBox binaries that can be created and generated using the Freescale BeeKit Wireless Connectivity Toolkit.

The users can use the BlackBox binaries in the same manner as it is described in Section 4.3, “Starting a RF4CE Network”. The Freescale Test Tool is used to send and receive data. The BeeStack Consumer Blackbox Interface document provides further details about host interface, protocol framing and command formatting.

This chapter shows only how to load the image to the target board.

4.4.1 Loading the Image to a MC1322x Target Board

1. From the Test Tool main menu, choose View->Firmware Loader->MC1322x Firmware Loader. The MC1322x Firmware Loader appears as shown in Figure 4-10.

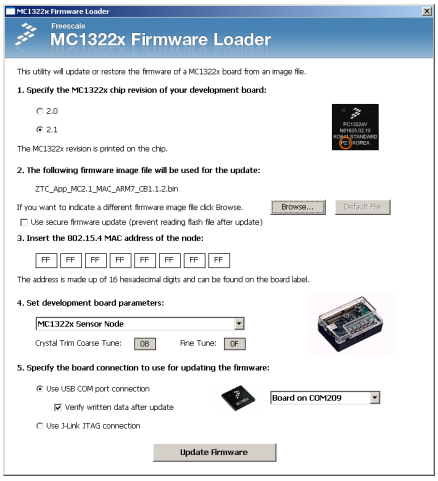

Figure 4-10. MC1322x Firmware Loader Window

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.

4.4.2 Loading the Image to an HCS08 Target Board

1. From the Test Tool main menu, choose View->Firmware Loader->HCS08 Firmware Loader. The HCS08 Firmware Loader appears as shown in Figure 4-11.

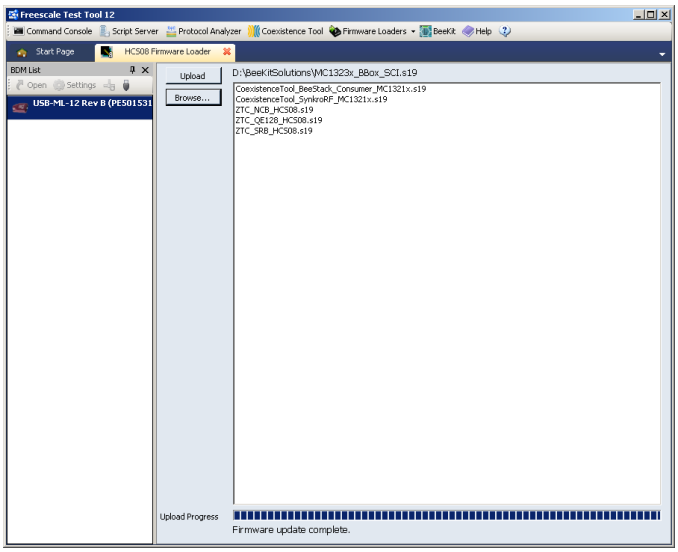

Figure 4-11. HCS08 Firmware Loader Window

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

Chapter 5
BeeStack Consumer Sample Applications

This chapter describes the BeeStack Consumer applications included in the BeeStack Consumer Codebase.

NOTE

Before performing any of the tasks in this chapter, users must read the first three chapters of this guide. Each application is created and generated from the BeeKit Wireless Connectivity Toolkit, selecting the BeeStack Consumer codebase, creating the desired project and choosing the project type "BeeStack Consumer Apps". To run the applications, users should export the solution, compile the projects and program the boards using the appropriate IDE.

Some of the applications interact with users through a menu displayed on a HyperTerminal application. Other applications need the Freescale Test Tool software to configure the device and to send commands. There are also some applications that work in conjunction with a PC application software for demo purposes.

5.1 Controller Node Application

This application implements the functionality of a ZigBee RF4CE controller node. It exercises features from the BeeStack Consumer network, the ZigBee Remote Control profile or from the Freescale Private profile. It interacts with users through a menu displayed on a UART terminal application. The ZigBee Remote Control and Freescale Private Profile features used in the application are chosen at compile time. The application includes support for accessing the following profile features:

- ZigBee Remote Control – Controller-side Push Button Pair
- FSL Private Profile – Fragmented data transmission
- FSL Private Profile – Fragmented data reception

This application implements the basic functions of a remote control. It allows the user to:

- reset the controller node
- start the controller node
- send discovery request
- pair and send data to a target node (e.g. Target Node Application)
- handle unpair requests

NOTE

This application can be used in conjunction with the Target Node Application. See Chapter 3, "Starting and Running a Simple RF4CE Network" to learn how to use the menu options.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

5.2 Target Node Application

This application implements the functionality of a ZigBee RF4CE target node. It exercises features from the BeeStack Consumer network, the ZigBee Remote Control profile or from the Freescale Private profile. It interacts with users through a menu displayed on a UART terminal application. The ZigBee Remote Control and Freescale Private Profile features to be used in the application are chosen at compile time. The application includes support for accessing the following profile features:

- ZigBee Remote Control – Target-side Push Button Pair
- FSL Private Profile – Fragmented data transmission
- FSL Private Profile – Fragmented data reception

This application implements the basic functions of a target node. It allows the user to:

- reset the target node
- start the target node
- handle pair requests
- send data to other nodes
- handle unpair requests

NOTE

This application can be used in conjunction with the Controller Node Application. See Chapter 3, "Starting and Running a Simple RF4CE Network" to learn how to use the menu options.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

5.3 ZTC Node Application

This application allows an external host to access the BeeStack Consumer stack features through a serial interface (SCI, SPI, I2C) and also to monitor the communication between MAC, network, profile layers and application. The MAC, network, ZigBee Remote Control Profile and Freescale Private Profile features used are chosen at compile time. The application code includes support for accessing all the MAC, stack and profile features.

NOTE

See Chapter 4, "Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software" to learn how to use this application with the Test Tool software.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

5.4 HECA Controller Node Application

This embedded application runs on MC1323x-RCM board and implements the functionality of a ZigBee RF4CE remote controller device which uses a keyboard, touchpad, accelerometer and LCD to enable the user to control a number of virtual target devices (TV, MP3 player, Home Status Control) that runs inside the HECA PC application and uses RF4CE BlackBox devices to make use of the RF4CE services and functionality.

For information about how to run the HECA Controller Node and the HECA PC applications, see the *BeeStack Consumer Home Entertainment Application Quick Start Guide* (BSCONHECAQSG).

5.5 Simple TV Application

This application implements the functionality of a ZigBee RF4CE TV node. It runs on a Freescale 13213-NCB or 1320x-NCB board and uses the LCD to display the information that a TV device would usually show on its screen. This application works with the Freescale Virtual Remote application. The Virtual Remote application emulates a remote control through a PC application. The PC application communicates with a USB port connected to an MC13213 SRB board. The board is running a BlackBox device application. The simple TV application uses the following profile features:

- ZigBee Remote Control – Target-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands
- FSL Private Profile – MenuOwner feature

The BlackBox application connected to the Virtual Remote uses the following profile features:

- ZigBee Remote Control – Controller-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands
- FSL Private Profile – MenuBrowser feature
- FSL Private Profile – MenuDisplayer feature

The Virtual Remote PC application and the binary for the BlackBox application it uses can be found in the following location: `Program Files\Freescale\Virtual Remote`. To run the Virtual Remote application setup, perform the following tasks:

1. Connect the MC13213 SRB board to the PC using a USB cable and load it with the S19 formatted file from the Virtual Remote application folder. See the BeeStack Consumer Quick Start Guide for more information about loading binary images to Freescale boards.

2. Power on the 13213-SRB and follow the installation instructions as displayed in the Found New Hardware wizard on the PC. If the drivers do not automatically load, they can be found in the following directory: `C:\Program Files\Freescale\Drivers`
3. Load the 13213-NCB board with the SimpleTV application image generated from BeeKit.
4. Reset both boards.
5. When the initialization of the 13213-NCB is complete, the text “TV off” appears on the 13213-NCB LCD. This shows that the TV application is started and that the TV is in standby mode and waiting for remote commands.
6. Launch the Virtual Remote PC application by clicking on Start ->Programs -> Freescale BeeKit -> Virtual Remote -> Virtual Remote. The Virtual Remote PC application appears.
7. Users must first pair the Virtual Remote application with the TV application that is running on the 13213-NCB as follows:
   a) Click on any of the soft keys of the Virtual Remote PC application to start the controller-side Push Button Pair process.
   b) Long press SW4 on the 13213-NCB board so the TV can start the target-side Push Button Pair process.
   c) After pairing is complete, the TV status appears on the Virtual Remote displaying the “TV off” string.
8. Click the power button on the Virtual Remote application. The 13213-NCB displays the volume level and channel. The Virtual Remote PC application displays the TV status (On) and the volume level and channel.
9. On the Virtual Remote application, click any of the volume, channel or number buttons and watch their status change as displayed on the 13213-NCB LCD and the Virtual Remote PC application display. Pressing the buttons on the 13213-NCB controls the Virtual Remote functions as follows:

- SW1 decreases volume
- SW2 increases volume
- SW3 decreases channel number
- SW4 increases channel number
- If the TV is off, any switch press turns it on
- If the TV is on, a long press of SW1 turns it off

10. On the remote, press the Menu key to access the SimpleTV menu over the air. Browse through the TV menu using the menuUp, menuDown, menuLeft and menuRight keys that surround the OK button. Press the 'Exit' key to exit the TV menu. If not accessed for more than 30 seconds, the TV menu will automatically exit.

This application is available on the MC1321x and MC1320x platforms.

5.6 BlackBox Controller Node Application

This application allows an external host to access the BeeStack Consumer stack features through a serial interface (SCI or I2C) and also to monitor the communication between network, profile layers and application. The network, ZigBee Remote Control Profile and Freescale Private Profile features to be used

are pre configured cannot be selected. From BeeKit, users can choose the communication interface to use (SCI or I2C), some communication parameters (baud rate, SCI port, I2C slave address), the MAC address of the device and other RF settings. The configured solution is exported by BeeKit as an s19 binary file and should be loaded to one of the 1321x-SRB or 1321x-NCB boards. See the BeeStack Consumer Quick Start Guide for more information about loading binary images into MC1321x Freescale boards.

The application includes support for accessing all the network features and the following profile services:

- ZigBee Remote Control – Controller-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands

This application is available on the MC1321x platform.

NOTE

See Chapter 4, "Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software" to learn how to access the enabled features using the Test Tool software.

5.7 BlackBox Target Node Application

This application allows an external host to access the BeeStack Consumer stack features through a serial interface (SCI or I2C) and also to monitor the communication between network, profile layers and application. The network, ZigBee Remote Control Profile and Freescale Private Profile features used are pre configured and cannot be selected. From BeeKit, users can choose the communication interface to use (SCI or I2C), some communication parameters (baud rate, SCI port, I2C slave address), the MAC address of the device and other RF settings. The configured solution is exported by BeeKit as an s19 binary file and should be loaded to one of the 1321x-SRB or 1321x-NCB boards. See the BeeStack Consumer Quick Start Guide for more information about loading binary images to MC1321x Freescale boards.

The application includes support for accessing all the network features and the following profile services:

- ZigBee Remote Control – Target-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands

This application is available on the MC1321x platform.

NOTE

See Chapter 4, "Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software" to learn how to access the enabled features using the Test Tool software.

5.8 BlackBox Node Application

This application allows an external host to access the BeeStack Consumer stack features through a serial interface (SCI or I2C) and also to monitor the communication between network, profile layers and application. The network, ZigBee Remote Control Profile and Freescale Private Profile features to be used are pre configured and are not selectable. The user can choose from Beekit the communication interface to use (SCI or I2C), some communication parameters (baud rate, SCI port, I2C slave address) and the MAC address of the. The configured solution is exported by Beekit as an s19 binary file and should be

loaded into one of the 1323x-RCM or 1323x-REM boards. Please see the Quick Start Guide document for more information about loading binary images into MC1323x Freescale boards.

The application includes support for accessing all the network features and the following profile services:

- ZigBee Remote Control – Controller-side Push Button Pair
- ZigBee Remote Control – Target-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands
- FSL Private Profile – Fragmented data transmission
- FSL Private Profile – Fragmented data reception
- FSL Private Profile – Poll - originator side
- FSL Private Profile – Poll - recipient side
- FSL Private Profile – Remote pair - originator side
- FSL Private Profile – Remote pair - recipient side
- FSL Private Profile – Over the air menus - browser side
- FSL Private Profile – Over the air menus - owner side
- FSL Private Profile – Over the air menus – display side

This application is available on the MC1323x platform.

NOTE

See Chapter 4, “Creating a RF4CE Network using the ZigBee Test Client (ZTC) Node application and the Test Tool software” to learn how to access the enabled features using the Test Tool software.

5.9 RF4CE/SynkroRF Repeater Node Application

This application implements the functionality of a ZigBee RF4CE and/or SynkroRF repeater node. The ZigBee RF4CE and/or SynkroRF repeater can repeat packets that are exchanged between nodes that are part of a particular SynkroRF or a ZigBee RF4CE network. For the repeater to know what network it should repeat packets for, it must acquire information about the nodes in that network. This is accomplished via a learning process that requires user interaction. After the learning process is complete, the repeater is ready to retransmit the packets that it hears being exchanged between the nodes it recognizes.

Using BeeKit, the RF4CE/SynkroRF repeater solution allows users to configure the MAC address of the device, the learn mode LQI threshold and other RF settings. The configured solution is exported by BeeKit as an s19 binary file and should be loaded to one of the 1321x-SRB, 1321x-NCB or MC1323X-REM, MC123x-RCM with MC13233C processor. See the BeeStack Consumer Quick Start Guide for more information about loading binary images into 1321x or MC1323x Freescale boards.

5.9.1 User Requested Learn Mode

The user can direct the repeater to enter the learn mode by performing a short press of SW1 on the board. The repeater signals that is has entered learn mode by blinking LED 1. The repeater remains in learn mode for 30 seconds or until users again press SW1. When in learn mode, the repeater expects to hear packets that have the following characteristics:

- Are transmitted between a single pair of devices. This constraint ensures that the repeater cannot accidentally learn nodes that are part of another network.
- Have the LQI greater than a preset threshold. This constraint ensures that the repeater can still learn new nodes even while devices from other networks are placing information over the air while the repeater is in learn mode.

The learn process ends with a Success status if all the packets that have been heard are coming from a single source node going to a single destination node. This status is displayed by the repeater by keeping LED1 on for two seconds after the learn process completes. The two nodes are then added to the repeater nodes table.

The learn process ends with a Failed status if any of the following occur:

- 30 seconds has elapsed since the user started the learn process and did not stop it.
- If no packet with an LQI greater than the preset threshold was heard during the learn state.
- If packets with an LQI greater than the preset threshold were heard but they were not coming from a single source node and going to a single destination node.

When the repeater learn state ends with a Failed status, LED1 is immediately turned off. In this, case no information is added to the repeater nodes table.

After placing the repeater in learn mode, users should send a command from the remote to only one of the devices it is paired with. Next, exit the repeater learn mode and monitor LED1 to see if the learn process was completed with successfully or if it failed. If it was successful, the repeater is able to repeat any future packet transmitted between the remote and the device regardless of the direction. The learning steps should be performed in the same manner for all the devices that users wants included in the repeater table.

NOTE

- In case the receiver is in the transmitter range, turn off receiver to allow transmitter to send on all RF4CE channels.
- In learning process use transmissions on multiple channels with acknowledge (ACK).

5.9.2 Dynamic Learn Mode

While running, the repeater is in a dynamic learn mode. If it hears packets coming/going from/to a known node, but going/coming to/from an unknown node, the repeater assumes that the unknown node is also part of the network and adds the node to the repeater node table. Future packets coming/going from/to the previously unknown node are now repeated.

5.9.3 Resetting the Repeater

The repeater node table is preserved even if the device is turned off then on again. If users want to clear the repeater node table, perform a long press of SW1 on the board.

5.9.4 Repeating Packet Mode

The repeater device will not retransmit any frame coming/going from a known node. The repeater only retransmits those packets that are retried by the source node for a preset number of times. The presence of the retries suggests that the destination device may not hear the packets due to the poor signal strength or presence of interference. While retransmitting packets, the repeater blinks LED 4. The repeater can retransmit four different packets exchanged between four different pairs of devices at any time.

This application is available on the MC1321x platform.

5.10 Over-The-Air Programmer (OTAP) Controller and Target Node Applications

The OTAP Controller and Target Node applications include the same features as the standard Controller and Target Node applications, but they are supplemented by over the air programming features that are part of the Freescale Private Profile. The purpose of both applications is to demonstrate the OTAP features by upgrading client device firmware with images received over the air from server devices. In the demonstration template applications, the OTAP Target Node plays the role of the server and the OTAP Controller Node plays the role of the client.

To store the firmware image, the OTAP client and server should have an external non-volatile memory storage device installed. For example, the 1320x-QE128 development boards provide an AT24C1024BW EEPROM 1MBit chip; for MC1322x demo application, both devices need to be attached to a 1 Mbit, serial FLASH memory M25P10-A from ST Microelectronics connected over the SPI to the MC1322x MCU.

The target application does not use menus displayed on a UART terminal console. This is because the target application interacts over the UART interface with the OTAP RF4CE module of the Test Tool 12

PC application as described in Section 5.10.5, “OTAP RF4CE Module from Test Tool 12”. User interaction is accomplished using switches 1 and 2 on the board itself.

When the server application is up and running, it receives client firmware images from TestTool 12 over the UART interface and places them in its external memory storage device (FLASH or EEPROM). The OTAP Target Node application provided as a server demonstration supports storing one single client image in external memory storage device. The server can service multiple clients, downloading the images all at the same time. To increase the reliability of this process, Freescale recommends increasing the number of big buffers in the server application’s pool. The gTotalBigMsgs_d define enables this capability and its location is as follows:

```
NwkApps\Configure\ApptoMacPhyConfig.h.
```

The OTAP Controller Node application interacts with users through a menu displayed on a UART terminal. The menu options are accessed as described in Chapter 3, “Starting and Running a Simple RF4CE Network”. Also, the “ q. Query Next Image Request >> “ menu option was added that allows the client node to interrogate the server node if there is a new upgrade image available.

The client receives over the air firmware updates from the server through the Freescale Private Profile OTAP feature and stores the received image in its external memory storage device (FLASH or EEPROM). The received image overrides the current image with the help of a bootloader application after a reset of the client device.

The Freescale Private Profile OTAP Client feature supports a configurable interval and number of frame retransmissions.This are described in Beestack Consumer Private Profile User’s Guide. Adjusting these parameters may prove useful when trying to perform over the air updates in noisy RF environments or when trying to download images from servers that service multiple clients at the same time.

The OTAP Controller and Target Node applications, and the Freescale Private Profile OTAP are interfaced with platform through OtapSupport.h

NOTE

- The OTAP Controller Node application template provided as a demonstration in the codebase, supports one single image to be loaded and booted to the external memory storage device.
- The OTAP Controller Node application template provided as a demonstration in the codebase supports updates with firmware images containing a pair table of the exact same size as the ones defined in original image.

5.10.1 Bootloader for S08 Based Platforms

The bootloader is part of the client project and it is delivered in source code, as a platform component. The bootloader is not using compiler's libraries provided by Freescale CodeWarrior and is not accessing any other components from the rest of the project, except *_Startup* function.

The bootloader is placed in internal flash memory, starting with address 0xFC00, and is using only1k of internal flash memory. The pages where the bootloader is placed are write protected.

A flag stored in the internal memory flash is used for checking the presence of an upgrade image in the external memory storage device. Another flag stored in the internal flash memory is used for completion of upgrade image copying process in internal memory flash. Both flags must be in an unprotected page from internal memory flash. If the copying process completion flag is not set, the bootloader will restart the copying process at the next reset.

The S08 bootloader executes (in order) the following functions:

1. Check at startup if an upgrade image is present in the external memory storage device.
2. Copy the upgrade image in the internal flash memory and preserve pages that are specified in the image bitmap.
3. Uncheck the upgrade image flag and set the copying process completion flag.

The upgrade image must contain the bootloader component.

The bootloader supports only the S08 wireless development boards defined in BeeKit, and it may serve as a template for a different configuration.

5.10.2 Bootloader for ARM7 (MC1322x) Based Platforms

The bootloader is an independent application provided in the codebase as a source code template. The bootloader must be loaded to the client's internal FLASH when the client device is programmed with its initial version of firmware. Loading both the bootloader and the actual OTAP Controller Application during the initial firmware programming is done using the MultiImageFlashProgramer batch file. The bootloader and its content and usage modes are described in more detail in Section 5.10.6, "MultiImageFlashProgrammer - ARM7(MC1322x) Based Platforms".

Once the bootloader is programmed and the client device is sent out to the field, the bootloader cannot be updated through an over the air update process, it can only be updated by flashing the device again. Because the bootloader is a separate application, it allows the client application to be developed independent of the bootloader.

The bootloader image is placed at address 0x00 in the internal FLASH and is the first application that is run after reset. It is copied by the MC1322x bootstrap ROM application at the start of the RAM and is given execution control. The bootloader relocates itself at the end of the RAM and starts executing from there. The first action is to check whether an image with the correct CRC is present in the external FLASH. If yes, the image is copied to the internal FLASH preserving the sectors not marked in the override bitmap field associated with the image. After the process completes successfully, the image is invalidated in the external FLASH. After this process, or if no image was available in external FLASH, the bootloader copies the image from the internal FLASH in the RAM and gives it execution control. The client application is placed in internal FLASH, starting with the beginning of the second FLASH sector. The image size of the client application should be less than 96k (RAM size) minus 600 octets. This 600 octets is the bootloader footprint module that is responsible for copying the client image from the internal FLASH to RAM.

5.10.3 OTAP Controller and Target Node Applications Steps for S08 Based Platforms

To run a demonstration where a client (OTAP Controller Node application) is updated with a new image from a server (OTAP Target Node application), a number of steps must be performed.

Export the OTAP Controller Node application and OTAP Target Node application from the 'BeeStack Consumer OTAP Demo Apps' project type. Select the S08 platform, depending on the hardware being used. Leave all other settings at their default values.

1. Connect both boards to two PC USB ports. Install the PC drivers for the VirtualComm ports detected by the operating system from the following location:
   `c:\Program Files\Freescale\Drivers`
2. Build the OTAP Target Node application and load it to the board that will act as the server.
3. Build the OTAP Client Node application and load it to the board that will act as the client.
4. Open a HyperTerminal application (Start/Programs/Applications), connect it to the Virtual Com Port installed for the client application and set its communication parameters as follows:
   - 19200 baudrate
   - 8 bits
   - No parity
   - 1
   - No flow control.
5. Reset both devices.
6. Pair the devices by pressing the 'b' key in the client's HyperTerminal window and short pressing switch '1' on server board.
7. Create an updated client image by modifying the order of the menu items. Make sure to increase the file version stored in the otapDemoFileVersion variable in the `NwkApps\NwkApp.c` file. Build the updated client application.
8. Load the updated image to the external FLASH of the server node as described in Section 5.10.5, "OTAP RF4CE Module from Test Tool 12". Set the override sector bitmap to preserve the FLASH sectors used by the NVM module. This prevents the pair table of the client from being erased. After loading completes, the server starts to automatically advertise the image. The client is then notified about the presence of the new image and because the hardware version is the same and the file version is newer, the over the air transfer process starts. The server node, provided as a template, can also send the Image Notify requests to advertise the image it has in its external FLASH each time that switch 2 on the board is pressed.
9. After the image is successfully downloaded to the client, the demonstration code provided in the client application automatically resets it after the delay period until the upgrade time that was set by users for the image passed.
10. After the client application resets, ensure that the order of the menu items displayed by the client in the HyperTerminal window was updated. Also, if the pair table was preserved, check that the information in the pair table of the client by pressing the 'L' key to display the pair table. This should contain the server node.

5.10.4 OTAP Controller and Target Node Applications Steps for ARM7 (MC1322x) Based Platforms

To run a demonstration where a client (OTAP Controller Node application) is updated with a new image from a server (OTAP Target Node application), a number of steps must be performed.

Export the OTAP Controller Node application and OTAP Target Node application from the 'BeeStack Consumer OTAP Demo Apps' project type. Select the MC13224 or MC13226 platform, depending on the hardware being used. Leave all other settings at their default values.

1. Connect both boards to two PC USB ports. Install the PC drivers for the VirtualComm ports detected by the operating system from the following location:
   `c:\Program Files\Freescale\Drivers`
2. Build the OTAP Target Node application and load it to the board that will act as the server.
3. Export the Bootloader application from the 'BeeStack Consumer OTAP Demo Apps' project type.
4. Build the OTAP Client Node application. Take the .out file from this build and write it over the application.out file in the MultiImageFlashProgrammer folder that resides in the same directory where the Bootloader was exported.
5. Program the client board using the MultiImageFlashProgrammer batch file located in the MultiImageFlashProgrammer folder, as described in Section 5.10.6, "MultiImageFlashProgrammer - ARM7(MC1322x) Based Platforms". The bootloader application does not need to be compiled as its .out file is already present in the MultiImageFlashProgrammer. Still, if users need to modify parts of the bootloader application, it must be compiled and its resulting.out file must be written over the bootloader.out file in the MultiImageFlashProgrammer folder before running the procedure described in Section 5.10.6, "MultiImageFlashProgrammer - ARM7(MC1322x) Based Platforms".
6. Open a HyperTerminal application (Start/Programs/Applications), connect it to the Virtual Com Port installed for the client application and set its communication parameters as follows:
   - 19200 baudrate
   - 8 bits
   - No parity
   - 1
   - No flow control.
7. Reset both devices.
8. Pair the devices by pressing the 'b' key in the client's HyperTerminal window and short pressing switch '1' on server board.
9. Create an updated client image by modifying the order of the menu items. Make sure to increase the file version stored in the otapDemoFileVersion variable in the `NwkApps\NwkApp.c` file. Build the updated client application.
10. Load the updated image to the external FLASH of the server node as described in Section 5.10.5, "OTAP RF4CE Module from Test Tool 12". Set the override sector bitmap to 1FFFFFFE to preserve the FLASH sectors used by the NVM module. This prevents the pair table of the client from being erased. After loading completes, the server starts to automatically advertise the image.

The client is then notified about the presence of the new image and because the hardware version is the same and the file version is newer, the over the air transfer process starts. The server node, provided as a template, can also send the Image Notify requests to advertise the image it has in its external FLASH each time that switch 2 on the board is pressed.

11. After the image is successfully downloaded to the client, the demonstration code provided in the client application automatically resets it after the delay period until the upgrade time that was set by users for the image passed.
12. After the client application resets, ensure that the order of the menu items displayed by the client in the HyperTerminal window was updated. Also, check that the information in the pair table of the client was preserved by pressing the ‘L’ key to display the pair table. This should contain the server node.

5.10.5 OTAP RF4CE Module from Test Tool 12

The OTAP RF4CE module in TestTool 12 (Figure 5-1) is used in this demonstration for transferring an image containing a client firmware upgrade and its associated parameters in the external memory storage device of the OTAP server. The sample server application provided as a template in the codebase assumes for MC1322X platforms that a 1 Mbit, serial FLASH memory M25P10-A from ST Microelectronics is attached over the SPI to the MCU of the server device. For S08 wireless platform, the sample server application assumes that a development board supported in codebase is used, all this boards contains a memory storage device (EEPROM) and drivers for it are implemented in platform component.

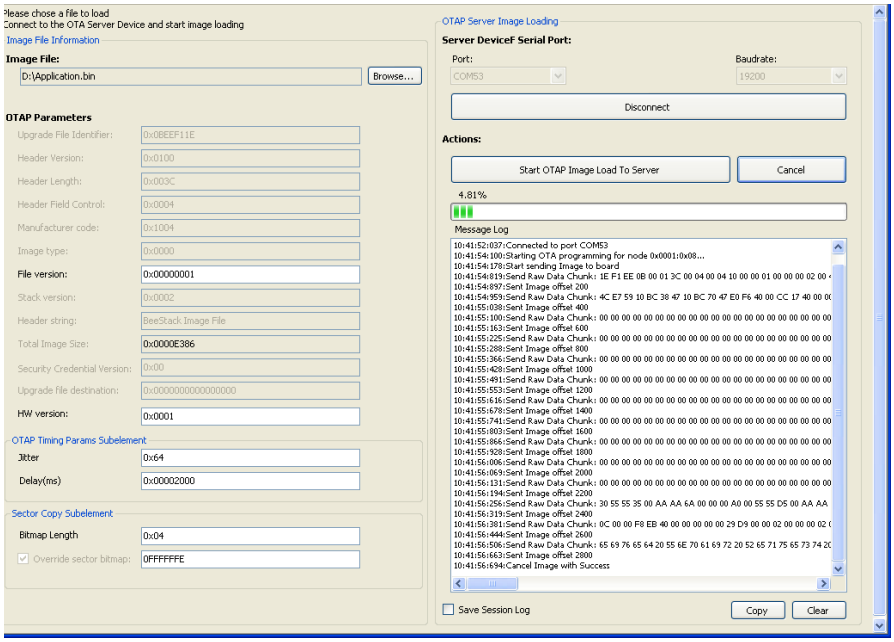

Figure 5-1. OTAP Graphic User Interface from Test Tool 12

To transfer an image and its associated parameters to the external server, perform the following steps:

1. Browse for the image file, which should be in binary format for MC1322x or a .S19 file for S08 platforms.

2. Set the image file version. Ensure that the file version provided here is an exact match of the file version used in the otapDemoFileVersion variable from the `NwkApps\NwkApp.c` file when the client image was built.
3. Set the image hardware version. Ensure that the hardware version provided here is an exact match to the hardware version used in the otapDemoHardwareVersion variable from the `NwkApps\NwkApp.c` file when the client image was built.
4. Set the jitter value between 0x01 and 0x64.
5. Set the Delay value - This is the delay until upgrade value and is provided in milliseconds.
6. Set bitmap length as follows:
   — For MC1322x demonstration, the value is 4. This means that the bitmap has 32 bits. Each bit corresponds to one MC1322x Flash sector which is 4KB in length.
   — For S08 the maximum bitmap length has 128 bits length for QE128, each bit in bitmap corresponds to one page from MCU internal flash.This pages are 512 bytes length for MC1321x and QE128, and 1024 bytes length for MC1323x.
7. Set the override sector bitmap as follows:
   — For MC1322x the rightmost bit in the bitmap corresponds to sector 0 (the one where the bootloader is placed) while the leftmost bit corresponds to last internal FLASH sector.
   — For S08 the right most bit correspond to first internal flash page while the leftmost bit corresponds to last page. A 1 bit means that the corresponding sector/page can be upgraded with new content. A 0 bit means that the corresponding sector should not be upgraded and its content is left untouched. The bits that are outside memory length are not use by bootloader and should be used as padding bits (in case of MC13233C the bitmap length is not a multiple of 8).
8. Set UART port number to the Virtual Com port where the server board is connected.
9. Set the baud rate - For this demo, the baudrate is 19200.
10. Press “Connect” and check the message log.
11. Press “Start OTAP Image Load To Server” and check the message log to ensure that the operation completed successfully.

During the transfer, the server application performs a CRC on the image. When the image transfer is complete, the resultant CRC is compared to the CRC provided by TestTool as an element associated with the image. If they both match, the process is considered success and complete.

The image is provided to the server application by TestTool as an OTA file (as described in ZigBee document 095264r00). The OTA file format is composed of a header, followed by a number of sub-elements. The header describes general information about the image, such as file version or hardware version. The RF4CE Private profiles use a couple of the fields defined in the header of an image, marked with an “E” as shown in the following table.

Table 5-1. OTA Header Fields

Octets	Data Types	File Names	Enable/Disabled
4	Unsigned 32-bit integer	OTA Upgrade File Identifier	D
2	Unsigned 16-bit integer	OTA Header Version	D

Table 5-1. OTA Header Fields (continued)

Octets	Data Types	File Names	Enable/Disabled
2	Unsigned 16-bit integer	OTA Header Length	D
2	Unsigned 16-bit integer	OTA Header Field Control	D
2	Unsigned 16-bit integer	Manufacturer Code	D
2	Unsigned 16-bit integer	Image Type	D
4	Unsigned 32-bit integer	File Version	E
2	Unsigned 16-bit integer	Zigbee Stack Version	D
32	Character String	OTA Header String	D
4	Unsigned 32-bit integer	Total Image Size(including header)	D
2	Unsigned 16-bit integer	Minimum Hardware Version	E
2	Unsigned 16-bit integer	Maximum Hardware Version	D

Header elements marked as disabled are not configurable from the PC application. The disabled fields are present in the OTA header file that is sent to the server over the UART, but are ignored by the server application.

Sub-elements in the file are used for:

- defining the actual image file, binary for MC1322x platforms and S19 for S08 platforms,
- the bitmap of the internal FLASH sectors for MC1322x platforms and of the internal FLASH pages for S08 platforms not to be upgraded
- OTAP demonstration specific information.

Sub-elements in the file are composed of a tag identifier followed by a length field and then by the data.

Table 5-2. Sub-element Format

Octets	2-bytes	4-bytes	Variable
Data	Tag ID	Length Field	Data

Sub-elements are generally specific to the manufacturer and the implementation. Table 5-3 details the tag identifiers used in this demonstration in the order taken from the OTA file built by the PC application.

Table 5-3. Tag Identifiers

Tag Identifiers	Description
0xF000	Override sector bitmap
0xF001	OTAP demo information
0x0000	Upgrade Information

The identifiers may be used to provide forward and backwards compatibility as new sub-elements are introduced. Existing devices that do not understand newer sub-elements may ignore the data. This feature is not implemented in this demonstration.

The override sector bitmap informs the client's bootloader which sectors/pages content must be updated in the internal FLASH. In the Test Tool 12 GUI, the right most bit of the bitmap corresponds to sector 0 or first page and the left most bit corresponds to the last internal flash sector/page. If the corresponding bit is set to '1', then the sector/page is erased.

For MC1322x platforms sector 0 is the sector where the bootloader is placed and it is protected by the bootloader even if it is set and sector 31 contains hardware parameters and is protected by the bootloader. For MC1322x platforms the application image will be placed starting with sector 1.

For S08 platforms the bootloader is placed starting with address 0xFC00, and the pages starting with this address are protected from platform internal flash configuration registers.

Even if protected sectors/pages exist, the entire image (containing the protected sectors/pages as well) must be downloaded to the server.

The OTAP demonstration specific information is described in Table 5-4.

Table 5-4. OTAP Demonstration Specific Information

Octets	Data Types	Name
1	Unsigned 8-bit integer	jitter
4	Unsigned 32-bit integer	delayUntilUpgrade
4	Unsigned 32-bit integer	crc

- The jitter is a value between 0x01 and 0x64. This value prevents flooding the server with simultaneous requests for new images from several clients. When receiving an ImageNotify indication from a server, the client should generate a random number between 0x00 and 0x64. If the generated number is less than or equal to the jitter provided by the server, the client is allowed to request the image.
- The delayUntilUpgrade value represents the amount of time in milliseconds that a client must wait until booting the new image after the over the air download process completes.
- The CRC field contains the 16 bit CRC-CCITT (0x1021) of the image, computed by TestTool 12.

The communication between the PC and the application uses a simple protocol with a frame format as described in Table 5-5.

Table 5-5. Uart frame format

Name	mStx	opcodeGroup	msgType	payloadLength	aPayload	checksum
Octets	1	1	1	2	variable	1
Description	Start of frame	Message group	Message type	Payload Length	payload	Frame checksum

OpcodeGroup values are 0xA3 for frames received from the PC application and 0xA4 for status responses sent to the PC application.

MsgType are used by the application to identify the commands received over the UART as follows:

- gOtapServerSetInternalFlashReq_c = 0x28 - Not used in this demo application

- gOtapServerSetStartImageReq_c = 0x29 - Informs the server application that the PC tool is starting the process for transferring an image to the server's external memory storage device.
- gOtapServerSetPushImageChunkReq_c = 0x2A - Informs the server application about a new chunk from the OTA file. In this demonstration application, the header and the sub-types fields come in one single UART packet to avoid extra processing on the server application node. The actual firmware image is split into the multiple UART packets.
- gOtapServerSetCommitImageReq_c = 0x2B - Informs the server application that the transfer process is complete.
- gOtapServerSetCancelImageReq_c = 0x2C - Cancels the transfer.
- PayloadLength represents the payload length provided in little endian format.
- The frame checksum is the xor value performed on all octets in the frame without mStx field.

The server application responds with a status message after each command received from the PC. In the case of success, the message type is the received command. In the case of an error, the message type value is gMessageError_c = 0xFE.

5.10.6 MultiImageFlashProgrammer - ARM7(MC1322x) Based Platforms

The MultiImageFlashProgrammer allows users to simultaneously load multiple images to the internal FLASH of the MC1322x MCU. In the OTAP's case, it can be used to load the client node with the original application and the bootloader. The MultiImageFlashProgrammer is located in the same folder as the Bootloader template application from where the codebase was exported. The MultiImageFlashProgrammer uses IAR C-SPY Command Line Utility and because of that, it requires that IAR Embedded Workbench Version 5.5 is installed on the PC.

To load the bootloader and the application to the client device, perform the following steps:

1. Overwrite the `application.out` file in the MultiImageFlashProgrammer folder with the `.out` file generated after building the client application. This will be the firmware that the client will receive.
2. Overwrite the `Bootloader.out` file in the MultiImageFlashProgrammer folder with the `.out` file generated after building the Bootloader application. This step is needed in case the bootloader template code provided in the codebase is modified.
3. Edit the `BootloaderAndApplicationLoader.bat` file to contain the correct paths to the location where the IAR IDE5.5 Embedded Workbench was installed on the PC.
4. Connect the Jlink JTAG connector to the client board and run the batch file.
5. The process is complete when the "Press any key to continue . . ." message appears.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x MC1323x, MC1322x.

5.11 ZID Controller Mouse Device Application

This application demonstrates the functionality of a mouse device. It exercises features from the BeeStack Consumer network, the ZigBee Input Device (ZID) profile and from the Push Button Pair layer. It interacts with users through a menu displayed on an UART terminal application, as shown in Figure 5-2:

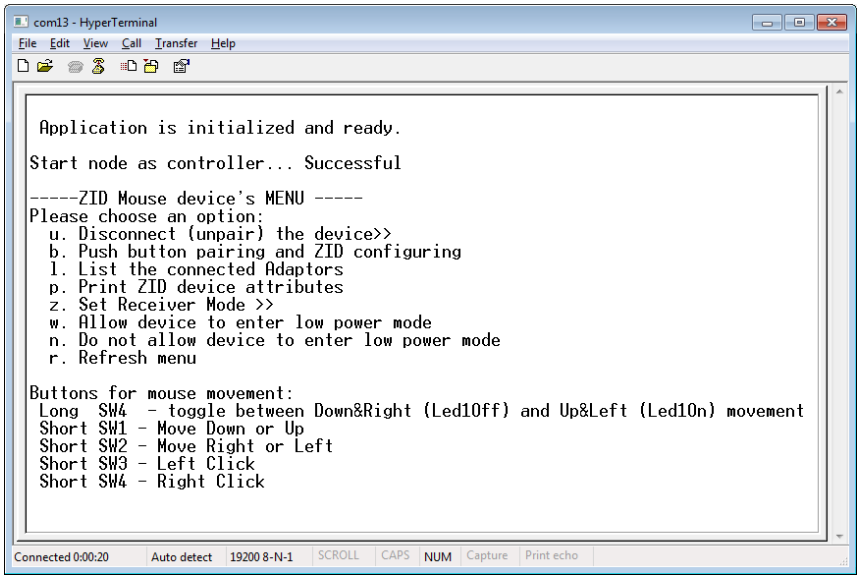

Figure 5-2. Mouse application menu

The application allows the user to perform the displayed actions and uses four keys (SW1 –SW4) on the board to simulate the mouse's movement or clicks. As shown in the displayed menu, each keypress causes an activity, such as "Move Down or Up" (short press SW1) or "Left Click" (short press SW3). When a short (less than one second) key press is performed, the device will send a mouse report over the air, signaling the user interaction to the remote node (the ZID Adaptor device). If "Left Click" or "Right Click" was pressed the application will also send a NULL report for click release. Performing a long press on SW4, the user can select the mouse movement: Down & Right or Up & Left. This will toggle LED1 on the same board, showing the selected movement: LED1 Off means Down & Right movement is selected, LED1 Off means Up & Left movement is selected.

NOTE

This application can be used in conjunction with the ZID Target Adaptor Device Application. See Chapter 3, "Starting and Running a Simple RF4CE Network" to learn how to use the menu options. This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

5.12 ZID Controller Keyboard Device Application

This application demonstrates the functionality of a keyboard device. It exercises features from the BeeStack Consumer network, the ZigBee Input Device (ZID) profile and from the Push Button Pair layer. It interacts with users through a menu displayed on an UART terminal application, as shown in Figure 5-3:

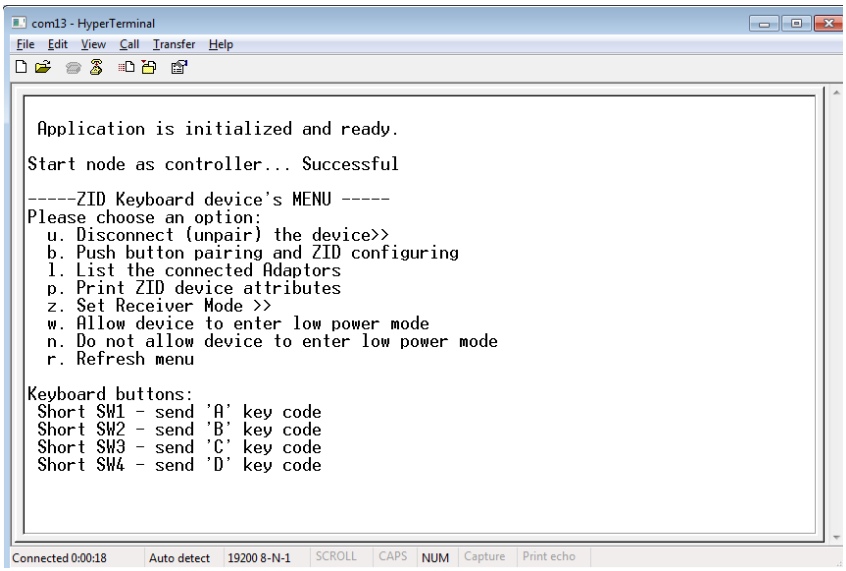

Figure 5-3. Keyboard application menu

As shown in the menu, the application allows the user to perform the displayed actions and uses four keys (SW1 –SW4) on the board to send some keyboard's key codes ("A", "B", "C" and "D"). When a short keypress is performed, the device will send a keyboard report over the air, signaling the user interaction to the remote node (the ZID Adaptor device). The application will also send a keyboard NULL report for key release.

NOTE

This application can be used in conjunction with the ZID Target Adaptor Device Application. See Chapter 3, "Starting and Running a Simple RF4CE Network" to learn how to use the menu options.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

5.13 ZID Target Adaptor Device Application

This application implements the functionality of an adaptor device. It exercises features from the BeeStack Consumer network, the ZigBee Input Device (ZID) profile and from the Push Button Pair layer. It interacts with users through a menu displayed on a UART terminal application, as shown in Figure 5-4:

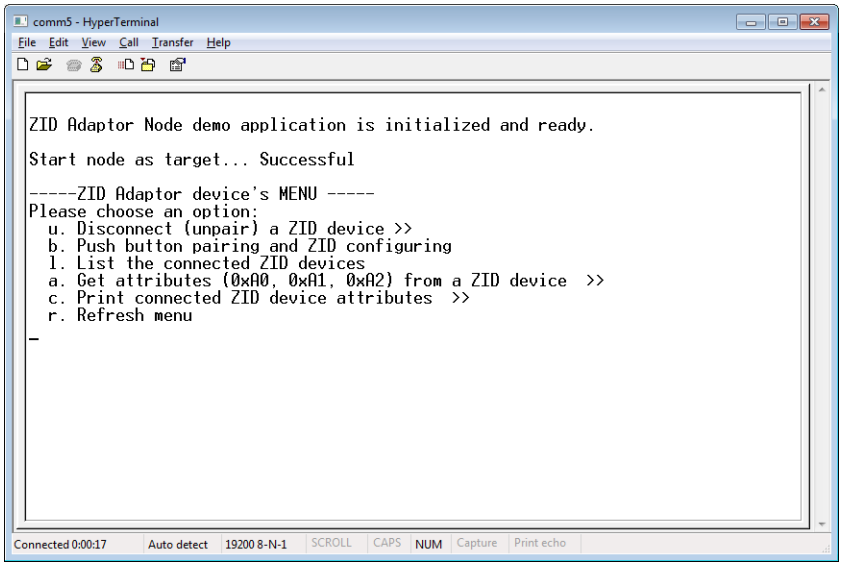

Figure 5-4. Adaptor application menu

As shown in the displayed menu, the application demonstrates the following ZID profile features:

- Connect a remote note using the push button pair process and ZID configuration stage.
- Disconnect (unpair) a remote node.
- Handle the received report data.
- Get some attributes from the remote node.

At this stage, this application doesn't interact with the USB class driver (refer to ZID Profile specification) to forward the received reports to a host (e.g. PC). It only handles the reports printing them over the UART.

NOTE

This application can be used in conjunction with the ZID Controller mouse device or ZID Controller Keyboard device applications. See Chapter 3, "Starting and Running a Simple RF4CE Network" to learn how to use the menu options.

This application is available on the MC1323x platform.

Chapter 6
BeeStack Consumer/SynkroRF Hybrid Sample Applications

The BeeStack Consumer/SynkroRF Hybrid sample applications, besides functionality provided by BeeStack Consumer network, the ZigBee Remote Control profile and the Freescale Private profile, provide SynkroRF network features. This allows the devices from different networks (RF4CE and SynkroRF) to establish connections using the pair procedure and communicate to each other. Each application is created and generated from BeeKit Wireless Connectivity Toolkit, selecting the BeeStack Consumer codebase, and choosing the project type BeeStack SynkroRF Hybrid Apps. To run the applications, users should export the solution, compile the projects and program the boards using the appropriate IDE.

6.1 Hybrid Controller Node Application

This application implements the functionality of a ZigBee RF4CE/SynkroRF hybrid controller node. It can exercise features from the BeeStack Consumer network, the SynkroRF network, the ZigBee Remote Control profile and the Freescale Private profile. This application can pair and exchange commands with a ZigBee RF4CE target device and a SynkroRF controlled device. It interacts with users through a menu displayed on a UART terminal application. The ZigBee Remote Control and Freescale Private Profile features used in the application are chosen at compile time. The application includes support for accessing the following profile features:

- ZigBee Remote Control – Controller-side Push Button Pair

This application implements the basic functions of a hybrid RF4CE/SynkroRF remote control. It allows the user to:

- Reset the controller node
- Start the controller node
- Pair with a RF4CE target node using Push Button Pair procedure
- Send and receive data to/from a paired RF4CE target node
- Pair with a SynkroRF controlled node using SynkroRF pair procedure
- Send and receive data to/from a paired SynkroRF target node
- Handle unpair requests

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

6.2 Hybrid Target Node Application

This application implements the functionality of a ZigBee RF4CE target node. It can exercise features from the BeeStack Consumer network, the SynkroRF network, the ZigBee Remote Control profile and the Freescale Private profile. This application can pair and exchange commands with a ZigBee RF4CE controller device and a SynkroRF controller device. It interacts with users through a menu displayed on a UART terminal application. The ZigBee Remote Control and Freescale Private Profile features used in the application are chosen at compile time. The application includes support for accessing the following profile features:

- ZigBee Remote Control– Target-side Push Button Pair

This application implements the basic functions of a hybrid RF4CE/SynkroRF target node. It allows the user to:

- Reset the target node
- Start the target node
- Pair with a RF4CE controller node using the Push Button Pair procedure
- Send and receive data to/from a paired RF4CE controller node
- Pair with a SynkroRF controller node using SynkroRF pair procedure
- Send and receive data to/from a paired SynkroRF controller node
- Handle unpair requests

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

6.3 Hybrid ZTC Node Application

This application allows an external host to access the BeeStack Consumer/SynkroRF hybrid stack features through a serial interface (SCI or I2C) and also monitors the communication between MAC, network, profile layers and application. The MAC, network, ZigBee Remote Control Profile and Freescale Private Profile features used are chosen at compile time. The application code includes support for accessing all the MAC, stack and profile features.

This application is available on the following platforms: MC1320x-QE96, MC1320x-QE128, MC1321x and MC1323x.

6.4 Hybrid Simple TV Application

This application implements the functionality of a ZigBee RF4CE/SynkroRF Hybrid TV node. It runs on a Freescale 13213-NCB board and uses the LCD to display information that a TV device would usually display on its screen. It works with the Freescale Virtual Remote application. The Freescale Virtual Remote application emulates the functionality of either a ZigBee RF4CE or a SynkroRF remote control through a PC application. The PC application communicates with a USB port connected to an MC13213 SRB board that is running a BlackBox device application. The ZigBee RF4CE/SynkroRF Hybrid Simple TV application uses the following profile features:

- ZigBee Remote Control – Target-side Push Button Pair
- ZigBee Remote Control – Sending and receiving commands

The Virtual Remote PC application and the binary for both the BeeStack Consumer and SynkroRF BlackBox applications it uses is found in the following location: `Program Files\Freescale\Virtual Remote`. To run the Virtual Remote application setup, perform the following tasks:

1. Connect the 13213-SRB board to the PC using a USB cable and load it with either the BeeStack Consumer or the SynkroRF BlackBox S19 formatted file from the Virtual Remote application folder. See the BeeStack Consumer Quick Start Guide for more information about loading binary images into Freescale boards.
2. Power on the 13213-SRB and follow the installation instructions as displayed in the Found New Hardware wizard on the PC. If the drivers do not automatically load, they can be found in the following directory: `C:\Program Files\Freescale\Drivers`
3. Load the 13213-NCB board with the Hybrid SimpleTV application image generated from BeeKit.
4. Reset both boards.
5. When the initialization of the 13213-NCB is complete, the text “TV off” appears on the 13213-NCB LCD. This shows that the TV application is started and that the TV is in standby mode and waiting for remote commands.
6. Launch the Virtual Remote PC application by clicking on Start ->Programs -> Freescale BeeKit -> Virtual Remote -> Virtual Remote. The Virtual Remote PC application appears.
7. Users must first pair the Virtual Remote application with the TV application that is running on the 13213-NCB as follows:
   a) Click on any of the soft keys of the Virtual Remote PC application. This starts the ZTC controller-side Push Button Pair process for a BeeStack Consumer BlackBox or a SynkroRF pair process if a SynkroRF BlackBox is used.
   b) If a Virtual Remote is using a BeeStack Consumer BlackBox, users should perform a long press of SW4 on the 13213-NCB board so the TV will start the target-side Push Button Pair process.
   c) If a Virtual Remote is using the SynkroRF BlackBox, do not perform any action on the SimpleTV, because it is already automatically accepting SynkroRF pair requests.
   d) After pairing is complete, the TV status appears on the Virtual Remote displaying the text “TV off”.
8. Click on the power button on the Virtual Remote application. The 13213-NCB displays the volume level and channel while the Virtual Remote PC application displays the TV status (On) and the volume level and channel.

9. On the Virtual Remote application, click any of the volume, channel or number buttons and watch their status change as displayed on theMC13213 NCB LCD and the Virtual Remote PC application display. Pressing the buttons on the 13213-NCB controls the Virtual Remote functions as follows:

- SW1 decreases volume
- SW2 increases volume

SW3 decreases channel number

- SW4 increases channel number
- If the TV is off, any switch press turns it on
- If the TV is on, a long press of SW1 turns it off

This application is available on the MC1320x-QE128 and MC1321x platforms.