Kinetis W MCUs: BLE+Thread Lab

Lab Hand Out

NXP Semiconductors, 6501 William Cannon Drive West, Austin, Texas 78735
www.nxp.com

Contents

1 Lab Overview

This lab will use two FRDM-KW41Z boards, one running the standard Thread Router Eligible End Device application, and the other board running a BLE+Thread hybrid application.

2 Prerequisites

The following items are needed to complete this hands-on lab. This has already been installed on your computer:

- Boards
  - Two FRDM-KW41Z
- Software
  - MCUXpresso SDK for FRDM-KW41Z (includes Thread stack): http://mcuxpresso.nxp.com
  - MCUXpresso IDE: http://nxp.com/mcuxpresso/ide
  - Terminal Software (like TeraTerm or PuTTY)
  - NXP Kinetis BLE Toolbox Smartphone App (on Google Play or Apple App Store)

3 Thread Router Eligible Device Application

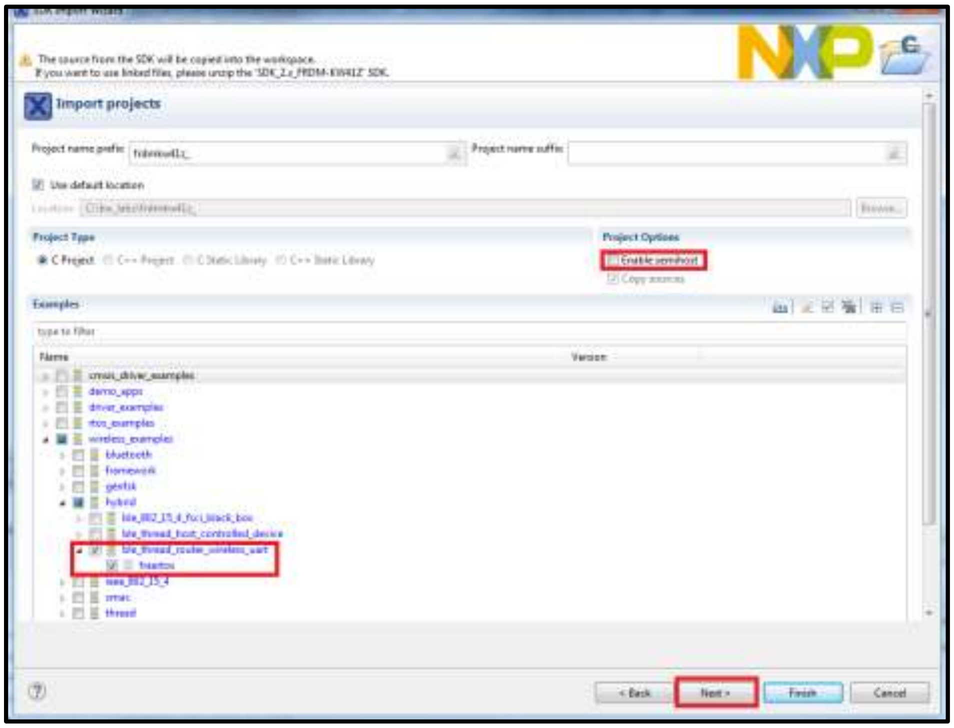
You should already have a board programmed with the Thread Router Eligible Device application. However if you do not, follow the instructions from the previous lab to program in the Router Eligible End Device (REED) example.

When flashing that project, ensure that the channel number and PSK_D have been modified to ensure you will be using your own unique network.

4 BLE+Thread Application

4.1 Configure the Project

1. Import the new BLE+Thread project using the same method as you did for the REED example, using the Import action in the Quickstart Panel. You will want to import the **ble_thread_wireless_uart** example found under the **wireless_examples->hybrid** category. Make sure to uncheck the **Enable Semihost** option on this page, and the **"Redirect SDK "PRINTF" to C library "printf""** option on the next screen.


The source from the SDK will be copied into the workspace.
If you want to use linked files, please unzip the 'SDK_2.x_FRDM-KW41Z' SDK.

Import projects

Project name prefix: frdmkw41z_   Project name suffix:

☑ Use default location

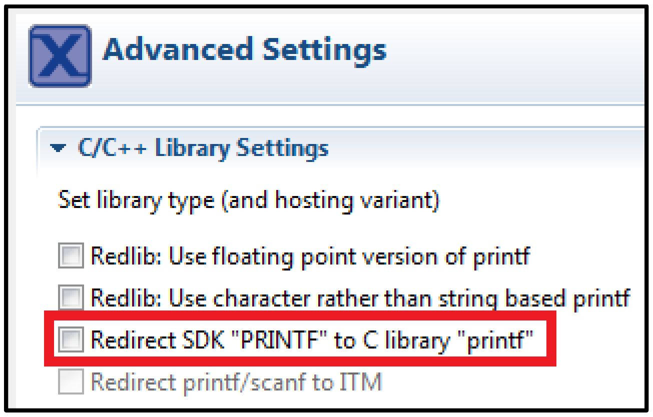
Location: C:\kw_labs\frdmkw41z_

Project Type: ● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options: ☐ Enable semihost ☑ Copy sources

Examples

Name	Version
cmsis_driver_examples	
demo_apps	
driver_examples	
rtos_examples	
wireless_examples	
bluetooth	
framework	
genfsk	
hybrid	
ble_802_15_4_fsci_black_box	
ble_thread_host_controlled_device	
☑ ble_thread_router_wireless_uart	
☑ freertos	
ieee_802_15_4	
smac	
thread	


Advanced Settings

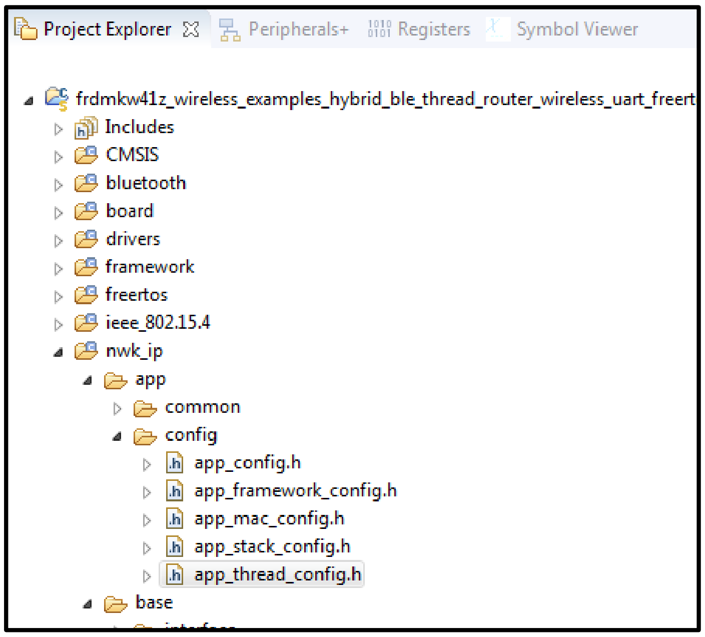
C/C++ Library Settings

Set library type (and hosting variant)

☐ Redlib: Use floating point version of printf
☐ Redlib: Use character rather than string based printf
☐ Redirect SDK "PRINTF" to C library "printf"
☐ Redirect printf/scanf to ITM

2. Now we will change the same configuration options you did for the REED lab to make your board unique for this lab, avoid wireless interference, and ensure you join your thread network, and not your neighbor's thread network.

3. Change the **THR_SCANCHANNEL_MASK** to something unique in **app_thread_config.h** so that your boards will not conflict with other boards in the class on the same channel. In a real system a channel scan would determine the least used channel, but we will hardcode it for this lab:
    a. File is located here:



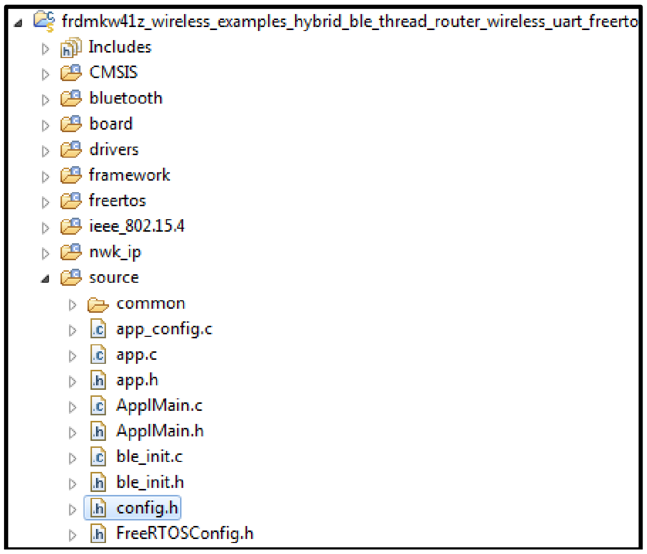
```
Project Explorer   Peripherals+   Registers   Symbol Viewer

◢ frdmkw41z_wireless_examples_hybrid_ble_thread_router_wireless_uart_freert
    ▷ Includes
    ▷ CMSIS
    ▷ bluetooth
    ▷ board
    ▷ drivers
    ▷ framework
    ▷ freertos
    ▷ ieee_802.15.4
    ◢ nwk_ip
        ◢ app
            ▷ common
            ◢ config
                ▷ app_config.h
                ▷ app_framework_config.h
                ▷ app_mac_config.h
                ▷ app_stack_config.h
                ▷ app_thread_config.h
        ◢ base
```

b. Edit the channel number (it's '25' by default) to match the **"Channel Number"** value that is printed on the sticker located on your FRDM-KW41Z box. After changing it, it would look similar to:

#define THR_SCANCHANNEL_MASK (0x00000001 << 19)

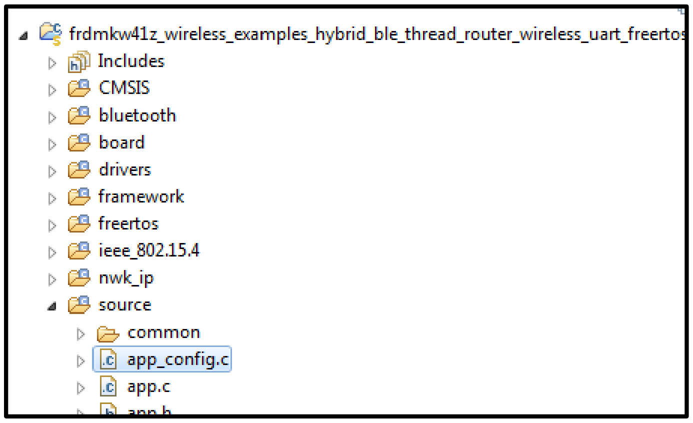
4. Also in that same **app_thread_config.h** file, change the **THR_PSK_D** to define to a unique PSKd for your particular set of boards, which is the password used when joining a Thread network.
    c. Append the value of the **"Unique Kit Number"** that can be found on the sticker on your FRDM-KW41Z box. Note you will also need to change the size.
    d. When done it should look similar to the following:
    #define THR_PSK_D {8, "THREAD12"}
5. Now we need to make the Bluetooth unique for your particular board.
6. Change the **BD_ADDR** to something unique in **config.h**, line 153, so that your boards will not conflict with other boards in the class.:
    • File is located here:



```
◢ frdmkw41z_wireless_examples_hybrid_ble_thread_router_wireless_uart_freerto
    ▷ Includes
    ▷ CMSIS
    ▷ bluetooth
    ▷ board
    ▷ drivers
    ▷ framework
    ▷ freertos
    ▷ ieee_802.15.4
    ▷ nwk_ip
    ◢ source
        ▷ common
        ▷ app_config.c
        ▷ app.c
        ▷ app.h
        ▷ ApplMain.c
        ▷ ApplMain.h
        ▷ ble_init.c
        ▷ ble_init.h
        ▷ config.h
        ▷ FreeRTOSConfig.h
```

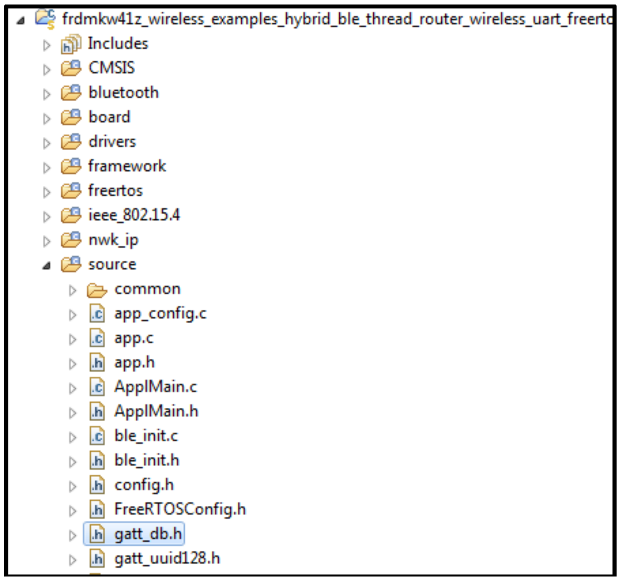
- Edit the most significant byte to match what is printed as the **"Unique Kit Number"** on the sticker located on your FRDM-KW41Z box. After changing it, it would look similar to: **#define BD_ADDR 0x12,0x01,0x01,0x9F,0x04,0x01**

7. First we'll change the advertising name seen by Android phones. Open app_config.c and modify the **gAdShortenedLocalName_c** structure, changing the "NXP_THR" string found on line 90. Remove the "THR" part to replace it with the **"Unique Kit Number"** printed on the sticker on your board. Make sure to update the length accordingly. This length is the amount of characters + 1
   - File is located here:



   - After changing it, it would look similar to:
     .length = 7,
     .aData = (uint8_t*)"NXP_12"

8. Now change the advertising name that is seen by iPhones. Open up gatt_db.h and replace the string **NXP_THREAD_SHELL** on line 8 with just **NXP_Unique#.** Make sure to update the length accordingly. This length is the amount of characters
   - File is located here:



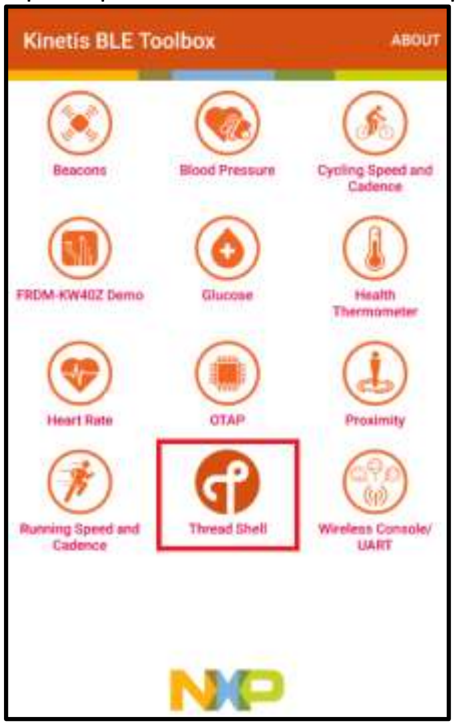
- After changing it, it would look similar to:

  VALUE(value_device_name, gBleSig_GapDeviceName_d, (gPermissionFlagReadable_c), 6, "NXP_12")

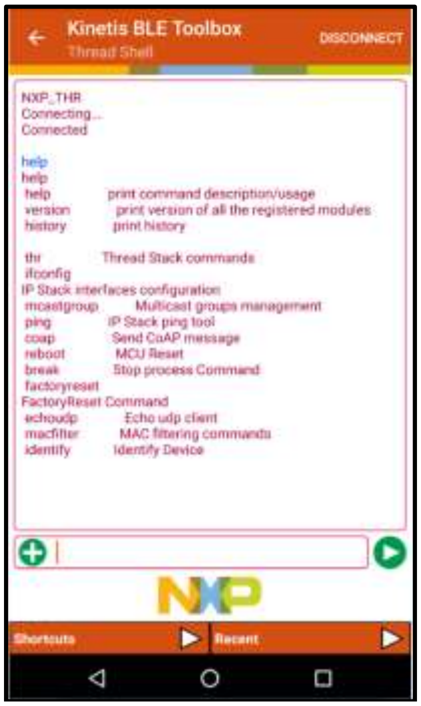
9. Now compile and download this application to **one** of the boards.
10. On the other board that has the Thread router_eligible_thread device app that you programmed in earlier, verify that the board has been factory reset (ie its network information has been cleared) by holding down one of the push buttons for 10 seconds. The blue and red LEDs should be flashing after releasing the button. Alternatively, you can also type "**factoryreset**" into the terminal.

4.2 Running the demo

11. Now you have one board, which we'll call R1, that is running the BLE+Thread app. The other board, which we'll call R2, is running only Thread. Both should be blinking red and blue LEDs rapidly.
12. Open up the Kinetis BLE Toolbox app and go to the "Thread Shell" option.



13. You should see your board listed with your custom name. Tap on it to connect which will take you to a terminal prompt.
14. Type "help" into that terminal prompt in the app, and you should see a list of all the Thread commands, just like you did in the previous lab in TeraTerm. Essentially we've created a terminal on your phone where you can run any Thread command.



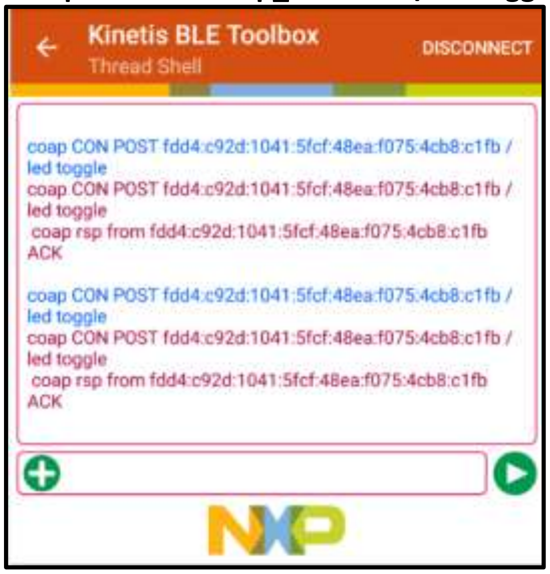
```
NXP_THR
Connecting...
Connected

help
help
 help           print command description/usage
 version          print version of all the registered modules
 history          print history

 thr            Thread Stack commands
 ifconfig
IP Stack interfaces configuration
 mcastgroup          Multicast groups management
 ping           IP Stack ping tool
 coap           Send CoAP message
 reboot           MCU Reset
 break           Stop process Command
 factoryreset
FactoryReset Command
 echoudp           Echo udp client
 macfilter          MAC filtering commands
 identify          Identify Device
```

15. Start a new Thread network by typing "**thr create**"

16. Open up a TeraTerm on your computer and connect to the second board (that is just running Thread) and join this new network by typing "**thr join**".
17. Take note of the ML64 address listed in the terminal for R2 by typing "**ifconfig**"
18. Inside the smartphone app terminal, type in the command to toggle the LED.
    "**coap CON POST <ip_address> /led toggle**"



```
coap CON POST fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb /
led toggle
coap CON POST fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb /
led toggle
 coap rsp from fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb
ACK

coap CON POST fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb /
led toggle
coap CON POST fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb /
led toggle
 coap rsp from fdd4:c92d:1041:5fcf:48ea:f075:4cb8:c1fb
ACK
```

19. Run the command a few more times to make the LED on R2 toggle on and off via the BLE app. Make use of the **Shortcuts** and **Recent** buttons in the app to save some time typing. You can also try getting the temperature like you did in the previous lab.