

# Over-the-Air (OTA) Update With the MSP430FR57xx

Mark Swanson

MSP430 Applications

## ABSTRACT

Over-the-air (OTA) updates or wireless firmware updates of the MSP430FR57xx family of devices enable service updates to be performed in the field using the ultra-low power capabilities of the MSP430™ microcontroller family and FRAM technology. The SimpliciTI™ protocol stack is used to implement sub-1GHz communication between the [MSP-EXP430FR5739](#) Experimenter Board and a [CC1111](#) USB RF access point. A simple graphical user interface (GUI) enables simple firmware selection and visual feedback on the progress of the wireless firmware update. This application report described use of the wireless update demonstration, design information of the wireless update process, and steps to begin custom development.

Application collateral and the software referenced in this document can be downloaded from [http://software-dl.ti.com/msp430/msp430\\_public\\_sw/mcu/msp430/FR5739OTA/latest/index\\_FDS.html](http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/FR5739OTA/latest/index_FDS.html).

## Contents

1	Introduction .....	3
2	Getting Started .....	5
3	Wireless Update .....	11
4	Wireless Update Enabled Firmware Development .....	20
5	Wireless Update Benchmarks .....	29
6	References .....	39

## List of Figures

1	Wireless Update System.....	3
2	MSP-EXP430FR5739 Experimenter Board .....	5
3	CC1101EMK868-915.....	5
4	CC1111F32 USB RF Access Point .....	6
5	CC1111 USB RF Access Point Connected to USB Port .....	7
6	MSP-EXP430FR5739 Control Center Launch Screen.....	7
7	MSP-EXP430FR5739 Control Center File Selection.....	8
8	MSP-EXP430FR5739 Control Center Start of Update.....	9
9	Hardware Initialization Sequence for Wireless Update .....	9
10	MSP-EXP430FR5739 Control Center WBSL Download Progress .....	10
11	MSP-EXP430FR5739 Firmware Download Progress .....	10
12	Program Flow for the Wireless Update.....	12
13	Code Composer Studio v5 Splash Screen.....	13
14	Code Composer Studio v5 Workspace Launcher.....	13
15	Import Existing CCS/CCE Eclipse Project .....	14
16	Browse Directory for Import CCS Projects.....	14
17	Import All Existing CCS Projects .....	15
18	Code Composer Studio v5 Build Configuration Selection .....	15

MSP430, SimpliciTI, Code Composer Studio are trademarks of Texas Instruments.  
IAR Embedded Workbench is a registered trademark of IAR Systems.  
All other trademarks are the property of their respective owners.

19	Code Composer Studio v5 Debug .....	16
20	IAR Embedded Workbench® for MSP430™ 5.30 Splash Screen.....	16
21	Add Existing IAR Project .....	17
22	Browse Directory for Add Existing IAR Project.....	17
23	IAR Embedded Workbench® for MSP430™ 5.30 Build Configuration Selection.....	18
24	IAR Embedded Workbench® for MSP430™ 5.30 Debug .....	18
25	Code Composer Studio v5 New CCS Project Selection .....	20
26	Code Composer Studio v5 Project Creation.....	21
27	New CCS Project With CBSL Load Image .....	21
28	Include Load Image in CCS Project .....	22
29	Post Build Step in CCS Project.....	23
30	IAR EW for MSP430 5.30 Create New Project Selection.....	24
31	IAR EW for MSP430 5.30 Project Creation.....	24
32	Select Project Name .....	25
33	Select Target Device.....	25
34	New IAR EW Project With CBSL Debug Image.....	26
35	Download Extra Image in IAR EW .....	26
36	Linker Output File Selection in IAR EW .....	27
37	Linker Configuration File Override in IAR EW .....	27
38	Current Measurement of a Wireless Update .....	29
39	Current Measurement of a Packet Transmission .....	30
40	Power Consumption Test Apparatus .....	31
41	Current Measurement of a Link Event.....	32
42	Current Measurement of a WBSL Transmission.....	33
43	Current Measurement of the First Firmware Packet.....	34
44	Current Measurement of a Reset Vector Packet.....	35

### List of Tables

1	CC1101 Average Current of a Link Event .....	32
2	MSP430FR5739 Average Current of a Link Event .....	32
3	CC1101 Average Current of a WBSL Transmission.....	33
4	MSP430FR5739 Average Current of a WBSL Transmission .....	33
5	CC1101 Average Current of the First Firmware Packet .....	34
6	MSP430FR5739 Average Current of the First Firmware Packet .....	34
7	CC1101 Average Current of Subsequent Firmware Packets .....	34
8	MSP430FR5739 Average Current of Subsequent Firmware Packets .....	35
9	CC1101 Average Current of a Reset Vector Packet .....	35
10	MSP430FR5739 Average Current of a Reset Vector Packet .....	35
11	CC1101 Average Current of a Wireless Update .....	36
12	MSP430FR5739 Average Current of a Wireless Update .....	36
13	FRAM and Flash MSP430 Wireless Update Comparison .....	37
14	Wireless Update MSP430FR5739 Memory Organization.....	37
15	Firmware Memory Sizes for Wireless Update .....	37
16	Radio Timing for Wireless Update .....	38
17	CC RF Configuration for Wireless Update .....	38

## 1 Introduction

The purpose of this document is to provide instructions on how to both use and continue development of a Wireless Update project on the MSP-EXP430FR5739 board (see [Figure 1](#)). FRAM technology on the MSP430FR57xx device family produces significantly reduced write times and power consumption when compared with flash-based memory. Utilizing the low-power characteristics of both the FRAM series of MSP430 devices and the CC1101 sub-1GHz transceiver, the application that is described in this document enables a wireless firmware update process that consumes ultra-low power and enables longer lifetime for embedded systems.



**Figure 1. Wireless Update System**

This document includes sections for the following functional topics:

- Use of the wireless update demonstration
- Description of the wireless update process and associated software projects
- Instructions to begin custom firmware development based on the wireless update firmware
- Benchmarks for the wireless update process

### 1.1 MSP430™ Bootstrap Loader (BSL)

The MSP430 BSL is a program built into an MSP430 device that enables communication with embedded memory in the MSP430 microcontroller during prototyping, final production, and while in service. Both the programmable memory, typically flash, and the data memory, RAM, can be modified by the BSL. Some devices shipped from TI contain a BSL residing either in ROM or flash depending on the device family.

In the case of the MSP430FR5739, the BSL resides in ROM memory and cannot be customized for different interfaces or protocols such as the SimpliciTI network protocol. Instead, the Wireless Update acts as an application which implements the communication with memory much like a BSL while residing in the programmable FRAM of the MSP430FR5739.

Further information regarding the BSL may be found in the following documents:

- *MSP430 Programming Via the Bootstrap Loader (BSL)* ([SLAU319](#))
- *Creating a Custom Flash-Based Bootstrap Loader (BSL)* ([SLAA450](#))

## 1.2 **SimpliciTI™ Network Protocol**

The SimpliciTI protocol is a simple low-power RF network protocol aimed at small RF networks. Such networks typically contain battery operated devices which require long battery life, low data rate and low duty cycle and have a limited number of nodes talking directly to each other or through an access point or range extenders. Access points and range extenders are not required but provide extra functionality such as storing and forwarding messages. With the SimpliciTI protocol, the MCU resource requirements are minimal, which results in reduced system cost.

The SimpliciTI protocol is designed for easy implementation and deployment out-of-the-box on several TI RF platforms such as the MSP430 family of low-power MCUs and the CC1xxx, CC25xx, and CC430 transceivers and SoCs. The sample applications run out-of-the-box on several hardware platforms including the CC1101EM that is used with this wireless update.

Further information regarding the SimpliciTI network protocol may be found at *SimpliciTI Compliant Protocol Stack* (<http://www.ti.com/tool/simpliciti>).

## 1.3 **Wireless Update**

The wireless update procedure allows the user to perform a firmware update of the MSP-EXP430FR5739 Experimenter Board (FRAM board) wirelessly using a CC1101 sub-1GHz RF transceiver. Use of an IDE, USB cable, and the eZ-FET Emulator are not required for the wireless update process to function.

The Core BSL (CBSL) which performs the essential functions for the wireless update process should be looked at as an application rather than a universal BSL. The CBSL is always invoked during a reset of the device and as a result the main application must be created in a specific way to allow proper CBSL functionality. In addition, the CBSL must be modified to comply with local RF regulatory restrictions.

The Wireless BSL (WBSL), which enables the LED routines on the FRAM board, allows for custom functions to be loaded into RAM temporarily during the wireless update procedure. WBSL functions make calls to the CBSL to perform memory write operations. Further functionality, such as LED routines, may be added to the WBSL to meet specific project needs.

## 2 Getting Started

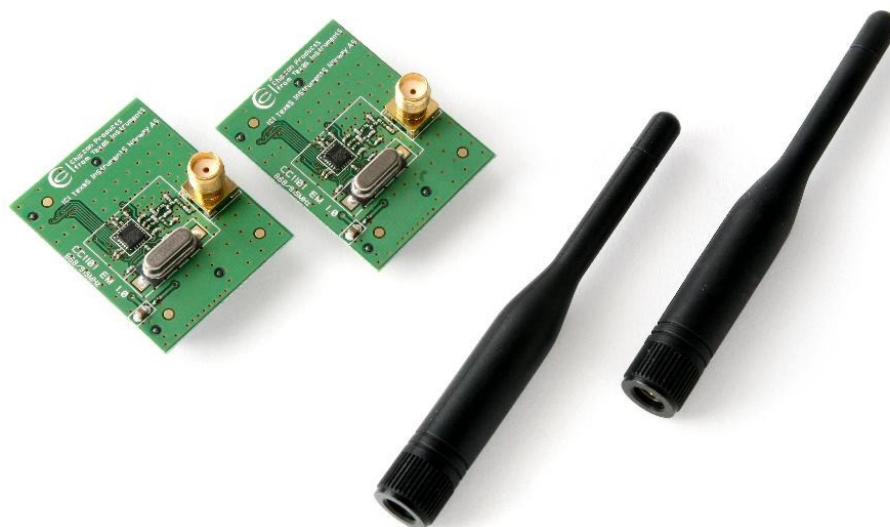
### 2.1 Hardware Requirements

The MSP-EXP430FR5739 Experimenter Board (see [Figure 2](#)) is a development platform for the MSP430FR57xx devices. It supports this new generation of MSP430 microcontroller devices with integrated Ferroelectric Random Access Memory (FRAM). The board is compatible with many TI low-power RF wireless evaluation modules such as the CC1101EM. The MSP430FR5739 is an ultra-low-power MSP430 MCU that features 16KB embedded FRAM.



**Figure 2. MSP-EXP430FR5739 Experimenter Board**

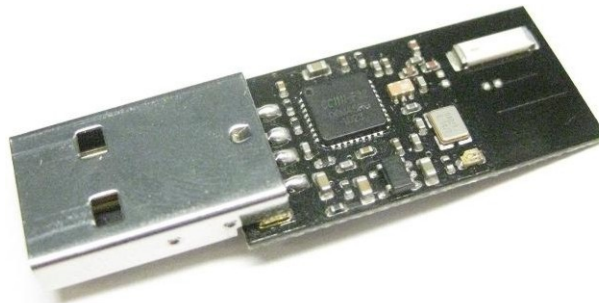
The CC1101 is a low-cost sub-1GHz transceiver designed for very low-power wireless applications. The circuit is mainly intended for the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency bands at 315, 433, 868, and 915 MHz, but can easily be programmed for operation at other frequencies in the 300- to 348-MHz, 387- to 464-MHz, and 779- to 928-MHz bands. The CC1101EMK (see [Figure 3](#)) includes two CC1101EM 868/915 MHz modules and antennas and can be used with the MSP430FR57xx Experimenter Board for software development, prototyping and testing using the MSP430 device.



**Figure 3. CC1101EMK868-915**



The CC1111Fx is a true low-power sub-1GHz system-on-chip (SoC) designed for low power wireless applications (see [Figure 4](#)). The CC1111Fx combines the excellent performance of the state-of-the-art RF transceiver CC1101 with an industry-standard enhanced 8051 MCU, up to 32 KB of in-system programmable flash memory, up to 4 KB of RAM, and many other powerful features. Full-speed USB 2.0 interface allows for quickly interfacing to a PC using the USB interface, and the high data rate (12 Mbps) of the USB interface avoids the bottlenecks of RS-232 or low-speed USB interfaces.



**Figure 4. CC1111F32 USB RF Access Point**

---

**NOTE:** The CC1111 USB RF access point and CC1101EMK must both be operating at a frequency of 915 MHz due to the RF parameters utilized by the Wireless Update. Use of other frequency components does not allow the Wireless Update to function properly.

---

## 2.2 How to Use the Wireless Update

### 2.2.1 Install Demo Application, Drivers, and Firmware

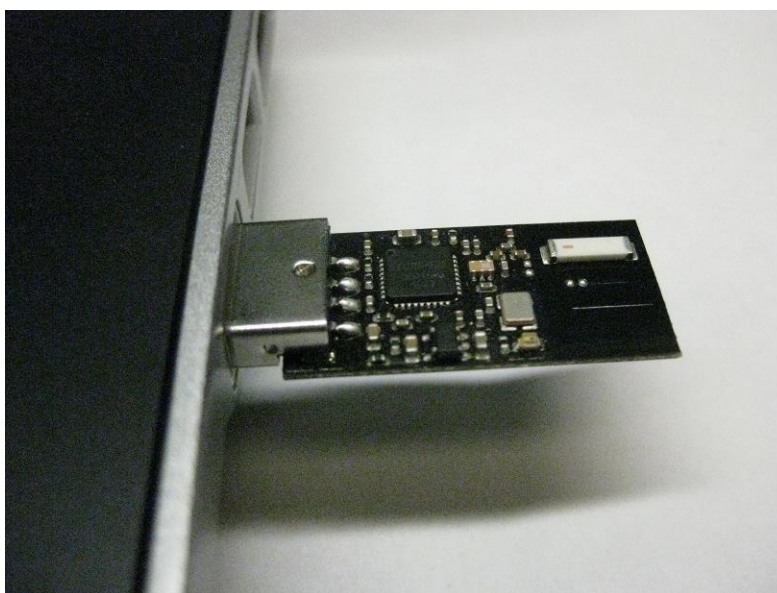
The Wireless Update software only supports Windows PCs.

1. Follow all installation instructions for the eZ430-Chronos PC software found in the *eZ430-Chronos Development Tool User's Guide* ([SLAU292](#)).
2. Download and install the [fr5739ota setup package](#) on the PC. It contains the following file structure to a single directory on the PC:
  - (a) bin
    - (i) EXP430FR5739\_OTA\_CC\_0\_5.exe – Wireless Updater Program
    - (ii) EXP430FR5739\_OTA\_BSL.ini – Specifies WBSL TI-TXT file for the Wireless Updater Program
    - (iii) eZ430\_Chronos\_CC.dll – Enables communication with CC1111 Access Point
    - (iv) MSP-EXP430FR5739\_OTA\_WBSL\_0\_01.txt – WBSL TI-TXT file
    - (v) Single\_Blink\_Demo.txt – Primary Demonstration application for the Wireless Update
    - (vi) Double\_Blink\_Demo.txt – Alternate demonstration application for the Wireless Update
  - (b) src
    - (i) Blink\_Demo – Source code for generating demonstration application output files for the Wireless Update
    - (ii) CCS – Code Composer Studio v5.1 specific project files
    - (iii) GUI Sources – Source code and associated files for the Wireless Updater Program
    - (iv) IAR – IAR Embedded Workbench for MSP430 5.30 specific project files
    - (v) MSP430FR5739\_OTA\_BSL – Source code for generating WBSL and CBSL output files for the Wireless Update

## 2.2.2 Running the Wireless Update

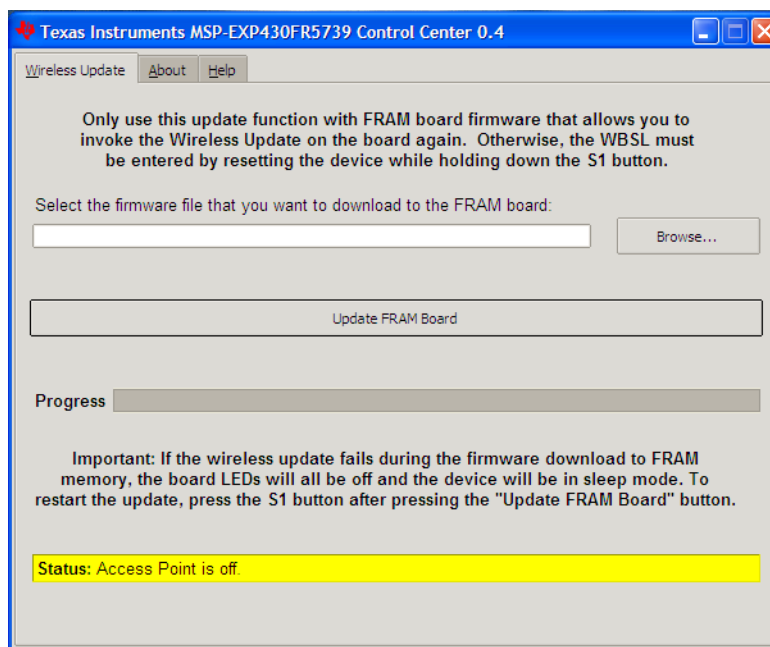
**NOTE:** Instructions for running the Wireless Update assume completion of the installation instructions and firmware which contains the CBSL already executing on the FRAM board. Instructions for downloading a demonstration application containing the CBSL are given in [Section 3.3](#).

1. Connect the USB RF access point to the PC (see [Figure 5](#)). Drivers for the access point should have been installed during installation of the eZ-430 Chronos PC software.



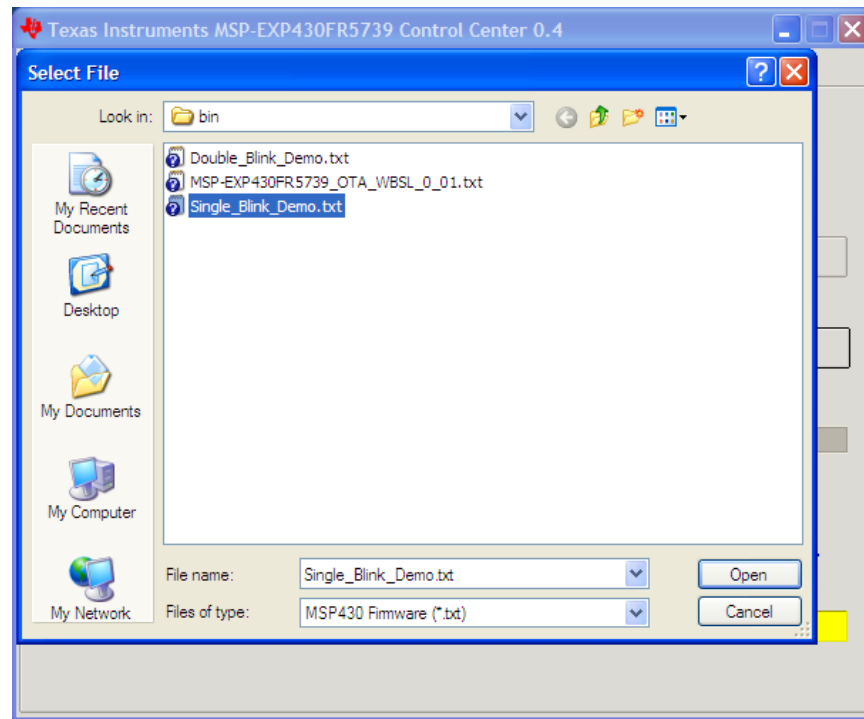
**Figure 5. CC1111 USB RF Access Point Connected to USB Port**

2. Run the MSP-EXP430FR5739 Control Center program from the installed EXE package bin directory on the PC.



**Figure 6. MSP-EXP430FR5739 Control Center Launch Screen**

3. Select an MSP430 firmware Image (.txt file) by clicking Browse (see [Figure 7](#)). Example firmware images for demonstration purposes are located in the installed EXE package bin directory on the PC.

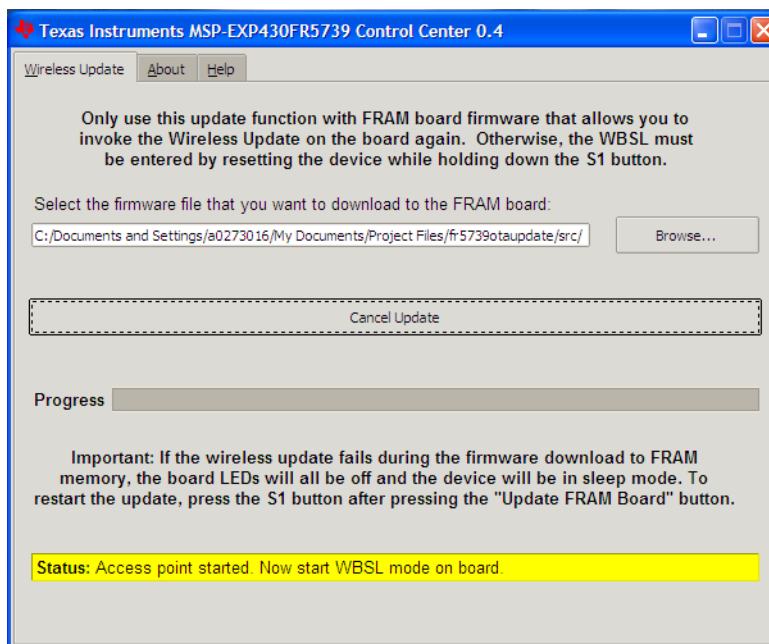


**Figure 7. MSP-EXP430FR5739 Control Center File Selection**

**NOTE:** The file to be downloaded to the watch must be in TI-TXT format to properly function with this update procedure. Firmware must reside within the available main application area of FRAM; otherwise, the update procedure fails due to boundary checks on the FRAM board during WBSL execution.

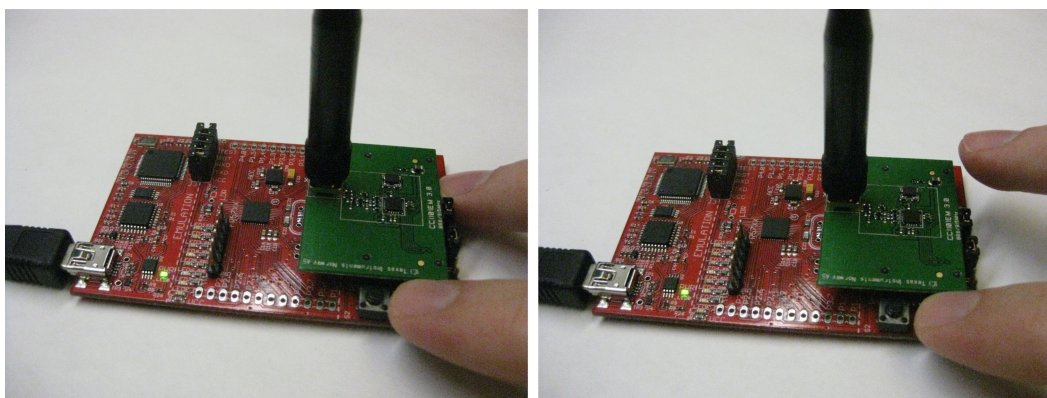
4. Activate the update mechanism on the PC by clicking Update FRAM Board. When the PC is ready, the Status line displays "Access point started. Now start WBSL mode on board." (see [Figure 8](#)).





**Figure 8. MSP-EXP430FR5739 Control Center Start of Update**

5. Reset the FRAM board by pressing and releasing the RST button while holding down the S1 button (see [Figure 9](#)). This invokes the CBSL on the FRAM board.



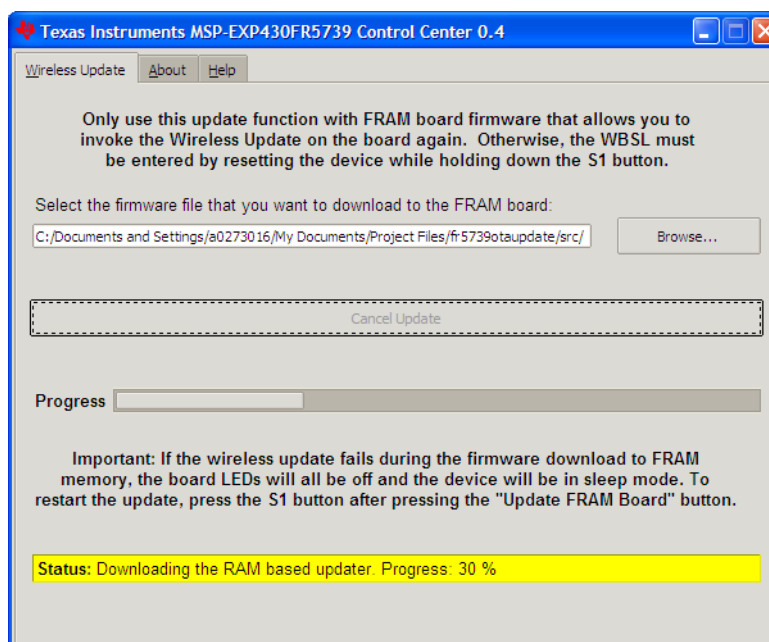
**Figure 9. Hardware Initialization Sequence for Wireless Update**

6. After a short time, the wireless update is started.  
At the beginning of the update, all board LEDs are off. During this time, the WBSL is downloaded from the PC to the RAM of the MSP430FR5739 (see [Figure 10](#)). This program contains all LED routines. Once this transfer completes, the code is executed from RAM. All LEDs are turned on in succession and then off in succession to indicate the start of execution from RAM. At this point, the download of the user-selected firmware is started. The first four LEDs indicate progress of firmware download, with all four LEDs being on when download is 100% complete. The fifth, sixth, and seventh LEDs indicate RF activity, device connectivity, and packet error, respectively (see [Figure 11](#)).

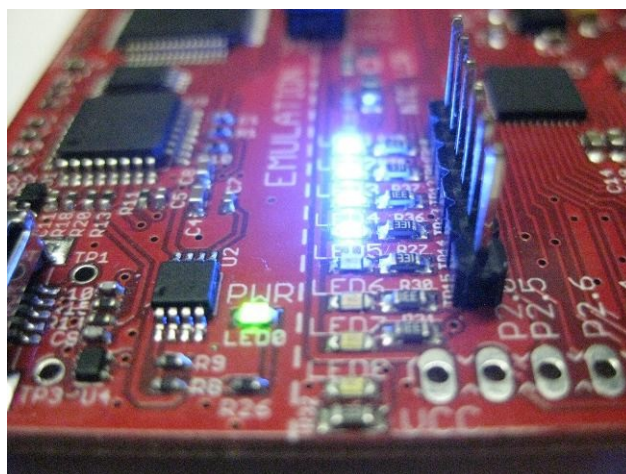
---

**NOTE:** If the update fails during the download of actual firmware, it can be activated again by pressing the S1 button on the FRAM board after the wireless update is made active in the Control Center by clicking on Update FRAM Board.

---



**Figure 10. MSP-EXP430FR5739 Control Center WBSL Download Progress**



**Figure 11. MSP-EXP430FR5739 Firmware Download Progress**

## 3 Wireless Update

### 3.1 Components of the Wireless Update Process

The lightweight radio protocol based on the SimpliciTI specification used by the *eZ430-Chronos Development Tool's* Wireless Update feature was modified for use with the CC1101 radio to minimize code size residing in FRAM. LED routines to handle progress information and communication link status indicators are downloaded over the air and executed in RAM. Use of the USB RF access point and CC1101 radio minimizes hardware requirements for the wireless update.

Primary components of the wireless update are:

- The CBSL residing in FRAM area of the MSP430FR5739
- The WBSL is transferred from PC to RAM area of MSP430FR5739
- The SimpliciTI network protocol residing on the RF access point (CC1111)
- The MSP-EXP430FR5739 Control Center application supporting the wireless update functionality on the PC

### 3.2 Wireless Update Procedure

The wireless update procedure functions as follows:

1. Wireless update functionality is activated in the MSP-EXP430FR5739 Control Center following a valid firmware image selection.
2. RF access point is ready and awaits request from the FRAM board.
3. CBSL is invoked on the FRAM board through a hardware device reset while the S1 button is held down.
4. FRAM board downloads WBSL from the MSP-EXP430FR5739 Control Center into the RAM area of the MSP430FR5739.
5. FRAM board executes the WBSL. An LED pattern is displayed indicating the start of execution from RAM after which the firmware image download begins. The LEDs show the progress and communication link status during WBSL execution.
6. A software device reset starts the new application on the FRAM board when the wireless update is complete.

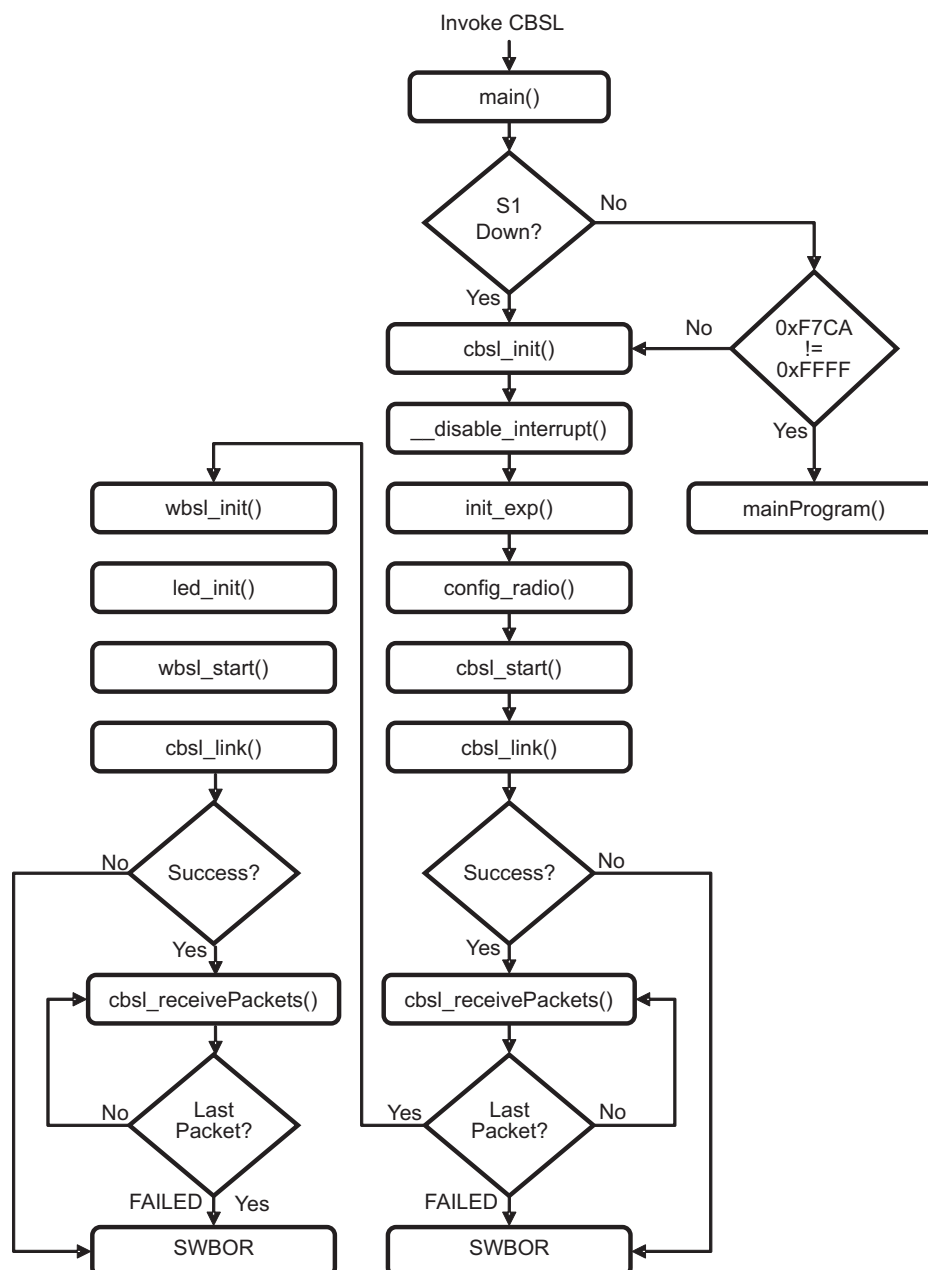
#### 3.2.1 Detailed Description of the Wireless Update Program Flow

To invoke the WBSL, the device must perform a system reset. Interrupt sources for a system reset may be found in the "Interrupt Sources, Flags, and Vectors" table of the *MSP430FR573x Mixed-Signal Microcontrollers* data sheet ([SLAS639](#)). After a system reset, the CBSL begins execution, and S1 of the FRAM board checks for forced execution of a wireless update. If S1 is not being pressed, then the main application begins execution if the reset vector of the main application has not been cleared to 0xFFFF. If S1 is being pressed, then the CBSL begins the wireless update process.

After starting the wireless update process, the CBSL on the FRAM board configures the CC1101 transceiver and attempts to link with a listening USB RF access point. If the link is successful, the FRAM board downloads the WBSL into the RAM area of the MSP430FR5739. Once completed, the WBSL attempts to link with the access point again to begin downloading the firmware update. If this link is successful, then the reset vector of the main application is cleared to 0xFFFF and the device begins receiving data packets. If either link fails, the device generates a reset to restart the CBSL, and the main application executes or the CBSL polls the S1 switch waiting for the user to initiate another wireless update if the reset of the main application has been cleared to 0xFFFF.

While each data packet containing the firmware update is received, the MSP430FR5739 reads data from the RXFIFO of the CC1101 over SPI, writes data to the main application area of FRAM, and checks that the FRAM write was successful. LED1 through LED4 on the FRAM board each indicate 25% progress of the firmware download, and LED5 through LED7 indicate radio activity, successful link, and link error respectively. When the firmware download is complete, the CC1101 radio is reset and the device resets to start the main application.

A program flow of the WBSL is shown in [Figure 12](#). A detailed flow of the USB RF access point can be found in the *eZ430-Chronos Development Tool User's Guide* Section 3.6.3 ([SLAU292](#)).



**Figure 12. Program Flow for the Wireless Update**

### 3.3 Wireless Update Firmware

Full project sources of the Wireless Update and demonstration applications are included in the installed EXE package. Modifications to the Wireless Update or demonstration applications may be performed in Code Composer Studio™ v5.1 or IAR Embedded Workbench® for MSP430 v5.30.

#### 3.3.1 Code Composer Studio™ v5.1

While having the FRAM board connected to the PC via the USB cable, the full steps to begin firmware development with CCS v5.1 are as follows:

1. Launch Code Composer Studio v5.1: Start > All Programs > Texas Instruments > Code Composer Studio > Code Composer Studio (see [Figure 13](#)).



**Figure 13. Code Composer Studio v5 Splash Screen**

2. Create new Workspace or open existing Workspace that does not yet contain FRAM projects from the installed EXE package (see [Figure 14](#)).

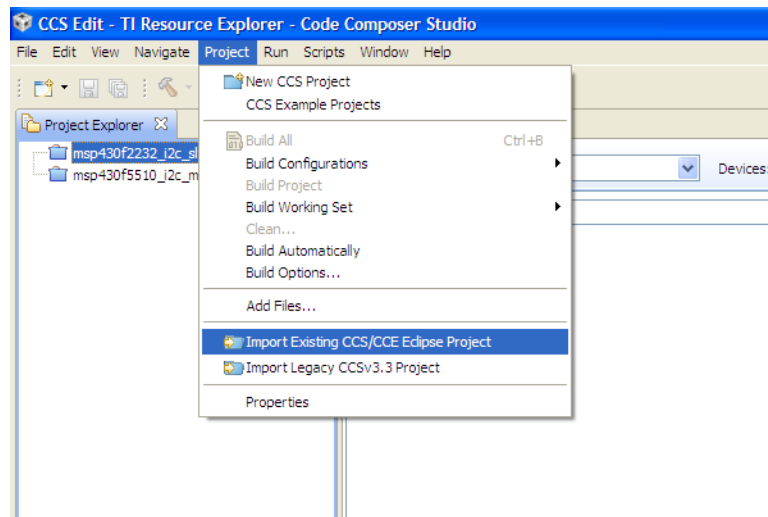
**NOTE:** Workspaces should not be located in a long path of directories as Windows only supports path names up to 255 characters. Keep in mind that the projects contain subdirectories.



**Figure 14. Code Composer Studio v5 Workspace Launcher**

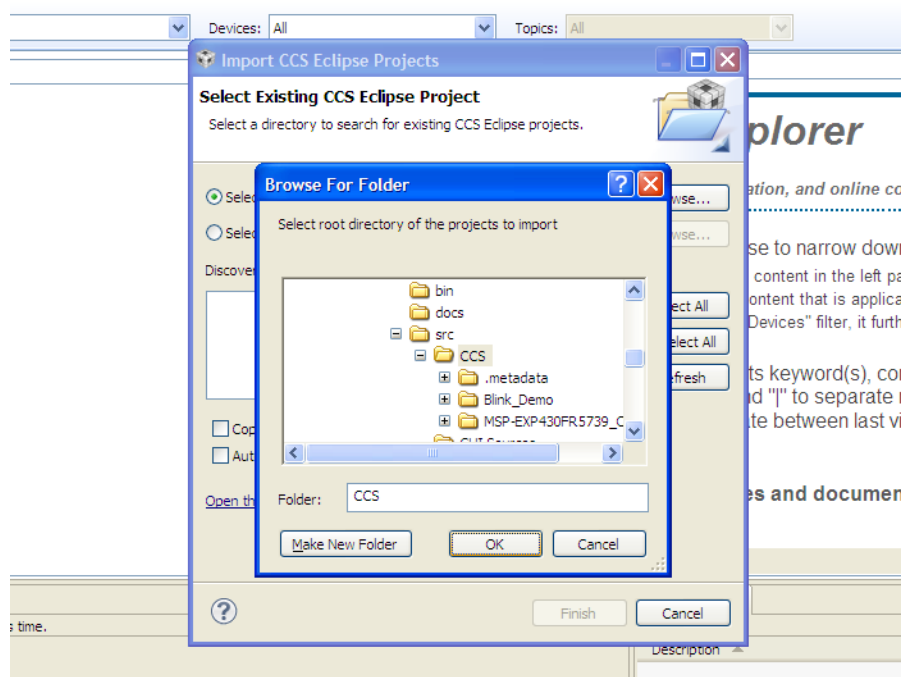


3. Import Wireless Update Project: Project > Import Existing CCE/CCS Eclipse Project (see [Figure 15](#)).



**Figure 15. Import Existing CCS/CCE Eclipse Project**

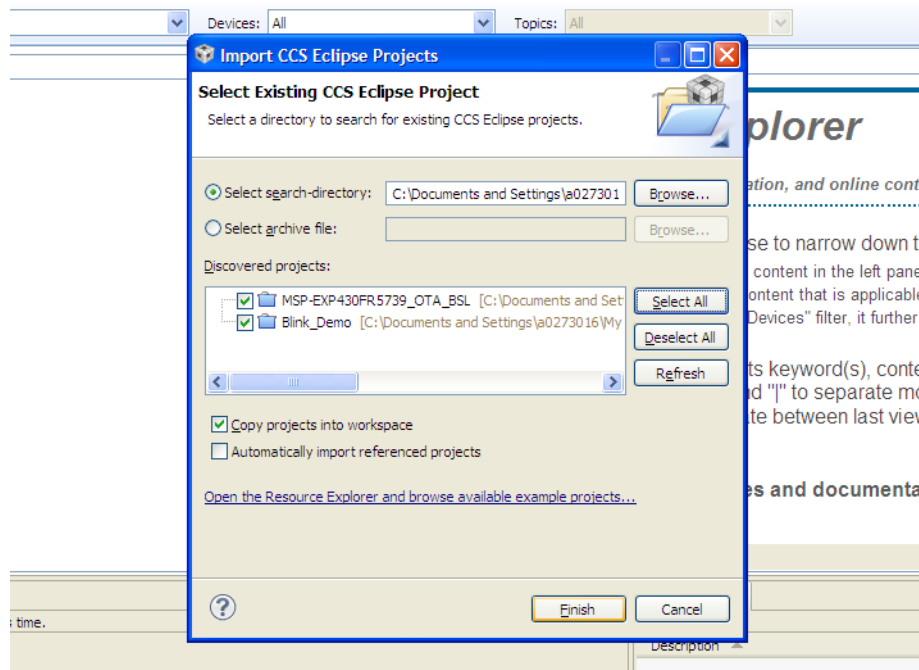
4. Click "Browse..." and browse to CCS within the installed EXE package. Confirm directory by clicking OK (see [Figure 16](#)).



**Figure 16. Browse Directory for Import CCS Projects**

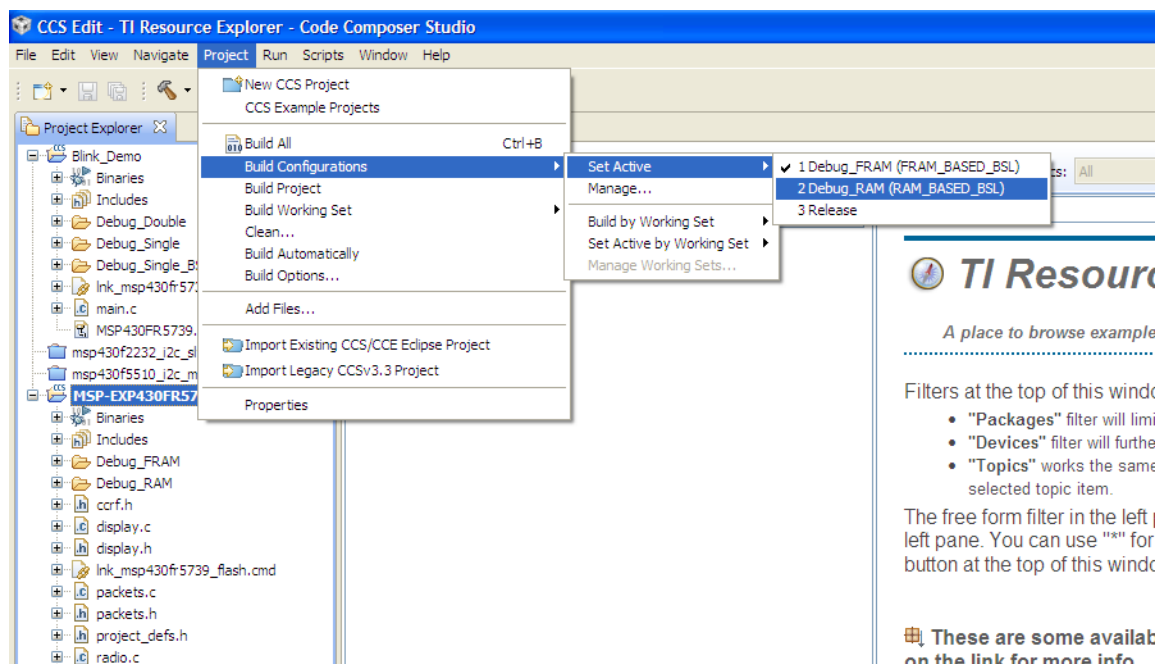
5. The Wireless Update project appears as "MSP-EXP430FR5739\_OTA\_BSL" and the demonstration application appears as "Blink\_Demo" within the Import CCS Eclipse Projects screen. Select Copy projects into workspace to create new copies of the projects.

- Click Select All then Finish to finalize the project import. The projects appear in the C/C++ Projects view of CCS (see Figure 17).



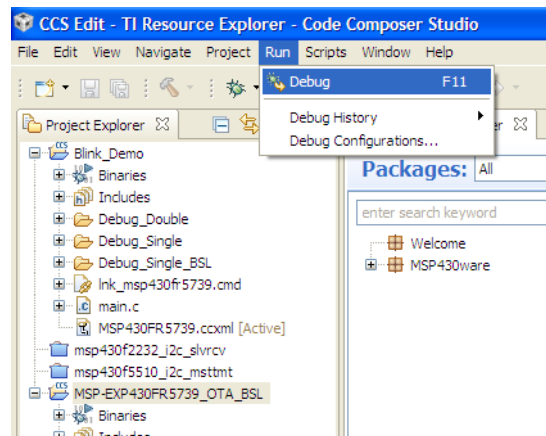
**Figure 17. Import All Existing CCS Projects**

- Select the desired project setting (see Figure 18). The Wireless Update project comes with two different project settings (build configurations), which can be selected in Project > Build Configurations > Set Active > Configuration: Debug\_FRAM or Debug\_RAM. The three different project settings, described in Section 4, of the demonstration application are: Debug\_Single, Debug\_Double, and Debug\_Single\_BSL.



**Figure 18. Code Composer Studio v5 Build Configuration Selection**

8. Go to Run > Debug to launch the debugger for the currently selected project (see [Figure 19](#)).

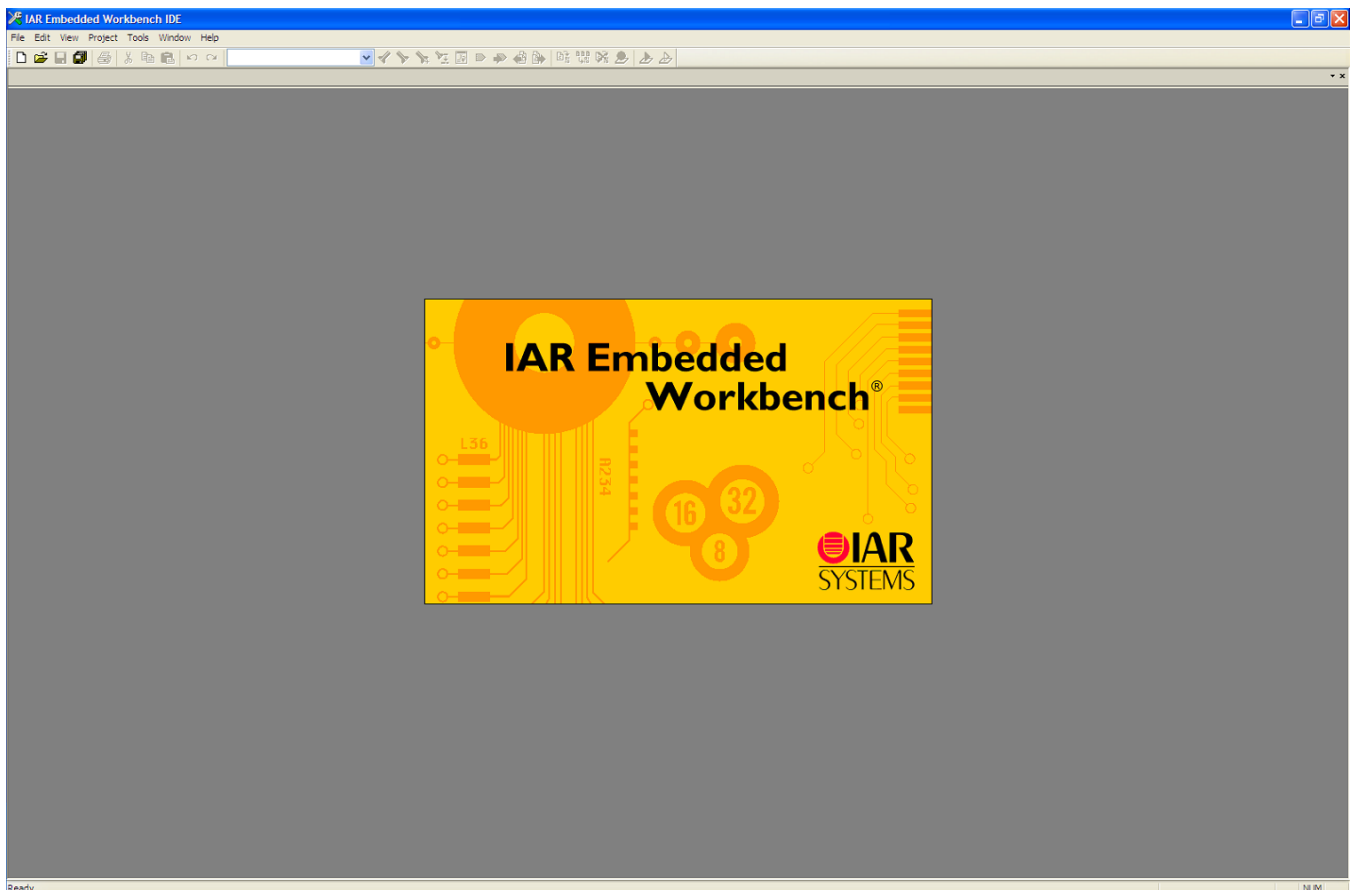


**Figure 19. Code Composer Studio v5 Debug**

### 3.3.2 IAR Embedded Workbench® for MSP430™ 5.30

While having the FRAM board connected to the PC via the USB cable, the full steps to begin firmware development with IAR Embedded Workbench for MSP430 5.30 are as follows:

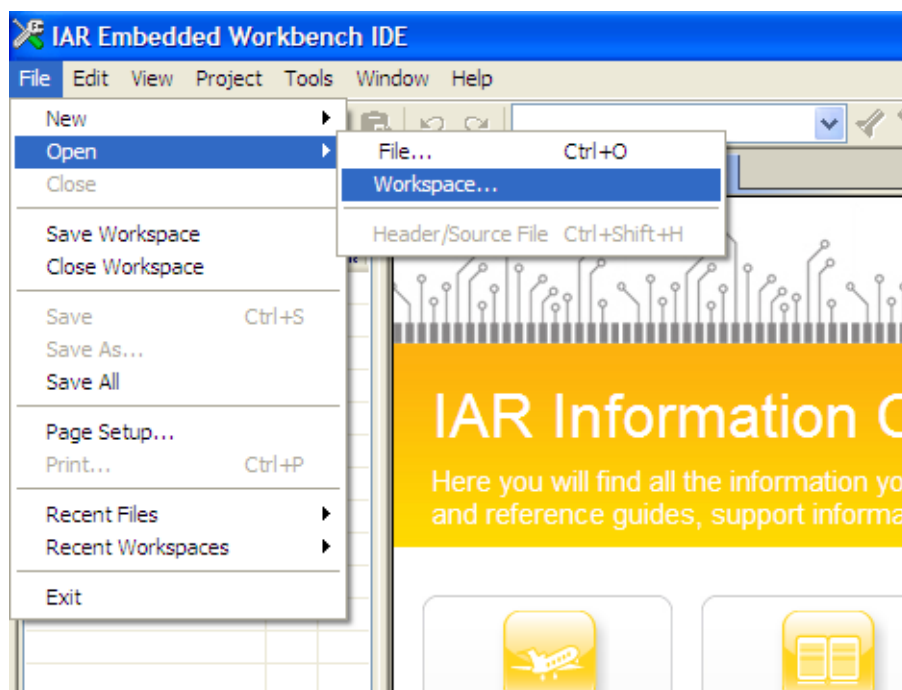
1. Launch IAR Embedded Workbench for MSP430 5.30: Start > All Programs > IAR Systems > IAR Embedded Workbench for MSP430 5.30 > IAR Embedded Workbench (see [Figure 20](#)).



**Figure 20. IAR Embedded Workbench® for MSP430™ 5.30 Splash Screen**

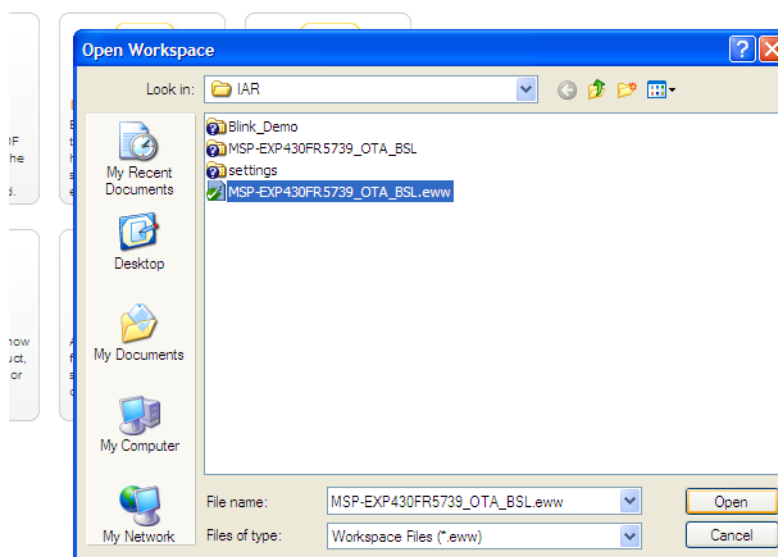
**NOTE:** Workspaces should not be located in a long path of directories as Windows only supports path names up to 255 characters. Keep in mind that the projects contain subdirectories as well.

2. Open Existing Wireless Update Workspace: File > Open > Workspace (see [Figure 21](#)).



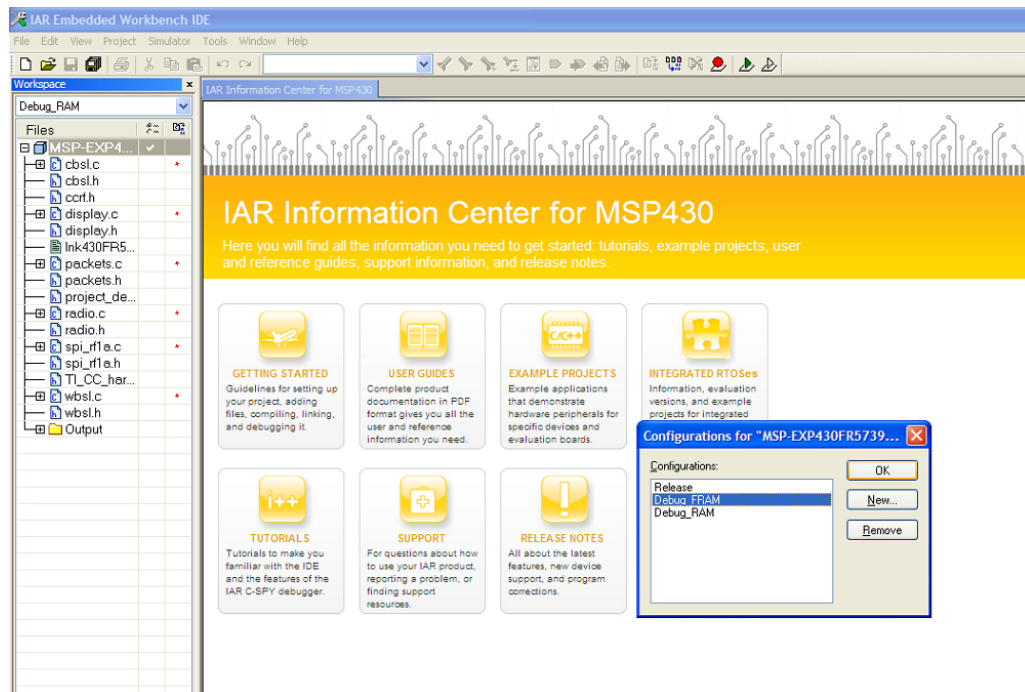
**Figure 21. Add Existing IAR Project**

3. Browse to IAR within the installed EXE package directory. Confirm the workspace by clicking Open after selecting the .eww file (see [Figure 22](#)).



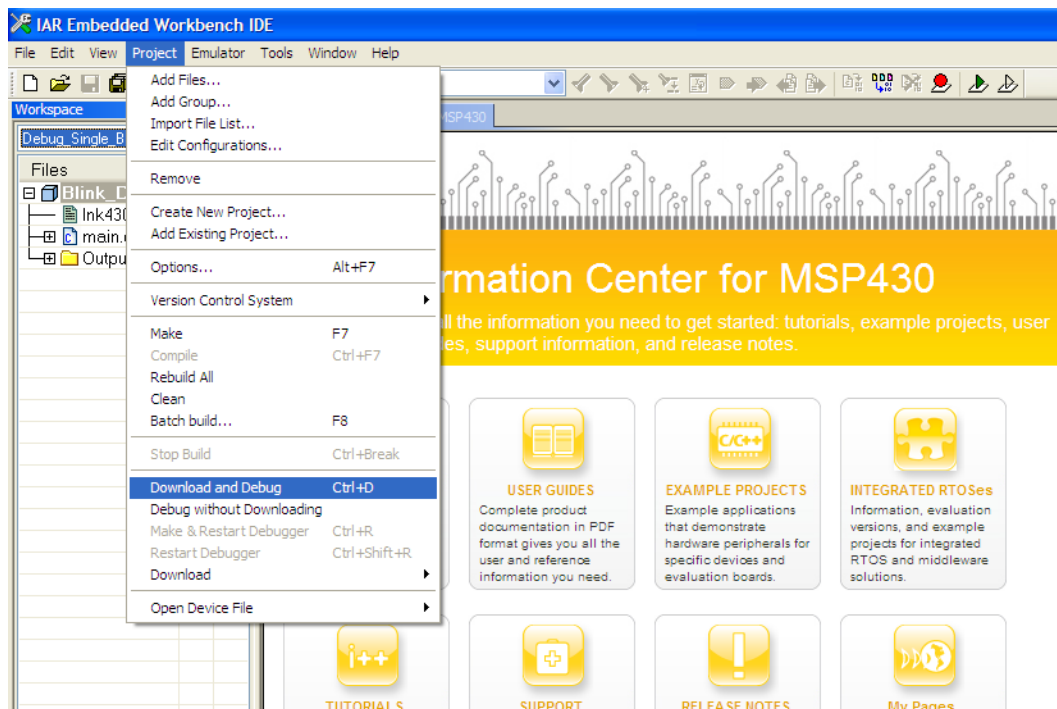
**Figure 22. Browse Directory for Add Existing IAR Project**

4. Select the desired project setting (see Figure 23). The Wireless Update project comes with two different project settings (build configurations), which can be selected in Project > Edit Configurations: Debug\_FRAM or Debug\_RAM > OK. The three different project settings of the demonstration application are: Debug\_Single, Debug\_Double, and Debug\_Single\_BSL.



**Figure 23. IAR Embedded Workbench® for MSP430™ 5.30 Build Configuration Selection**

5. Go to Project > Download & Debug to launch the debugger for the currently selected project (see Figure 24).



**Figure 24. IAR Embedded Workbench® for MSP430™ 5.30 Debug**



### 3.4 Wireless Update GUI

Full project sources of the MSP-EXP430FR5739 Control Center are included in the installed EXE package. The graphical user interface is based on Tcl/Tk. The Control Center consists of a Tcl/Tk executable, .ini file, and DLL from the Chronos Control Center software which handles the communication with the USB RF access point. More details on the eZ430 Chronos Control Center DLL are accessible from the *eZ430-Chronos Development Tool User's Guide* Section 3.6 ([SLAU292](#)). Use of an .ini file to select the WBSL TI-TXT file allows for a simpler user interface while still allowing modifications to the WBSL TI-TXT file for future updates or custom development.

Development can be performed with a text editor such as Notepad on Windows. Execution of a Tcl script (\*.tcl) requires Tcl/Tk to be installed. Download Tcl/Tk for Windows for free from <http://www.activestate.com/activetcl>. Once Tcl/Tk is installed on the computer, a double click on a Tcl script starts the interpreter and executes the script. Alternatively, start the Tcl interpreter wish.exe from the installation directory and execute the script with the "source" command. If the script filename contains blanks, put it into quotes.

A wrapping tool is needed to compile all Tcl/Tk components into a standalone executable. The freeware tool Freewrap (<http://freewrap.sourceforge.net>) may be used for that purpose.

1. Open Command Prompt
2. Navigate to the Freewrap directory.

```
c:\>cd c:\Tcl\freewrap
```

3. Start Freewrap with the complete path to the TCL script to be wrapped.

```
c:\Tcl\freewrap>freewrap.exe "C:\Tcl\Project\EXP430FR5739_CC_x_x.tcl"
```

A standalone executable, which only requires the corresponding Windows DLL, is generated. The DLL can be used on any PC running a Windows OS and does not require Tcl/TK to be installed.

## 4 Wireless Update Enabled Firmware Development

### 4.1 Main Application Development

The Wireless Update project was developed specifically to minimize the impact on main application development. Necessary changes to a project which integrate the CBSL are performed only once at project creation, and have no further impact on application development. Projects which integrate the CBSL have full debug capabilities and use of all available interrupts. Available FRAM area for code use is restricted since the CBSL resides in FRAM, but the only other impact on main application execution or development is marginally increased startup time during CBSL execution following a reset.

After following the steps outlined in the IDE specific instructions in [Section 4.1.1](#) (CCS) or [Section 4.1.2](#) (IAR Embedded Workbench), no further steps are necessary for integration of the CBSL with a main application. The CBSL binaries are loaded into the output file for the main application during the linking phase for each build of the project.

Development of a main application intended for use with the MSP-EXP430FR5739 Control Center can skip including the CBSL image file in the main application project as described in the following sections. The linker command file restricts the main application code to valid memory ranges for use with the wireless update. Not including the CBSL image file prevents invalid memory ranges appearing in the final TI-TXT image file selected with the MSP-EXP430FR5739 Control Center.

Demo\_Blink, a simple P1.0 toggle demonstration application to blink an LED, contains three build configurations. Debug\_Single and Debug\_Double both do not include the CBSL and are intended for programming a device using the MSP-EXP430FR5739 Control Center. Debug\_Single\_BSL does include the CBSL and is intended for programming a device using JTAG or Spy-Bi-Wire rather than the MSP-EXP430FR5739 Control Center.

#### 4.1.1 Using Code Composer Studio™ v5

1. Follow instructions for Code Composer Studio v5.1 under *Wireless Update Firmware* to build the MSP-EXP430FR5739\_OTA\_BSL project using the Debug\_FRAM project setting.
2. Launch Code Composer Studio v5.1: Start > All Programs > Texas Instruments > Code Composer Studio > Code Composer Studio
3. Create new Workspace or open existing Workspace. Create new project for main application development: File > New > CCS Project (see [Figure 25](#)).

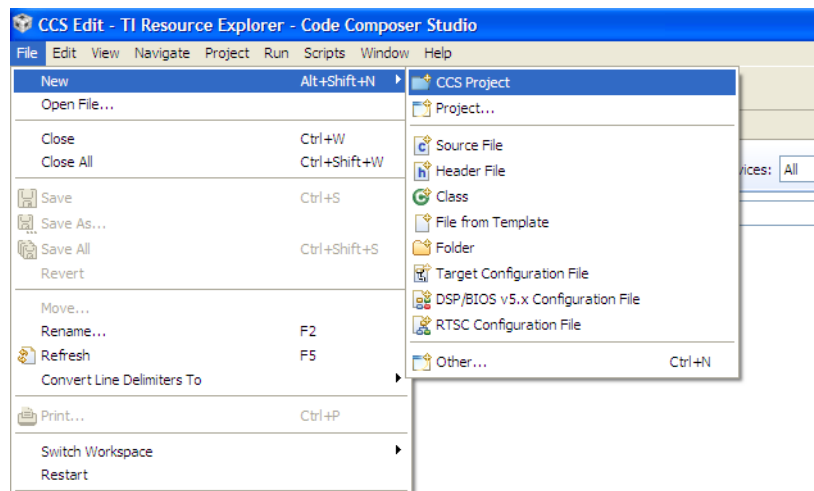
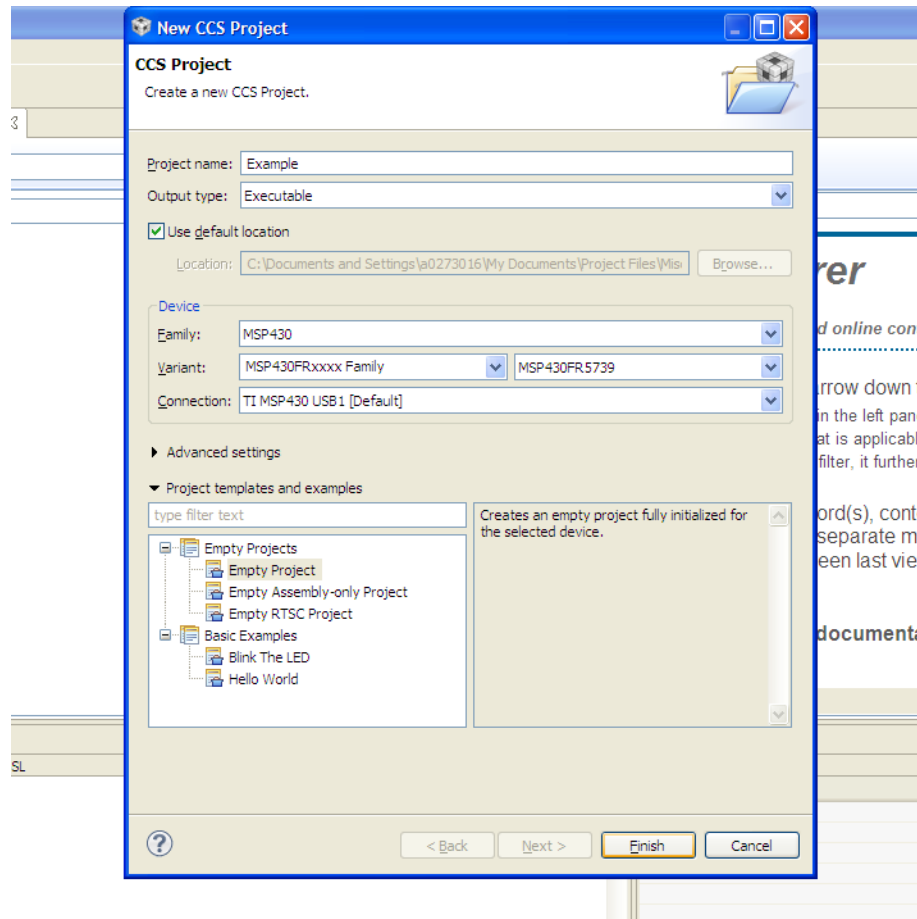


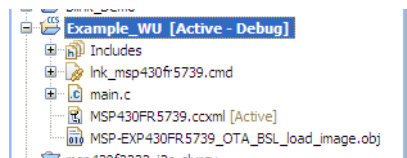
Figure 25. Code Composer Studio v5 New CCS Project Selection

4. Select a name for the project unique to the Workspace. Create an empty MSP430FR5739 project: Variant > MSP430FRxxxx Family > MSP430FR5739 > Finish (see [Figure 26](#)).



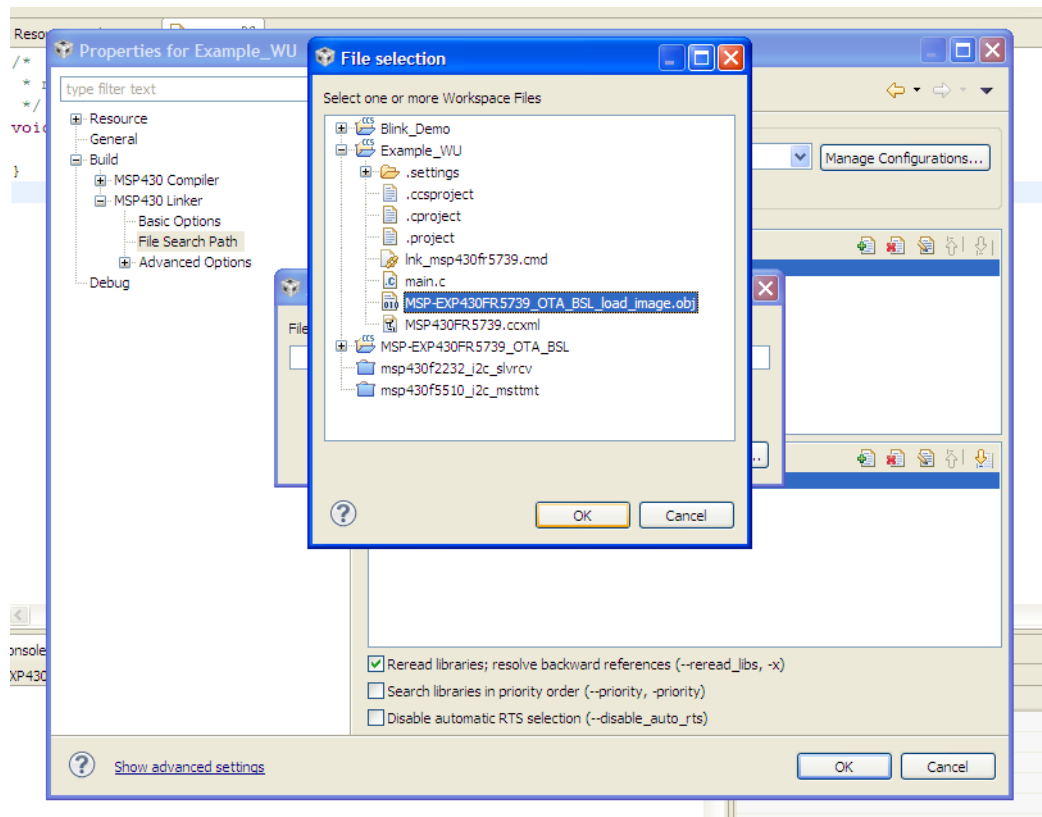
**Figure 26. Code Composer Studio v5 Project Creation**

5. Copy the MSP-EXP430FR5739\_OTA\_BSL\_load\_image.obj file from the MSP-EXP430FR5739\_OTA\_BSL\Debug\_FRAM folder into the newly created project folder (see [Figure 27](#)).



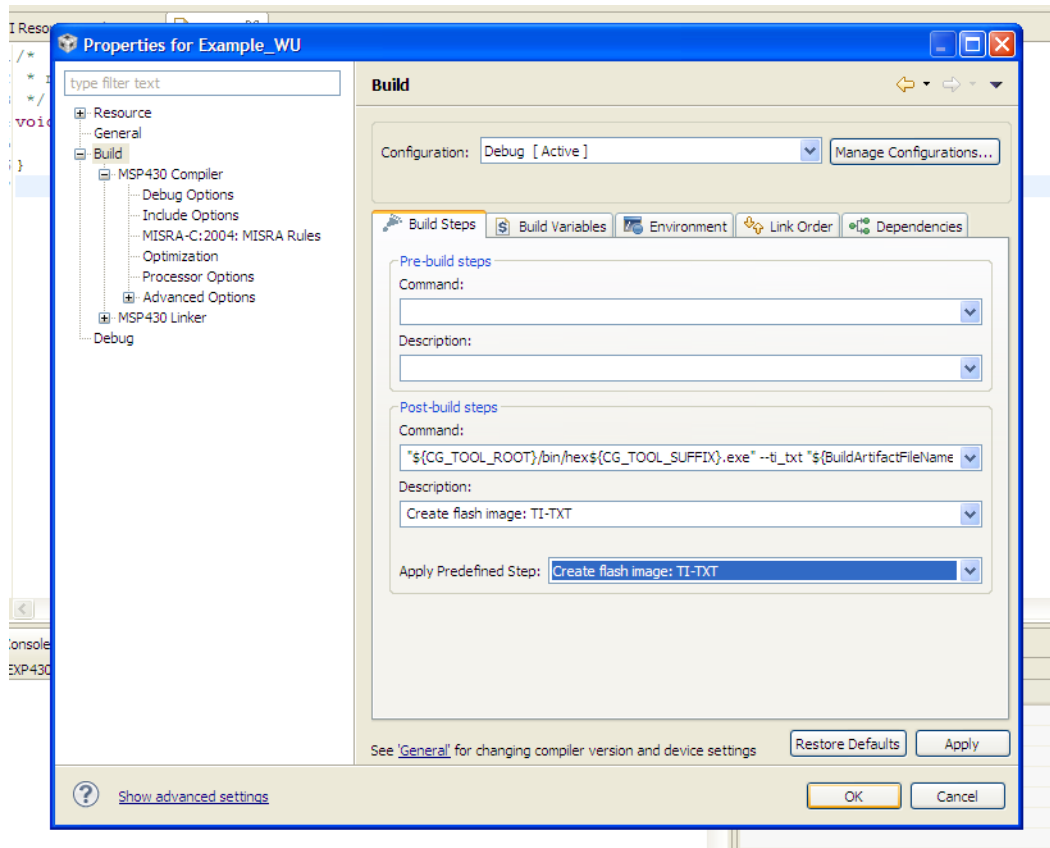
**Figure 27. New CCS Project With CBSL Load Image**

6. Include the image file in the main application project: Project > Properties > Build > MSP430 Linker > File Search Path > Include library file or command file as input > Add... > Workspace... > Expand main application project folder > MSP430FR5739\_OTA\_BSL\_load\_image.obj > OK > OK > OK (see Figure 28).



**Figure 28. Include Load Image in CCS Project**

7. Add the build step to create a TI-TXT flash image post-build: Project > Properties > Build > Apply Predefined Step > Create flash image: TI-TXT > OK (see [Figure 29](#)).



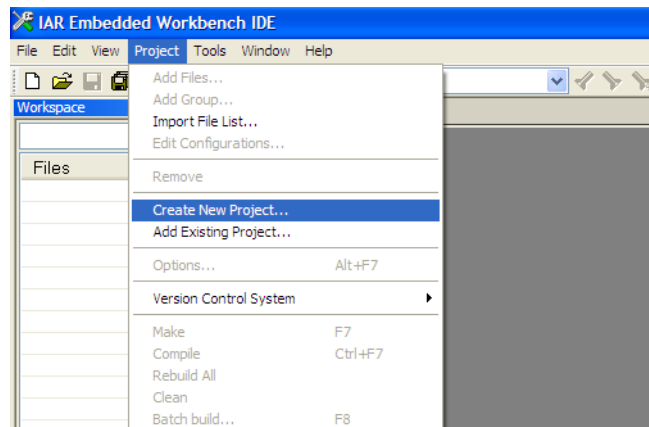
**Figure 29. Post Build Step in CCS Project**

8. Delete the `Ink_msp430fr5739.cmd` linker command file from the main application project folder. Copy the `Ink_msp430fr5739.cmd` linker command file from the Blink\_Demo project into the project folder.



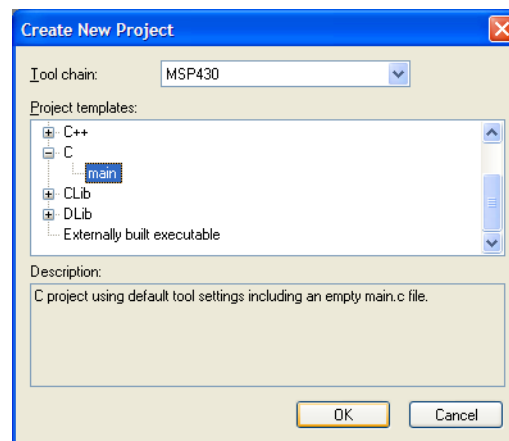
#### 4.1.2 Using IAR Embedded Workbench® for MSP430 5.30

1. Follow instructions for IAR Embedded Workbench for MSP430 5.30 under [Section 3.3](#) to build the MSP-EXP430FR5739\_OTA\_BSL project using the Debug\_FRAM project setting.
2. Launch IAR Embedded Workbench for MSP430 5.30: Start > All Programs > IAR Systems > IAR Embedded Workbench for MSP430 5.30 > IAR Embedded Workbench
3. Create new Workspace or open existing Workspace. Create new project for main application development: Project > Create New Project... (see [Figure 30](#)).



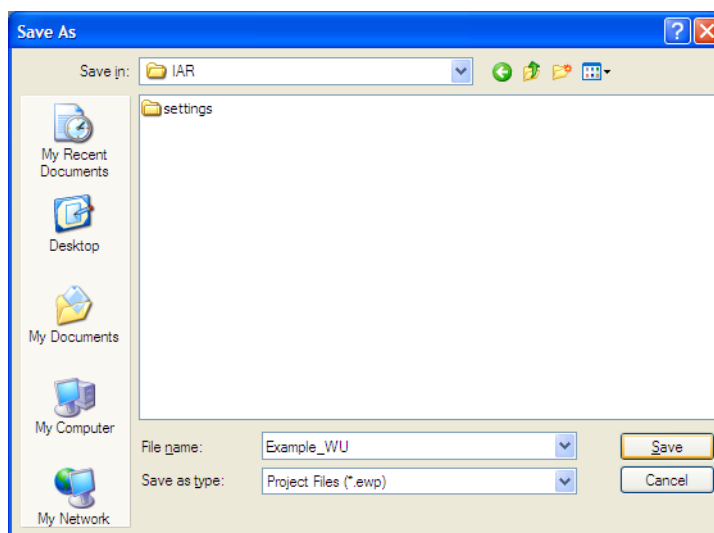
**Figure 30. IAR EW for MSP430 5.30 Create New Project Selection**

4. Select Tool chain: MSP430 and Project templates: C > main > OK (see [Figure 31](#)).




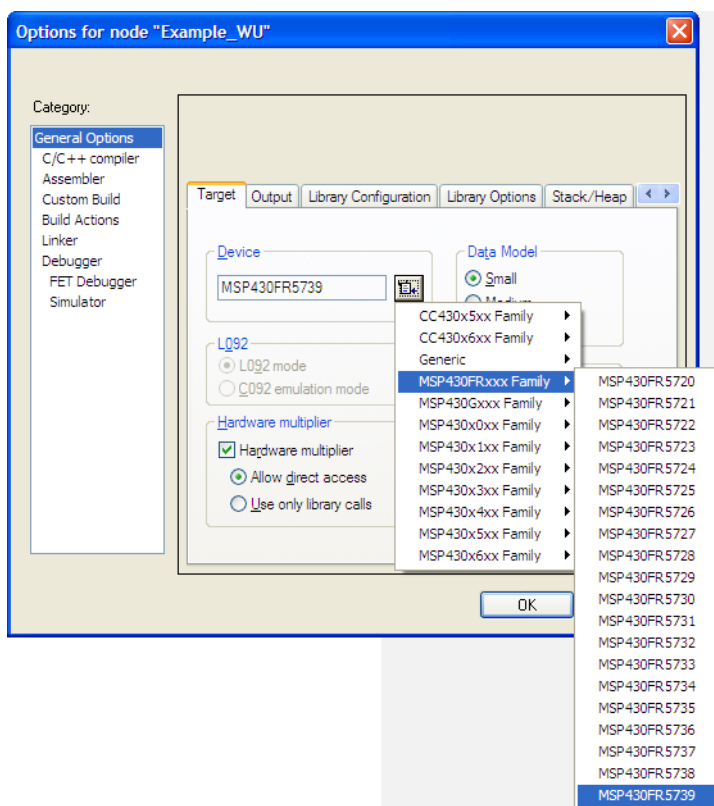
**Figure 31. IAR EW for MSP430 5.30 Project Creation**

5. Select a name for the project unique to the Workspace and save (see [Figure 32](#)).



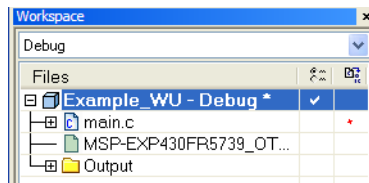
**Figure 32. Select Project Name**

6. Set the target device of the newly created project to MSP430FR5739: Project > Options... > General Options > Target > Device >  > MSP430FRxxx Family > MSP430FR5739 (see [Figure 33](#)).



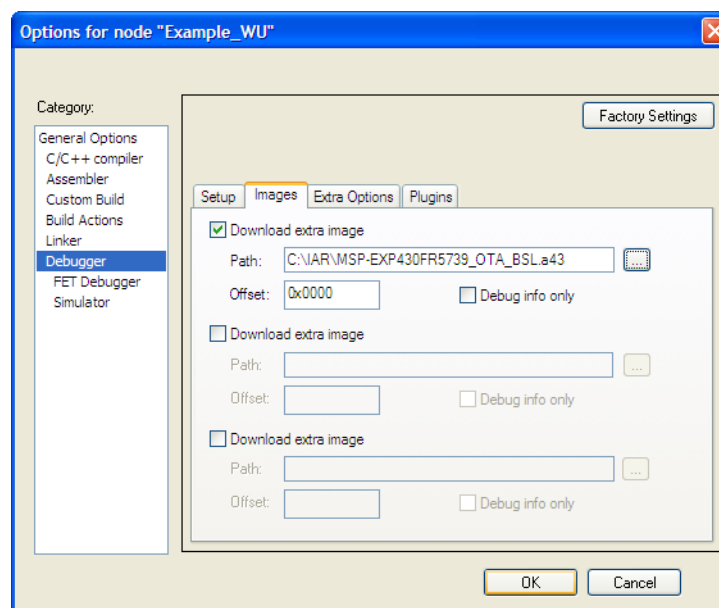
**Figure 33. Select Target Device**

7. Copy the MSP-EXP430FR5739\_OTA\_BSL.d43 file from the MSP-EXP430FR5739\_OTA\_BSL\Debug\_FRAM\Exe project folder into the newly created project folder (see [Figure 34](#)).



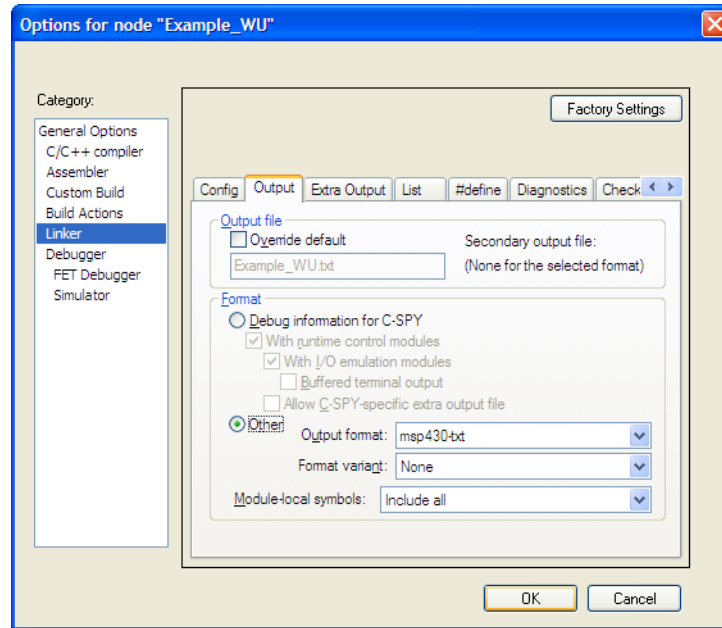
**Figure 34. New IAR EW Project With CBSL Debug Image**

8. Include the debug image file in the main application project: Project > Options... > Debugger > Images > Download extra image > Path: ... > Select MSP-EXP430FR5739\_OTA\_BSL.d43 > Open > Offset: 0x0000 > OK (see [Figure 35](#)).



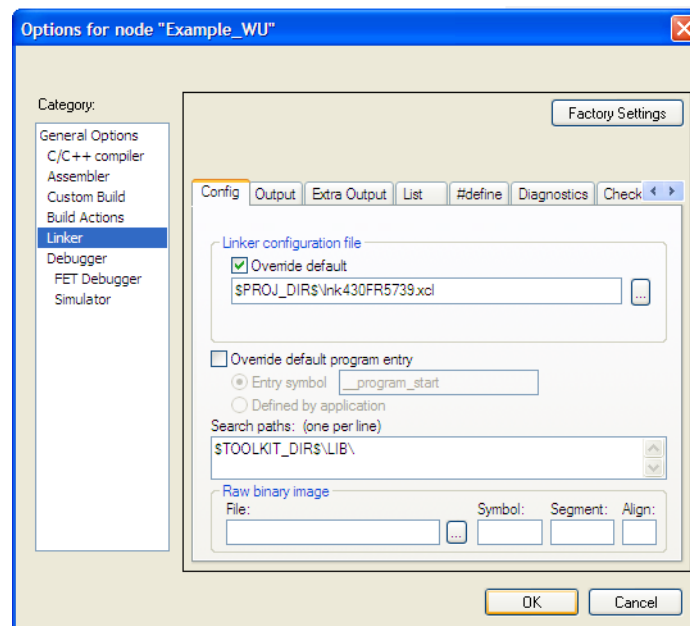
**Figure 35. Download Extra Image in IAR EW**

9. Add the Linker step to create a TI-TXT flash image during linking: Project > Options... > Linker > Output > Other > Output format: msp430-txt > OK (see [Figure 36](#)).



**Figure 36. Linker Output File Selection in IAR EW**

10. Copy the Ink430FR5739.xcl linker configuration file from the Blink\_Demo project into the newly created project folder. Add the linker configuration file to the project: Project > Add files... > Ink430FR5739.xcl > Open. Override default linker configuration file: Project > Options... Linker > Config > Override default > "\$PROJ\_DIR\$\Ink430FR5739.xcl" > OK (see [Figure 37](#)).



**Figure 37. Linker Configuration File Override in IAR EW**

## 4.2 Wireless Update BSL Development

The Wireless Update reduces FRAM area code size while maintaining readability and functional organization. Hardware-specific definitions have been organized to aid in migration to different hardware platforms. Functional routines such as radio control, display, packet handling, CBSL, and WBSL functions have been separated to aid functional-specific modifications. Development for the CBSL and WBSL is performed using the same source code.

Following the steps outlined in *Wireless Update Firmware* and only selecting the MSP-EXP430FR5739\_OTA\_BSL project creates a new project folder for Wireless Update development. Build configurations for the RAM and FRAM select custom post-build steps which allow specific output files to be created depending on which project setting is selected.

### 4.2.1 Linker Command/Configuration File

Development of the CBSL and WBSL may require modifications to the linker command file to allow more FRAM area code space for newly developed Wireless Update firmware. To avoid repeated modifications to the linker command file it is recommended that the FRAM or CODE section is used expanded for development on both the CBSL and WBSL. Once new firmware development has completed, the linker command file can then be modified to allow the CBSL to reside precisely within the FRAM or CODE section limits.

### 4.2.2 CCS v5.1 Post-Build Steps

Post-build steps for the two project settings of the MSP-EXP430FR5739\_OTA\_BSL allow for output files to be created for use with the MSP-EXP430FR5739 Control Center or included in main application projects utilizing the Wireless Update.

Debug\_RAM project settings output a TI-TXT file which only contains the WBSL residing within the RAM area of the MSP430FR5739 device. The TI-TXT file may then replace the existing WBSL .txt file used by the MSP-EXP430FR5739 Control Center. Add the following line of code to the post-build step to create the specific TI-TXT file: Project > Properties > Build > Post-build steps > Command:

```
"${CG_TOOL_ROOT}/bin/hex${CG_TOOL_SUFFIX}.exe" --ti_txt "${BuildArtifactFileName}" -o
"${BuildArtifactFileName}.txt" -order MS -romwidth 16 -exclude .cio -exclude .sysmem
-exclude .cinit -exclude .pinit -exclude .const -exclude .text -exclude .bss
-exclude .stack -exclude .reset
```

Debug\_FRAM project settings output a load image which may be included in main application projects to link the CBSL into the output file of the main application. The load image is linked into the main application's output file which allows for debug instances to load the main application and CBSL to the device seamlessly to significantly aid development. Add the following line of code to the post-build step to create the specific load image file: Project > Properties > Build > Post-build steps > Command:

```
"${CG_TOOL_ROOT}/bin/hex${CG_TOOL_SUFFIX}.exe" --load_image "${BuildArtifactFileName}" -o
"${BuildArtifactFileName}_load_image.obj" --section_name_prefix=wbsl -exclude .bss
-exclude .stack -exclude .wbsl
```

### 4.2.3 IAR EW for MSP430 5.30 Linker Configuration

Linker configurations for the two project settings of the MSP-EXP430FR5739\_OTA\_BSL allow for output files to be created for use with the MSP-EXP430FR5739 Control Center or included in main application projects utilizing the Wireless Update.



Debug\_RAM linker configurations output a TI-TXT file which only contains the WBSL reading within the RAM area of the MSP430FR5739 device. The TI-TXT file may then replace the existing WBSL .txt file used by the MSP-EXP430FR5739 Control Center. Change the linker output format to create the specific TI-TXT file:

Project > Options > Linker > Output > Other > Output format: msp430-txt

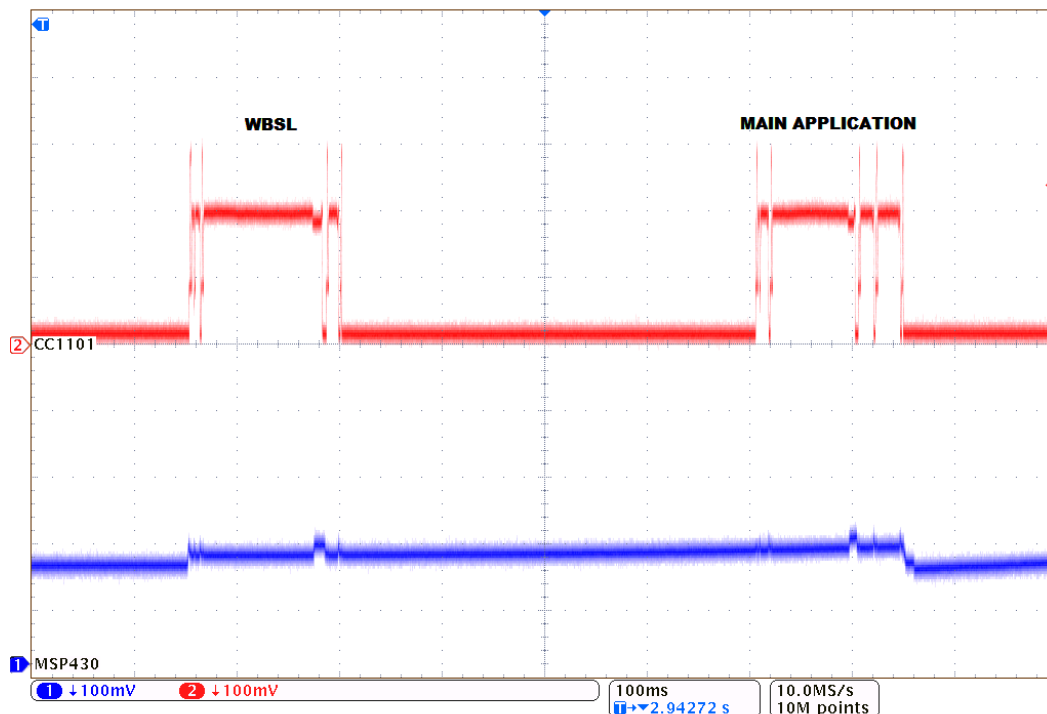
Debug\_FRAM linker configurations output a debug image file which may be included in main application projects to download the CBSL image to the device during debugging of the main application. The debug image is downloaded with the main application's output file which allows for debug instances to load the main application and CBSL to the device seamlessly to significantly aid development. Debug image files are the default linker output file format.

## 5 Wireless Update Benchmarks

### 5.1 Wireless Update Power Consumption

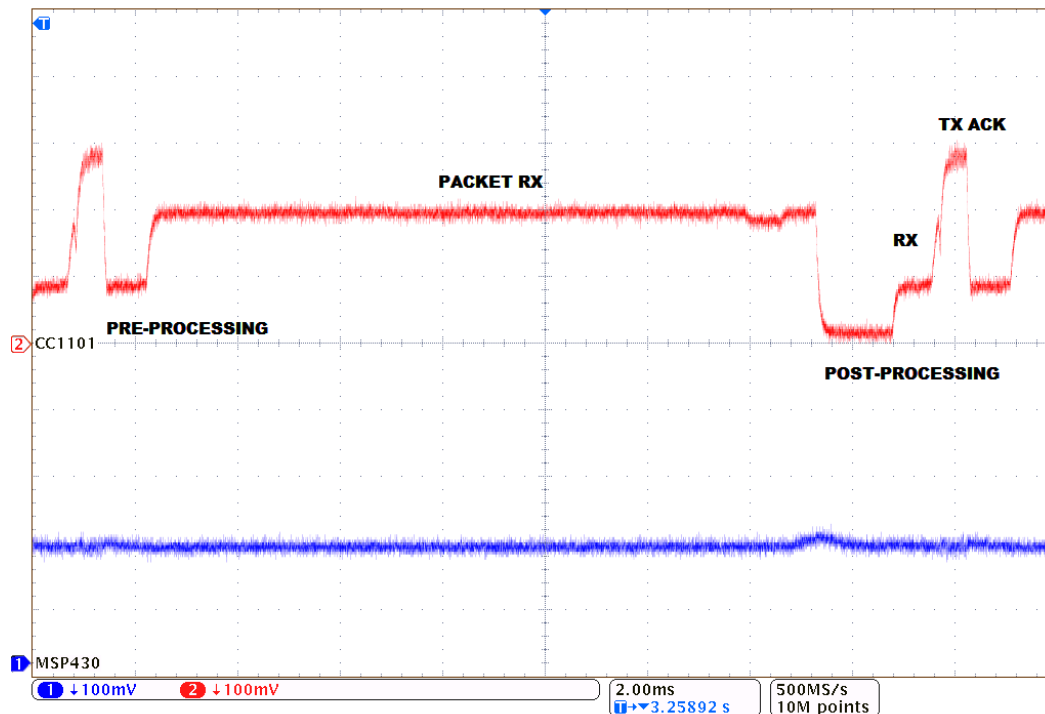
The ultra-low power capabilities of FRAM allow for the MSP430FR5739 to perform a wireless update using less power than a flash-based device. The increased write speed of FRAM also reduces the total time necessary for the CC1101 transceiver to be in an active state. Total power consumption for a specific wireless update procedure is dependent on main application size, but the following analysis captures key power components to the overall procedure and allows for adjustable power consumption estimates (see [Figure 38](#)).

Comparison of power consumption using the Wireless Update procedure with that of other RF communication is not possible using only one metric. Peak current may be used to rate a device's power consumption, but during a wireless update the device only consumes current at the peak level while it is transmitting. Throughout a wireless update, the device only transmits for a fraction of the procedure.



**Figure 38. Current Measurement of a Wireless Update**

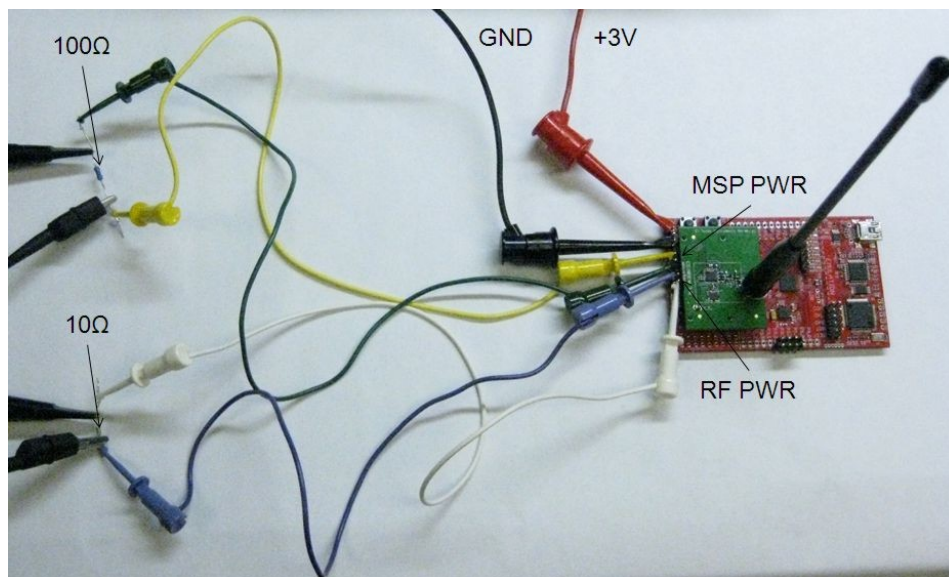
Transmitting, receiving, sleeping, and waking from sleep are some of the several states in which the device operates throughout the procedure. Each state is active for a different amount of time, and switching between states may take varying amounts of time. All of these factors must be taken into account if an accurate power consumption result is to be found (see [Figure 39](#)). Average current factors both amount of time spent in each state and an individual state's power consumption. Power consumption results for the Wireless Update use average current, because the metric uses multiple sources of information. See *Measuring Bluetooth® Low Energy Power Consumption* ([SWRA347](#)) for more details on current consumption measurements.



**Figure 39. Current Measurement of a Packet Transmission**

### 5.1.1 Test Apparatus

Testing for current consumption of the CC1101 transceiver and the MSP430FR5739 was done using the MSP-EXP430FR5739 Experimenter Board's RF PWR and MSP PWR headers respectively. Two shunt resistors were connected across the headers using jumper wires, and voltage drops across the shunt resistors were measured using an oscilloscope to calculate current consumption. A regulated DC power supply was used to externally power the board to avoid excess current draw from the USB and eZ-FET. All jumpers connecting the EMULATION circuitry were removed during testing (see [Figure 40](#)).



**Figure 40. Power Consumption Test Apparatus**

### 5.1.2 Test Procedure

The GPIO pins connecting the MSP430FR5739 with the 3 axis accelerometer and NTC thermistor were set to output low to effectively disconnect these external devices. LED functionality was removed by modifying the Wireless Update project. The Double\_Blink demonstration application was used as a primary source for test results. Through the use of an empty array residing in FRAM, the Double\_Blink demonstration application was also used as a secondary source except with 0x00 filling the remainder of main application memory. The procedure outlined in *Running the Wireless Update* was used to perform current consumption testing.

### 5.1.3 Test Calculations

Calculations for current consumption are completed using Ohm's law. A 10-Ω and 100-Ω resistor are used for the RF and MSP PWR headers, respectively. Substituting these resistor values into Ohm's law, we find the following two equations for instantaneous current consumption of the CC1101 transceiver and the MSP430FR5739 device:

$$I_{CC1101} = \frac{V_{Shunt,CC1101}}{R_{Shunt,CC1101}} = \frac{V_{Shunt,CC1101}}{10\ \Omega} = \frac{V_{Shunt,CC1101}}{10}$$

$$I_{MSP430} = \frac{V_{MSP430}}{R_{Shunt,MSP430}} = \frac{V_{MSP430}}{100\ \Omega} = \frac{V_{MSP430}}{100} \quad (1)$$

A weighted average is used to find the average current for each state of the CC1101 transceiver and MSP430FR5739 device.

#### 5.1.3.1 Link Event With Access Point

Linking with the USB RF access point occurs twice during every wireless update. The discovery reply packet issued by the CC1101 to initiate a link is identical for each link event. The initialization packet to begin WBSL or firmware packet transmission is also identical for each wireless update and is included in the link analysis. Current consumed by the CC1101 and MSP430FR5739 between Wireless Updates is similar as a result of the consistencies between link events (see [Figure 41](#), [Table 1](#), and [Table 2](#)).

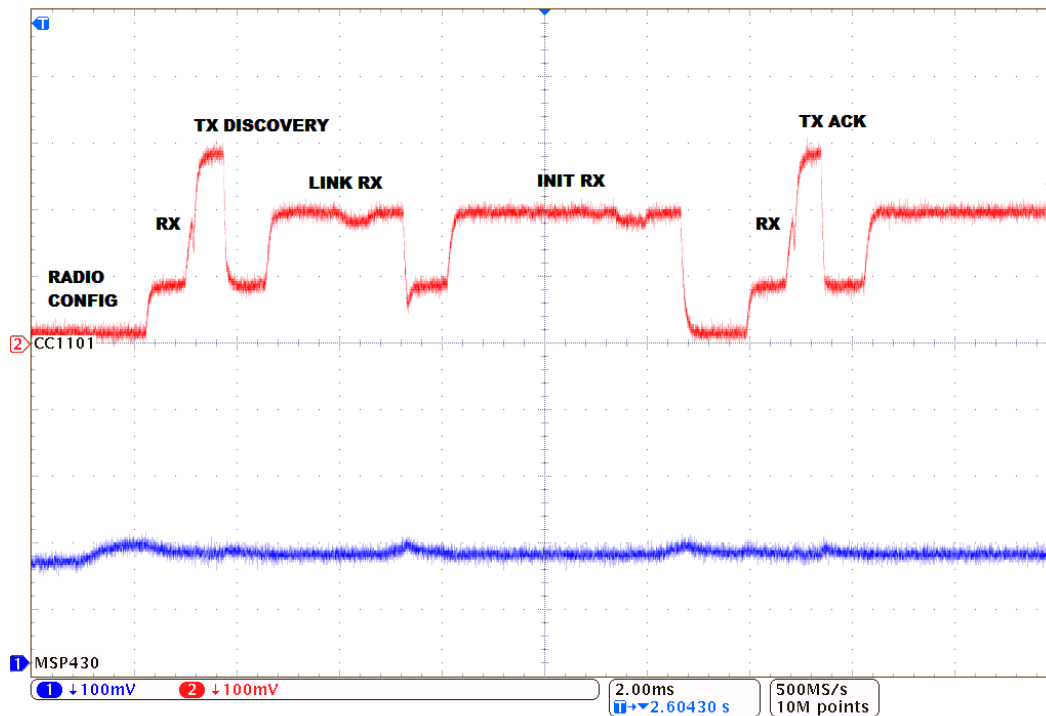


Figure 41. Current Measurement of a Link Event

Table 1. CC1101 Average Current of a Link Event

CC1101 State	Time (ms)	Average Current (mA)
RADIO CONFIG	1.13	1.4
TX DISCOVERY	1.65	15.8
RX LINK	3.54	16.7
RX INIT	6.54	11.1
TX ACK	1.60	14.7

Table 2. MSP430FR5739 Average Current of a Link Event

MSP430FR5739 State	Time (ms)	Average Current (mA)
RADIO CONFIG	1.13	1.78
TX DISCOVERY	1.65	1.72
RX LINK	3.54	1.71
RX INIT	6.54	1.70
TX ACK	1.60	1.70

### 5.1.3.2 WBSL Transmission

The WBSL is transmitted one time during every Wireless Update. The WBSL data packet(s) received by the CC1101 does not change after development for a specific wireless update procedure is completed, but it may be different from what is shown in the following analysis. Similar to each link event, the current consumed by the CC1101 and MSP430FR5739 for separate wireless updates is comparable (see [Figure 42](#), [Table 3](#), and [Table 4](#)).

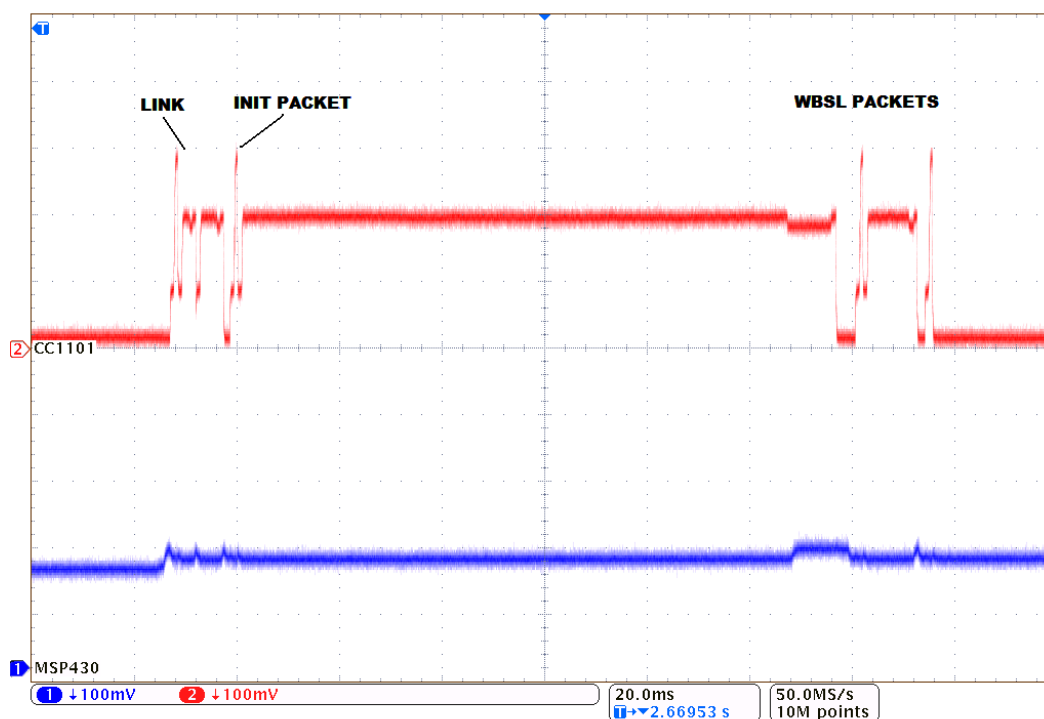


Figure 42. Current Measurement of a WBSL Transmission

Table 3. CC1101 Average Current of a WBSL Transmission

CC1101 State	Time (ms)	Average Current (mA)
Link Event	14.46	12.7
First WBSL Packet	122.04	19.0
Second WBSL Packet	13.49	16.6

Table 4. MSP430FR5739 Average Current of a WBSL Transmission

MSP430FR5739 State	Time (ms)	Average Current (mA)
Link Event	14.46	1.71
First WBSL Packet	122.04	1.71
Second WBSL Packet	13.49	1.71

### 5.1.3.3 Firmware Packet Transmission

Data packets containing firmware data and code may be much larger and result in higher current consumption than those required to link with the access point or transmit the WBSL. Due to the varied nature of these packets, current consumption varies based on the application. Analysis is performed on the largest data packet for worst-case results. Due to changes in USB RF access point timings, further analysis is performed for data packets which are received after the initial data packet has been received. These packets contain a smaller delay than the one seen during RX PACKET in [Figure 43](#) and [Table 5](#) through [Table 8](#).

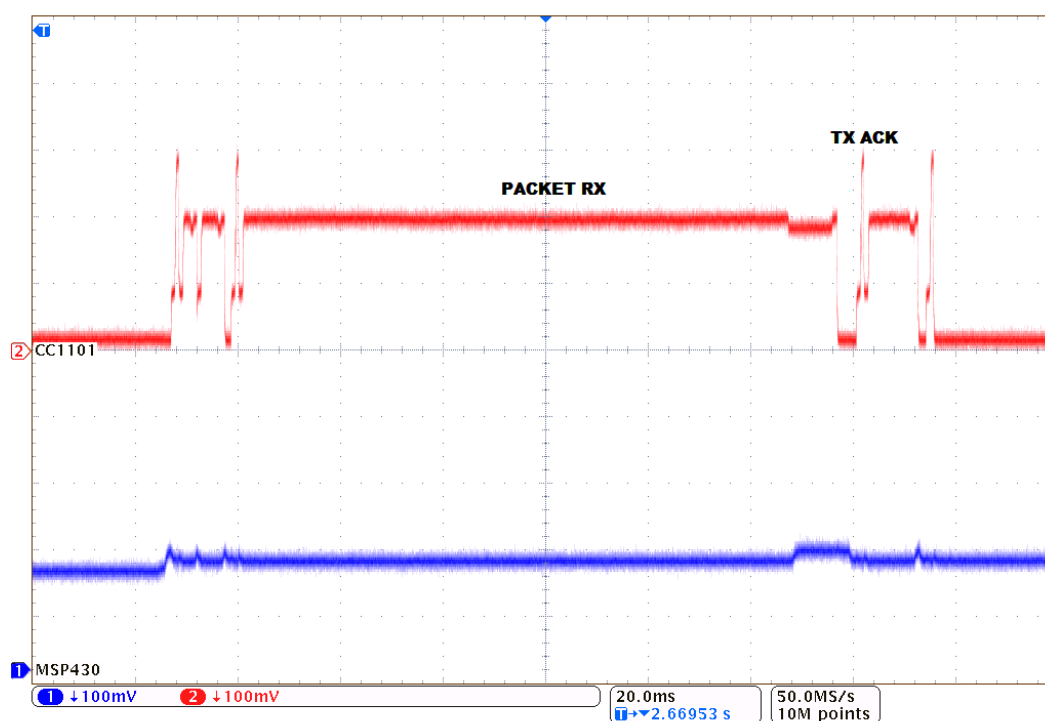


Figure 43. Current Measurement of the First Firmware Packet

Table 5. CC1101 Average Current of the First Firmware Packet

CC1101 State	Time (ms)	Average Current (mA)
RX PACKET	120.44	19.1
TX ACK	1.60	15.4

Table 6. MSP430FR5739 Average Current of the First Firmware Packet

MSP430FR5739 State	Time (ms)	Average Current (mA)
RX PACKET	120.44	1.71
TX ACK	1.60	1.70

Table 7. CC1101 Average Current of Subsequent Firmware Packets

CC1101 State	Time (ms)	Average Current (mA)
RX PACKET	107.41	19.1
TX ACK	1.60	15.4

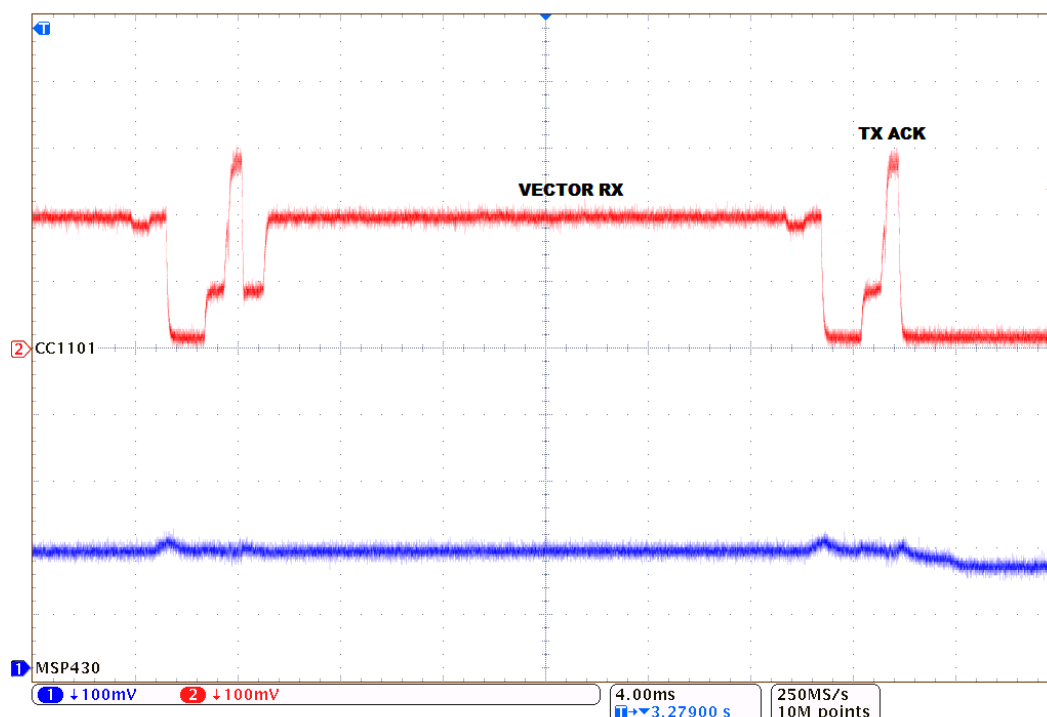


**Table 8. MSP430FR5739 Average Current of Subsequent Firmware Packets**

MSP430FR5739 State	Time (ms)	Average Current (mA)
RX PACKET	107.41	1.71
TX ACK	1.60	1.70

#### 5.1.3.4 Reset and Interrupt Vector Packet Transmission

Data packets containing the reset vector or various interrupt vectors are very small and result in lower current consumption. Variations in size may exist if two interrupt vectors reside adjacent to each other in FRAM, but analysis is only performed on a single interrupt vector sized packet (see [Figure 44](#), [Table 9](#), and [Table 10](#)).



**Figure 44. Current Measurement of a Reset Vector Packet**

**Table 9. CC1101 Average Current of a Reset Vector Packet**

CC1101 State	Time (ms)	Average Current (mA)
RX VECTOR	24.01	18.1
TX ACK	1.54	15.2

**Table 10. MSP430FR5739 Average Current of a Reset Vector Packet**

MSP430FR5739 State	Time (ms)	Average Current (mA)
RX VECTOR	24.01	1.70
TX ACK	1.54	1.70

### 5.1.4 Wireless Update Results and Comparisons

Analysis for current consumption of the Wireless Update is performed using the data shown in [Section 5.1.3.4](#) and results from a Wireless Update using the Blink\_Demo project for firmware data packet and interrupt vector packet counts. Weighted averages were once again used to find the overall Average Current. Further analysis is performed to develop an equation to calculate average current based on the firmware data packet and interrupt vector data packet counts for a custom main application (see [Table 11](#) and [Table 12](#)).

**Table 11. CC1101 Average Current of a Wireless Update**

Wireless Update State	Time (ms)	Average Current (mA)
WBSL Transmission	149.99	18.2
AP Processing Application Packets	404.45	1.4
Firmware Packet	122.04	19.0
Timer Interrupt Vector	25.55	17.9
Reset Interrupt Vector	25.55	17.9
TOTAL	727.58	8

**Table 12. MSP430FR5739 Average Current of a Wireless Update**

Wireless Update State	Time (ms)	Average Current (mA)
WBSL Transmission	149.99	1.71
AP Processing Application Packets	404.45	1.70
Firmware Packet	122.04	1.71
Timer Interrupt Vector	25.55	1.70
Reset Interrupt Vector	25.55	1.70
TOTAL	727.58	1.70

Using the data shown previously, the following equation can be used to estimate total average currents for both the CC1101 and MSP430FR5739 device during a Wireless Update for a custom main application. The first term is the weighted average current for all states which are consistent between Wireless Updates. Following terms are application dependent, with the firmware packet count calculation shown in [Equation 2](#) and the interrupt vector count calculated by summing the number of distinct interrupt vectors used in the application.

$$x_{\text{packets}} = \frac{\text{Firmware Code Size (bytes)}}{255 \text{ bytes}}$$

$$I_{\text{AVG,CC1101}} \text{ (mA)} = \frac{3753.39 + (2318.76 \times x_{\text{packets}}) + (457.35 \times x_{\text{interrupts}})}{579.99 + (122.04 \times x_{\text{packets}}) + (25.55 \times x_{\text{interrupts}})}$$

$$I_{\text{AVG,MSP430}} \text{ (mA)} = \frac{987.48 + (208.69 \times x_{\text{packets}}) + (43.44 \times x_{\text{interrupts}})}{579.99 + (122.04 \times x_{\text{packets}}) + (25.55 \times x_{\text{interrupts}})} \quad (2)$$

Flash write speed and power consumption values were taken from the *MSP-EXP430FR5739 FRAM Experimenter Board User's Guide* ([SLAU343](#)). Section 2.2.3.1 *The Math Behind Mode 2* explains that 512 bytes write time for a flash device is roughly equivalent to 51.2 ms. Halving this value to reach 256 bytes is equivalent to a maximum sized firmware packet and results in a total write time of 41.6 ms when segment erase time is included. Total write time for a maximum sized firmware packet using FRAM during a Wireless Update was found to be 2.3 ms. Flash-based MCUs also require increased average currents reaching 2.2 mA for high write speeds, resulting in the Firmware Packet's Average Current to increase from 1.71 mA to 1.85 mA and the Firmware Packet's Time to increase from 122.04 ms to 161.34 ms. Comparison for an application of maximum size (52 firmware packets) with consideration of the timing and average current changes for flash-based devices is shown in [Table 13](#).

**Table 13. FRAM and Flash MSP430 Wireless Update Comparison**

Memory Technology	Total Time (ms)	Average Current (mA)
FRAM	7073.66	1.70
Flash	8995.22	1.84

## 5.2 MSP430FR5739 Memory Organization

Memory organization for the Wireless Update project is primarily driven by the need to place the CBSL outside of the device BSL memory (see [Table 14](#)). The MSP430FR5739 device uses ROM instead of flash for this memory area, unlike MSP430x5xx/MSP430x6xx devices. The CBSL controls the reset vector of the device to enable forced execution of the CBSL prior to main application execution if the S1 button is held down. Main application memory space is placed adjacent to the CBSL, and RAM is shared between main application firmware and Wireless Update firmware.

**Table 14. Wireless Update MSP430FR5739 Memory Organization**

Memory Region	Start – End Address	Size	Owner
CBSL Reset Vector (FRAM)	0xFFFF – 0xFFFE	0x0002 = 2 bytes	CBSL
Device Interrupt Vectors (FRAM)	0xFFFD – 0xFF80	0x007E = 126 bytes	Main Application Firmware. This region can be written to by the WBSL. The main application uses this space for program execution.
CBSL Firmware (FRAM)	0xF7B6 – 0xFF7F	0x07CA = 1994 bytes	CBSL
Main Application Firmware (FRAM)	0xC200 – 0xF7A7	0x35A8 = 13736 bytes	Main Application Firmware. This region can be written to by the WBSL. The main application uses this space for program execution.
RAM	0x1FFF – 0x1C00	0x0400 = 1024 bytes	CBSL, WBSL, Main Application. This region can be written to by the CBSL. The WBSL firmware uses this space for program execution. Both the CBSL and main application may use this space for general purpose memory.
Device Description Information (TLV) (FRAM)	0x1A7F – 0x1A00	0x0080 = 128 bytes	Shared
Information Memory (FRAM)	0x19FF – 0x1800	0x0200 = 512 bytes	Shared
Device BSL Memory (ROM)	0x17FF – 0x1000	0x800 = 2048 bytes	Shared
Peripherals	0x0FFF – 0x0000	0x1000 = 4096 bytes	Shared

## 5.3 Firmware Memory Size

Due to the CBSL residing in FRAM, available code size for the main applications is reduced (see [Table 15](#)). Use of the WBSL reduces the impact on FRAM size reduction by placing enhanced functionality temporarily in RAM during a wireless update. Maximum available FRAM for the main application is calculated by subtracting the area taken by the CBSL from the total FRAM area of the device. The size of the CBSL and WBSL are measured using the optimizations that minimized FRAM area code size (--opt\_level=2, --opt\_for\_speed=0). Table 15 summarizes address ranges and sizes for the CBSL, WBSL, and main application.

**Table 15. Firmware Memory Sizes for Wireless Update**

Firmware	Start Address	End Address	Size
WBSL <sup>(1)</sup>	0x1C00	0x1E4F	0x0250 = 592 bytes
CBSL	0xF7A8	0xFF7F	0x07D8 = 2008 bytes

<sup>(1)</sup> Address range and size does not include RESET vector located at 0xFFFE with size 0x0002 = 2 bytes.

**Table 15. Firmware Memory Sizes for Wireless Update (continued)**

Firmware	Start Address	End Address	Size
Main Application <sup>(2)</sup>	0xC200	0xF7A7	0x2E3A = 13736 bytes

<sup>(2)</sup> Address range and size includes RESET vector located at 0xF7CA and assumes that all available space is used for the application.

## 5.4 Radio Timing

While the CC1101 is transmitting or receiving, a large amount of power is being consumed by the radio. Embedded applications that need to conserve power should rarely perform wireless updates to avoid excess power consumption. Timings were calculated using an oscilloscope and a GPIO signal on the MSP-EXP430FR5739 Experimenter Board which indicated when certain stages of the wireless update were operating (see [Table 16](#)).

**Table 16. Radio Timing for Wireless Update**

Wireless Update Stages	Time (ms)
Radio configuration – Link with access point	5.20
Radio configuration – WBSL download complete	300
Radio configuration – First main application packet received	2600
Radio configuration – Main application <sup>(1)</sup> download complete	8500

<sup>(1)</sup> An empty array was used to fill the entire main application memory space for maximum download time calculation. Time varies based on main application size.

## 5.5 RF Parameters

Configuration of the CC1101 radio is done through a SPI interface which accesses registers on the CC1101 device (see [Table 17](#)). Communication with the RF USB access point is enabled with the RF parameters used by the WBSL. As mentioned in Chapter 3, the RF parameters need to be changed in order to meet local RF regulatory restrictions for final end-products. Description of each value was found using SmartRF Studio 7. More details can be found in the *SmartRF Studio 7 Tutorial*.

**Table 17. CC RF Configuration for Wireless Update**

CC RF Configuration Register	Value
FSCTRL1	0x0C
FSCTRL0	0x00
SYNC1	0xD3
SYNC0	0x91
FREQ2	0x22
FREQ1	0xB1
FREQ0	0x3B
CHANNR	0x14
WBSL_SETTING_PATABLE	0x8B
MDMCFG4	0x2D
MDMCFG3	0x3B
MDMCFG2	0x13
MDMCFG1	0x22
MDMCFG0	0xF8
DEVIATN	0x62
FREND1	0xB6
FREND0	0x10
MCSM2	0x07
MCSM1	0x3C

**Table 17. CC RF Configuration for Wireless Update (continued)**

CC RF Configuration Register	Value
MCSM0	0x18
WOREVT1	0x87
WOREVT0	0x6B
WORCTL	0xF8
FOCCFG	0x1D
BSCFG	0x1C
AGCCTRL2	0xC7
AGCCTRL1	0x00
AGCCTRL0	0xB0
FSCAL3	0xEA
FSCAL2	0x2A
FSCAL1	0x00
FSCAL0	0x1F
FSTEST	0x59
TEST2	0x88
TEST1	0x31
TEST0	0x09
PTEST	0x7F
AGCTEST	0x88
FIFOTHR	0x07
IOCFG2	0x06
IOCFG1	0x1B
IOCFG0	0x09
PKTCTRL1	0x05
PKTCTRL0	0x45
ADDR	0xAD
PKTLEN	0xFF

## 6 References

1. *MSP430x5xx and MSP430x6xx Family User's Guide* ([SLAU208](#))
2. *MSP430FR573x Mixed-Signal Microcontrollers* ([SLAS639](#))
3. *MSP430FR572x Mixed-Signal Microcontrollers* ([SLASE35](#))
4. *CC1101 Low-Power Sub-1 GHz RF Transceiver* ([SWRS061](#))
5. *eZ430-Chronos Development Tool User's Guide* ([SLAU292](#))
6. *MSP-EXP430FR5739 FRAM Experimenter Board User's Guide* ([SLAU343](#))

Revision History

Changes from Original (January 2012) to A Revision	Page
<ul style="list-style-type: none"> <li>Changed title of document ..... 1</li> <li>Added "Over-the-air (OTA) updates..." to first sentence in abstract..... 1</li> </ul>	

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.



## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)