



Porting Freescale 802.15.4 MAC and Bee-Stack to the Wi.Freestar Module

Overview

This document explains the basic steps required to port the Freescale 802.15.4 MAC and Bee-Stack for use on the Wi.Freestar module. It does not attempt to explain the inner workings of the Freescale 802.15.4 MAC, Freescale Bee-Stack, or Wi.Freestar module.

The instructions defined here demonstrate adding the Wi.Freestar hardware definition to the 802.15.4 MAC PHY layer as well as the Freescale sample Home Controls Lighting project. The revision of Bee-Stack used was 1.0-1.1.0, which can be acquired from Freescale and uses revision 1.06 of the Freescale 802.15.4 MAC.

It is assumed the reader is familiar with C language programming development on the Freescale MC9S08 family of microcontrollers using the Metrowerks CodeWarrior IDE, and that Bee-Stack has been installed on the development system in the default directory. The version of CodeWarrior used here was 3.1, Full Edition. The Wi.Freestar module PCB revisions supported are B and C.

The three steps to porting Bee-Stack to the Wi.Freestar module are:

1. Port the 802.15.4 MAC PHY layer to the Wi.Freestar module. The result is a new library which is later included in Wi.Freestar projects utilizing the Freescale 802.15.4 MAC or Bee-Stack.
2. Modify the specific Bee-Stack ZigBee project source files required to handle any associated I/O.
3. Test the functionality of the resulting executable.

There is also a section containing information concerning development with the Wi.Freestar module.

It is a good idea to back up the Bee-Stack folder before beginning the port, thereby preserving the option to compare or restore files as well as perform another port in the future. This will result in an additional folder such as:

C:\F8W\FS-1.0-1.1.0\COPY of Bee-Stack

Step 1 – Porting the 802.15.4 MAC PHY Layer

The code for the PHY layer resides in C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\1.06. The first action in this step is to add a new target for the Wi.Freestar module. Begin by opening project:

Freescale_802.15.4_PHY_SW_1.06_MC_V31.mcp

in folder:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\1.06\mcp

and create a new target for the Wi.Freestar module. To do this (see **Figure 1**):

1. Choose the Targets tab in the project window.
2. From the Project menu, select Create Target.
3. Enter an appropriate name for the Wi.Freestar module library (such as **802.15.4_PHY_FS01.Lib** as shown).
4. Clone the existing target **802.15.4_PHY_RD01.Lib** (as shown).



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

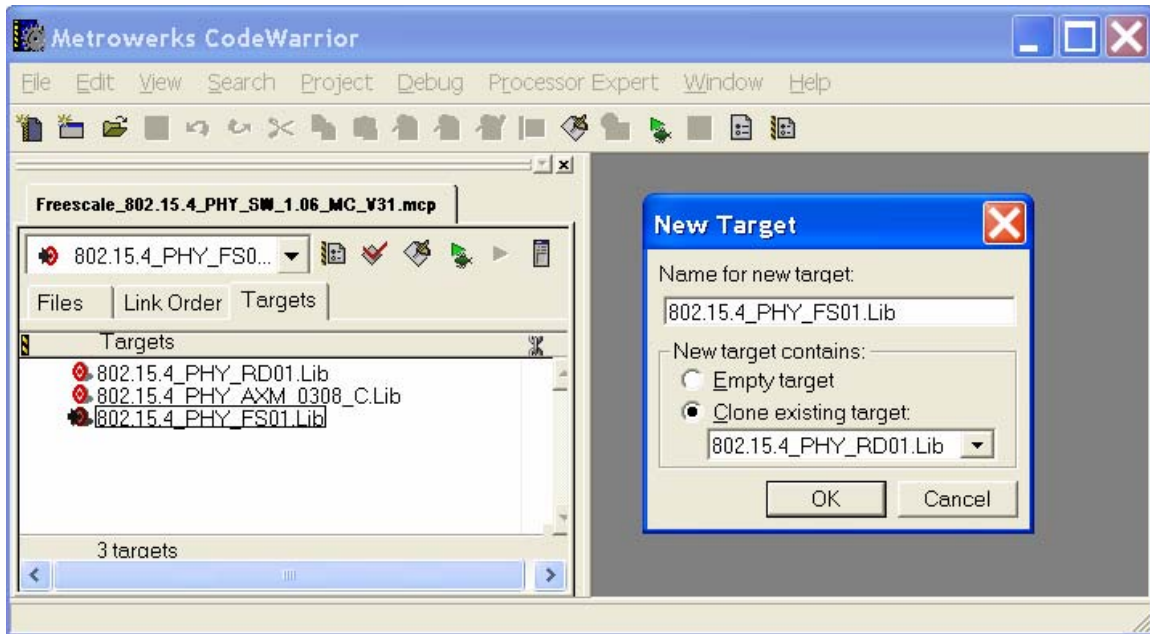


Figure 1 – Creating the Wi.Freestar Target (target already created in example above)

After creating the target, select it in the project window drop down list and open the Settings dialog (choose the icon to the right of the target drop down list or from the Edit menu, item 802.15.4_PHY_FS01.Lib Settings...). Modify the Libmaker settings as shown in **Figure 2** and enter **..\Libs\802.15.4_PHY_FS01.Lib** as the Library Filename for the output of the library build. This will place the resulting library in this folder:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Libs

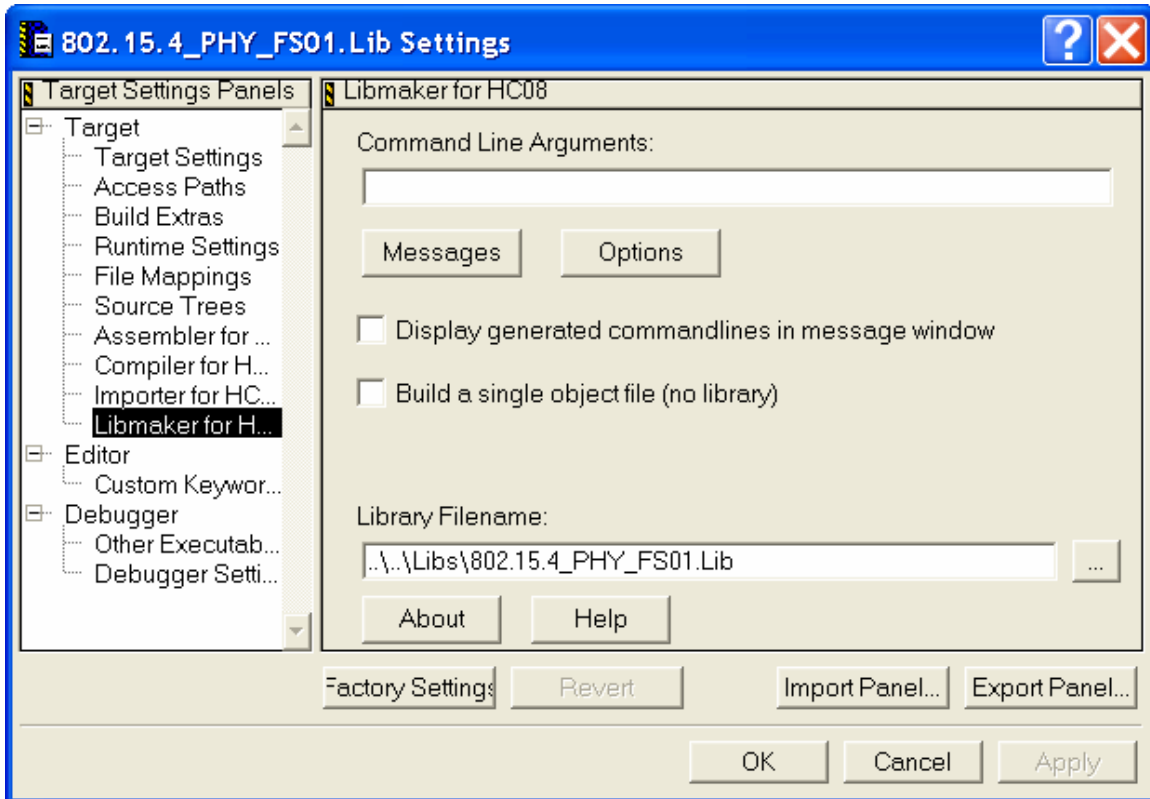


Figure 2 – Setting Up Options for Libmaker

The Compiler for HC08 options can also be accessed from the Settings Dialog. Locate the command line argument **-DTARGET_SARD** and replace it with **-DTARGET_FS01** (per **Figure 3**) to differentiate the Wi.FreeStar module build. This completes changes to the project settings.

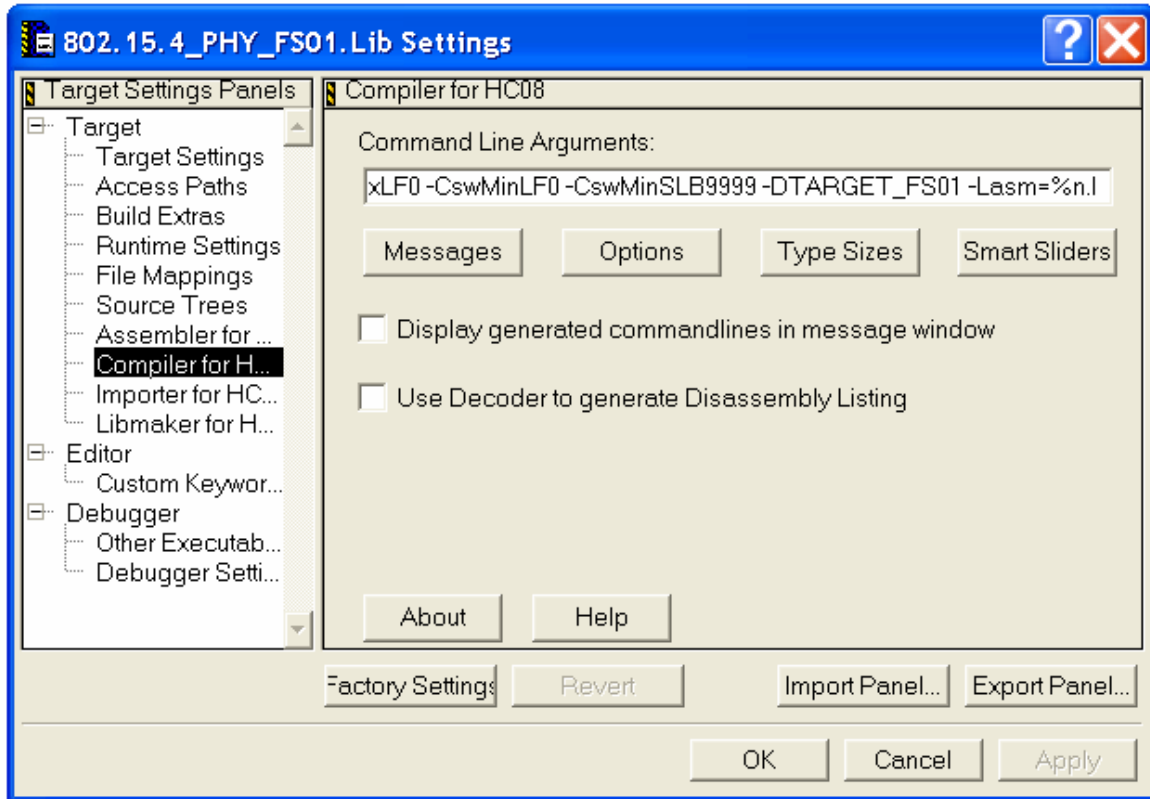


Figure 3 – Setting Up Options for Compiler

Now the source files can be modified for the correct hardware platform.

Begin by opening Target.h, which can be found under the Ghdr group under the Files tab of the project window. Near line number 30 is a line similar to the following:

```
#if !(defined(TARGET_DIG528_2) || defined(TARGET_DIG536_2) || defined(TARGET_DIG534_1)
|| defined(TARGET_AXIOM_GB60) || defined(TARGET_RD01) || defined(TARGET_FS01))
```

Modify it so it appears as above, with the “ || defined(TARGET_FS01)” at the end. This is to avoid the PHY compilation from defaulting to the incorrect platform.

Below this section:

```
#if defined TARGET_DIG536_2 || defined TARGET_DIG528_2 || defined TARGET_RD01 // 13192-
SARD or 13192-EVB
...
(code removed)
...

#endif TARGET_DIG536_2 || TARGET_DIG528_2 || defined TARGET_RD01
```

insert the following code:



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronix.com

```

#if defined TARGET_FS01 // Wi.Freestar

// Define HW pin mappings
#define ABEL_PORT1      PTCB
#define ABEL_ATT_PIN    (1<<2)
#define ABEL_RxTx_PIN   (1<<3)
#define ABEL_RESET_PIN  (1<<4)

#define ABEL_PORT2      PTBD
#define ABEL_GPIO1_PIN  (1<<4)
#define ABEL_GPIO2_PIN  (1<<5)

#define ABEL_PORT3      PTDD
#define ABEL_ANT_SWITCH_PIN_TXEN (1)
#define ABEL_ANT_SWITCH_PIN_RXEN (1<<1)
#define ABEL_PA_EN      (1<<2)

// Define HW port mappings (see App_Target.h)
//
// we will initialize our I/O elsewhere, so just affect the pins required by the MAC

// Port A - not used by PHY layer. Reinitialize in application if needed.
//
// start as output low
#define mSETUP_PORT_A //PTAD = 0x00;\
                    //PTAPE = 0x00;\
                    //PTADD = 0xff; // Set as output

// Port B - used by PHY layer. Reinitialize in application if needed.
//
// set unused analog inputs as output low
#define mSETUP_PORT_B PTBD &= 0xB3;\
                    PTBPE &= 0xB3;\
                    PTBDD |= 0x4C;\
                    PTBD &= ~(ABEL_GPIO1_PIN | ABEL_GPIO2_PIN);\
                    PTBPE &= ~(ABEL_GPIO1_PIN | ABEL_GPIO2_PIN);\
                    PTBDD &= ~(ABEL_GPIO1_PIN | ABEL_GPIO2_PIN); // Set as input

// Port C - used by PHY layer. Reinitialize in application if needed.
//
// start LEDs as output high
#define mSETUP_PORT_C PTCB |= 0xc0;\
                    PTCPE &= 0x3f;\
                    PTCDD |= 0xc0;\
                    PTCB &= ~(ABEL_RESET_PIN | ABEL_ATT_PIN | ABEL_RxTx_PIN);\
                    PTCPE &= ~(ABEL_RESET_PIN | ABEL_ATT_PIN | ABEL_RxTx_PIN);\
                    PTCDD |= (ABEL_RESET_PIN | ABEL_ATT_PIN | ABEL_RxTx_PIN);

// outputs

// Port D - used by PHY layer. Reinitialize in application if needed.
//
// start internal pins as output low
#define mSETUP_PORT_D PTDD &= 0x1F;\
                    PTDPE &= 0x1F;\
                    PTDDD |= 0xE0;\
                    PTDD &= ~(ABEL_ANT_SWITCH_PIN_TXEN |
ABEL_ANT_SWITCH_PIN_RXEN | ABEL_PA_EN);\
                    PTDPE &= ~(ABEL_ANT_SWITCH_PIN_TXEN |
ABEL_ANT_SWITCH_PIN_RXEN | ABEL_PA_EN);\
                    PTDDD |= (ABEL_ANT_SWITCH_PIN_TXEN |
ABEL_ANT_SWITCH_PIN_RXEN | ABEL_PA_EN); // outputs

// Port E - used by PHY layer. Reinitialize in application if needed.
//
// start internal pins as output low

```



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronix.com

```

#define mSETUP_PORT_E    PTED  &= 0x3F;\
                        PTEPE &= 0x3F;\
                        PTEDD |= 0xC0; // Set as output

// Port F - not used by PHY layer.  Pins internal on device.
//
// start internal pins as output low
#define mSETUP_PORT_F    PTFD   = 0x00;\
                        PTFPE = 0x00;\
                        PTFDD = 0xff; // Set as output

// Port G - used by PHY layer for clock input and background debug mode.
//
// start internal pins as output low
#define mSETUP_PORT_G    PTGD   &= 0x0F;\
                        PTGPE &= 0x0F;\
                        PTGDD |= 0xF0; // Set as output

// // for RF test modes only:
// #define MC13192_CE      PTED_PTED2
// #define MC13192_CE_PORT PTEDD_PTEDD2
// #define SETUP_PA_ENABLE { PTDDD_PTDDD2 = 1; }

// Define HW macros - DO NOT EDIT
#define HWAssertAbelReset  ABEL_PORT1 &= ~ABEL_RESET_PIN; // Reset = 0;
#define HWDeAssertAbelReset ABEL_PORT1 |=  ABEL_RESET_PIN; // Reset = 1;

#define RxTxEnable         ABEL_PORT1 |=  ABEL_RxTx_PIN; // RxTxEnable = 1;
#define RxTxDisable       ABEL_PORT1 &= ~ABEL_RxTx_PIN; // RxTxDisable = 0;

#define AttEnable          ABEL_PORT1 |=  ABEL_ATT_PIN; // Attention Enable = 1;
#define AttDisable        ABEL_PORT1 &= ~ABEL_ATT_PIN; // Attention Disable = 0;

#define IsAbelActive()     (ABEL_PORT2 & ABEL_GPIO1_PIN)
#define IsAbelCrcOk()     (ABEL_PORT2 & ABEL_GPIO2_PIN)
#define IsAbelCcaBusy()   (ABEL_PORT2 & ABEL_GPIO2_PIN)

#define RX_ANTENNE_ENABLED { ABEL_PORT3 &= ~(ABEL_ANT_SWITCH_PIN_TXEN |
ABEL_PA_EN); ABEL_PORT3 |= (ABEL_ANT_SWITCH_PIN_RXEN); }
#define RX_DISABLE_LNA
#define TX_ANTENNE_ENABLED { ABEL_PORT3 &= ~(ABEL_ANT_SWITCH_PIN_RXEN);
ABEL_PORT3 |= (ABEL_ANT_SWITCH_PIN_TXEN | ABEL_PA_EN); }
#define TX_DISABLE_PA     { ABEL_PORT3 &= ~(ABEL_PA_EN); }

#define PHY_TYPE "LSR_FS1"

#endif TARGET_FS01

```

This defines the pin connections between the microcontroller and the MC13192 radio. Save the file and copy it from this folder:

C:\F8W\Fs-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\1.06\Src\Ghdr

to the following folder:

C:\F8W\Fs-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Src\Ghdr

Note: The Freescale MAC and Bee-Stack use two separate but identical copies of this file. It is important to keep these files in sync for Bee-Stack applications should any changes need to be made later.

The above code changes are sufficient to port the Freescale PHY to the Wi.FreeStar module. However, in order to better utilize the power saving feature of the microcontroller, a few more changes must be made.



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronic.com

Locate the routine **InitPHYIO** in the file:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\1.06\Src\Code\Phy\PHY_HW_Setup.c

Under the following lines:

```
// Setup port D
mSETUP_PORT_D
```

insert this code fragment:

```
#ifdef mSETUP_PORT_E
mSETUP_PORT_E
#endif
#ifdef mSETUP_PORT_F
mSETUP_PORT_F
#endif
#ifdef mSETUP_PORT_G
mSETUP_PORT_G
#endif
```

The final action is building the PHY library. Ensure the 802.15.4_PHY_FS01.Lib target is selected in the project window drop down list and build the project using the corresponding project window Make toolbar button, or by making sure the project is currently active and then selecting Make from the Project menu or pressing F7. If all is well, the project will compile without errors or warnings. If this is not the case, fix the issues and try again.

Upon successfully completing this step, the PHY library will be created in this folder:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Libs

The above will only need to be carried out again if there is a change in pin function.

Close the CodeWarrior IDE.

Step 2 – Porting the Bee-Stack ZigBee Project Code

In this document, we will use the Freescale Home Control Lighting example to verify a successful port. This includes altering both projects SLC03394 and SRC03391. The process involves using the new Wi.Freestar module 802.15.4 PHY library and modifying existing header and source files to suit the module.

First port the coordinator. The project file can be found in:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\Projects\HomeLighting\SLC03394\MC13192

And the project file is:

ZStack_SLC03394.mcp

Open the project file. Note that this project may need to be converted to the latest version of the IDE.

Begin by creating a new target for the Wi.Freestar module. To do this (see **Figure 1** for reference, but use the text shown below):

1. Choose the Targets tab in the project window.
2. From the Project menu, select Create Target.



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

3. Enter an appropriate name for the Wi.Freestar module coordinator (such as **Coordinator - FS01**).
4. Clone the existing target **Coordinator - GT60 - DIG536**.

After creating the target, select it from the project window drop down list and open the Settings dialog (choose the icon to the right of the target drop down list or from the Edit menu, item Coordinator - FS01 Settings...). Modify the Linker settings as shown in **Figure 4** and enter **ZStack_SLC03394 Coordinator - GT60 FS01.abs** as the Application Filename for the output of the build.

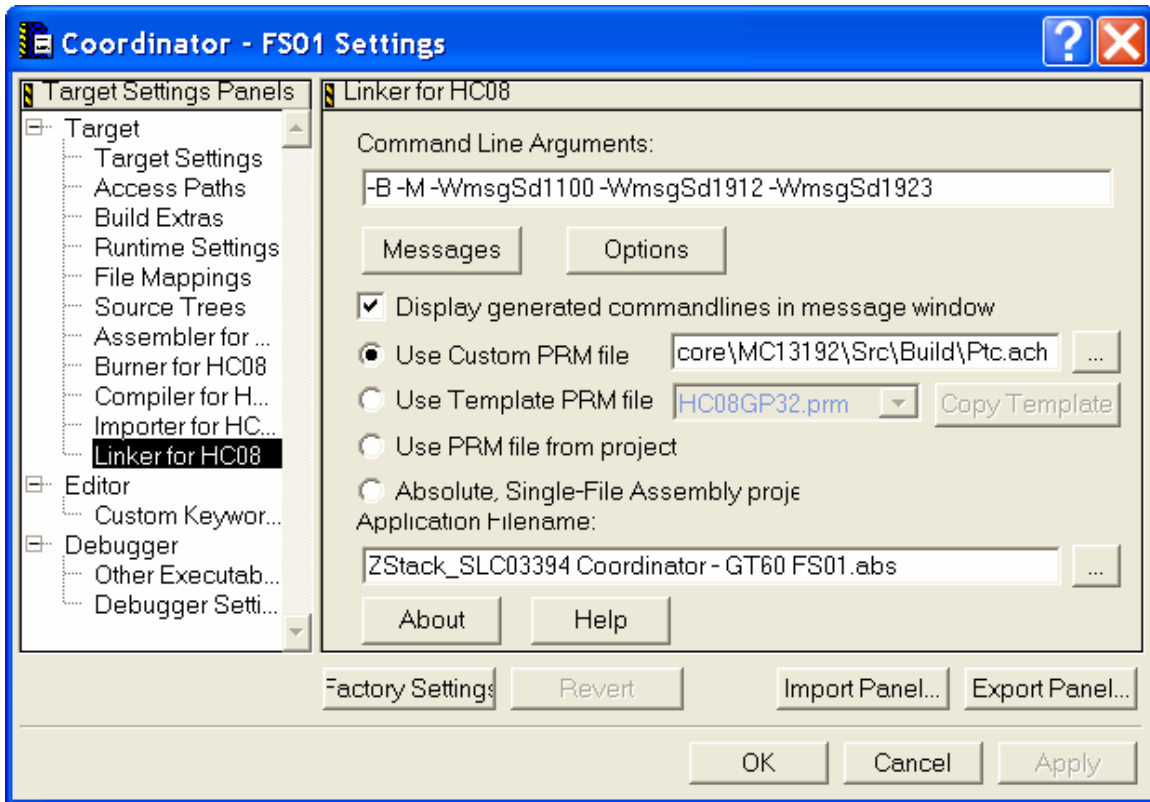


Figure 4 – Setting Up Options for Linker

As before, modify the compiler command line arguments (see **Figure 3** for technique), replacing both arguments **-DTARGET_DIG536_2** and **-DGT60_SARD** with **-DTARGET_FS01** and **-DGT60_FS01**.

Select the Files tab in the project window, open the MAC folder, and then the Libs subfolder (refer to **Figure 5**). Right click on the Libs folder and select Add Files..., then browse to

802.15.4_PHY_FS01.Lib

which should have been placed in the following directory:

C:\F8W\Fs-1.0-1.1.0\Bee-Stack\MAC_core\Mc13192\Libs

Select this library by clicking to the right of it in the target column (column directly to the right of the Data column with a red symbol above it) until a dot appears. In a similar fashion, click on the target column dots as necessary until **only** the 802.15.4_PHY_FS01.Lib and 802.15.4_MAC_FFDPNBV.Lib are included (dots in the target columns indicate they are included, while no dot and “n/a” in the Code and Data columns indicate they are not). Refer to **Figure 5**.



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

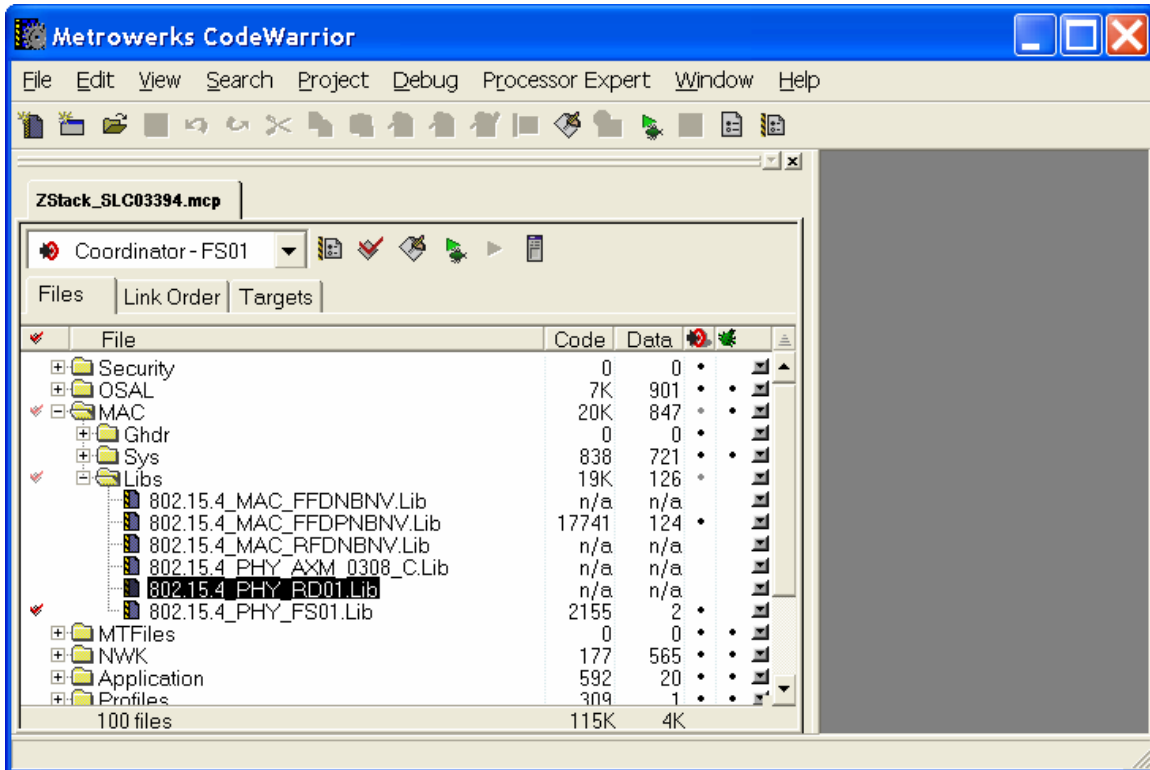


Figure 5 – Adding the New PHY Library

Now that the project settings are complete, the necessary source files specific to the Wi.Freestar module may be modified. Note that in the majority of the cases, this consists of locating the text

-DTARGET_DIG536_2 and -DGT60_SARD

and adding or modifying sections to correspond with

-DTARGET_FS01 and -DGT60_FS01.

Here is a list of the affected files for the ported version:

```
C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Src\Ghdr\Target.h
C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Src\Ghdr\App_Target.h
C:\F8W\FS-1.0-1.1.0\Bee-Stack\OSAL\MTSPCI.c
C:\F8W\FS-1.0-1.1.0\Bee-Stack\ZMain\OnBoard.h
C:\F8W\FS-1.0-1.1.0\Bee-Stack\ZMain\OnBoard.c
```

File C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Src\Ghdr\Target.h should have already been copied over from the 802.15.4 MAC conversion. However, App_Target.h (in the same directory) requires some editing. It can be opened by double clicking its name under the Ghdr subfolder in the MAC folder within the project window (Files tab).

Near line number 35 in App_Target.h is a line similar to the following:

```
#if !(defined(TARGET_DIG528_2) || defined(TARGET_DIG536_2) || defined(TARGET_DIG534_1)
|| defined(TARGET_AXIOM_GB60) || defined(TARGET_RD01) || defined(TARGET_FS01))
```



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

Modify it so it appears as above, with the “ || defined(TARGET_FS01)” at the end. This is to avoid the Bee-Stack compilations from defaulting to the incorrect platform.

Below this section:

```
#if defined TARGET_DIG536_2 || defined TARGET_DIG528_2 || defined TARGET_RD01 // 13192-
SARD or 13192-EVB
...
(code removed)
...

#endif TARGET_DIG536_2 || TARGET_DIG528_2 || defined TARGET_RD01
```

insert the following code:

```
/******
//*****

#if defined TARGET_FS01 // Wi.Freestar

#define LED_PORT PTCB
#define LED1_PIN (1<<6)
#define LED2_PIN (1<<7)
#define LED_MASK (LED1_PIN | LED2_PIN)

#define LED1ON LED_PORT &= ~LED1_PIN;
#define LED1OFF LED_PORT |= LED1_PIN;
#define LED1TOGGLE LED_PORT ^= LED1_PIN;

#define LED2ON LED_PORT &= ~LED2_PIN;
#define LED2OFF LED_PORT |= LED2_PIN;
#define LED2TOGGLE LED_PORT ^= LED2_PIN;

#define LED3ON
#define LED3OFF
#define LED3TOGGLE

#define LED4ON
#define LED4OFF
#define LED4TOGGLE

// PCB switches defines
#define SWITCH_PORT 0
#define mSWITCH1_MASK 0
#define mSWITCH2_MASK 0
#define mSWITCH3_MASK 0
#define mSWITCH4_MASK 0
#define mSWITCH_MASK (mSWITCH1_MASK | mSWITCH2_MASK | mSWITCH3_MASK | mSWITCH4_MASK)

#define mSWITCH_PORT_GET ((SWITCH_PORT & mSWITCH_MASK) ^ mSWITCH_MASK)

// for testing
#undef LED3ON
#undef LED3OFF
#undef LED3TOGGLE
#undef LED4ON
#undef LED4OFF
#undef LED4TOGGLE
#undef SWITCH_PORT
#undef mSWITCH1_MASK
#undef mSWITCH2_MASK
#undef mSWITCH3_MASK
```



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

```
#undef mSWITCH4_MASK

#define LED3ON      LED1ON
#define LED3OFF     LED1OFF
#define LED3TOGGLE  LED1TOGGLE
#define LED4ON      LED2ON
#define LED4OFF     LED2OFF
#define LED4TOGGLE  LED2TOGGLE
#define SWITCH_PORT PTAD
#define mSWITCH1_MASK 0x40
#define mSWITCH2_MASK 0x80
#define mSWITCH3_MASK 0
#define mSWITCH4_MASK 0

// Define HW port mappings
// Port A - not used by PHY layer. Reinitialize in application if needed.
// #define mAPP_SETUP_PORT_A //PTAD = 0x00;\
// //PTAPE = mSWITCH_MASK;\
// //PTADD = 0x00; // Set as input
// for testing:
#define mAPP_SETUP_PORT_A PTAD &= ~mSWITCH_MASK;\
PTAPE |= mSWITCH_MASK;\
PTADD &= ~mSWITCH_MASK; // Set as input

// Port B - used by PHY layer. Remember to AND or OR new values to preserve original
settings.
#define mAPP_SETUP_PORT_B //PTBD = 0x00;\
//PTBPE = 0x00;\
//PTBDD = 0x00;

// Port C - used by PHY layer. Remember to AND or OR new values to preserve original
settings.
//
// PHY initializes LED ports as output
#define mAPP_SETUP_PORT_C //PTCD = 0x00;\
//PTCPE = 0x00;\
//PTCDD = 0x00;

// Port D - not used by PHY layer. Reinitialize in application if needed.
#define mAPP_SETUP_PORT_D //PTDD = 0x00;\
//PTDPE = 0x00;\
//PTDDD = LED_MASK;

// Port E - not used by PHY layer. Reinitialize in application if needed.
#define mAPP_SETUP_PORT_E //PTED = 0x00;\
//PTEPE = 0x00;\
//PTEDD = 0x00;

// Port F - not used by PHY layer. Reinitialize in application if needed.
#define mAPP_SETUP_PORT_F //PTFD = 0x00;\
//PTFPE = 0x00;\
//PTFDD = 0x00;

#endif TARGET_FS01
```

To help alleviate confusion, LED numbers referenced in the code will be named “LEDx” and physical LEDs residing on the Wi.Freestar module will be named “FS_LEDx” in this document.

Note that since the Wi.Freestar module only has two LEDs, LED1 and LED3 have been paired, as have LEDs LED2 and LED4. LED1 and LED3 correspond to FS_LED2 (red) on the Wi.Freestar module, and LED2 and LED4 correspond to FS_LED3. Also, in order to simplify testing with the Home Controls Lighting projects, SWITCH1 has been routed to PTA6 and SWITCH2 to PTA7. The LED and switch



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronix.com

functions are up to the user to select and could be handled by serial command, etc. Also note that in order to use the SWITCHx functionality as it stands for the demo, an appropriate momentary push button circuit will have to be connected in such a way as to mimic the switch actuation on the SARD and EVB boards.

The next file to modify is C:\F8W\FS-1.0-1.1.0\Bee-Stack\OSAL\MTSPCI.c, located under the OSAL folder. Near the top of the file are lines similar to:

```
#ifndef WIN32
  #if defined ( GB60_REVB ) || defined ( GB60_REVC ) || defined ( GT60_ARD ) || defined
( GT60_SARD ) || defined ( GT60_EVB ) || defined ( GT60_FS01 )
    #include "gb60_io.h"
  #endif
#endif
```

Make sure it matches the above, with the “ || defined (GT60_FS01)” at the end to include the appropriate microprocessor header file.

Further down in the file is routine **MT_SerialInit**. Within this routine are lines similar to the following:

```
#if defined( GB60_REVB ) || defined( GB60_REVC ) || defined( GT60_SARD ) || defined(
GT60_EVB ) || defined( GT60_FS01 )
  // Setup port E bit 0 (TDX) as output and set low
  // Enable the pull up on the TDX line
  __asm BSET 0,0x11;
#endif
```

Modify it so it matches the above text, with the “ || defined (GT60_FS01)” at the end.

Next open file C:\F8W\FS-1.0-1.1.0\Bee-Stack\ZMain\OnBoard.h, found within the ZMain folder. As with the other source file modifications presented here, these assume changes are made from the beginning to the end of the file.

Locate the text (near line 68):

```
// Serial Ports Settings
#if defined (GT60_ARD) || defined (GT60_SARD)
  #if ((defined (ZAPP_P1) && defined (ZTOOL_P1)) ||\
      (defined (ZAPP_P1) && defined (ZTOOL_P2)) ||\
      (defined (ZAPP_P2) && defined (ZTOOL_P1)) ||\
      (defined (ZAPP_P2) && defined (ZTOOL_P2)))
    #error Only 1 Port available for ARD and SARD
  #endif
#endif
```

and replace it with:

```
// Serial Ports Settings
#if defined (GT60_ARD) || defined (GT60_SARD) || defined (GT60_FS01)
  #if ((defined (ZAPP_P1) && defined (ZTOOL_P1)) ||\
      (defined (ZAPP_P1) && defined (ZTOOL_P2)) ||\
      (defined (ZAPP_P2) && defined (ZTOOL_P1)) ||\
      (defined (ZAPP_P2) && defined (ZTOOL_P2)))
    #error Only 1 Port available for ARD, SARD, and Wi.Freestar module
  #endif
#endif
```



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronic.com

Just a few lines below is the text (now near line 83):

```
#if defined (GT60_SARD)
  #if defined (ZTOOL_P2) || defined (ZAPP_P2)
    #error port 2 is not available for SARD
  #endif
#endif
```

which should be replaced with:

```
#if defined (GT60_SARD) || defined (GT60_FS01)
  #if defined (ZTOOL_P2) || defined (ZAPP_P2)
    #error port 2 is not available for SARD or Wi.Freestar module
  #endif
#endif
```

Under the section:

```
/* SARD */
#if defined (GT60_SARD)
  #if defined (SERIAL_XFER)
    #define ZAPP_P1
  #else
    #define ZTOOL_P1
  #endif
#endif
```

add (now near line 121):

```
/* Wi.Freestar module */
#if defined (GT60_FS01)
  #if defined (SERIAL_XFER)
    #define ZAPP_P1
  #else
    #define ZTOOL_P1
  #endif
#endif
```

Further down is a comment (near line 304):

```
// Eval board switchs (keys)
```

Under the

```
#elif defined( GT60_SARD ) || defined ( GT60_EVB )
  #define EVAL_SW_MASK 0x3C
  #define EVAL_SW4 0x20
  #define EVAL_SW3 0x10
  #define EVAL_SW2 0x08
  #define EVAL_SW1 0x04
```

code block within this section, add (near line 323):

```
#elif defined( GT60_FS01 )
  #define EVAL_SW_MASK 0x00
  #define EVAL_SW4 0x00
  #define EVAL_SW3 0x00
  #define EVAL_SW2 0x00
  #define EVAL_SW1 0x00
```



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronic.com

```
// for testing:
#undef EVAL_SW_MASK
#undef EVAL_SW4
#undef EVAL_SW3
#undef EVAL_SW2
#undef EVAL_SW1
#define EVAL_SW_MASK 0xc0
#define EVAL_SW4 0x00
#define EVAL_SW3 0x00
#define EVAL_SW2 0x80 // PTA7
#define EVAL_SW1 0x40 // PTA6
```

before the `#else` that ends that code block.

Finally open file C:\F8W\FS-1.0-1.1.0\Bee-Stack\ZMain\OnBoard.c, found within the ZMain folder. Again, assume changes are made from the beginning to the end of the file.

Locate the text similar to the following (near line 29):

```
#elif defined( GT60_SARD ) || defined ( GT60_EVB ) || defined ( GT60_FS01 )
```

and modify it to include the “ `|| defined (GT60_FS01)` ” at the end.

Around line 102 is the following line:

```
#elif defined( GT60_SARD ) || defined( GT60_EVB )
```

Under this section that defines the light and buzzer signal controls for the SARD and EVB, add the following section (near line 118):

```
#elif defined( GT60_FS01 )
#define TurnOnLED1 __asm bclr 6,0x08 // port C.6
#define TurnOnLED2 __asm bclr 7,0x08 // port C.7
#define TurnOnLED3 TurnOnLED1
#define TurnOnLED4 TurnOnLED2
#define TurnOffLED1 __asm bset 6,0x08
#define TurnOffLED2 __asm bset 7,0x08
#define TurnOffLED3 TurnOffLED1
#define TurnOffLED4 TurnOffLED2

#define TurnOnBuzzer
#define TurnOffBuzzer

#define TurnOnBigLight
#define TurnOffBigLight
```

before the last `#else` ending this block of code.

Near line 151 is a section that begins as follows:

```
#if defined ( FS_RX_FLOWCONTROL )
// RX RTS Initialize
```

which defines the serial port flow control (if enabled). Under the `GT60_SARD` section add the following lines (near line 163):



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

```
#elif defined ( GT60_FS01 )
#define SERIAL_PORT1_RX_RTS_PORTENABLE    __asm bclr 0,0x09 // PTC0 - PTCPE
#define SERIAL_PORT1_RX_RTS_PORTDIRECTION __asm bset 0,0x0b // PTC0 - PTCDD
#define SERIAL_PORT2_RX_RTS_PORTENABLE    // NO Flow Control
#define SERIAL_PORT2_RX_RTS_PORTDIRECTION // NO Flow Control
```

Near line 180 is a section that begins with the following comment:

```
// RX RTS Enable
```

Under the GT60_SARD code within this section, add the following lines (near line 187):

```
#elif defined ( GT60_FS01 )
#define SERIAL_PORT1_RX_RTS_ENABLE    __asm bclr 0,0x08 //PTC0
#define SERIAL_PORT2_RX_RTS_ENABLE    // NO Flow control
```

Similarly, right below it is a section that begins with the following comment:

```
// RX RTS Disable
```

Again, under the GT60_SARD section add the following lines (near line 205):

```
#elif defined ( GT60_FS01 )
#define SERIAL_PORT1_RX_RTS_DISABLE __asm bset 0,0x08 //PTC0
#define SERIAL_PORT2_RX_RTS_DISABLE // NO Flow control
```

In routine **OnBoard_ProcessEvent** are lines similar to the following:

```
#if defined ( FS_RX_FLOWCONTROL )
#if defined ( GT60_SARD ) || defined( GT60_FS01 )
```

Modify the second line to include the “ || defined (GT60_FS01)” at the end as above.

Update routine **InitBoard** by finding the following section of code beginning with the comment:

```
// Set the direction for the LED Port
```

and add these lines for the Wi.Freestar module under the GT60_SARD and GT60_EVB block:

```
#elif defined( GT60_FS01 )
PTCDD |= 0xc0; // port C.6 and C.7
```

At the start of routine **KBInit** is a section to initialize the keyboard port. Under the GT60_SARD and GT60_EVB block add the following for the Wi.Freestar module:

```
#elif defined ( GT60_FS01 )
// no keyboard
// for testing demo, use PTA6 and PTA7
PTAPE |= 0xc0; // pullups for 6 & 7
PTADD &= 0x3f; // 6 & 7 inputs
```

Within routine **OnBoard_GetKeys** is code to process the key inputs for all platforms. This has been enabled for the testing of the Home Control Lighting demo, but can be disabled by altering which portion of the following code has been commented. Add these lines under the GT60_SARD and GT60_EVB section:



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronics.com

```
#elif defined( GT60_FS01 )
// keys = 0; // no keys (if this applies)
// for demo testing:
keys = ~keys & EVAL_SW_MASK;
```

The last routine to modify is **SerialSetRxFlow**. Within the SERIAL_PORT1 conditional compilation section are two lines like the following:

```
#if defined ( GT60_SARD )
```

Modify both of them to include the Wi.Freestar module as an option:

```
#if defined ( GT60_SARD ) || defined ( GT60_FS01 )
```

That completes the code modifications for Bee-Stack. The next step is to build the coordinator application.

Make sure the Coordinator - FS01 target is selected in the project window drop down list and build the project using the corresponding project window Make toolbar button, or by making sure the project is currently active and then selecting Make from the Project menu or pressing F7. If all is well, the project will compile without errors or warnings. If this is not the case, fix the issues and try again. Keep in mind that the Freescale Home Controls Lighting projects could build with an abundance of informational messages. These can be ignored or displayed in the Errors & Warnings output window by selecting the third button from the left above the text window.

Once the project modifications have been completed successfully, the application can be run on a target Wi.Freestar module. Connect the Background Debugger device to the 6-pin PROGRAM DEBUG header on the Wi.Freestar Interface Board with a module inserted, noting the Pin 1 position, as would be done on the Freescale SARD or EVB boards. Download and run the application as normal. FS_LED2 should light to indicate a network has been formed (this LED is referenced as LED3 in the code). The Freescale routine **ZDO_NetworkFormationConfirmCB** in ZDO\ZDApp.c can be modified to produce a different result indicating a network has formed.

Note that the first time the coordinator and end device modules are powered up after a code download it may be necessary to activate the SWITCH1 input twice (low pulses, FS_LED2 should light, then FS_LED3) to program a random MAC address. This is a function of the Freescale Home Controls Lighting application -- each node on the network must have a unique MAC address for binding to take place.

Fortunately, the majority of the porting work has been completed. Close the CodeWarrior IDE.

The other side of the application is the End Device. The project file can be found in:

```
C:\F8W\FS-1.0-1.1.0\Bee-Stack\Projects\HomeLighting\SRC03391\MC13192
```

And the project file is:

```
ZStack_SRC03391.mcp
```

Open the project file. Note that this project may need to be converted to the latest version of the IDE.

Once again, begin by creating a new target for the Wi.Freestar module. To do this (see **Figure 1** as an example, but use text shown below):

1. Choose the Targets tab in the project window.
2. From the Project menu, select Create Target.



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronix.com

3. Enter an appropriate name for the Wi.Freestar module coordinator (such as **End Device - FS01**).
4. Clone the existing target **End Device - GT60 - DIG536**.

After creating the target, select it from the project window drop down list and open the Settings dialog (choose the icon to the right of the target drop down list or from the Edit menu, item End Device - FS01 Settings...). Modify the Linker settings as shown in **Figure 4** and enter **ZStack_SRC03391 End Device - GT60 FS01.abs** as the Application Filename for the output of the build.

As before, modify the compiler command line arguments (see **Figure 3** for technique), replacing both arguments **-DTARGET_DIG536_2** and **-DGT60_SARD** with **-DTARGET_FS01** and **-DGT60_FS01**.

Select the Files tab in the project window, open the MAC folder, and then the Libs subfolder (refer to **Figure 5**). Right click on the Libs folder and select Add Files..., then browse to

802.15.4_PHY_FS01.Lib

which should have been placed in the following directory:

C:\F8W\FS-1.0-1.1.0\Bee-Stack\MAC_core\MC13192\Libs

Select this library by clicking to the right of it in the target column (column directly to the right of the Data column with a red symbol above it) until a dot appears. In a similar fashion, click on the target column dots as necessary until **only** the 802.15.4_PHY_FS01.Lib and 802.15.4_MAC_RFDPNBNV.Lib are included (dots in the target columns indicate they are included, while no dot and "n/a" in the Code and Data columns indicate they are not). Refer to **Figure 5** for a visual reference but substitute the symbols mentioned above. The end device is the light switch in this application, and therefore uses the Reduced Function Device (RFD) MAC library.

Make sure the End Device - FS01 target is selected in the project window drop down list and build the project using the corresponding project window Make toolbar button, or by making sure the project is currently active and then selecting Make from the Project menu or pressing F7. If all is well, the project will compile without errors, though there may be a warning in function **SRC03391_set_OnOffSRC**. If this is not the case, fix the issues and try again. Keep in mind that the Freescale Home Controls Lighting projects could build with an abundance of informational messages. These can be ignored or displayed in the Errors & Warnings output window by selecting the third button from the left above the text window.

Once the project modifications have been completed successfully, the application can be run on a target Wi.Freestar module. Connect the Background Debugger device to the 6-pin PROGRAM DEBUG header on a second Wi.Freestar Interface Board with a module inserted, noting the Pin 1 position, as would be done on the Freescale SARD or EVB boards.

Reset the Coordinator/Light interface module to form the network. Then download and run the End Device/Light Switch application as normal. FS_LED2 should light to indicate the end device has joined the network (this LED is referenced by LED3 in the code). Note that the Freescale routine **ZDO_StartRouterConfirmCB** in ZDO\ZDApp.c can be modified to produce a different result indicating a network has formed.

Keep in mind that the first time the coordinator and end device modules are powered up after a code download it may be necessary to activate the SWITCH1 input twice (FS_LED2 should light, then FS_LED3) to program a random MAC address. This is a function of the Freescale Home Controls Lighting application -- each node on the network must have a unique MAC address for binding to take place. Unless this step is completed, the devices will not form or join a network.



Step 3 – Final Testing of the Port

Once both the Light and Light Switch boards have FS_LED2 lit, they are ready to be bound. Activate the SWITCH2 input on both boards. FS_LED3 should light on each module, indicating the coordinator and end device are linked. Both boards may have to be powered down and the process repeated to get the demo to function and the devices to bind. Once this has occurred, activating SWITCH1 on the End Device/Light Switch module will toggle FS_LED3 on the Coordinator/Light module. Note that the first activation may not appear because it turns the light on and FS_LED3 will already be lit as an indication of binding. This is due to the double function of the LEDs as a result of pairing LED2 and LED4.

Notes for Further Development

For the 802.15.4 MAC, the microcontroller pin initialization is completed for all unused pins as well as pins used by the radio interface (file Target.h). This leaves maximum flexibility for user applications utilizing the Freescale 802.15.4 MAC on a Wi.Freestar module with the same PHY library. This also means it is up to the user to initialize all I/O available off the module as they see fit for their given application. Generally speaking, any I/O that will not be used should be left physically unconnected and initialized to drive a low output. This, as well as other I/O initialization, can be performed by modifying the changes presented here. Two uses for this method include fixed Wi.Freestar module I/O utilization for all applications as well as varying the I/O initialization and creating unique PHY libraries for each product utilizing the module. These will gain simplicity in initialization (reduced code size, etc.), but will give up PHY library use flexibility. Any method is valid as long as the I/O made available at the Wi.Freestar module connections is properly initialized.

The Bee-Stack demo program initializes a few more available I/O lines for switch inputs, etc. (App_Target.h). This is an example of performing further I/O initialization.

It is very important to note that in order to maintain the module's FCC certification, output power level must be controlled. In particular, no transmission is permitted at supply voltages greater than 3.65V, at least at the maximum output power. These voltages are beyond allowable limits and, therefore, are not specified. Nevertheless, it is still possible for a high voltage to be applied to the board. An example could be a fresh or freshly charged battery pack as the module supply.

The Wi.Freestar module provides a method to determine the module supply voltage through analog input AD1P7 (connected to the output of an onboard 1.25V reference). With the supply voltage at pin V_{REFH} and with pin V_{REFL} tied to ground, the following table lists the ADC readings for this analog input:

Volts (V)	ADC result (decimal)
3.30	387
3.40	376
3.50	365
3.55	360
3.60	355
3.65	350
3.70	346

As shown above, transmission should not be allowed if the ADC result is less than 350 (decimal).



905 Messenger Lane Moore, OK 73160 - P 405.794.7730 - F 405.794.7477 - www.radiotronix.com

If the voltage supplied to the Wi.Freestar module will never reach this maximum level and failsafe measures are in place to ensure this, transmission is always possible. However, there are still maximum allowed output power levels, and these limits particularly affect the upper channels.

Maximum MC13192 Power Level Settings per Channel

The Freescale MC13192 used in the Wi.Freestar module has an output power level register which accepts values 0x00 through 0x0F (0 through 15). Using these values, the current maximum power level settings for 100 mW FCC compliance are as follows:

For channels 0-12 (2405 MHz - 2465 MHz), the maximum setting is 0x0d.

For the remaining channels:

Channel	Maximum Power Level Value
13 (2470 MHz)	0x09
14 (2475 MHz)	0x07
15 (2480 MHz)	0x03

These are the maximum levels, and they can be reduced.

The code fragment used to adjust the MC13192 register for the 802.15.4 MAC revision 1.06 is as follows:

```

unsigned char    ucTempPowerLevel;
aspMsg_t        aspPowLvl;

ucTempPowerLevel = GetAdjustedPowerLevel(ucCurrentChannel);

// MC13192 power level 0 to 15
aspPowLvl.msgType = gAppAspSetPowerLevelReq_c;
aspPowLvl.appToAspMsg.msgData.aspSetPowerLevelReq.powerLevel = ucTempPowerLevel;
// Valid values: [0...0xF]
if ( gSuccess_c == MSG_Send(APP_ASP, &aspPowLvl) )
{
    // okay
}
else
{
    // not okay
}

```