Freescale Semiconductor
Application Note

Document Number: AN4730
Rev. 0, 5/2013

ZigBee Wireless UART Used with the FreeMASTER Application

Karel Povalac

Contents

1 Introduction 1
1.1 General Description 2
2 How to Create a Wireless UART Application 2
2.1 Generating an Application 2
2.2 Modification of the Application 6
3 Use of Wireless UART Application 8
3.1 Basic FreeMASTER Settings 9
3.2 General Wireless UART Example 10
4 Conclusion 12
5 References 12

1 Introduction

This document describes the use of the ZigBee Wireless UART application with the FreeMASTER Run-Time debugging tool. The solution allows wireless data transmission over a serial port. This has the advantage of providing a non-galvanic connection to replace wired serial data communications in applications that require galvanic isolation. Example applications include electricity meters directly connected to the mains, motor control units, or other high-voltage devices. Wireless UART communication channel can also be used for bootloader applications. Uploading a new firmware image is described in the Freescale application note AN2295, available from Freescale.com.

© Freescale Semiconductor, Inc., 2013. All rights reserved.

1.1 General Description

This application note shows a simple way to build a ZigBee wireless UART application. The Freescale BeeKit software Graphical User Interface (GUI) is used to do this. The BeeKit Wireless Connectivity Toolkit may be downloaded from Freescale website free of charge. The software package is also supplied on the CD accompanying wireless connectivity development kits. BeeKit provides a simple GUI approach to configuring network settings, allowing you to concentrate on building the application. BeeKit helps reduce development time for developers lacking extensive networking experience. You are able to set configuration parameters that will control the setup and execution behavior of the wireless link within your application. The configuration parameters are validated inside the BeeKit to ensure all values provided are within acceptable ranges prior to the generation of a workspace. This functionality provides a mechanism for you to configure and validate your network parameters without having to navigate through multiple source files of a complex ZigBee system.

In addition to the GUI, the BeeKit toolkit includes a comprehensive codebase of wireless networking libraries, application templates, and sample applications. The codebases include Freescale's RF4CE and BeeStack (ZigBee/ZigBee Pro) protocol stacks and preconfigured application samples and templates. After you have completed the configuration of your wireless solution, BeeKit allows you to export the solution to an integrated development environment (IDE). Within the development tool, you are able to modify, build, and debug the application code. The project described below also has some additional features added, to allow starting, joining, and binding to the ZigBee network after generating a BeeKit solution. After the project is successfully built, you can download it onto two development boards. The last part of this application note discusses what adaptations are required for the project to work with FreeMASTER. Basic settings of the FreeMASTER Run-Time tool are also covered.

2 How to Create a Wireless UART Application

Prior to the project generation, you must download and install the BeeKit Wireless Connectivity Toolkit on a Windows PC, or run an installation file from the BeeKit CD-ROM. The latest version is available from the link below:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=BEEKIT_WIRELESS_CONNECTIVITY_TOOLKIT

The BeeKit Wireless Connectivity Toolkit is a comprehensive package of wireless networking libraries, application templates, and sample applications. The BeeKit GUI, part of the BeeKit Wireless Connectivity Toolkit, allows you to create, modify, and update various wireless networking implementations. The provided link also contains a BeeKit Wireless Connectivity Toolkit User's Guide. The guide describes how to use and install the Freescale BeeKit Toolkit.

2.1 Generating an Application

After you have successfully installed BeeKit and you've read the user's guide, you can begin generating a wireless UART application. After starting Freescale BeeKit software, select the appropriate codebase (File -> Select Codebase). For example, the latest ARM7 BeeStack Codebase is version 3.0.12 for Freescale MC1322x devices. Select the codebase and press the Set Active button. The active codebase is ready for use now.

The target project to be generated will act as a wireless bridge between two devices. The PC will form one side of the bridge and second part will be an embedded electricity meter. The functionality is described in Figure 1 below.

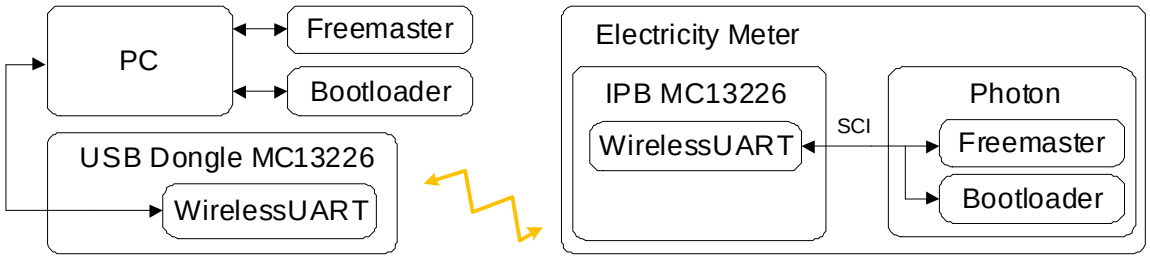

Figure 1. Block Diagram of Wireless UART FreeMASTER Example

Of course, the application described may be used in a different way as a typical wireless bridge solution. The example application will able to wirelessly communicate between the PC FreeMASTER and the embedded FreeMASTER application. The bootloader application may also be used.

Select File -> New Project to create a new solution. The wireless UART application is included inside the “Other Zigbee Applications” section. Figure 2 is a screen shot of the Project Selection step. Fill in appropriate project and solution names (mentioning the codebase version used is good practice) with a location before continuing. The project will consist of two nodes inside the network.

The first screen is just a welcome to the BeeStack configuration wizard. Select “Next” to continue. You are asked to make a number of parameter settings in the coming configuration screens. Select the MCU version and the target hardware board. The SRB board could be a good example to choose. (Selecting a user defined target board is also possible.) The next screen allows for the enabling of different platform modules on the development board. Select at least LEDs and Keyboard options for development purposes.

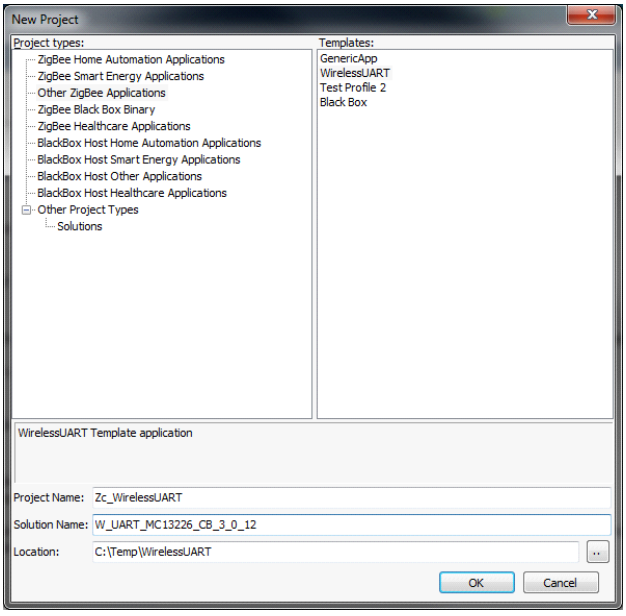

Figure 2. New Project Window for Wireless UART Application

On the next screen it is very important to enable the UART port on USB (see Figure 3). The Zigbee Test Client (ZTC) over the UART can be left disabled. The ZTC is a BeeStack UART communication protocol. The wireless UART application does not use the ZTC functionality.

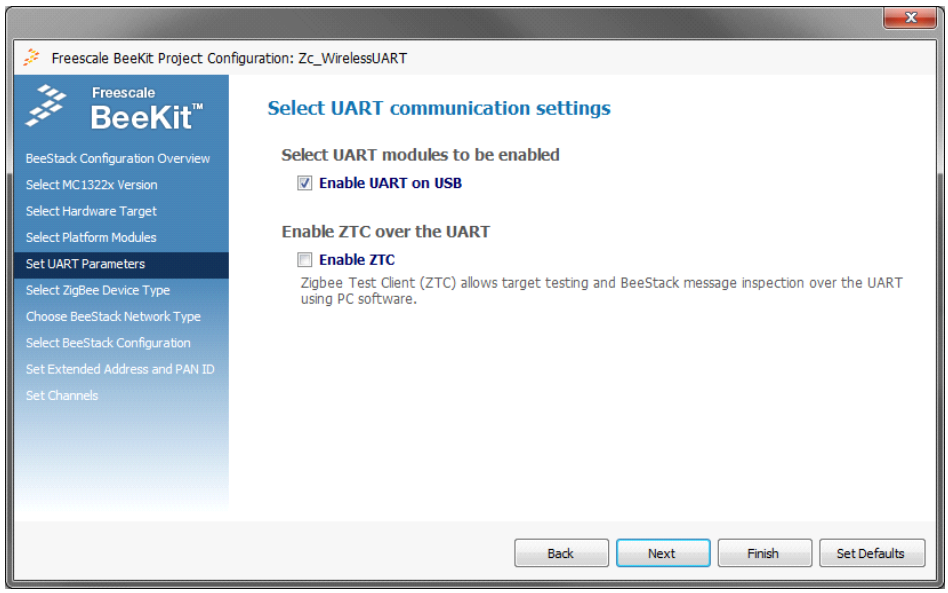

Figure 3. UART Communication Settings

The ZigBee device type of the target application must be selected on the following screen, where the appropriate choice is Coordinator. The ZigBee network can only have one Coordinator. For the second part of the project, you must select either a “Router” or “End Device” for the other node.

In the next step, the BeeStack network configuration is set (see Figure 4). Select an option without security because larger ZigBee packet overhead results from the implementation of security keys.

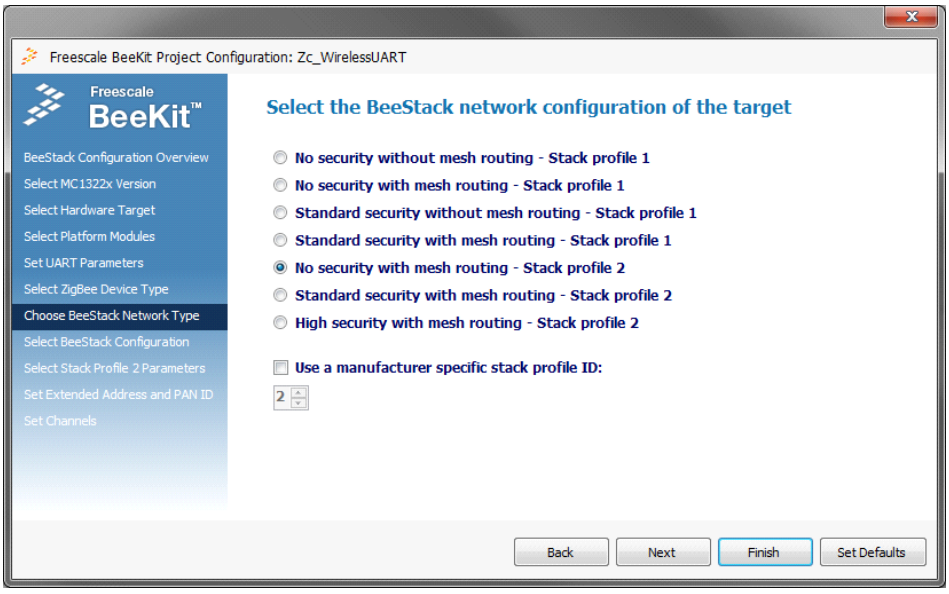

Figure 4. BeeStack Network Configuration Settings

The next two screens may be left unchanged with the default BeeStack settings. One of the last important options is the extended address and PAN ID information. The MAC address must be unique for each and every 802.15.4 node. The PAN ID will be the same for the Coordinator and the Router (and other nodes) in a ZigBee network. In the last page, selection of the channels for the wireless communication between the nodes is made. Press “finish” to close the wizard.

The solution explorer window contains a project with some basic settings files. The definition for the Coordinator has now been completed. The next step will follow the same procedure to configure the second node. You must add a new project solution into the existing explorer window. Select Solution -> Add Project. The same project selection pop-up window will appear as shown in Figure 2. The application template will again be WirelessUART. The second application settings will be mostly the same except for the ZigBee device type and MAC address settings. Choose “Router” as the 2nd ZigBee device type. A new unique MAC address is required. When the process settings are finished, the Solution Explorer should contain two separate projects as seen in Figure 5.

Platform modules of both nodes contain the UART interface settings. The UART1 half hardware flow control should be disabled. You don’t need to make any other modifications at this stage of the project. You can choose a different UART port number if needed. The solution can be quickly validated now (Solution -> Validate Solution) and exported to the IAR (Solution -> Export and Open Solution in IAR Embedded Work Bench). The BeeKit codebases currently supported are IAR version 5.5.

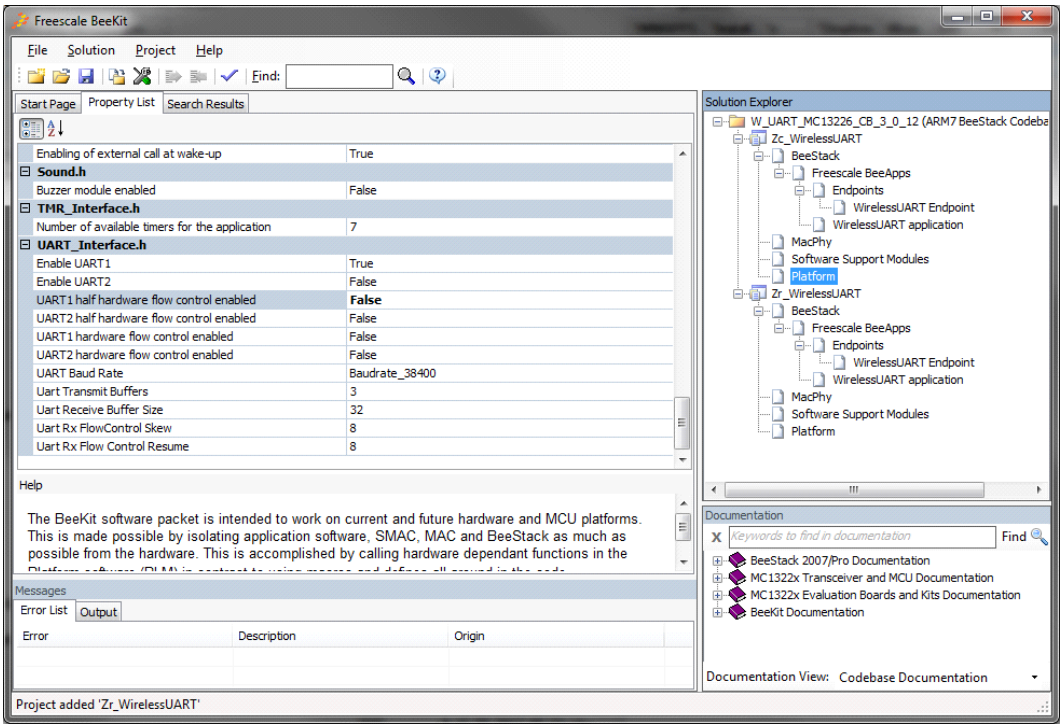

Figure 5. Freescale BeeKit Project Window

The export settings window is displayed in Figure 6. The project will be exported and immediately open in the IAR workspace after you press OK. If the project is not immediately opened in the IAR tool, you must open the project manually. The IAR project workspace file is saved with *.eww extension. Open the workspace manually by File->Open->Workspace. Or, you can drag and drop the IAR workspace file to get the same result.

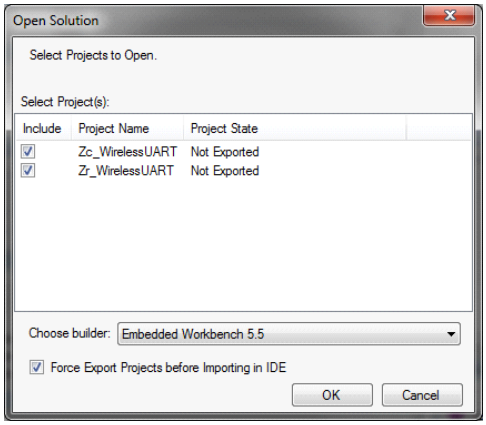

Figure 6. Export Settings Window

The BeeKit solution generation part of the project is complete and further modifications will now be made directly in the source code of the ZigBee application project.

2.2 Modification of the Application

In the previous section, the basic ZigBee project was created and imported into IAR tools. Some adaptation and additional features should now be included to improve the functionality. Modification of the Coordinator project will be addressed first.

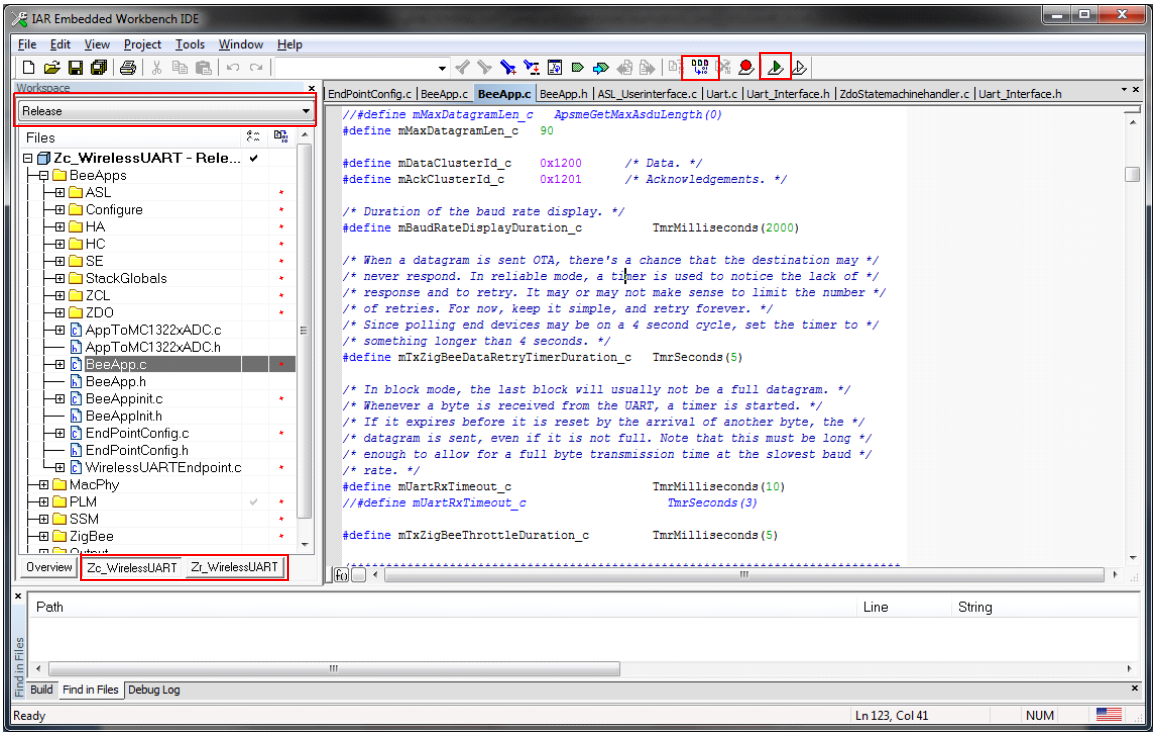

Figure 7. IAR Program Window

Figure 7 shows the main IAR window. Exported projects (coordinator or router) can be chosen in workspace. Currently, the figure displays the coordinator for wireless UART available in the released version of building code. Press the MAKE or COMPILE button to start the compilation process. After the compilation is done, it can be downloaded into the development kit. Press the DOWNLOAD and DEBUG button to do this. Important buttons are marked red in the Figure 7. The debug interface is needed for downloading a created project into the development platform.

2.2.1 Modification of Coordinator Source Code

To begin, create a ZigBee network automatically. This is handled in the initialization function. Open the source file BeeApp.c (located in the workspace project directory structure as follows: //W_UART_MC13226_CB_3_0_12\Zc_WirelessUART\BeeApps\). The Initialization function of the Zigbee application is called BeeAppInit() by adding the following call at the end of BeeAppInit() function.

```
BeeAppHandleKeys(gKBD_EventSW1_c);  // Create the network after the startup
```

The same modification will also be needed in the router device. The stand-alone application of Wireless UART is also included in BeeApp.c file. A few settings must be modified to work correctly with the FreeMASTER software and bootloader applications. The changes mainly concern the maximal length of data packet, UART timeouts, and transmission modes. The important changes in BeeApp.c are summarized below.

```
#define mMaxDatagramLen_c                90
#define mUartRxTimeout_c                 TmrMilliseconds(10)
#define mTxZigBeeThrottleDuration_c      TmrMilliseconds(5)
static bool_t mInCharMode = FALSE;       /* T = char mode, F = block mode. */
static bool_t mInReliableMode = FALSE;   /* T = reliable, F = unreliable. */
```

Some further changes are important to include, as outlined in the accompanying source file BeeApp.c in AN4730SW.zip. The best way is to compare the original file with the modified one.

2.2.2 Modification of Router Source Code

The same modifications of the application file BeeApp.c are required for the router, as already described above for the coordinator.

In addition the router device has to send bind request commands to the coordinator after it has joined the Zigbee network. All of this modification is done in

//Zr_WirelessUART\BeeApps\ASL\ASL_Userinterface.c. Reference is made to the accompanying source file in AN4730SW.zip for a detailed description of the ASL_Userinterface.c modifications. There is a new call back function (BindDeviceTimerCallBack) that manages the binding requests.

The best way of highlighting the changes in the attached files is to compare the original source files with the included ones and apply the changes. There are also a few sections that were commented out. These changes mainly touch the settings using the buttons and LEDs. All of these settings can be done without external inputs. Namely the changes are related to the baud rate settings, mode of transmission, and LED states.

When the modification process of the source files is done, you can compile and download the code to the development boards.

3 Use of Wireless UART Application

After the wireless UART application is created and modified according to the previous steps, it can be used and connected as shown in Figure 1 for the FreeMASTER application or more general solution as displayed in Figure 10. Thus both devices (coordinator and router) communicate over the serial port, which can be used according to the designer's needs. In effect, wherever a wired serial communication link is currently used, it could be replaced with a two node wireless communication link such as this. The default settings are defined in Table 1.

The project was created for two development sensor boards for simplicity. However, the same solution can be used for other development or customer defined boards that employ MC1322x integrated circuits. Solutions based on other than MC1322x wireless microcontroller platforms must be created with different BeeStack libraries.

Table 1. Default Wireless UART Port Settings

Communication port on MC1322x	UART1
Bitrate	34800 bps
Hardware flow control	Disabled

After the development boards are successfully connected with serial port (TX/RX) pins and powered on, the coordinator will create a ZigBee network automatically. Also the router node will join the network and bind with appropriate communication clusters. The joining procedure takes approximately 5 seconds and after that, the wireless serial link is ready for half-duplex serial communication. The wireless UART example is now ready for use as a replacement for wired communications.

3.1 Basic FreeMASTER Settings

To implement a wireless UART with the FreeMASTER run time debugging tool, you must run a FreeMASTER driver on the embedded side. In addition, the FreeMASTER software must be installed on the computer. The driver description and explanation is covered by the FreeMASTER Serial Communication Driver User's Guide [4] included in the installation of FreeMASTER. The installation also contains a few examples for different microcontrollers. A detailed description of how to run a FreeMASTER application is outside the scope of this document.

After the project with FreeMASTER driver is ready to use, the FreeMASTER application can be started on the PC. You may add and watch variables and put them into the window as needed.

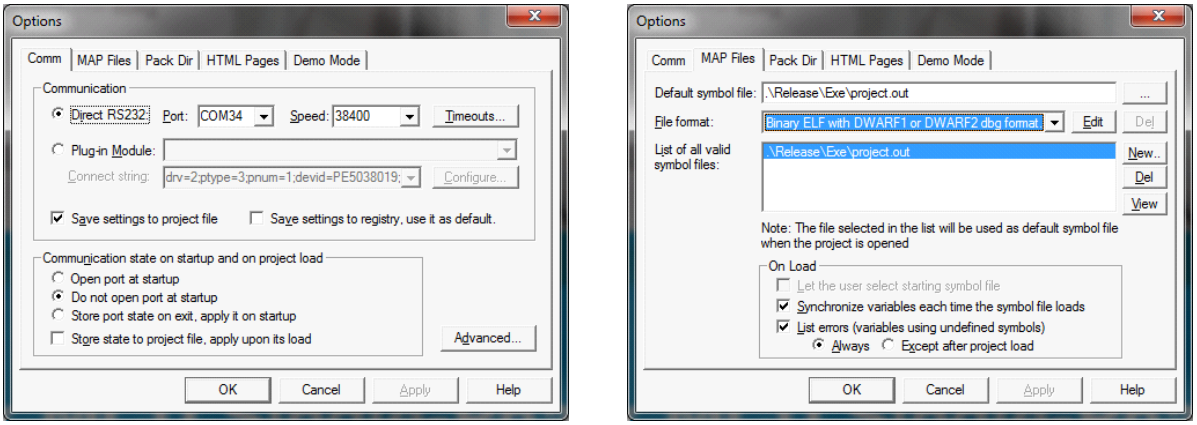

Figure 8. FreeMASTER Options Windows

The main setting for the project is under the Project -> Options. The setting window is displayed in Figure 8. The communication port has to be selected with the appropriate speed. The same value is setup on the PC side, embedded coordinator, and also on the embedded router. Timeouts may be kept the same as the default values (100 ms). The second list sets up the map file of the embedded application. The path to the map file has to be specified in Default symbol file field. All the changes are confirmed by pressing OK.

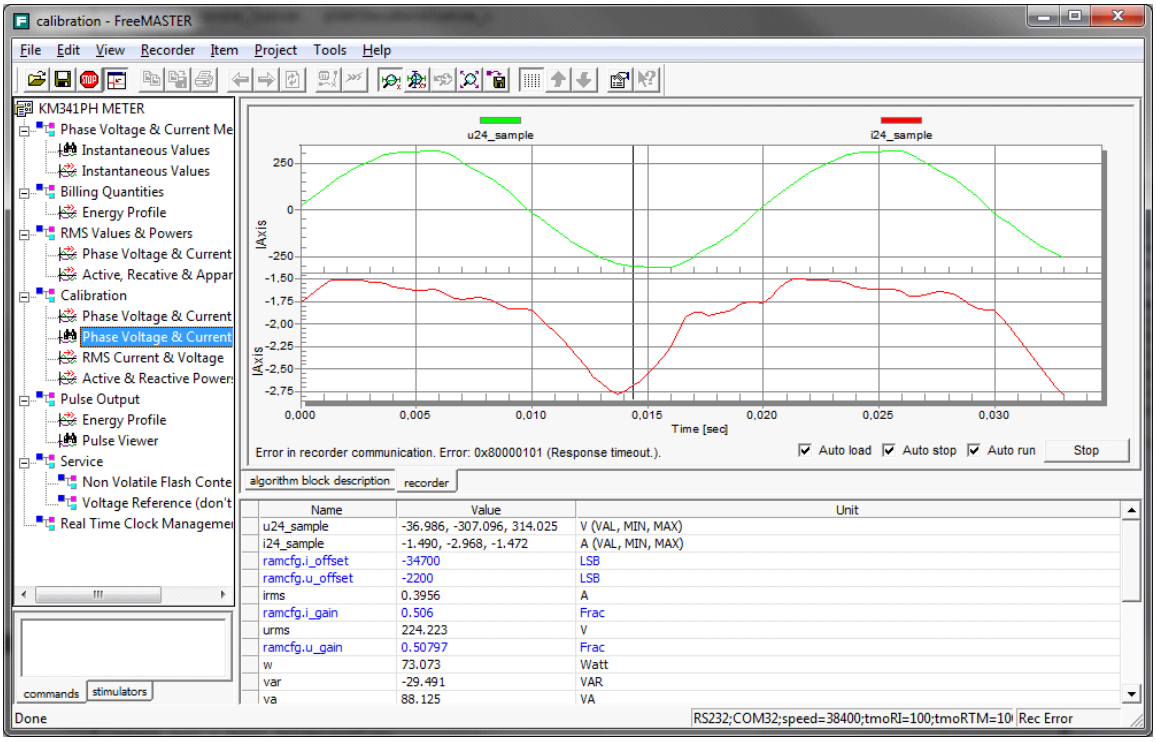

Figure 9. Example of FreeMASTER Application Window

The STOP button controls the start and stop of the communication. When the COM port is opened, the data are retrieved and displayed in the window. The basic example can be seen in Figure 9. Updated variables are displayed in the FreeMASTER window.

3.2 General Wireless UART Example

The general wireless UART example uses three ZigBee development boards. The example shows possibilities of PC communication over the wireless UART with embedded devices. Two nodes cover the wireless UART link and the third node works as ZTC enabled client. The scenario shows the Figure 10. Freescale TestTool is used as simple a demonstration of the wireless UART communication. The software is included in BeeKit installation and easy to use.

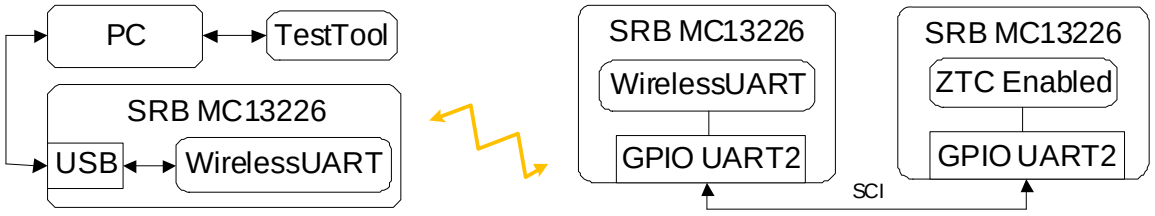

Figure 10. General Wireless UART Example

The Command Console can display ZTC data packets on UART port. The port setup must be done in advance (port number, baudrate). Each ZTC command is shown in the right hand side window (See Figure 11). Transmitted and received data packets prove successful communication over the wireless UART.

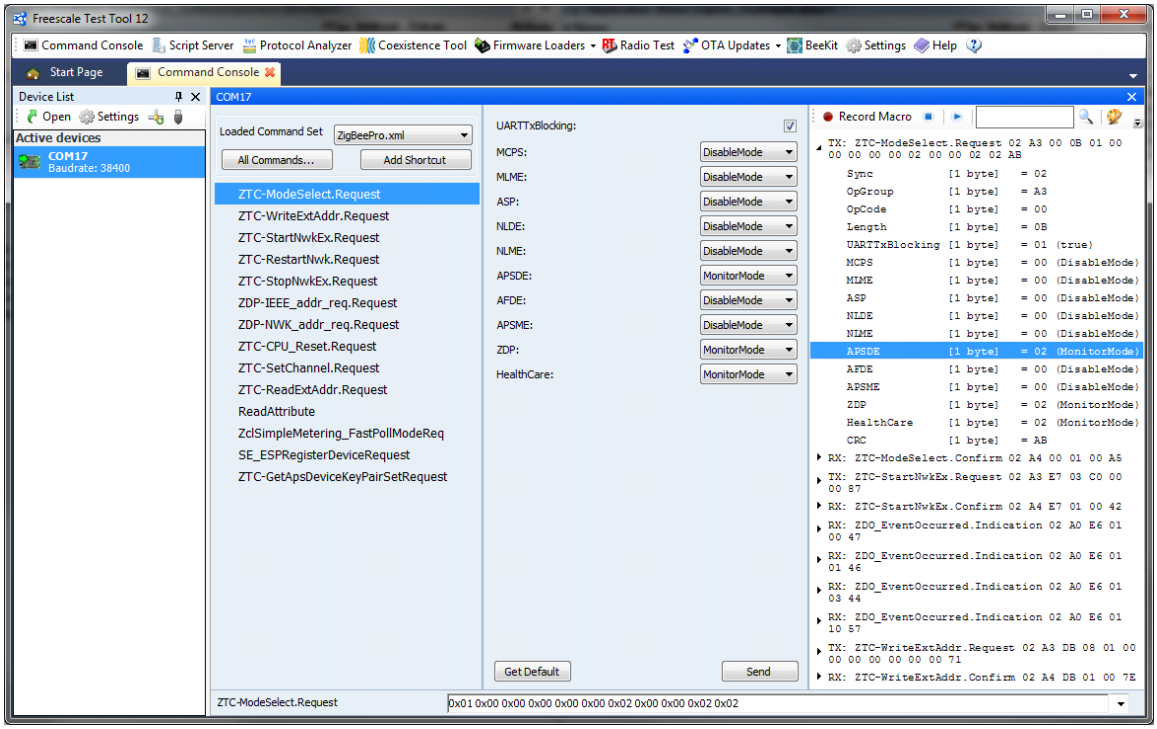

Figure 11. Freescale TestTool Program

The third node can be generated as an arbitrary solution from BeeKit. The node must have enabled ZTC communication over the UART. The ZTC can be enabled during the BeeKit project creation or can be found in BeeKit project under the Software Support Modules (ZTC Enabled - TRUE). The sensor node as a Freescale development kit has the UART2 port directly connected on GPIO pins, which are easily accessible. The GPIO uses second UART port that can be enabled in the UART_Interface.h file. Recommended settings are shown below.

```
#define gUart1_Enabled_d            FALSE
#define gUart2_Enabled_d            TRUE
```

The modification must be accepted also on the wireless UART node that will be connected with embedded client over the GPIO pins. The situation is shown in Figure 10.

4 Conclusion

This application note describes the step-by-step process of building a wireless UART application for point-to-point transmission usage. This project will be particularly useful in situations requiring serial communication between two devices that require galvanic isolation (for instance, electricity meters). This application note has shown how to generate a template project in BeeKit, consisting of a ZigBee Coordinator and Router. It then showed how to adapt the generated project source code to operate with the FreeMASTER application or to make a quick test with the Freescale TestTool. Modified source files for the embedded application are described and included to accompany the application note in the download AN4730SW.zip available from Freescale.com.

5 References

[1] FreeMASTER - Real-time debug monitor and data visualization tool. Freescale fact sheet available online.

[2] Developer’s Serial Bootloader for M68HC08, HCS08, ColdFire and Kinetis MCUs, AN2295. Freescale application note available online.

[3] BeeKit Wireless Connectivity Toolkit User’s Guide. Freescale document available online. Document Number: BKWCTKUG.

[4] FreeMASTER Serial Communication Driver User’s Guide. Freescale document.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

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

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+,Energy Efficient Solutions logo, PowerQUICC, QorIQ, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. CoreNet, Layerscape, QorIQ Qonverge, QUICC Engine, Tower, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© 2013 Freescale Semiconductor, Inc.

Document Number: AN4730
Rev. 0
5/2013