SynkorRF™ Network Demonstration Application

User's Guide

Document Number: SYNKRORFNAUG
Rev. 2.0
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 6
Creating a SynkroRF BlackBox Binary

About This Book

The *SynkroRF Network Application User's Guide* explains how to install and run the SynkroRF Network applications included in Freescale's BeeKit Wireless Connectivity Toolkit for SynkroRF applications.

Audience

This user's guide is intended for use with the SynkroRF Network example applications in BeeKit. Anyone needing to demonstrate the applications or to become familiar with their behavior should read this guide. SynkroRF application developers should read this guide along with all other SynkroRF 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 SynkroRF network.

Chapter 2 Creating Projects – Describes how to create SynkroRF projects using BeeKit. As an example, Controller and Controlled Node applications are created

Chapter 3 Controller and Controlled Node Applications - Details how to use the Controller and Controlled Node applications.

Chapter 4 Wireless UART Application - Describes how to create and use the SynkroRF Wireless UART application.

Chapter 5 Over-the-air-programmer (OTAP) Controller and Controlled Node Applications - Describes the OTAP client and server functionality as exemplified in OTAP Controller and Controlled Node applications

Chapter 6 BlackBox Node Application - Details how to create a BlackBox downloadable application image using BeeKit.

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

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
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 SynkroRF 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
RCM	Remote Controller Board - a Freescale MC1323X based board
REM	Remote Extender Board - a Freescale MC1323X based board
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
SRB	Sensor Reference Board, a Freescale development board
SCI	Serial Communication Interface. This is a hardware serial port on the HCS08. With the addition of an external level shifter, it can be used as an RS232 port
SPI	Serial Peripheral Interface. This is a serial port intended to connect integrated circuits that are together on one circuit board
SSP	Security Service Provider, a ZigBee stack component
Stack	SynkroRF protocol stack
Toggle	A toggle switch moves from one state to its other state each time it is toggled. For instance, if the first toggle takes the switch to Off, the next toggle will be to On,

	and the one after that will be to Off again. In the applications that this document describes, the switches are momentary push buttons with no memory of their states. The HCS08 maintains each switch 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 ZigBee specification

References

The following documents were referenced to build this document.

1. Freescale *SynkroRF Software Reference Manual*, Document SYNKRORM.
2. The data sheets for the MC13193, MC13203, MC1321x, MC1323x radios
3. Freescale *MC9S08GB/GT Data Sheet*, Document MC9S08GB60, December 2004

Revision History

The following table summarizes revisions to this guide since the previous release (Rev. 1.0).

Revision History

Location	Revisions
Throughout	Minor changes to accommodate March 2 release.

Chapter 1
Introduction

The Freescale BeeKit Wireless Connectivity Toolkit includes a set of example applications for the SynkroRF feature set using the generic controller and controlled node applications, supplemented by the Wireless UART and OTAP 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 SynkroRF Codebases, and either CodeWarrior for the HCS08, version 10.1 for HCS08 based development or IAR Embedded Workbench version 5.20 for Codebases supporting only ARM7 MC13224/MC13225 based development or version 5.40 or 5.50 for codebases supporting both ARM7 MC13226 and MC13224/MC13225. They also require a P&E Multilink Background Debug Mode pod for the HCS08 or a JLink JTAG pod for ARM7 to program the development boards. This document assumes that all of these are correctly installed.

This user's guide assumes familiarity with the purpose and major features of SynkroRF 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.

1.1 What This Document Describes

This document explains how to:

- Get the example SynkroRF 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 SynkroRF to a platform that BeeKit does not already support
- Learn about SynkroRF.

Chapter 2
Creating Projects 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 SynkroRF network project and configure the individual devices.

1. Start BeeKit.
2. If another Codebase (MAC, SMAC or BeeStack) is selected, perform the following:
   a) Select File -> Select Codebase… or click the "Select Other Codebase..." link.
   b) Select the SynkroRF 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…
4. Select the SynkroRF Apps project type from the left side of the window.

5. As shown in Figure 2-1, 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: SynkroRF
   Location: BeeKitSolutions (sub directory on host PC)

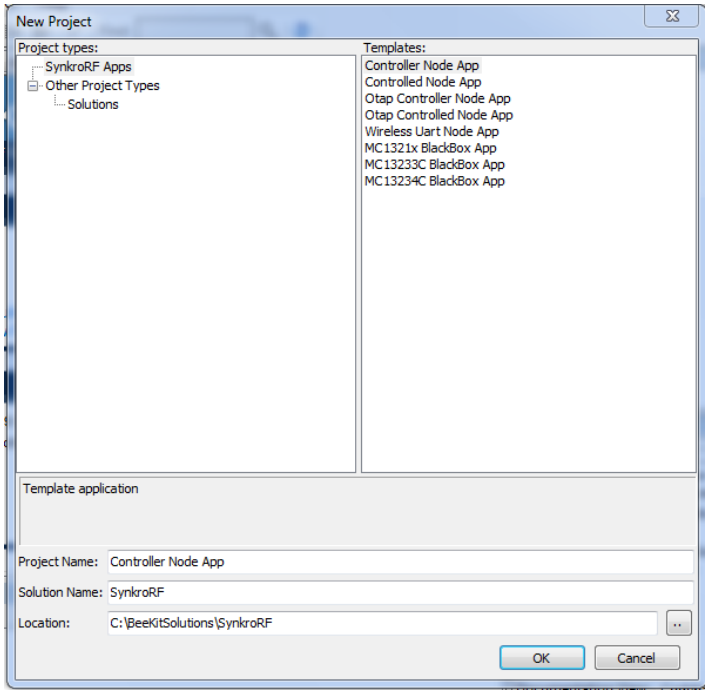

Figure 2-1. Project/Template Select

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 2-2.

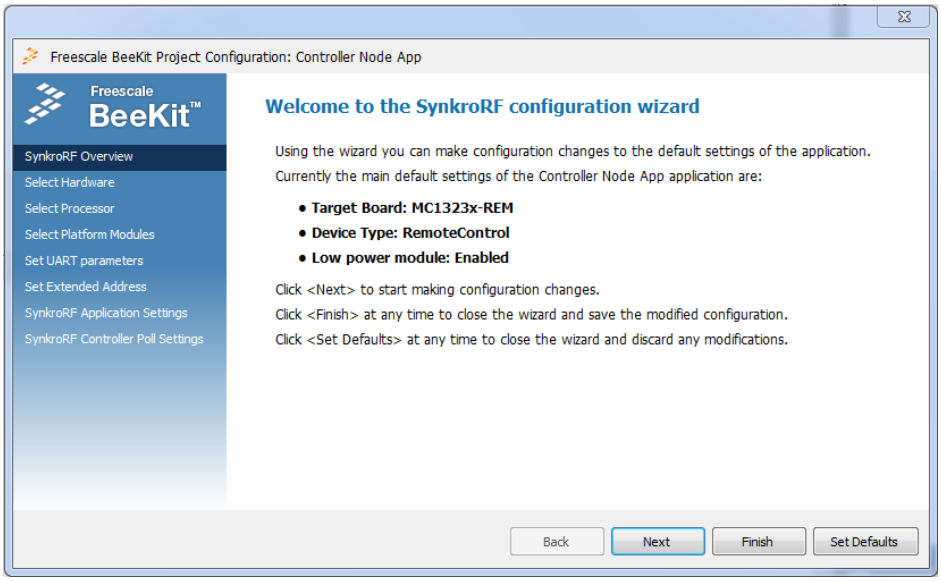

Figure 2-2. BeeKit Project Wizard Welcome Page

Review the current project settings. In this example, the board is a MC1323x REM, the node type is RemoteControl, and the low power module is enabled. This is the default setting for HCS08 SynkroRF codebase (The default settings depend on codebase platform and the project type).

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

- Click the Next button takes users on a tour of some common custom options. This choice is described in Section 2.1.2, “Custom Configuration Options”.
- If at any point the currently selected options plus the remaining options set to default are acceptable, pressing Finish closes the wizard and saves the configuration options.
- By pressing the Set Defaults button, all common options revert to their default values.
- After clicking the Finish button, users are presented with the complete list of BeeKit options. Each option can be modified here, even if it was already configured using the wizard.

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 2-3 for the ARM7 codebase version and Figure 2-4 for the HCS08 codebase version.

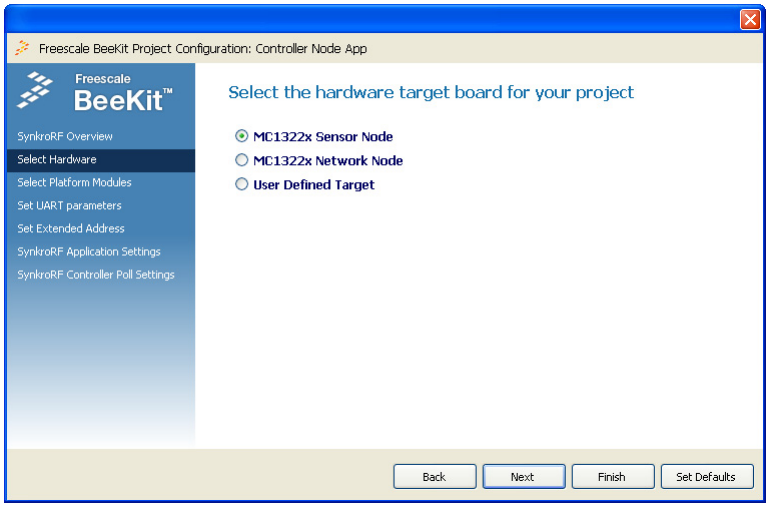

Figure 2-3. Hardware Target Page for ARM7 SynkroRF Codebase

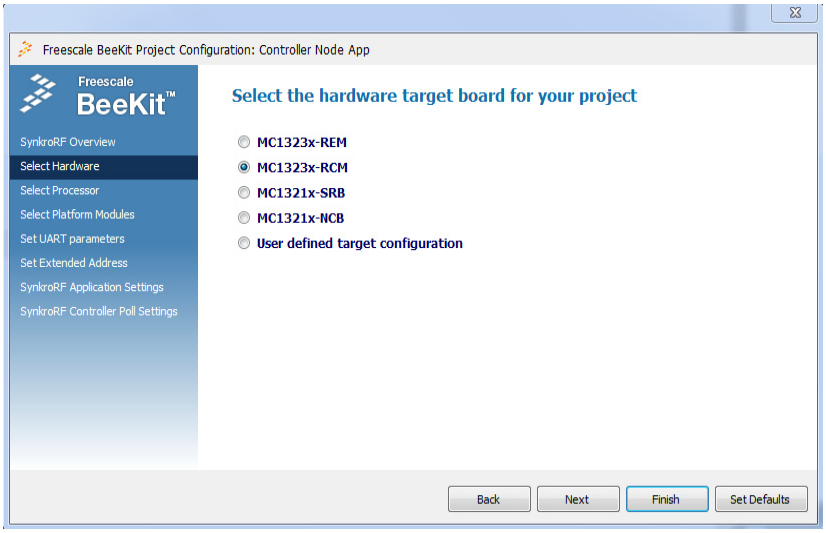

Figure 2-4. Hardware Target Page for HCS08 SynkroRF Codebase

1. Change the device to a specific platform. This example uses the MC1322x Sensor Node for ARM7 codebases and the 1323x-RCM for the HCS08 codebases.

2. Click on the Next button. In case of MC1323x boards the select processor page will appear as shown in Figure 2-5. For the boards different than MC1323x boards The Platform Modules page appears as shown in Figure 2-6.

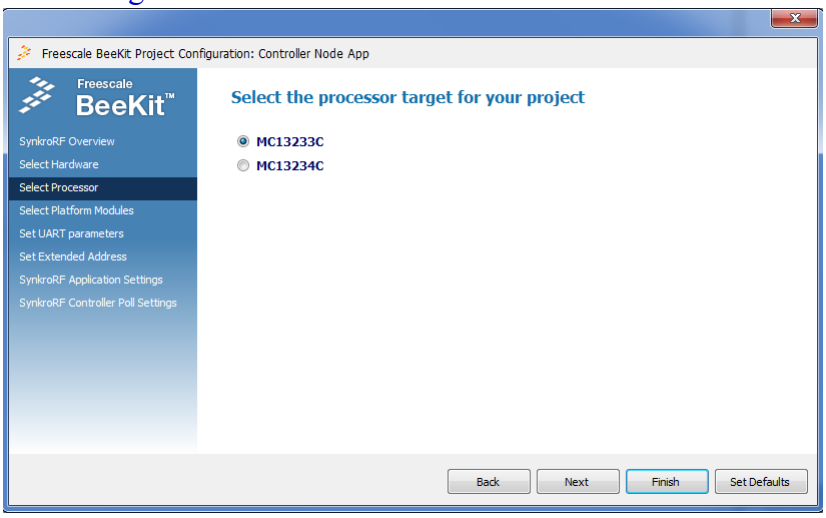

Figure 2-5. Select Processor Page

3. Select the processor target for your project, in this case MC13233C. Click on the Next button.The Platform Modules page appears as shown inFigure 2-6

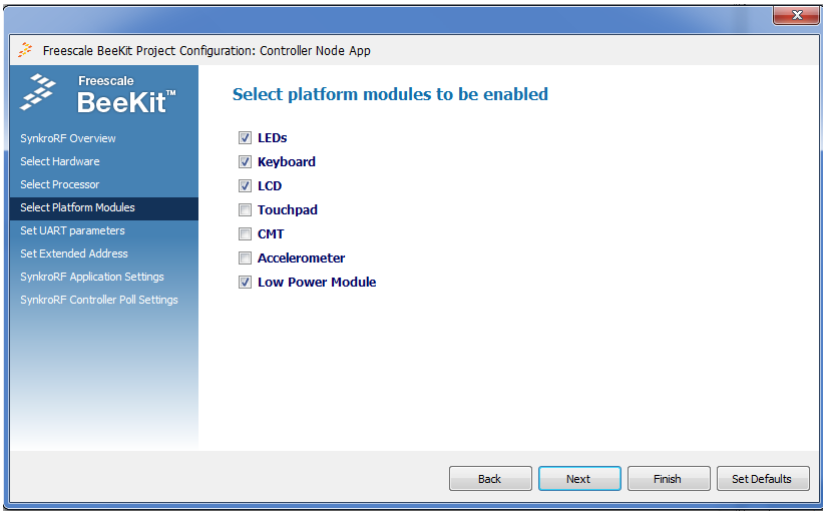

Figure 2-6. Select Platform Modules Page

4. Leave the default platform modules settings unchanged. LEDs and Keyboard modules on the board are enabled. Alternatively, users can enable the LCD, the SynkroRF application prints some information on it if it is enabled. Enabling the low power module allows the controller node to go into sleep node, if allowed by the application. Click Next to go to the UART Parameters page shown in Figure 2-7.

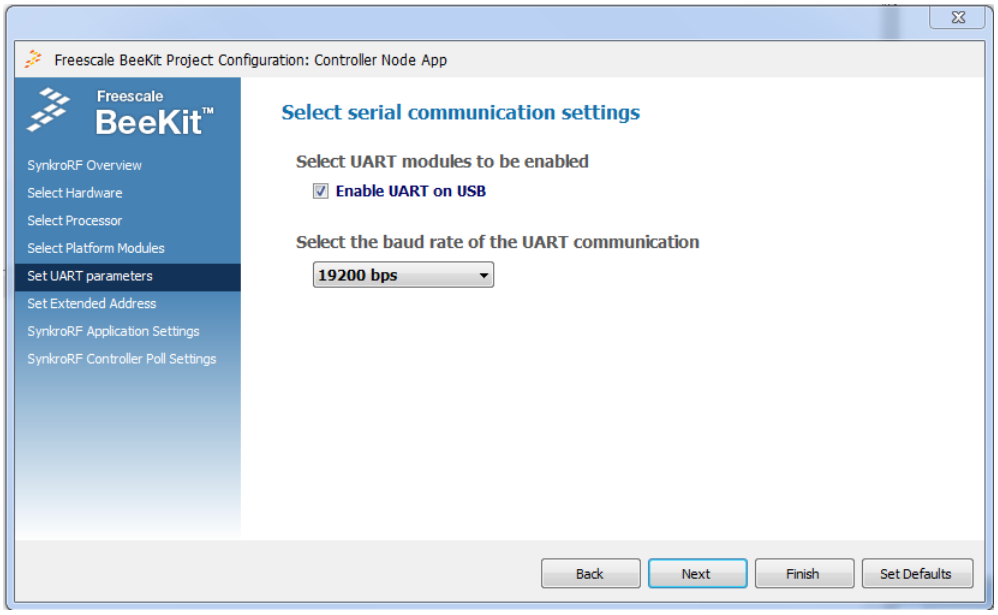

Figure 2-7. UART Parameters Page

6. Do not change the default UART settings. The UART module is enabled on the USB port of the board.
7. Click Next to go to the Extended Address configuration wizard page as shown in Figure 2-8.

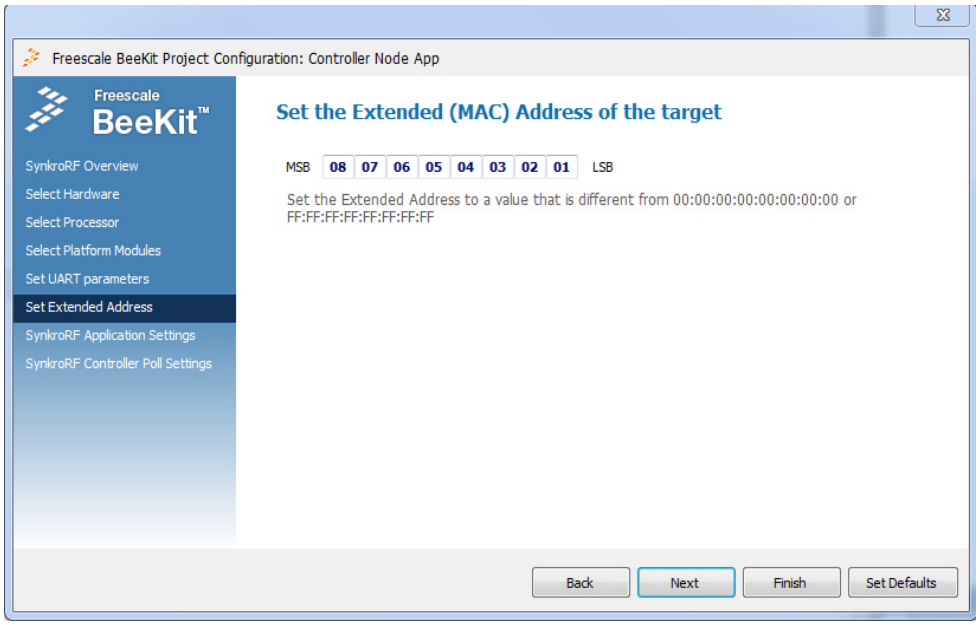

Figure 2-8. Extended Address configuration page

8. Users can leave the default address, or enter the full MAC address from the label on the development board.
9. Click Next. The SynkroRF Application Settings page appears as shown in Figure 2-9.

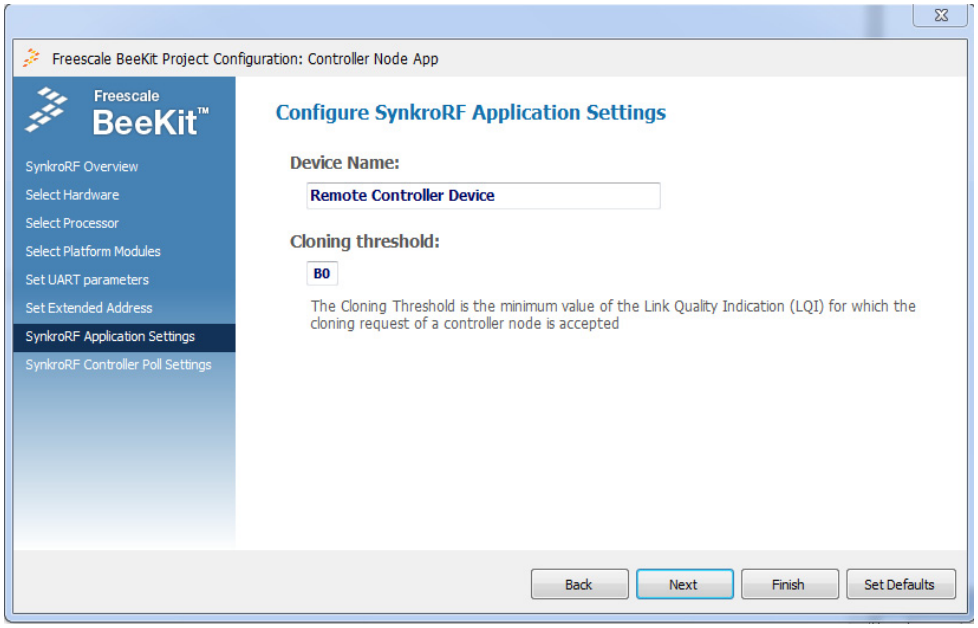

Figure 2-9. SynkroRF Application Settings page

10. Choose a name for the device (users can leave the default) and select a cloning LQI threshold.
11. Click Next to go to the SynkroRF Controller Poll Settings configuration wizard page shown in Figure 2-10.

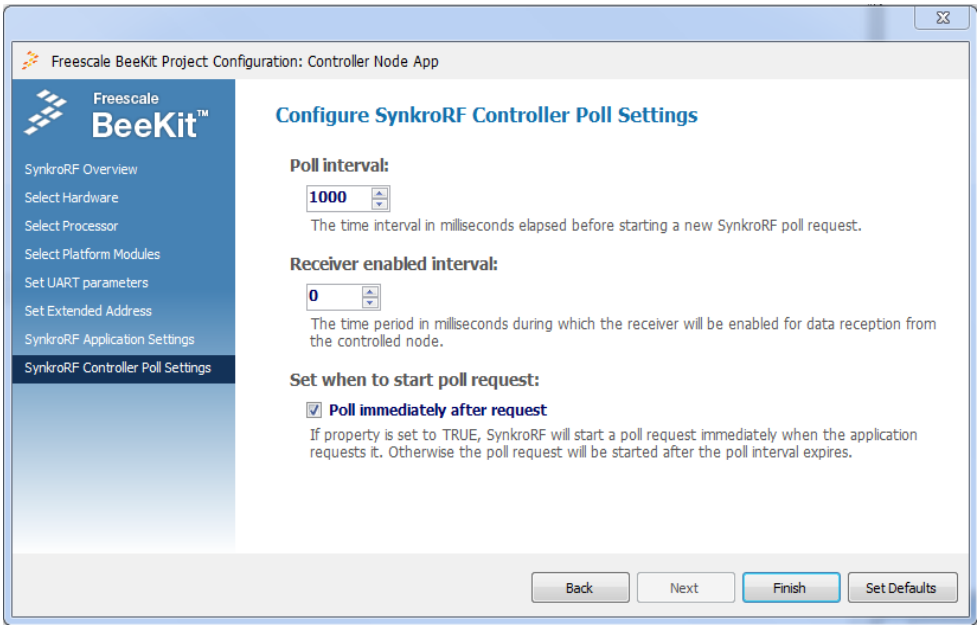

Figure 2-10. SynkroRF Controller Poll Settings page

12. Configure the poll interval, the receiver enabled interval and choose whether to start polling immediately after the poll request from the application.

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

2.1.3 Creating Additional Devices

A SynkroRF network consists of at least one controller node and at least one controlled node. Create additional devices by performing the following tasks:

1. From the BeeKit main menu, Select Solution -> Add Project…
2. This opens the Add Project window.
3. Select ZigBee Home Automation Applications. (The same applications selected for the coordinator.) However, this time, select the Controlled 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. Notice that the current settings for the switches for the controlled node are different than those for the controller node. In particular, the cloning threshold configuration screen is replaced with the pairing threshold configuration screen, as shown in shown in Figure 2-11.

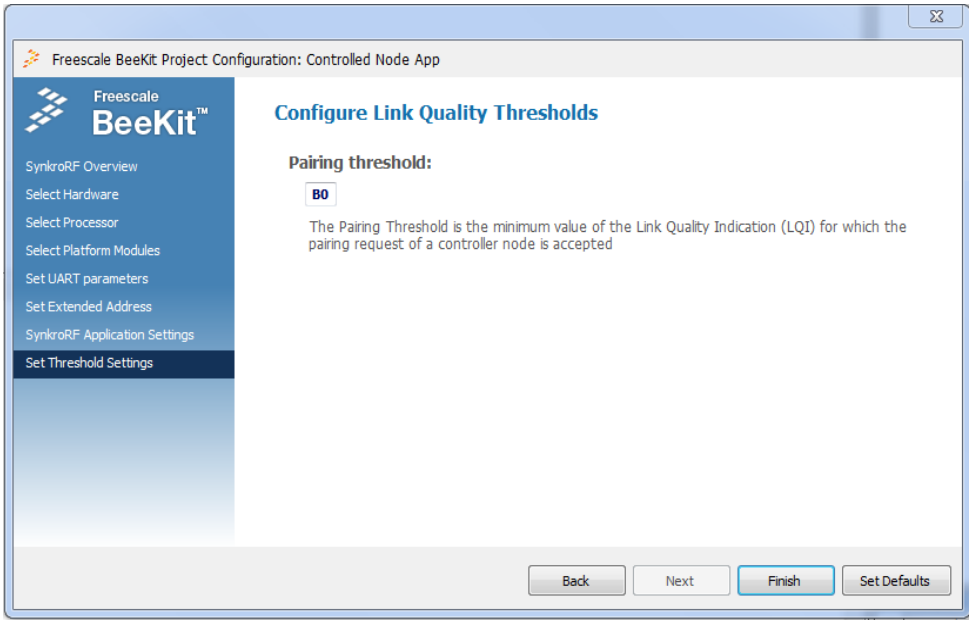

Figure 2-11. Pairing Threshold configuration window

5. After selecting Finish for the switch application, the BeeKit main window appears as shown in Figure 2-12.

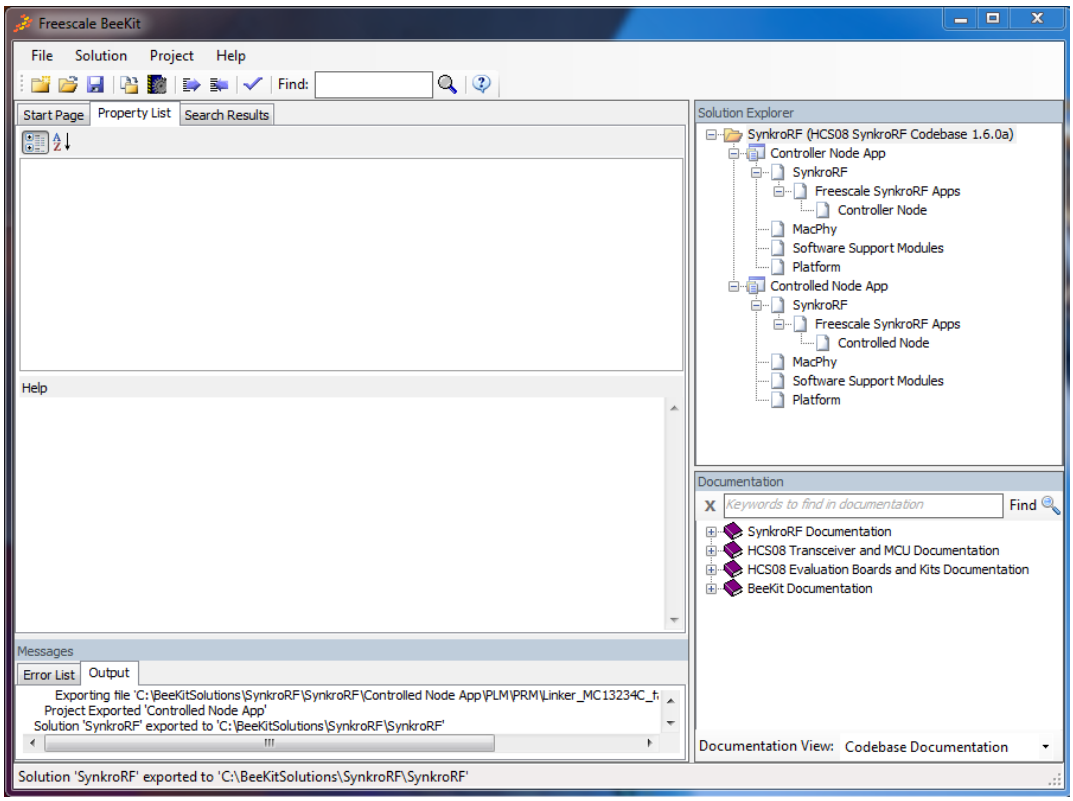

Figure 2-12. BeeKit Main Window

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

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

2.1.4 Exporting Created BeeKit Projects

After all the devices to be used in the network are created as BeeKit Projects and saved as a BeeKit Solution, the files must be exported into a format for importing into either the IAR Embedded Workbench or CodeWarrior IDE used for compiling and debugging. To export the saved solution, perform the following tasks:

1. From the Solution menu, select Export Solution. BeeKit verifies the internal consistency of the configuration, looking for such errors as two endpoints with the same number. If the verification succeeds, the window that opens displays all the created devices, each with a checked box as shown in Figure 2-13.

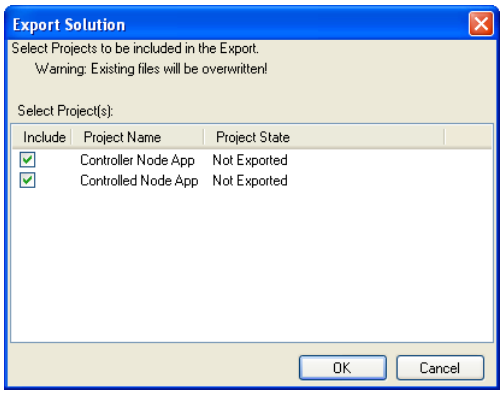

Figure 2-13. Export BeeKit Project Solution Window

2. Click on the OK button to start the export process. While BeeKit executes this, it displays the window shown in Figure 2-14, and the steps it takes scroll by in the Messages window pane.

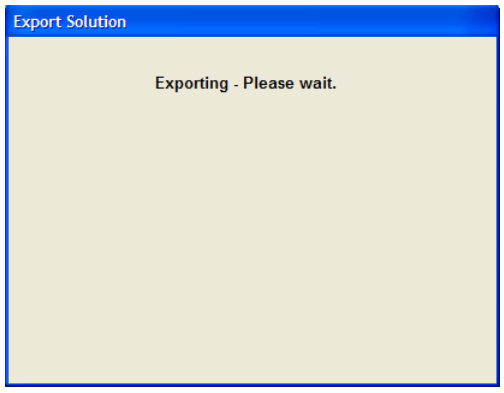

Figure 2-14. Please Wait – Exporting Project Window

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.

2.2 Importing Projects into and IDE

2.2.1 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 tool bar as shown in Figure 2-15 and export the solution in IAR EWB.

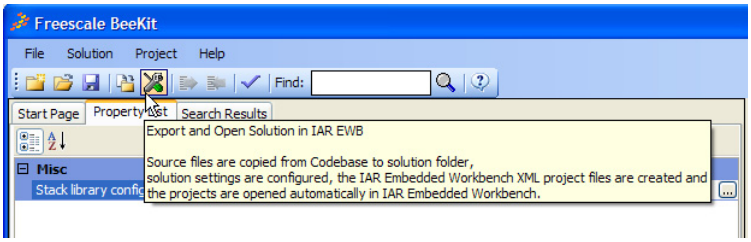

Figure 2-15. Exporting and Opening the Solution in Embedded Workbench

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 2-16 shows the projects of the solution previously created and imported into the IAR EWB.

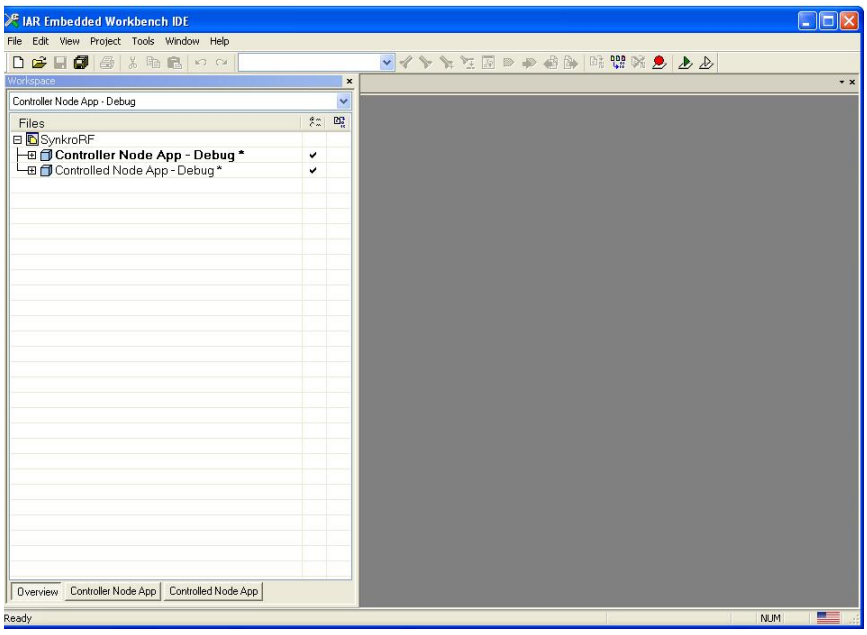

Figure 2-16. Solution Projects Imported in IAR Embedded Workbench

2.2.2 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 is 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` work space file created during solution export.

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

1. Start CodeWarrior and open the workspace.
2. Select File -> Import…The Import menu appears as shown in Figure 2-17.
3. Click Next.

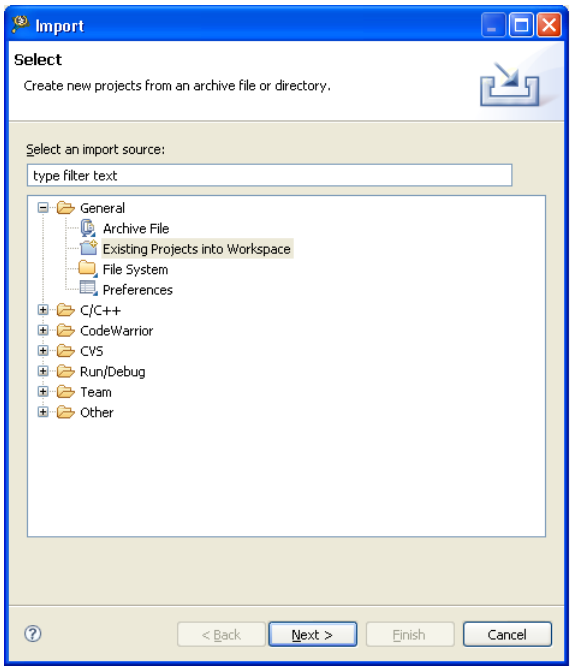

Figure 2-17. CodeWarrior Import Project…

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

```
C:\BeeKitSolutions\SynkroRF
```

5. CodeWarrior searches this folder and all subfolders for valid projects and displays the results as shown in Figure 2-18.

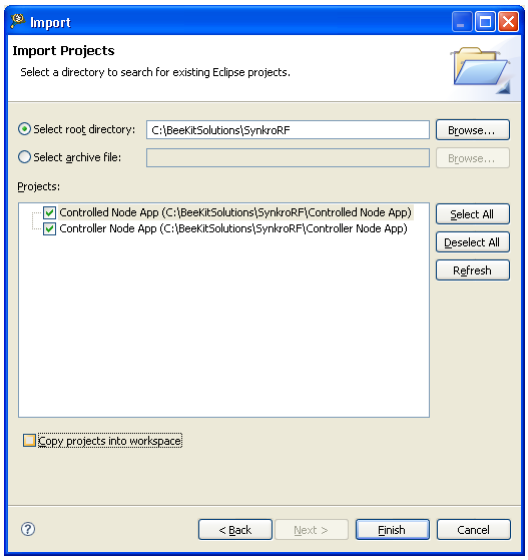

Figure 2-18. CodeWarrior Import Projects Window

6. Make sure that the “Copy projects into workspace” is unchecked and click Finish.
7. CodeWarrior opens the project files as shown in Figure 2-19.

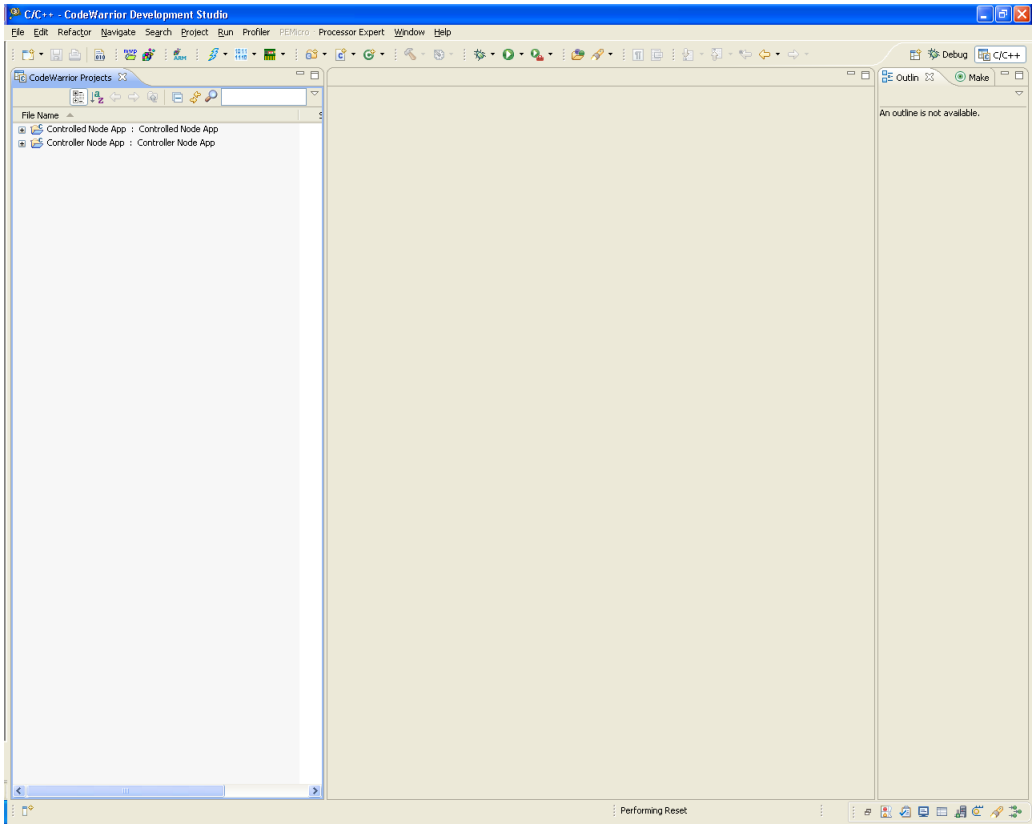

Figure 2-19. CodeWarrior Controller Node App Project Window

2.3 Building a Code Image Using the IDEs

2.3.1 Building a Code Image Using Embedded Workbench

To build the application binary in IAR Embedded Workbench perform the following tasks:

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 (if the selected project is already active this option is grayed out).
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 2-20.

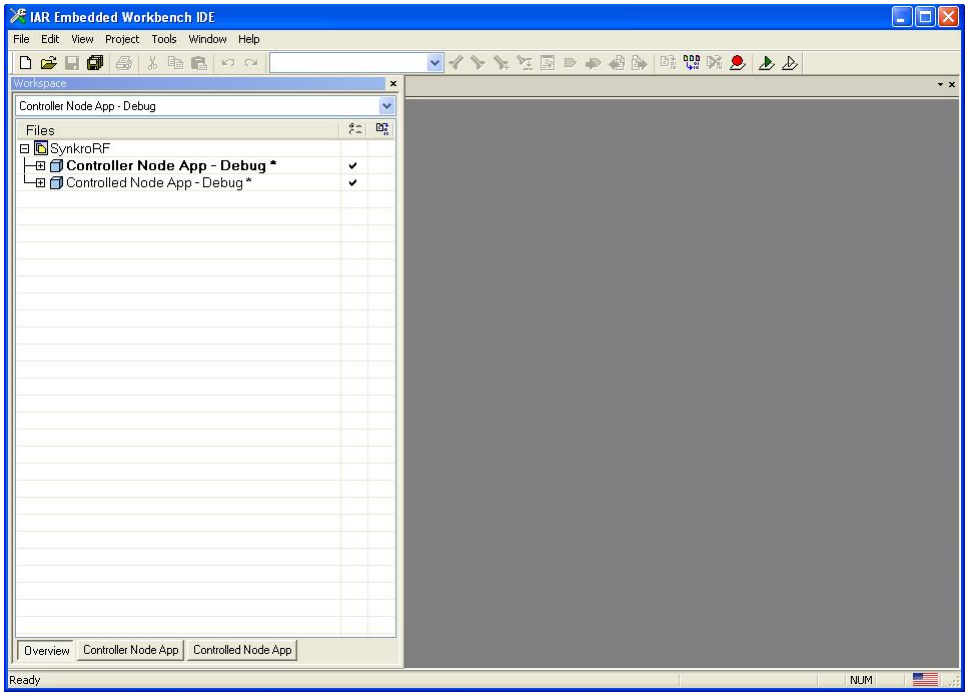

Figure 2-20. IAR EWB Showing the Build Messages Panel

2.3.2 Building a Code Image Using CodeWarrior

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

1. Right click on the target and select Build.
2. Click the “Build” icon.
3. From the menu select Project -> Build

2.4 Loading the Code Image to a ZigBee Device

2.4.1 Loading the Code Image to a MC1322x Board via JLink

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

1. Connect the JLink pod to the computer using a USB cable.
2. Turn on the MC1322x evaluation board.
3. Connect the JLink ribbon cable to the JTAG pins on the evaluation board. Align pin 1 of the JTAG port which is marked with a white dot with the blue ribbon wire of the JLink.
4. Download the compiled image to the board in one of three ways:
   a) Press the Debug hot key Ctrl+D.
   b) From the Project menu select Download and Debug (Embedded Workbench 5.20 (no later) or Debug (previous versions).
   c) Click the Debug icon in the main tool bar.
5. Wait until the board's FLASH memory is written. When this is complete, the IAR EWB debugging window appears as shown in Figure 2-21.

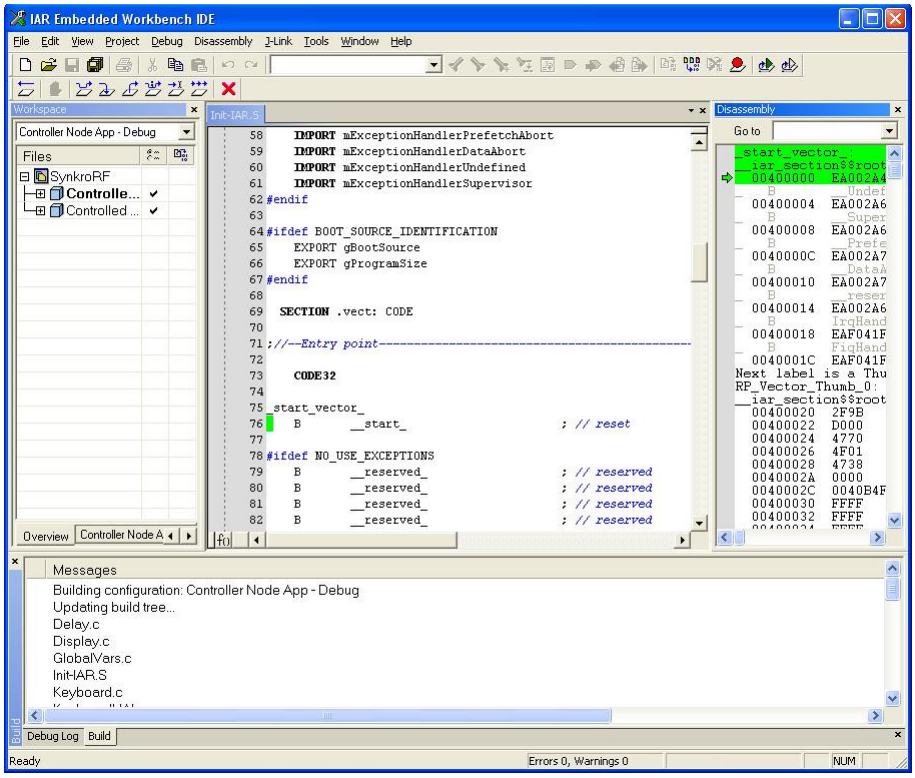

Figure 2-21. IAR EWB Debugging Window

7. Stop the debugger by pressing Ctrl+Shift+D or by choosing "Stop Debugging" from the Debug menu.

8. Right click on the Controller Node application in the Workspace Overview panel and select “Set as Active” from the context menu.
9. Repeat the steps as shown in Section 2.4.1, “Loading the Code Image to a MC1322x Board via JLink” to load the code for the other application into another board.

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

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

1. Connect the P&E BDM pod to the host computer using the USB cable. A lighted blue LED indicates the BDM has power and a successful USB connection.
2. Connect the BDM pod to the device. Align pin 1 of the BDM port connector with the red wire of the flat cable connector
   - 1321x-NCB — The connector is marked “BDM”, and pin 1 has a white dot beside it
   - 1320x-QE128EVB — The connector is labeled J9, and pin 1 is the one next to the label
   - 1323x-MRB — The connector is labeled J4 BDM and pin 1 is marked with ‘1’
3. Turn on the board.
4. The amber LED on the BDM lights 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. Download the compiled image to the board in one of two ways
   a) Click the “Debug” icon
   b) From the menu select Run -> Debug

   CodeWarrior opens the Debug view as shown in Figure 2-22.

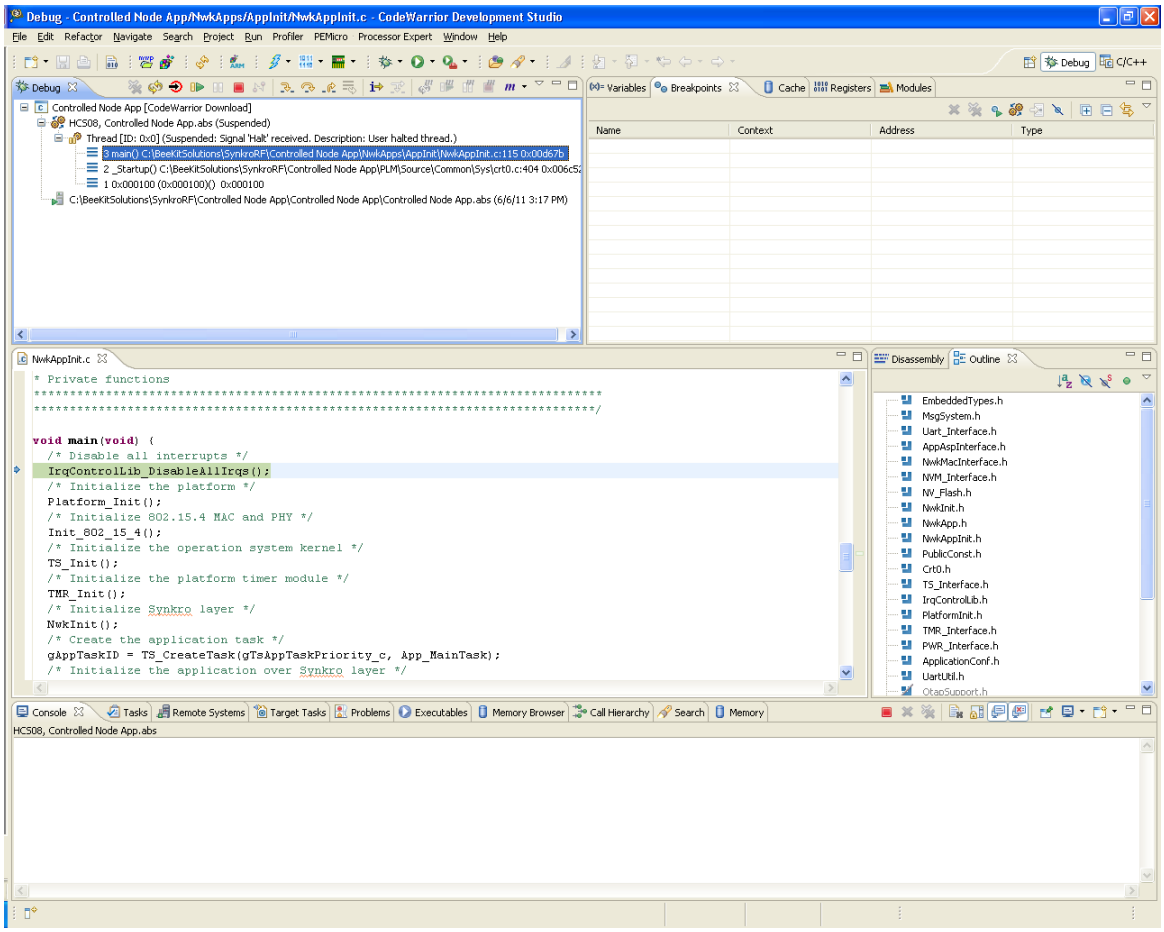

Figure 2-22. CodeWarrior Debug View

Chapter 3
Starting and Running a Simple SynkroRF Network

This chapter moves through the steps required to establish a small SynkroRF network using the two nodes as programmed in this guide. The network consists of the SynkroRF controller and the SynkroRF controlled nodes.

3.1 Starting the Network

1. Connect to the boards from the PC over UART using HyperTerminal or some other terminal emulator, one terminal window for each board.

   The baud rate used is the rate that was set in the BeeKit configuration wizard (default 19200), 8 data bits, no parity bits, 1 stop bit, no flow control. The COM port used depends on the devices present on the PC.
2. To determine the actual COM ports in use, open the Windows Device Manager, and under the Ports (COM & LPT) option, look for two devices labeled either “Freescale ZigBee/802.15.4 MAC COM Device” or “USB Serial Port”, as shown in Figure 3-1. (The COM ports shown in Figure 3-1 are different on every PC).

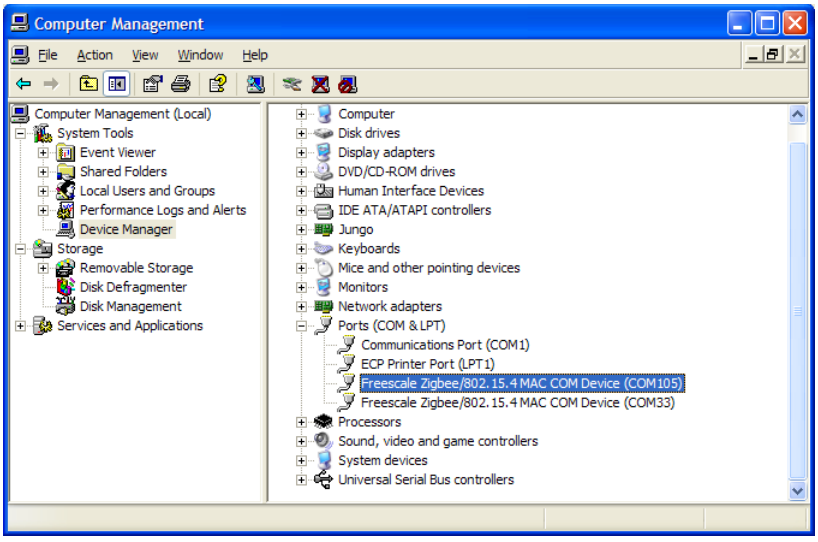

Figure 3-1. COM Ports in Device Manager

3. Reset both boards.

   After reset, the greeting message “Press a switch on board to start the application” should be displayed in each board’s terminal window as shown in Figure 3-2. The LEDs should also be flashing in a pattern.

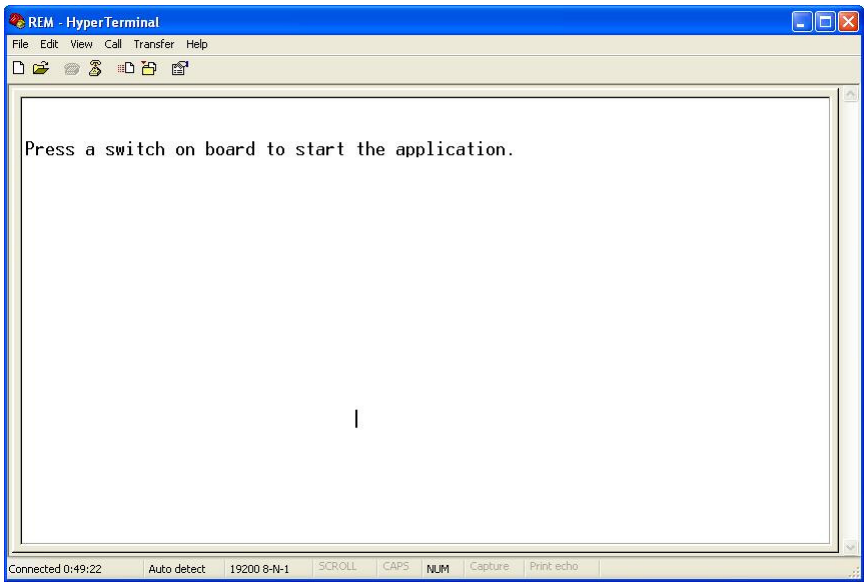

Figure 3-2. UART Greeting Message

4. Press any switch to display the main application menu.

   The menu should be printed on the terminal and the leds should have stopped flashing. If this did not happen, and users are employing MC1323x boards, check that the jumpers for the keyboard matrix buttons are set. The controller’s main menu is shown in Figure 3-3.

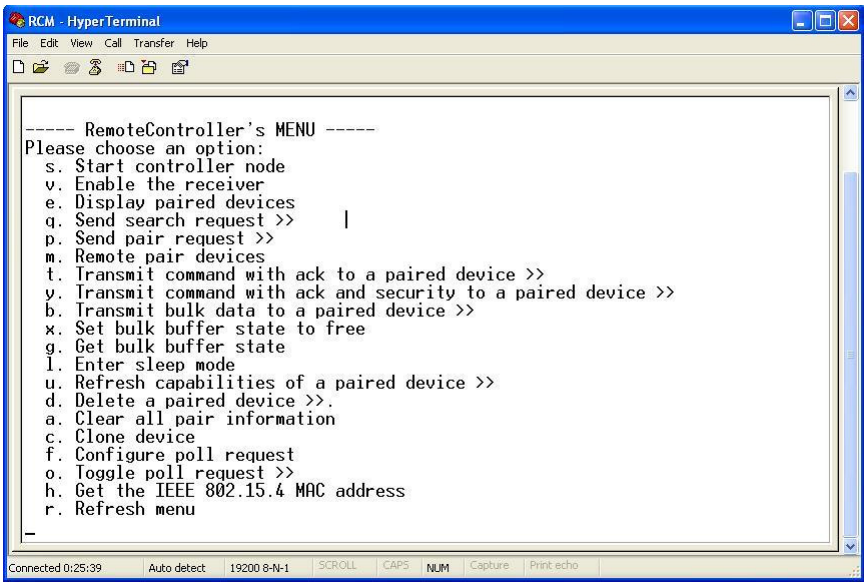

Figure 3-3. Controller Node Main Menu

The controlled node's main menu is shown in Figure 3-4

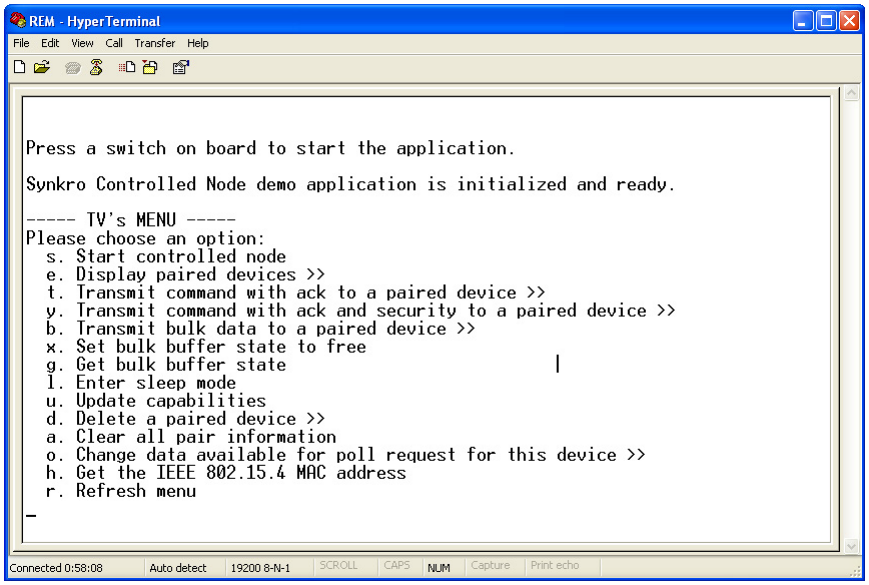

Figure 3-4. Controlled Node Main Menu (TV device type)

5. Each action in the menu has an associated key displayed just before the action. To perform a specific action, press its corresponding key for the appropriate device's terminal window.
6. Start both devices by pressing the "s" key in both windows.
7. To pair the remote controller with the TV, press the "p" key on the controller terminal, then press "1" to pair with a TV device. If the boards are close enough (depending on the pairing LQI threshold set for the controlled node in BeeKit), the boards pair.

3.2 Exchanging data

1. To transmit a command to another device, press the "t" key, then select the receiver device from the list of paired devices. Each of the nodes in the example network is paired with only one other device, so press "1".

The menu also allows users to access other functionality of the SynkroRF stack. Some of these options may not be available, depending on the previously set BeeKit configuration options. For example, pressing "l" to enter sleep mode displays the line "Low power module not included", if LPM support was not configured in BeeKit.

Chapter 4
Wireless UART Application

This chapter shows how to create a SynkroRF Wireless UART application using the Freescale BeeKit Wireless Connectivity Toolkit.

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

The Wireless UART network consists of two nodes, a controller device and a controlled device, which are pre-paired. No pairing is done when the application is started, the application starts directly in transmission mode.

The following scenario is based on using a HCS08 SynkroRF codebase.

4.1 Application Setup

4.1.1 Creating the Wireless UART BeeKit Project

1. Open BeeKit and select the SynkroRF codebase.
2. Select the SynkroRF Apps from the left side of the window, and Wireless UART app template from the right side of the window, as shown in Figure 4-1

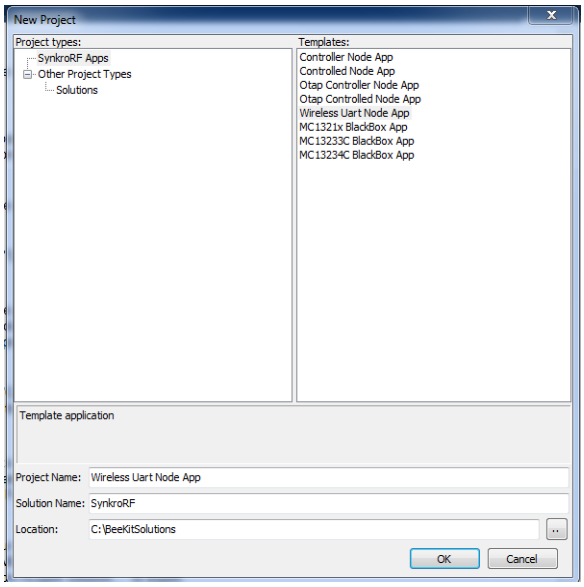

Figure 4-1. BeeKit Wireless UART App Template

3. Name the project “Wireless UART Node Controller App”.
4. Configure Wireless UART Node as “RemoteControl” SynkroRF Device Type.
5. Add a second project to the solution (see Section 2.1.3, “Creating Additional Devices”). This is also the Wireless UART App as shown in Step 2.
6. Name the project “Wireless UART Node Controlled App”
7. Configure Wireless UART Node as “TV” SynkroRF Device Type, as shown in Figure 4-2

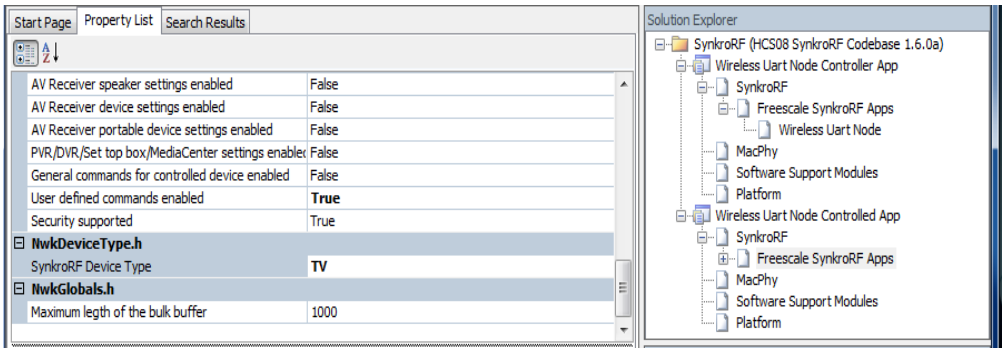

Figure 4-2. Configure Wireless Uart Node as TV

8. Export the solution, as shown in Section 2.1.4, “Exporting Created BeeKit Projects”

4.1.2 Building the code images and loading them into the boards.

This process is identical to that described in Chapter 2, “Creating Projects with BeeKit”. See Section 2.2, “Importing Projects into and IDE”, Section 2.3, “Building a Code Image Using the IDEs” and Section 2.4, “Loading the Code Image to a ZigBee Device”.

4.2 Wireless UART Setup and Operation

The following sections show how to identify and setup the UART/USB virtual COM ports, set up HyperTerminal or some other terminal emulator, start the Wireless UART application, form the network, and use the Wireless UART application.

4.2.1 Setting up the UART/USB Virtual Com Ports

Connect two UART terminals to both boards as described in the beginning of Section 3.1, “Starting the Network”.

4.2.2 Using the Wireless UART Application

Once the Wireless UART application is loaded to boards and two terminal windows are open to each of the boards, the Wireless UART application is ready to use.

1. Type some text in one of the HyperTerminal as shown in Figure 4-3.

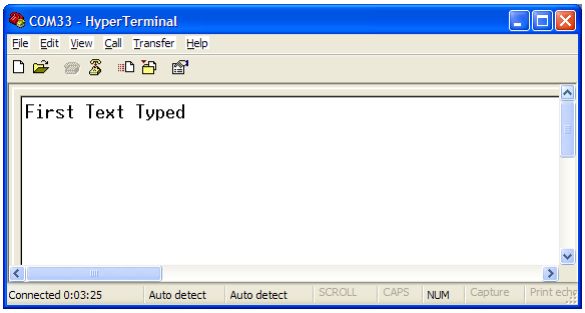

Figure 4-3. Text Typed in First HyperTerminal Window

2. The typed text is displayed as output in the second HyperTerminal as shown in Figure 4-4

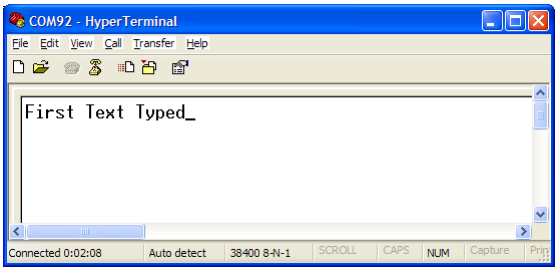

Figure 4-4. Text Echoed in Second HyperTerminal Window

3. Pressing any key on a board places that board into sleep mode. A second key press wakes it up. When a device is sleeping, LED 1 is on.

Chapter 5
Over the Air Programmer (OTAP) Application

The OTAP Controller and Controlled Node applications are an enhancement of the standard generic Controller and Controlled Node applications designed to showcase the OTAP functionality within SynkroRF. 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 Controlled 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.

The controlled application does not use menus displayed on a UART terminal console. This is because the controlled application interacts over the UART interface with the OTAP module of the Test Tool 12 PC application as described in Section 5.5, “OTAP Module from Test Tool 12”. User interaction is accomplished using switches 1 and 2 on the board itself.

The client receives over the air firmware updates from the server and stores the received image in its external memory storage device. The received image overrides the current image with the help of a bootloader application after a reset of the client device.

When the server application is up and running, it receives client firmware images from Test Tool 12 over the UART interface and places them in its external memory storage device. The OTAP Controlled Node application provided as a server demonstration supports storing one single client image. The server can service multiple clients, downloading the images all at the same time.

The OTAP Controller and Controlled Node applications are interfaced with platform through OtapSupport.h

NOTE

1. OTAP functionality is available on the following platforms: MC1321x and MC1323x.
2. The OTAP Controller Node demo application supports one single upgrade image in the external memory storage device.
3. 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.

5.1 OTAP Process Flow

Figure 5-1 shows the OTAP process flow and the frames exchanged over the air. The process proceeds in several steps as follows:

1. The server sends out Image Notify messages, announcing the availability of a new image for download. This step is optional.
2. The client requests a new image and includes its hardware version and its current firmware version in the request frame.The server analyzes the request, decides whether is has a suitable image for the client or not and sends an appropriate response.
3. If the server has indicated in Step 2 that an upgrade image is available, the client requests transmission of the image, block by block. The server is stateless and the client always requests a block of a maximum size from a specified offset and the server responds with that particular block. The client writes each received block into external memory.
4. Once the client has received the entire image it notifies the server and the server provides and acknowledgment. The server's response includes a delay value in milliseconds. The client should not reboot with the new firmware image until the specified delay time has passed.

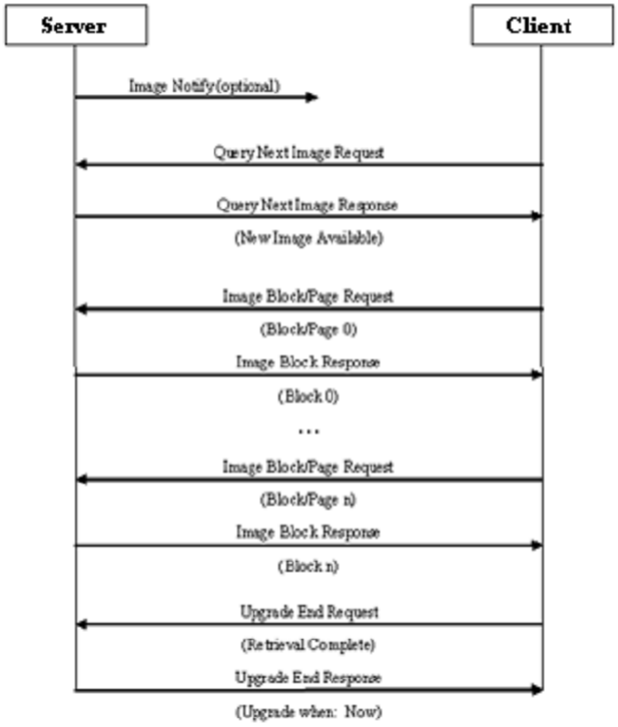

Figure 5-1. OTAP Process Flow

5.2 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 functionality:

- Check at startup if an upgrade image is present in the external memory storage device;
- Copy the upgrade image in the internal flash memory and preserve pages that are specified in the image bitmap;
- Unset 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.

5.3 Application setup

5.3.1 Creating the OTAP Application BeeKit Project

1. Open BeeKit and select the SynkroRF codebase.
2. Select the SynkroRF Apps from the left side of the window, and OTAP controller template from the right side of the window, as shown in Figure 5-2.

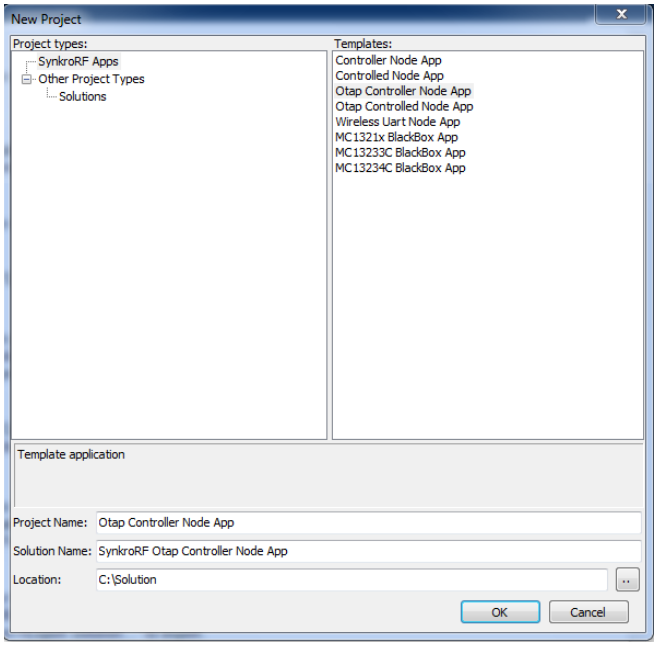

Figure 5-2. OTAP Controller Node application template

3. Name the project “Controller App”.
4. Follow the steps in the configuration wizard. The steps are identical to those described in Chapter 2, “Creating Projects with BeeKit”, Section 2.1.1, “Basic Options” and Section 2.1.2, “Custom Configuration Options”.
5. Add a second project to the solution (see Section 2.1.3, “Creating Additional Devices”). Select the SynkroRF Apps from the left side of the window, and OTAP Controlled app template from the right side of the window, as shown in Figure 5-3.

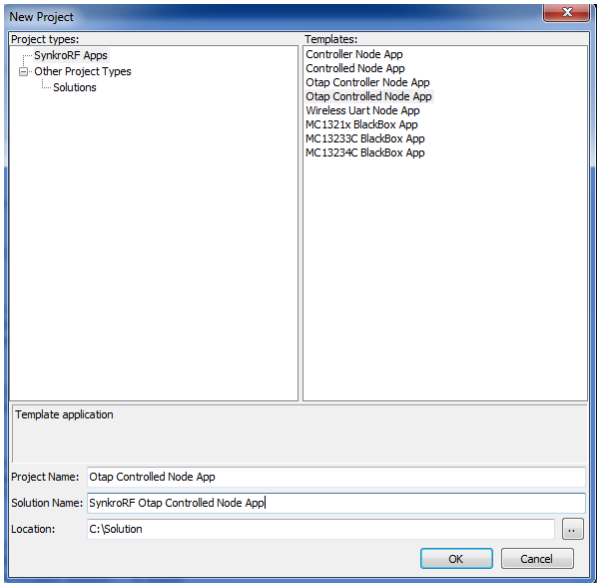

Figure 5-3. OTAP Controlled Node application template

6. Name the project “Controlled app”.
7. Export the solution as shown in Chapter 2, “Creating Projects with BeeKit”, Section 2.1.4, “Exporting Created BeeKit Projects”.

5.3.2 Building the Code Images and Loading to the Boards

This process is identical to that described in Chapter 2, “Creating Projects with BeeKit”. See Section 2.2, “Importing Projects into and IDE”, Section 2.3, “Building a Code Image Using the IDEs” and Section 2.4, “Loading the Code Image to a ZigBee Device”.

5.4 Running the OTAP Application

The OTAP Controller application is almost identical to the generic application described in Chapter 3, “Starting and Running a Simple SynkroRF Network”. The difference is the “Query next image” command was added which is accessed using the “j” key. This command starts the OTAP process.

User interaction with OTAP Controlled Node is accomplished using switches 1 and 2 on the board and with Section 5.5, “OTAP Module from Test Tool 12”.

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

1. Connect both boards to two Pc USB ports, build applications and load the boards.
2. Connect OTAP Controller node to a HyperTerminal or some other terminal emulator, and start application.
3. Start OTAP Controlled Node application by short pressing any switch on the board, and after few seconds press short switch 1.
4. Pair the devices from controller menu.
5. Load the updated image to the external FLASH of the server node as described in Section 5.5, “OTAP Module from Test Tool 12”. At the end of a successful download, the client is notified, over the air, about the new image.
6. The client may query for a new image by pressing ‘j’ key. Select the target paired node to copy the image from. A status bar displays the transmission progress on both nodes. Figure 5-4 shows the progress on OTAP client.

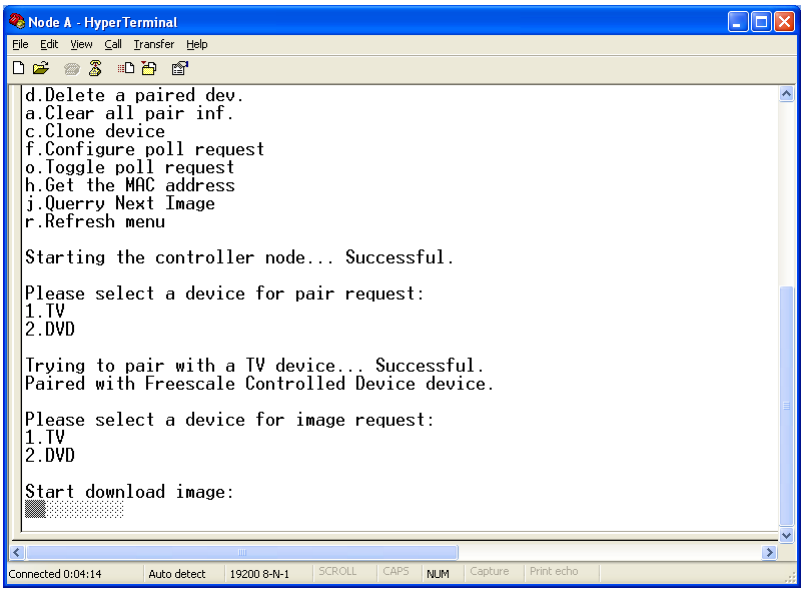

Figure 5-4. OTAP Client New Image Download

OTAP Client saves the received program image in its external EEPROM.

7. 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.

5.5 OTAP Module from Test Tool 12

The OTAP module in Test Tool 12 (Figure 5-5) 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 assumes that a development board supported in codebase is used, all this boards contains a memory storage device (EEPROM, FLASH) and drivers are implemented in platform component.

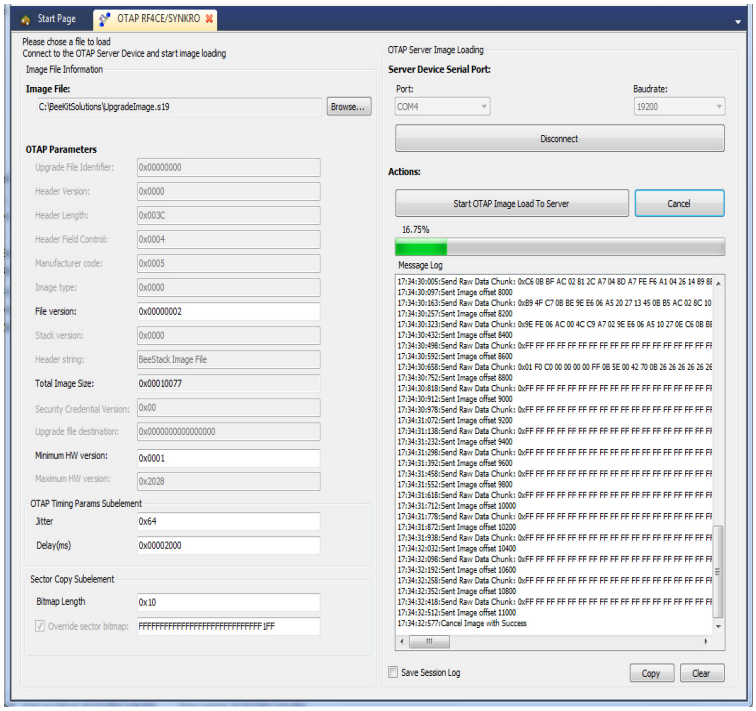

Figure 5-5. OTAP Graphic User Interface from Test Tool 12

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

1. Browse for the.S19 image file.
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 -> Each bit in bitmap corresponds to one page from MCU internal flash.This pages are 512 bytes length for MC1321x and 1024 bytes length for MC1323x.
7. Set the override sector bitmap. The right most bit in the bitmap corresponds to sector 0 (the one where the bootloader is placed) while the left most bit corresponds to last internal flash sector (page for S08). A 1 bit means that the corresponding sector 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 Test Tool 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 Test Tool 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 OTAP applications 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
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 cannot be configured 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.S19 image files, the bitmap of the internal FLASH pages not to be upgraded, and 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 pages content must be updated in the internal FLASH. In the Test Tool 12 GUI, the right most bit of the bitmap corresponds to first page and the left most bit corresponds to the last internal flash page. If the corresponding bit is set to '1', then the sector is erased.

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

Even if protected pages exist, the entire image (containing the protected 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 Test Tool 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

Table 5-5. UART Frame Format

Octets	1	1	1	2	variable	1
Description	Start of frame	Message group	Message type	Payload Length	payload	Frame check-sum

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.

12. 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.

Chapter 6
Creating a SynkroRF BlackBox Binary

This chapter shows how to create a SynkroRF BlackBox binary using the Freescale BeeKit Wireless Connectivity Toolkit and how to download the image to the target boards.

The SynkroRF Black box Reference Manual Provides further details about host interface, protocol framing and command formatting.

6.1 SynkroRF BlackBox Binary Generation

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

1. Start the BeeKit Wireless Connectivity Toolkit.
2. Select the appropriate SynkroRF codebase.
3. From the menu, select File->New Project… The New Project window appears as shown in Figure 6-1.

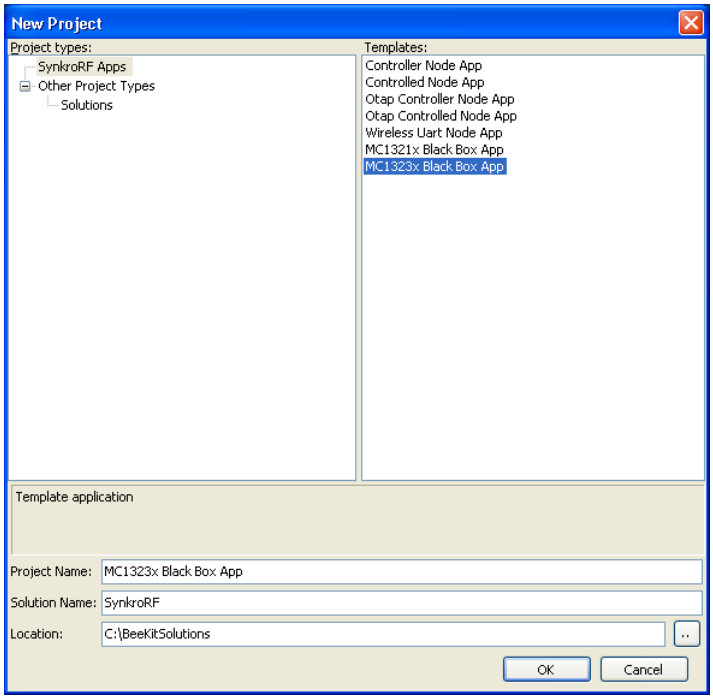

Figure 6-1. BeeKit New Project Window

4. From the left side of the New Project, select SynkroRF Apps (Figure 6-1).

5. Select the appropriate black box template for the board from the right side of the window. This example uses an MC1323x board.
6. Enter a name, solution location and project name at the bottom of the New Project window.
7. Click OK to create the project.

   The New Project window closes and the SynkroRF Configuration window appears as shown in Figure 6-2.

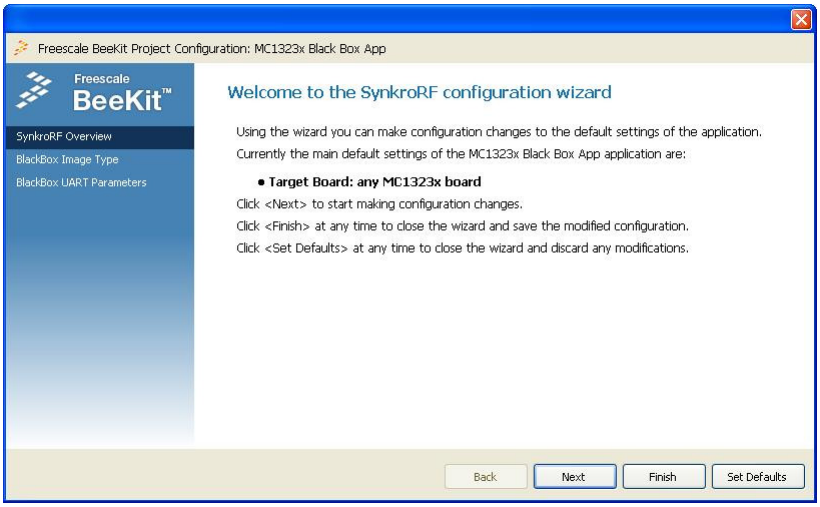

Figure 6-2. SynkroRF Configuration window

8. Click on the Next button. The SynkroRF BlackBox Image Type window appears as show in Figure 6-3.

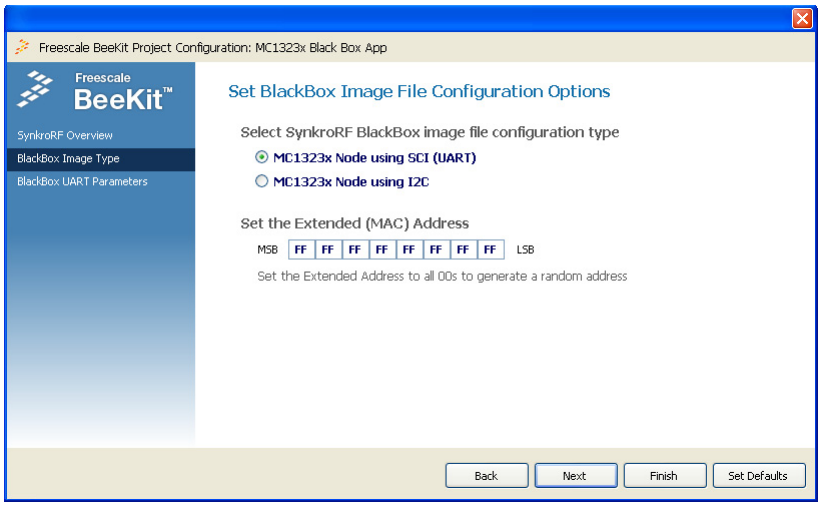

Figure 6-3. SynkroRF Black Box image type window

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

If a UART based image was selected, the connection settings window allow users to set the UART baud rate, as shown in Figure 6-4.

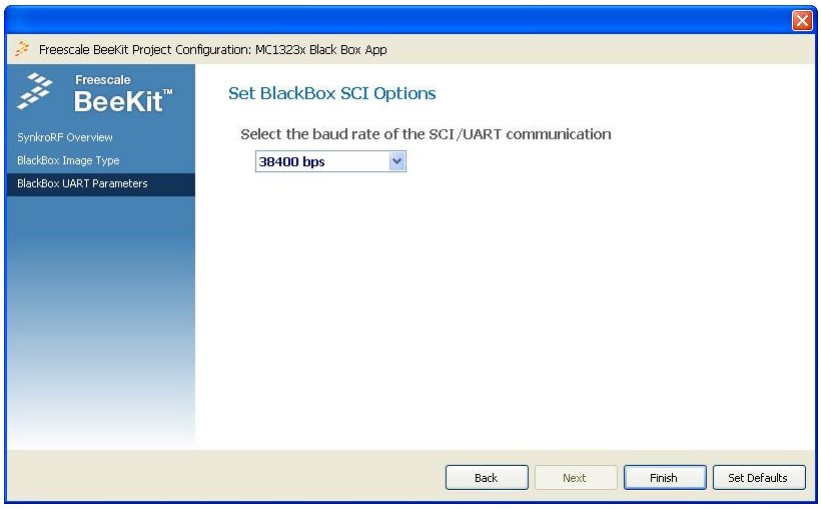

Figure 6-4. UART Settings Window

If an I2C interface is selected, the connection settings window allows setting the I2C slave address, as shown in Figure 6-5.

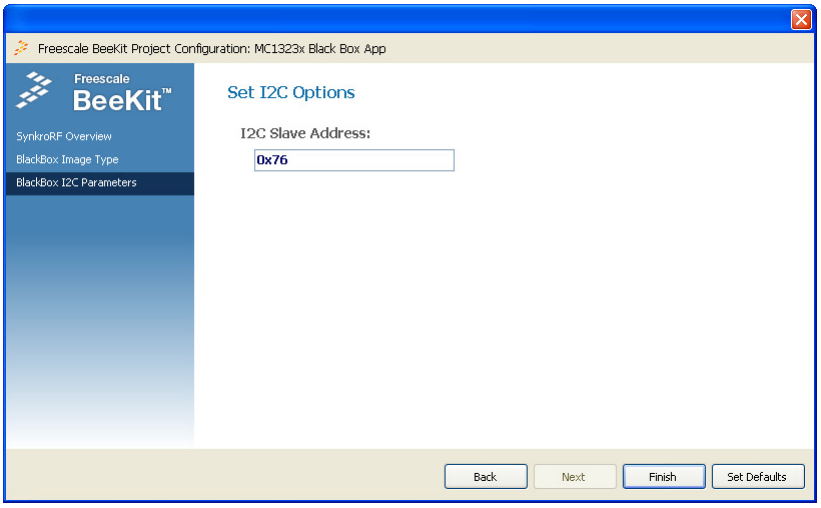

Figure 6-5. I2C Settings Window

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

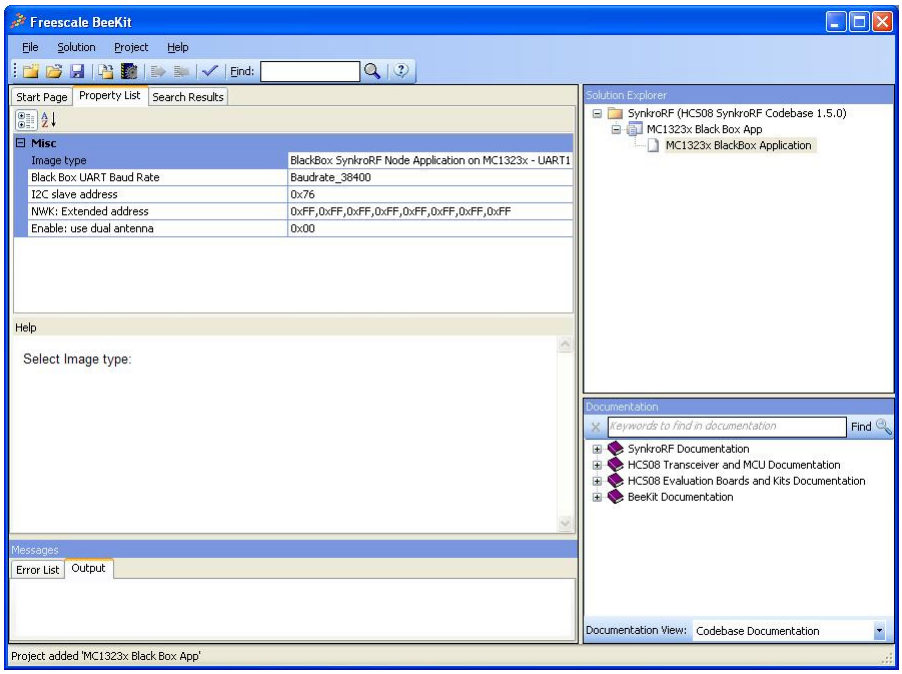

Figure 6-6. BeeKit Main Window (BlackBox Project)

12. From the menu select Solution\Export Solution. The Export Solution window appears, as shown in Figure 6-7.

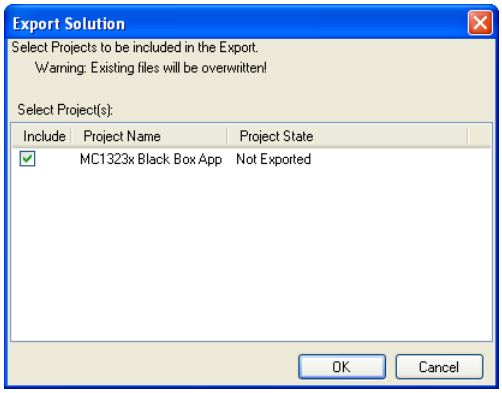

Figure 6-7. Export Solution Window

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

6.2 Uploading the BlackBox Image to the Target Board

If the target board is based on the MC1322x, then load the BlackBox image using the Freescale TestTool Firmware Uploader. If the target board is QE128 based, use Freescale CodeWarrior.

6.2.1 Uploading the Image to a MC1322x Target Board

1. From the Test Tool tool bar, choose Firmware Loader->MC1322x Firmware Loader. The MC1322x Firmware Loader appears as shown in Figure 6-8.

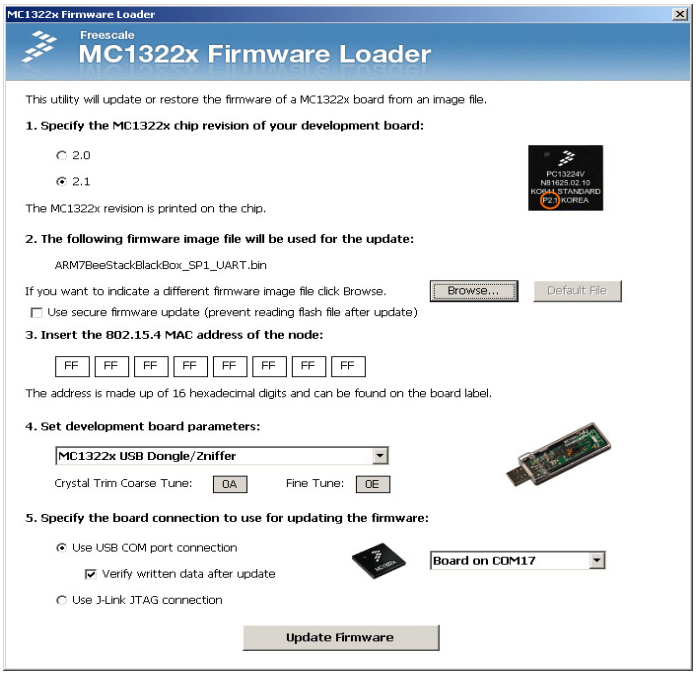

Figure 6-8. MC1322x Firmware Loader Window

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

6.2.2 Uploading the Image on a HCS08 based Target Board

1. Connect the BDM to the target board.
1. From the Test Tool tool bar, choose Firmware Loader->HCS08 Firmware Loader. The HCS08 Firmware Loader appears as shown in Figure 6-9.

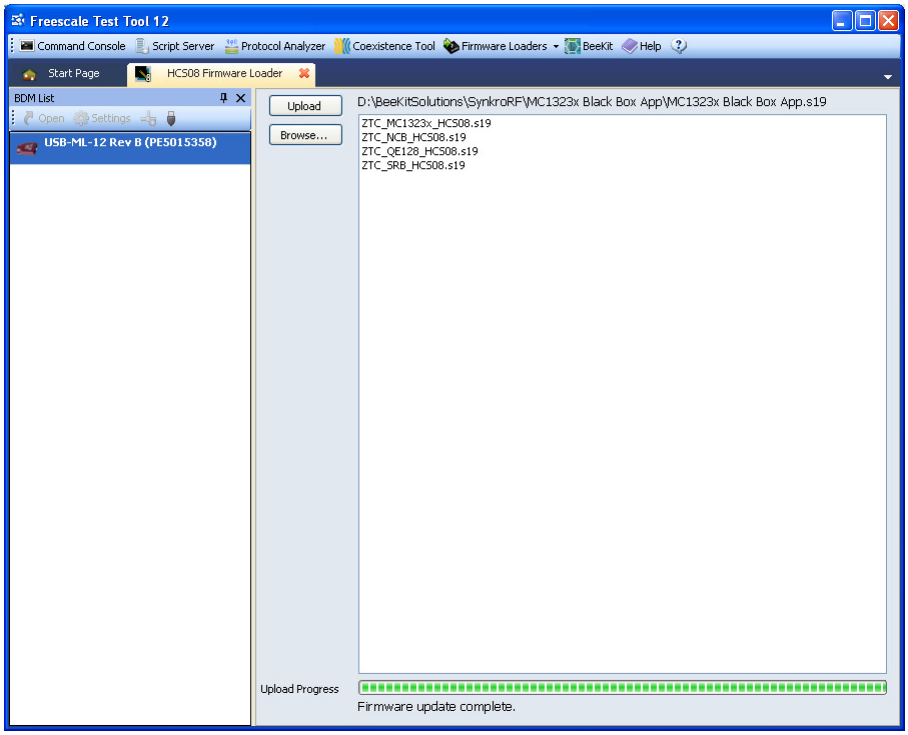

Figure 6-9. HCS08 Firmware Loader Window

2. Click the Browse button and browse to the location of the firmware file to be uploaded.
3. Click the Upload button.
4. A progress bar at the bottom of the window displays the uploading progress. When the upload is complete, the message “Firmware update complete” appears at the bottom of the window.

6.3 Running the Black Box application

See the Freescale *BeeStack Consumer Black Box Interface User’s Guide for* more information about SynkroRF and BeeStack Consumer black box functionality.