AN242

2.4 GHz ZigBee™ Network API Programming Example Guide

1. Introduction

The 802.15.4/ZigBee™ Development Board can be used for demo purposes or as a platform for firmware development. This document describes a code example using the ZigBee Network layer Application Programming Interface (API). The code example is a simple example that demonstrates ZigBee network formation. This code example may be used as a starting point to develop a ZigBee network application.

The 802.15.4/ZigBee Development Board comes with firmware installed for a ZigBee or 802.15.4 demo, as specified in the kit contents. To use the 802.15.4/ZigBee Development Board for code development, you will have to erase the existing firmware. If desired, the demo code can later be restored by downloading the appropriate hex file.

The Network Blinky example requires three or more boards to demonstrate network address allocation. The ZigBee Development Kit includes six boards. Using all six boards provides a very effective demo.

1.1. Connecting to the IDE

The 802.15.4/ZigBee development board can be used with the Silicon Laboratories Integrated Development Environment (IDE) and the universal serial bus (USB) debug adapter as shown in Figure 1.

- Connect the USB Debug adapter to the PC.
- Connect the ten pin ribbon cable to the 802.15.4/ZigBee development board (J2).
- Connect the 9 V universal ac/dc Adapter to the 2.1 mm power supply jack (J1) on the 802.15.4/ZigBee Development Board.

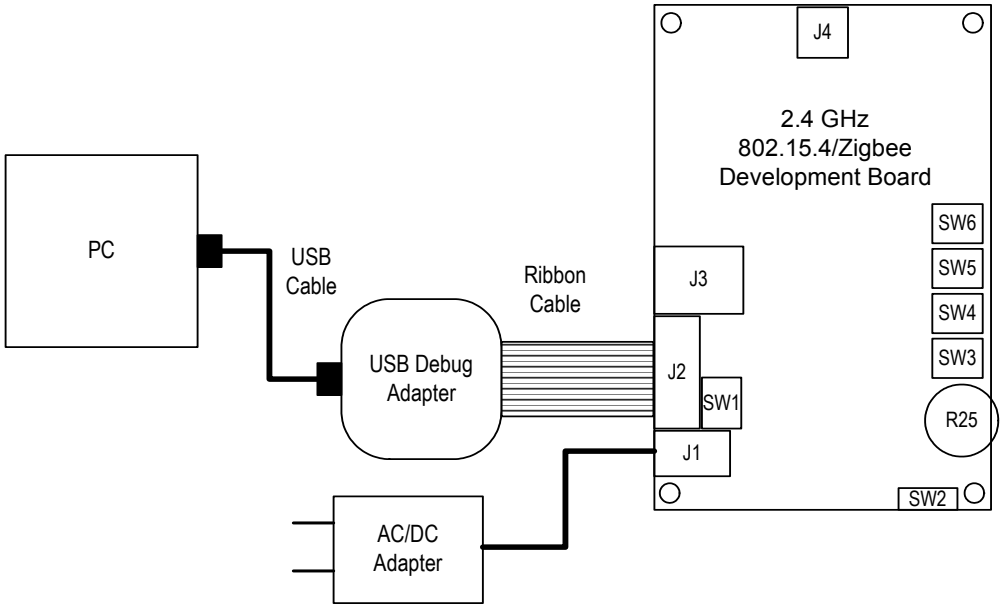

Figure 1. Programming Setup

2. Downloading the Firmware

Launch the Silicon Laboratories IDE. From the Options menu select the connection options sub-menu. Select the radio button for USB Debug Adapter. If the USB Debug Adapter is greyed out, make sure that the USB Debug Adapter is connected to an active USB port. Select the JTAG radio button for the Debug interface. Click OK to close the dialog box.

From the Debug menu, select Connect or click on the connect icon in the toolbar. The status bar on the bottom of the IDE window should display the connection status “Target:C8051F121”. This indicates that the debug adapter has successfully connected to the C8051F121 and is ready to download code.

The Intel hex file for the demo firmware is located in the following directory:

\SiLabs\MCU\Examples\ZigBee\NWK_Blinky\

To download the firmware, select “download object file...” from the debug menu. Do **not** erase the entire code space. This will erase the 64-bit Extended Unique Identifier stored in Flash memory.

Click on “Browse…” In the browse dialog box, find the “Files to List” pull-down menu and select “Intel - Hex”, then browse to the location of the NWK_Blinky.hex file. The demo file will be downloaded into Flash.

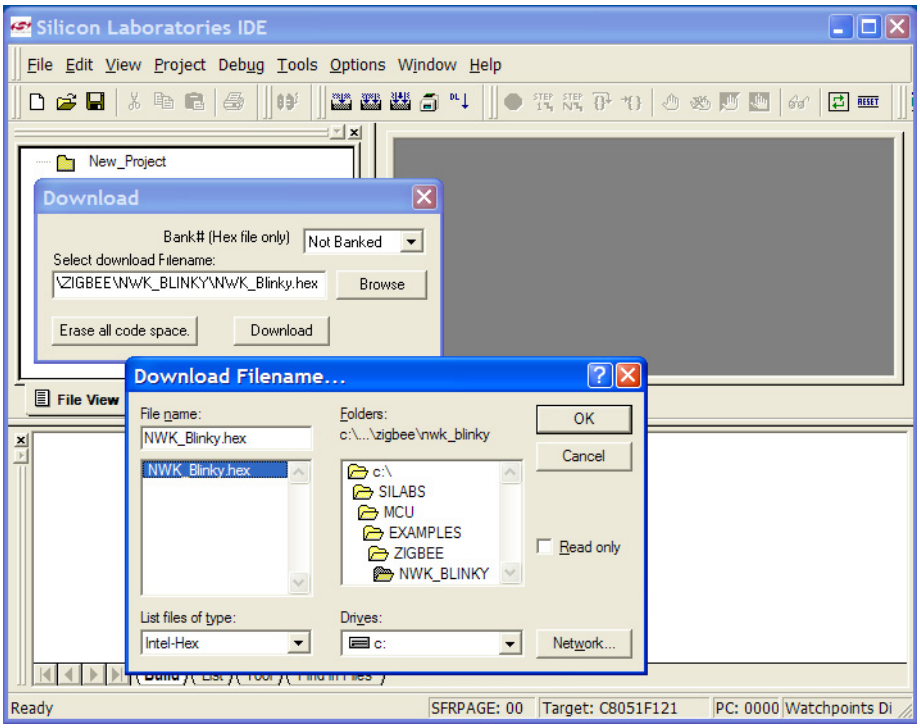

Figure 2. Downloading Firmware

Select “Go” from the debug menu or click on the green Go button in the toolbar to run the program. Three of the color LEDs on the board should blink. This indicates the code has been downloaded and is functional. Click on the red stop button to halt execution.

Select “Disconnect” from the Debug menu. Now unplug the power connection 10-pin header. Repeat this procedure for each board you wish to reprogram.

If you inadvertently erase the EUI you can modify the EUI directly from the IDE. Connect to the 802.15.4/ZigBee Development Board. Select “Code Memory” from the View>Debug Windows menu. Type “1FBF6” in the code address box. Click in the code memory window just left of the first byte of address FBF6. Enter the EUI from the sticker on the back of the board in little endian format. The least significant byte of the EUI should be in location 0x1FBF6 and the most significant byte should be in location 0x1FBFF.

3. NWK Blinky Demo Description

Once all of the boards have been programmed with the Network Blinky firmware, it is possible to demonstrate the function of the blinky code. The demo consists of two to eight 802.14.5/ZigBee Development Boards operating in standalone mode with no connection to the PC.

One of the boards is designated as the ZigBee Coordinator (ZC). The firmware is the same for the ZigBee Coordinator and the other devices. The first step is to configure the ZigBee Coordinator for one of three network topologies- Rangy, Bushy, or Kempt. The network parameters for these three network topologies are defined in Table 1. These three topologies are test cases from the ZigBee Level 2 inter-operability test procedure. A detailed explanation of the network parameters can be found in the ZigBee Network specification. The Cardinality column describes the maximum network configuration using six nodes. A ZigBee network will consist of a ZigBee Coordinator (ZC) and some combination of ZigBee Routers (ZR) and ZigBee End Devices (ZED).

Table 1. Topology Network Parameters

	Max Routers	Max Children	Max Depth	Color	Cardinality
Rangy	1	1	7	Green	ZC, 5xZR
Kempt	3	5	7	Yellow	ZC,2xZR,3xZED
Bushy	7	14	3	Amber	ZC,5xZED

The topology is selected by pressing SW6 on the ZigBee development board. Pressing the button once will cause the Green LED D6 to blink. This will select a Rangy network. Pressing a second time will cause the Yellow LED D7 to blink. This will select the Kempt network. Pressing a third time will cause the Amber LED to blink. This will select the Bushy network topology.

Pressing SW5 will cause the ZigBee coordinator to form a ZigBee network of the chosen topology. The color LED previously selected will stop blinking and illuminate continuously.

Pressing SW4 on any of the five remaining boards will cause it to join the ZigBee network as a router. Once the join process has completed the board will display the least significant byte of the network address in binary on D11–D14.The binary addresses are illustrated in Table 2.

Table 2. Binary Network Addresses

Decimal	Hex	Binary	LEDs			
			D13	D12	D11	D10
0	0x00	0000				
1	0x01	0001				
2	0x02	0010				
3	0x03	0011				
4	0x04	0100				
5	0x05	0101				
6	0x06	0110				
7	0x07	0111				
8	0x08	1000				
9	0x09	1001				
10	0x0A	1010				
11	0x0B	1011				
12	0x0C	1100				
13	0x0D	1101				
14	0x0E	1110				
15	0x0F	1111				

Pressing SW3 on any of the five remaining boards will cause it to join the ZigBee network as an end device. Once the join process has completed, the board will display the least significant byte of the network address in binary on D11–D14. Joining as an end device will terminate one branch of the tree and effect the network addresses of subsequent join operations. If a ZigBee router or end device fails to join a network it will blink red LED D9.

Once the board has been configured the push buttons can be used for a secondary function. The secondary function of push button SW6 is to send data. Pushing SW5 on the ZigBee coordinator will send broadcast data to all of the ZigBee boards. Pushing SW5 on a router or end device will send data to the ZigBee coordinator.

Once the network has been formed SW6 can be used to leave the network. Pushing SW6 on a router or end device will cause the device to leave the network. Pushing SW6 on the ZigBee coordinator will dissolve the network.

The function of the buttons are summarized in Table 3.

Table 3. Push Button Functions

	Primary Function	Secondary Function	
SW6	Select Network Profile	ZC	Dissolve Network
	Rangy, Kempt, or Bushy	ZR or ZED	Leave Network
SW5	Start as ZigBee Coordinator	ZC	Broadcast Data
	Form Network	ZR or ZED	Send Data to ZC
SW4	Start as ZigBee Router		
	Join Network		
SW3	Start as End Device,		
	Join Network		

4. Rebuilding the Project

The source code for the Network Blinky demo is included with the development kit. The project folder includes source files for the essential main and NWK_Blinky modules, a library file which contains the ZigBee networking code, and all of the header files required to rebuild the project. The project also contains a Silicon Laboratories IDE workspace file that can be used to edit and build the file.

The library file is a special library to facilitate building the project with the demo tools. The project can be rebuilt using the included demonstration tools with a 4 kB linker limit. The library itself does not count against the 4 kB linker limit. Thus, it is possible to rebuild the project using the demo tools. The demo tools are for evaluation purposes only. The user is limited to non-commercial use only as specified in the license agreement. The user will have to purchase a full set of development tools for commercial development.

- Connect the 802.15.4/ZigBee Development Board as shown in Figure 1.
- Connect the USB debug adapter or the serial adapter or to the PC.
- Connect the ten pin ribbon cable to the 802.15.4/ZigBee development board.
- Connect the 9 V ac/dc adapter to the 2.1 mm power supply jack on the 802.15.4 Development Board.
- Launch the Silicon Laboratories IDE.
- From the “Project” menu select “Open Project...”.
- Browse to the NWK__Blinky folder:
  `SiLabs\MCU\Examples\Zigbee\NWK_Blinky\`
- Open the workspace file, “NWK_Blinky.wsp”.
- From the File View tab, double-click on the source file. `NWK_Blinky.c`.
  This will open the application source window in the editor pane. Expand this window to fill the middle pane.
- From the Debug menu, select “Rebuild Project”.

The IDE will compile and link all files in the project. The IDE should display the linker status as shown in Figure 3.
`LINK/LOCATE RUN COMPLETE. 0 WARNINGS, 0 ERRORS.`

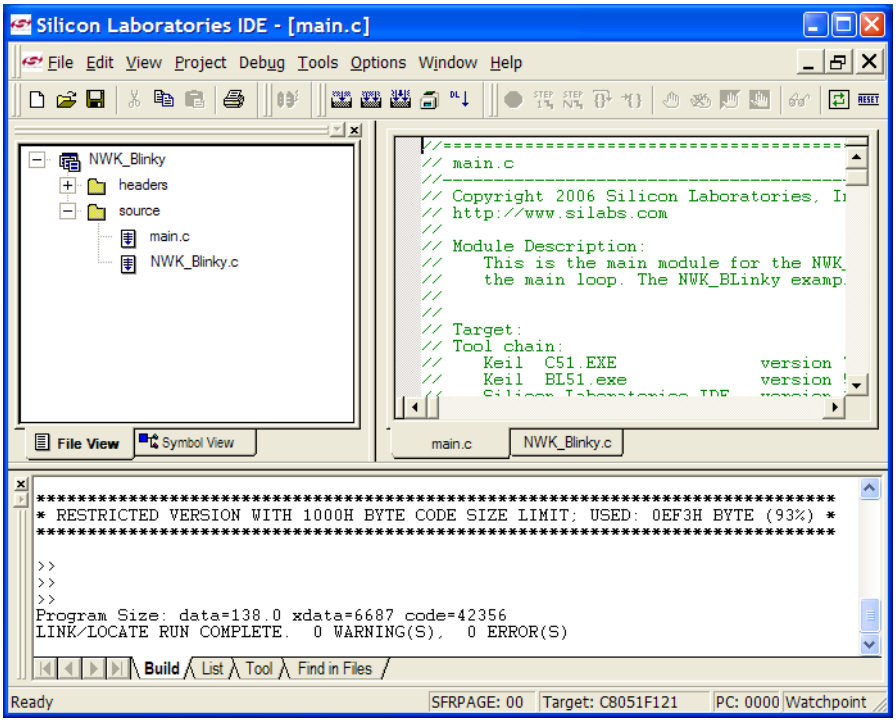

Figure 3. Build Complete, No Errors

- Select “Connect” from the “Debug” menu.
- Then select “Download” also from the debug menu.

The IDE will download the code to the target board. From the File View tab, double-click on the source file. `main.c`. This will open the `main.c` source window in the editor pane. Scroll down in the `main.c` window to locate the `main()` function. Locate the first line of code in the main function. Right click on this line and chose Insert-Breakpoint from the pop-up menu. Click on the green Go button or select Go from the debug window. Code execution will stop at the breakpoint as shown in Figure 4

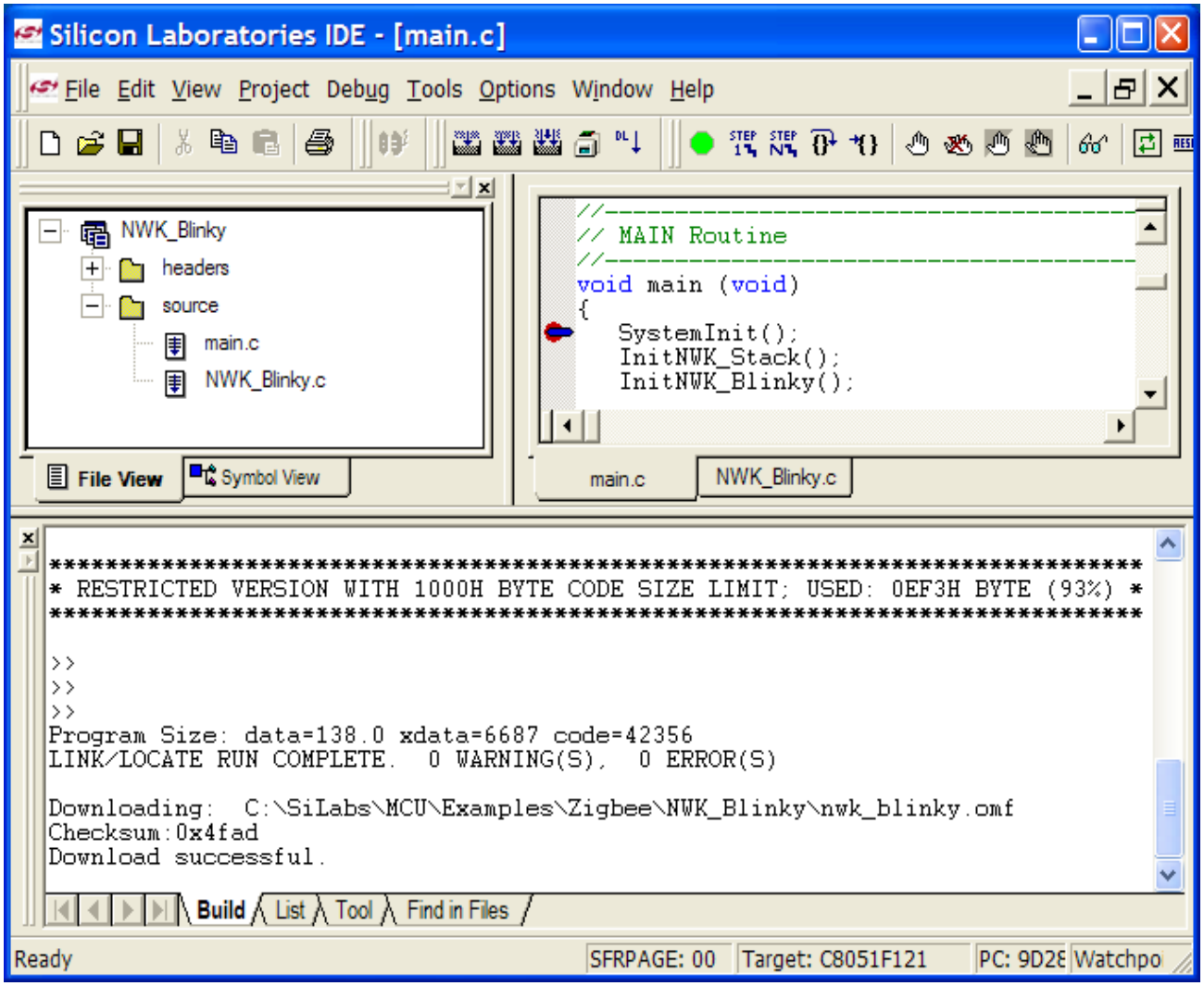

Figure 4. Run to Main

5. Using the Library

The project file includes special provisions to include the ZigBee stack library file in the list of files to be linked. The library file is located in the following directory:

```
Silabs\MCU\Examples\ZigBee\NWK_Blinky\
```

Three external obj/lib files have been added to the build. To view or alter the files to be linked:

- Select Target Build Configuration" from the Project menu.
- Click on the "Customize" button.
- Select the "Files to Link Tab".

Any source files added to the project will automatically add the corresponding object file. Any object or library file not associated with a source file must be added manually. In the NWK_Blinky example you should see three additional files added to the build:

```
ZigbeeNWK.LIB
hal_int0_isr.obj
hal_pca0_isr.obj
```

In addition to the library file there are two object files for essential interrupt service routines that have been added to the project. This ensures that all interrupt driven code will be included from the library.

If you would like to add the library file to another project follow these steps:

- Select "Target Build Configuration" from the Project menu
- Click on the "Customize" button
- Select the "Files to Link Tab"
- Click on the Add External OBJ
- Select "All Files" from the "List files of Type" pull-down menu
- Browse to the library file location
- Select the ZigbeeNWK.lib file
- Repeat for the two interrupt service routine OBJ files

The process for adding the ZigBee library to a project is illustrated in Figure 5.

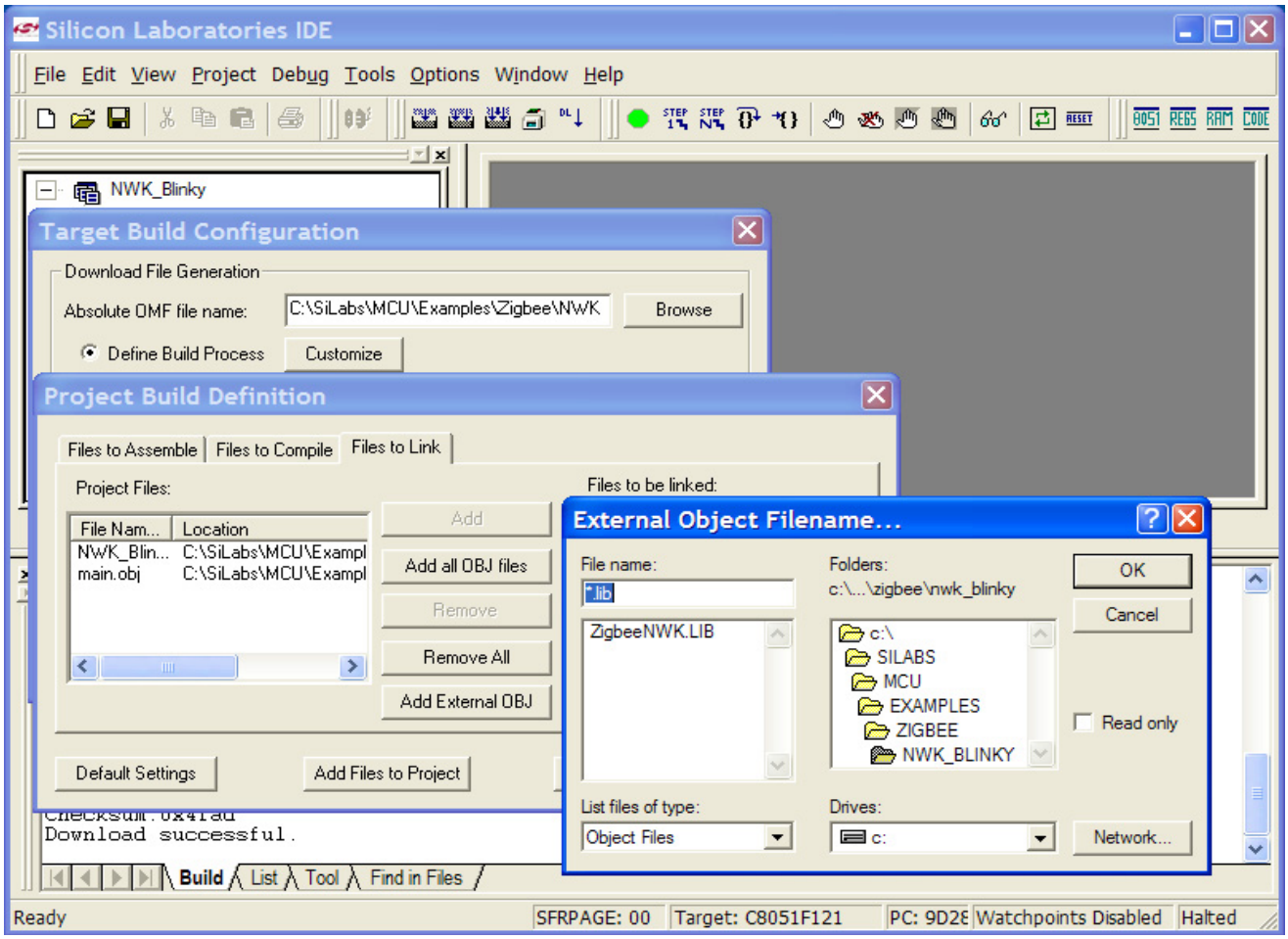

Figure 5. Including an External Library File

6. Library Limitations

- The ZigBee NWK library is restricted to products using Silicon Laboratories MCUs.
- Source code is not provided for the ZigBee Stack.
- Direct Access to the lower layers is not supported.
- The ZigBee NWK library currently only supports the C8051F121.
- The ZigBee NWK library currently does not support code banking.
- The NWK Blinky example uses most of the available code size for the evaluation tools.

The Keil evaluation tools are for evaluation purposes only. The user is limited to non-commercial use only as specified in the license agreement. The user will have to purchase a full set of development tools for commercial development.

Because the ZigBee stack is distributed as a library file, there are certain restrictions associated with the nature of library files. The library itself is a collection of compiled modules. The modules have been compiled specifically for the C8051F121 and the ZigBee development kit hardware. Due to licensing restrictions, source code is not distributed.

Because the NWK Blinky example code uses most of the code size available for the evaluation tools, it may be impractical to make substantial changes without a full version of the Keil tools. Many ZigBee NWK level applications may require more that 4 kB of code. In any case, the user is obligated under the license agreement to purchase a full license for commercial distribution. The purpose of the evaluation tools and this example code is for evaluation purposes only.

Only the NWK function calls are supported. Access to the internal functions and the lower MAC and PHY layers are not supported. An 802.15.4 MAC will be available in a separate 802.15.4 development kit.

Please refer to the ZigBee NWK specification for a functional description of the ZigBee Network layer and NWK primitives. Please refer to "AN241: NWK API Users Guide" for additional information on using the library NWK function calls.

DOCUMENT CHANGE LIST

Revision 0.1 to Revision 0.2

- Updated paths throughout the document.
- Updated screenshots to reflect new paths throughout document.

CONTACT INFORMATION

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701

www.silabs.com