RF430CL330H Target Board

User's Guide

TEXAS
INSTRUMENTS

Contents

List of Figures

Code Composer Studio is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

RF430CL330H Target Board

1   RF430CL330HTB Description

The RF430CL330H target board is an evaluation platform for the RF430CL330H NFC Type 4B Tag Platform supporting the following features:

- I2C or SPI Interface
- ISO14443B Protocol
- NFC Tag Type-4B Compliant
- 3K bytes available for NDEF message storage
- Through holes of all pins for debug and development
- On board antenna or optional external antenna connection (remove R3 and R4 for external antenna)

### 1.1   If You Need Assistance

If you have any feedback or questions, support for NFC devices and the RF430CL330H is provided by the Texas Instruments Product Information Center (PIC) and the TI E2E Forum (https://community.ti.com/forums/12.aspx). Contact information for the PIC can be found on the TI web site at http://support.ti.com.

2   Default Configuration

As shipped, the RF430CL330HTB hardware is configured for I2C mode (CS pin = 0, pull up resistors on SDA/SCL) with address 0x28(E0, E1, E2 = 0). The board must be connected to a host controller in order to set up the NDEF message and properly set registers. In this document, communication with the MSP-EXP430FR5739 experimenter's board is discussed, but the connections will be the same when different host controllers are implemented.

3   Hardware Description and Schematic

The RF430CL330HTB allows for the developer to become familiar with the RF430CL330H NFC Type 4B Tag Platform IC. The board can be used with many different TI microcontroller development platforms that use the Samtec EM headers. If the headers are not available on a specific microcontroller platform, through holes are available to allow easy access to all pins. The target board features a PCB antenna that wraps around the outside edge of the PCB. To connect an external antenna, R3 and R4 zero Ω resistors must be removed to disconnect the on board antenna and the external antenna coil can connect to the ANT1 and ANT2 pins available on JP1. The schematic and layout can be seen in Figure 1 and Figure 2.
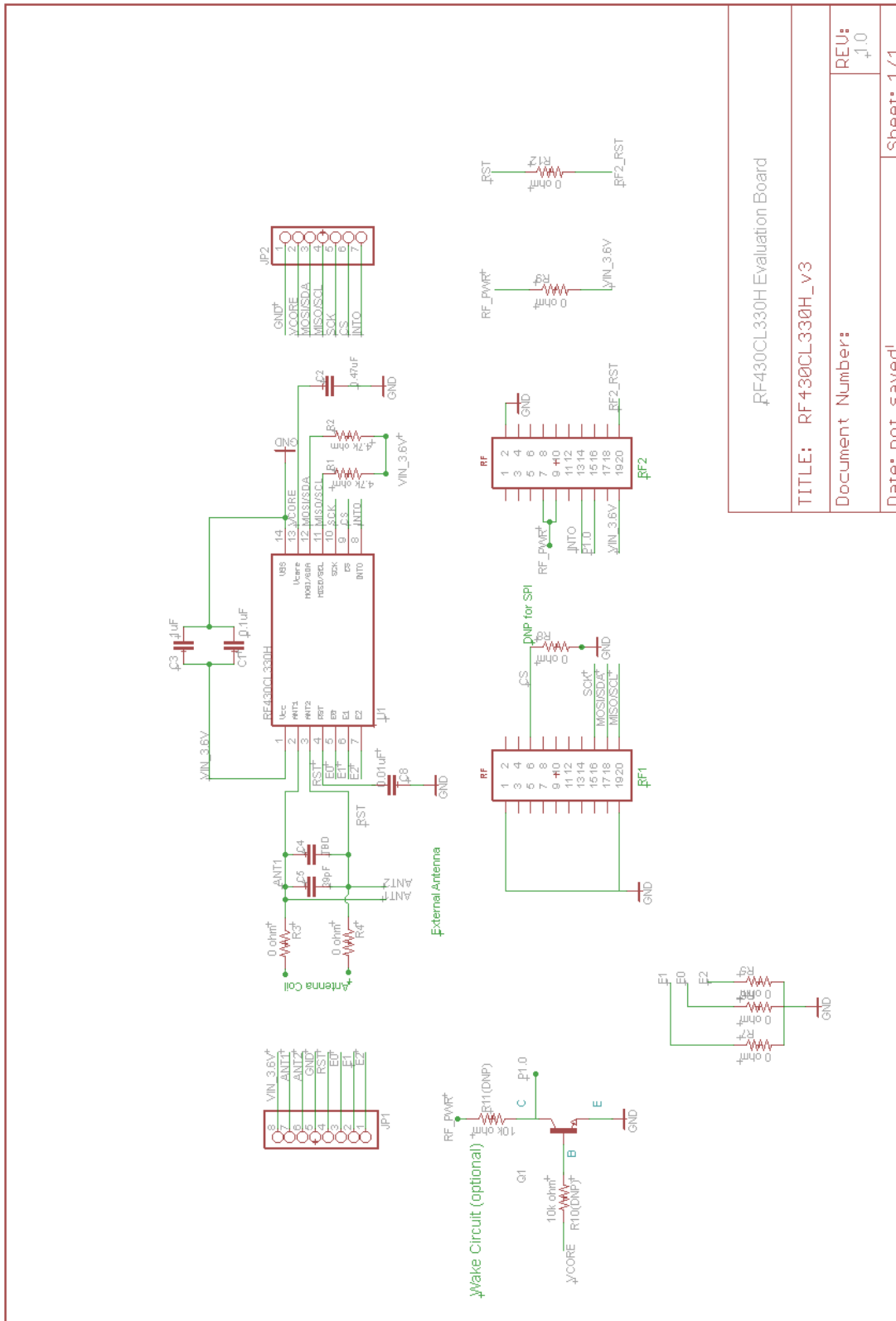
Figure 1. RF430CL330HTB Schematic
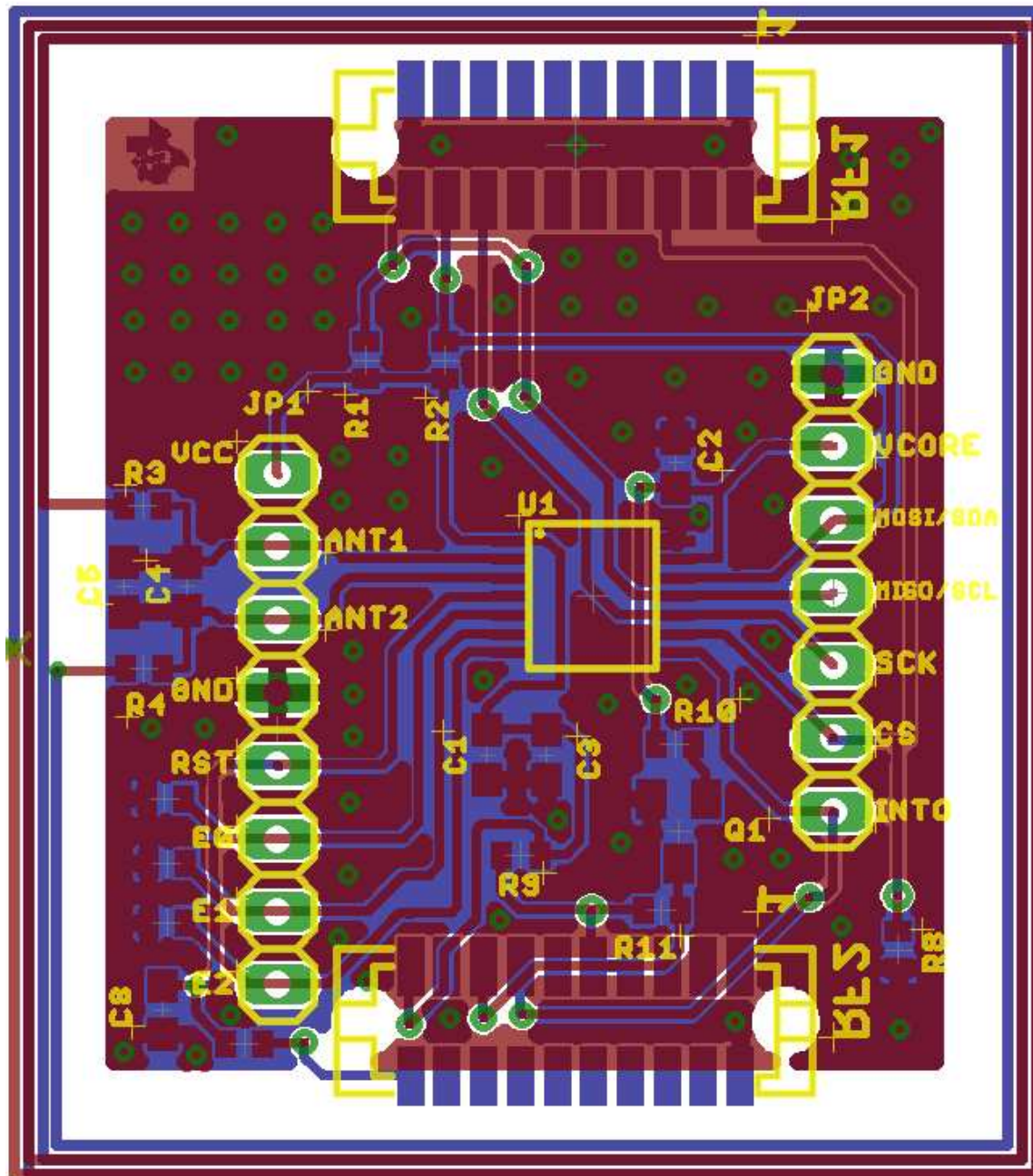
Figure 2. RF430CL330HTB Layout

4   Operation as an NFC Forum Type 4B Tag Platform

4.1   Command Sequence Between NFC Reader/Writer and Type 4B Tag Platform

Figure 3 shows the communication between the NFC Reader and Writer, referred to as the proximity coupling device (PCD) and the RF430CL330H, referred as the proximity integrated circuit card (PICC). All NFC communication is handled internally by the RF430CL330H and does not require any interaction from the host controller.
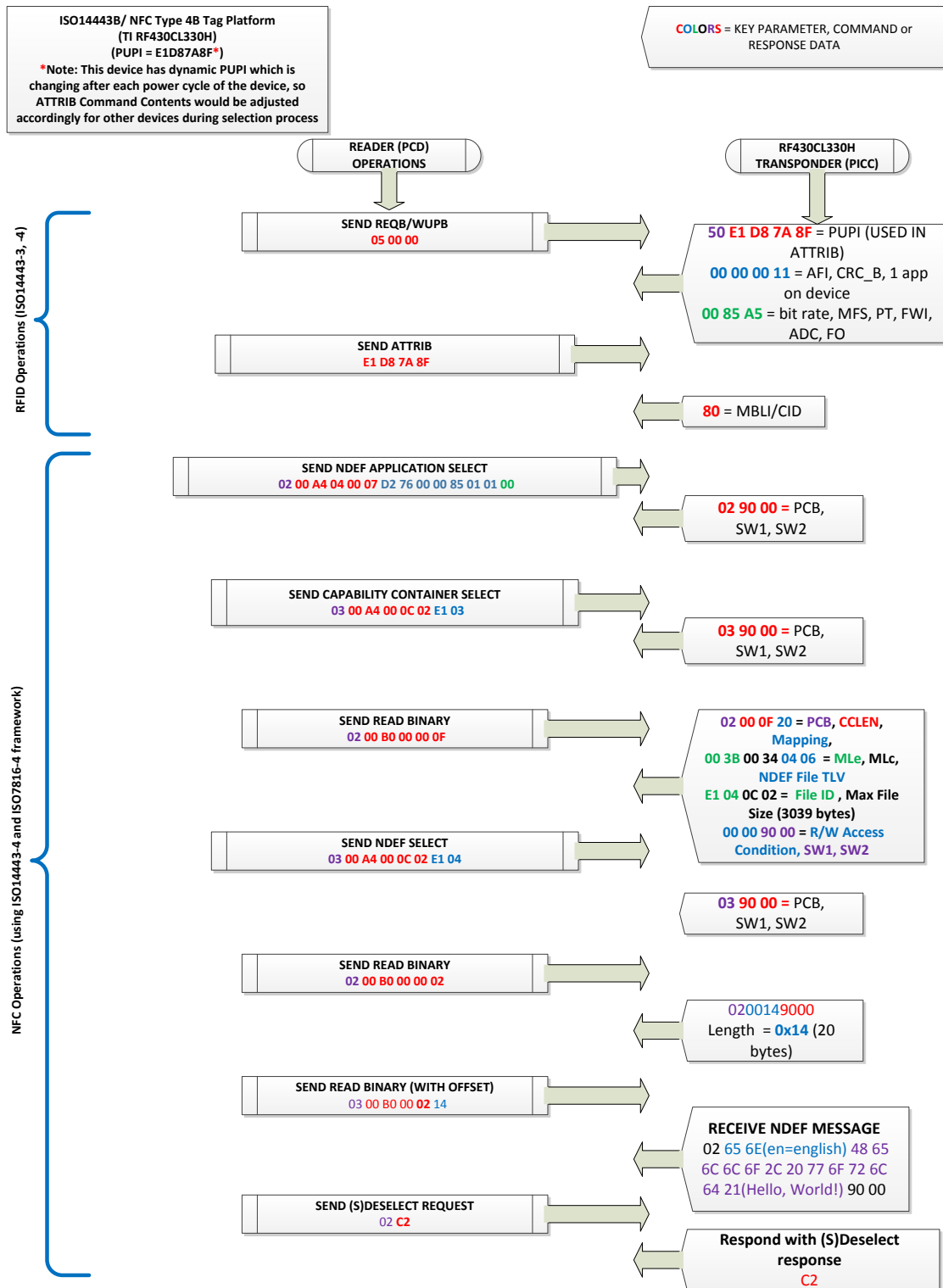
Figure 3. NFC Forum Type 4B Tag Platform Command Sequence

5   Updating the NDEF Message via Serial Interface

The RF430CL330H NDEF memory can be accessed via NFC or serial interface. This section discusses updating of the NDEF message via serial interface.

5.1   Typical Usage Example

1. Write capability container and message(s) into the NDEF memory (starting from address 0) via the serial interface.
2. Enable interrupts if required (End of Read and End of Write to indicate completion of read and write procedure).
3. Configure the interrupt pin INTO as needed and enable the RF interface. RF430CL330H is now ready to communicate via NFC.
4. Wait for interrupt signaled via INTO.
5. Disable RF interface (but keep INTO settings unchanged).
6. Read interrupt flag register to determine interrupt source(s).
7. Clear interrupt flags. INTO should go back into inactive state.
8. Read and modify NDEF memory as needed.
9. Enable RF interface again (keeping INTO settings unchanged) and continue with "Wait for interrupt" ().

5.2   NDEF Code Examples

In the example project, the NDEF message can be found in RF430.h. This project example uses a simple "Hello World!" message for demonstration purposes. To change this message, the guidelines found in the NFC Data Exchange Format specification must be followed. Additional information on the formatting of the NDEF message can also be found in the corresponding NFC Forum record type definition specification. Section 5.3 – Section 5.4 show examples of how this code can be modified for different message types.

5.3   Text Type NDEF message

```
#define RF430_DEFAULT_DATA        {
/*NDEF Tag Application Name*/                                                        \
0xD2, 0x76, 0x00, 0x00, 0x85, 0x01, 0x01,                                           \
/*Capability Container File ID*/                                                     \
0xE1, 0x03,                                                                          \
0x00, 0x0F, /* CCLEN */                                                              \
0x20,       /* Mapping version 2.0 */                                                \
0x00, 0x3B, /* MLe (49 bytes); Maximum R-APDU data size */                           \
0x00, 0x34, /* MLc (52 bytes); Maximum C-APDU data size */                           \
0x04,       /* Tag, File Control TLV (4 = NDEF file) */                              \
0x06,       /* Length, File Control TLV (6 = 6 bytes of data for this tag) */  \
0xE1, 0x04, /* File Identifier */                                                    \
0x0B, 0xDF, /* Max NDEF size (3037 bytes of useable memory) */                       \
0x00, /* NDEF file read access condition, read access w/o any security */      \
0x00, /* NDEF file write access condition, write access w/o any security */    \
/* NDEF File ID */                                                                   \
0xE1, 0x04,                                                                          \
/* NDEF File for Hello World  */                                                     \
0x00, 0x14, /* NLEN; NDEF length (20 byte long message) */                           \
0xD1,       /* Record Header */                                                      \
0x01, 0x10, /*type length, payload length */                                         \
0x54,       /* T = text */                                                           \
0x02,       /* Status Byte (2 byte language code) */                                 \
0x65, 0x6E, /* 'e', 'n', */                                                          \
/* Payload Data 'Hello, world!' */                                                   \
0x48, 0x65, 0x6C, 0x6C, 0x6f, 0x2c, 0x20,                                            \
0x77, 0x6f, 0x72, 0x6c, 0x64, 0x21                                                   \
}
```

5.4 Bluetooth Handover Type NDEF Message

```
#define RF430_DEFAULT_DATA         {                                              \
/*NDEF Tag Application Name*/                                                      \
0xD2, 0x76, 0x00, 0x00, 0x85, 0x01, 0x01,                                         \
/*Capability Container File ID*/                                                   \
0xE1, 0x03,                                                                        \
0x00, 0x0F,   /* CCLEN */                                                          \
0x20,         /* Mapping version 2.0 */                                            \
0x00, 0x3B,   /* MLe (49 bytes); Maximum R-APDU data size */                       \
0x00, 0x34,   /* MLc (52 bytes); Maximum C-APDU data size */                       \
0x04,         /* Tag, File Control TLV (4 = NDEF file) */                          \
0x06,         /* Length, File Control TLV (6 = 6 bytes of data for this tag) */ \
0xE1, 0x04,   /* File Identifier */                                                \
0x0B, 0xDF,   /* Max NDEF size (3037 bytes of useable memory) */                   \
0x00, /* NDEF file read access condition, read access w/o any security */         \
0x00, /* NDEF file write access condition, write access w/o any security */        \
/* NDEF File ID */                                                                 \
0xE1, 0x04,                                                                        \
0x00, 0x44,   /* NLEN; NDEF length (68 byte long message) */                       \

0xD2,         /* MB=1b, ME=1b, CF=0b, SR=1b, IL=0b, TNF=010b */         \

0x20,         /* Record Type Length: 32 octets */                      \

0x21,         /* payload length: 33 octets */                          \

/* Record Type Name: application/vnd.bluetooth.ep.oob */               \

0x61, 0x70, 0x70, 0x6C, 0x69, 0x63, 0x61, 0x74, 0x69, 0x6F, 0x6E, 0x2F, \

0x76, 0x6E, 0x64, 0x2E, 0x62, 0x6C, 0x75, 0x65, 0x74, 0x6F, 0x6F, 0x74, \

0x68, 0x2E,0x65, 0x70, 0x2E, 0x6F, 0x6F, 0x62,                          \

0x21, 0x00,   /* OOB optional data length: 33 bytes */                  \

0x06, 0x05, 0x04, 0x03, 0x02, 0x01, /* bluetooth device address:        \

01:02:03:04:05:06 (example address only) */                            \

0x0D,         /* EIR Data Length: 13 octets */                          \

0x09,         /* EIR Data Type: Complete Local Name */                  \

0x48, 0x65, 0x61, 0x64, 0x53, 0x65, 0x74, 0x20, 0x4E, 0x61,0x6D, 0x65,  \

/* Bluetooth Local Name: HeadSet Name */                               \

0x04,         /* EIR Data Length: 4 bytes */                            \

0x0D,         /* EIR Data Type: Class of device */                      \

0x04, 0x04, 0x20,  /* Class of Device: 0x20:Service Class= Audio,       \

0x04:Major Device Class=Audio/Video,                                   \
0x04: Minor Device Class=Wearable Headset Device */                    \
0x05,  /* EIR Data Length: 5 octets */                                 \

0x03,  /* EIR Data type: 16-bit Service Class UUID list (complete) */   \

0x1E, 0x11, 0x0B, 0x11  /* 16-bit Service Class UUID list (complete);   \

0x111E – HFP-HF, 0x011B – A2DP-SNK */                                  \
}
```
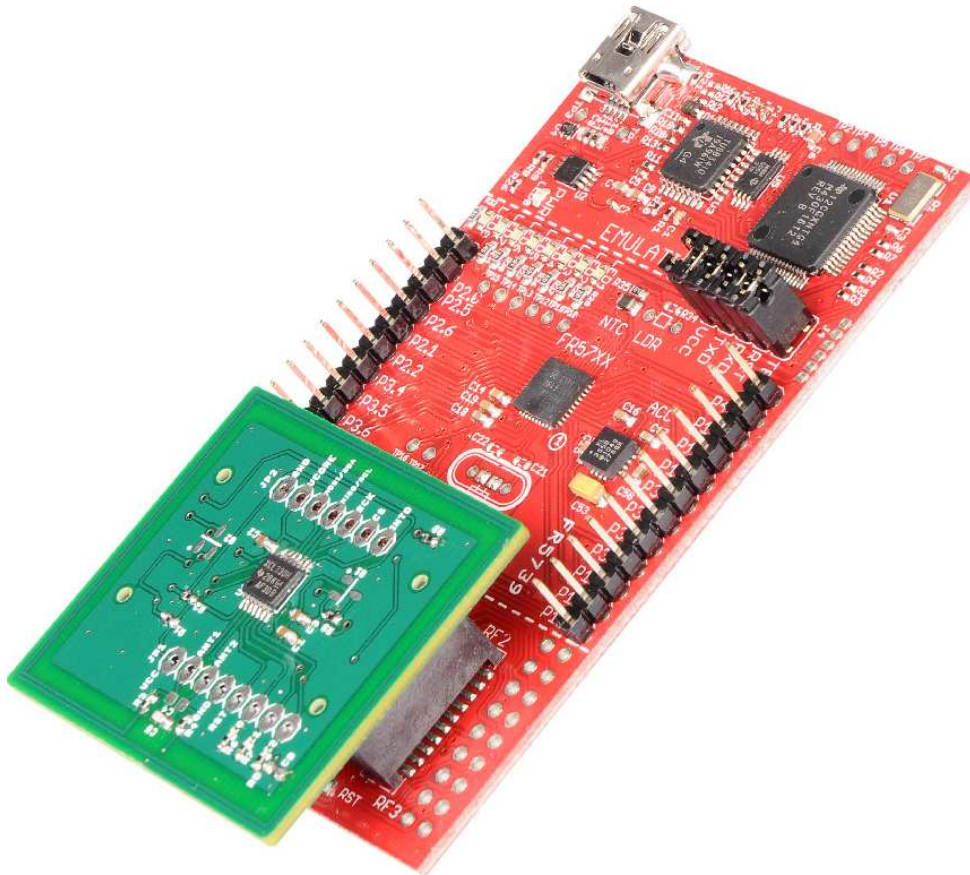
6   Setting Up the MSP-EXP430FR5739 + RF430CL330HTB

Example code is available for the MSP430FR5739 experimenter's board at the link below. This code shows how to initialize the device, write an NDEF message and handle the interrupt at the end of a read or write operation. This example requires the MSP-EXP430FR5739 and the RF430CL330HTB. The RF430CL330H target board must be connected to the RF1 and RF2 headers on the MSP430 experimenter's board as shown in Figure 4.

MSP430FR5739 example code: http://www.ti.com/litv/zip/sloc290

Figure 4. MSP-EXP430FR5739 + RF430CL330HTB

To compile and run the code, Code Composer Studio™ (CCS) must first be installed and the example project extracted to a known directory. Code Composer Studio can be downloaded from the link below. Once Code Composer Studio is installed, a workspace needs to be selected in your desired directory. This directory does not need to be the same as where the project was extracted.

Code Composer Studio Download: http://processors.wiki.ti.com/index.php/Download_CCS
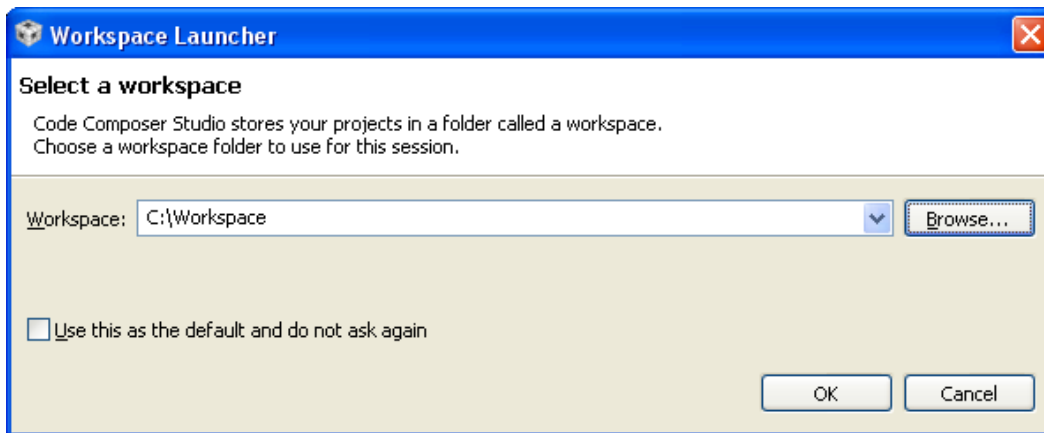
Figure 5. Workspace Selection

After the workspace is selected, go to "Project" → "Import Existing CCS Eclipse Project" (see Figure 6), and then browse to the location of the project. The project will be displayed in the "Discovered projects" box. Select the project and click finish.
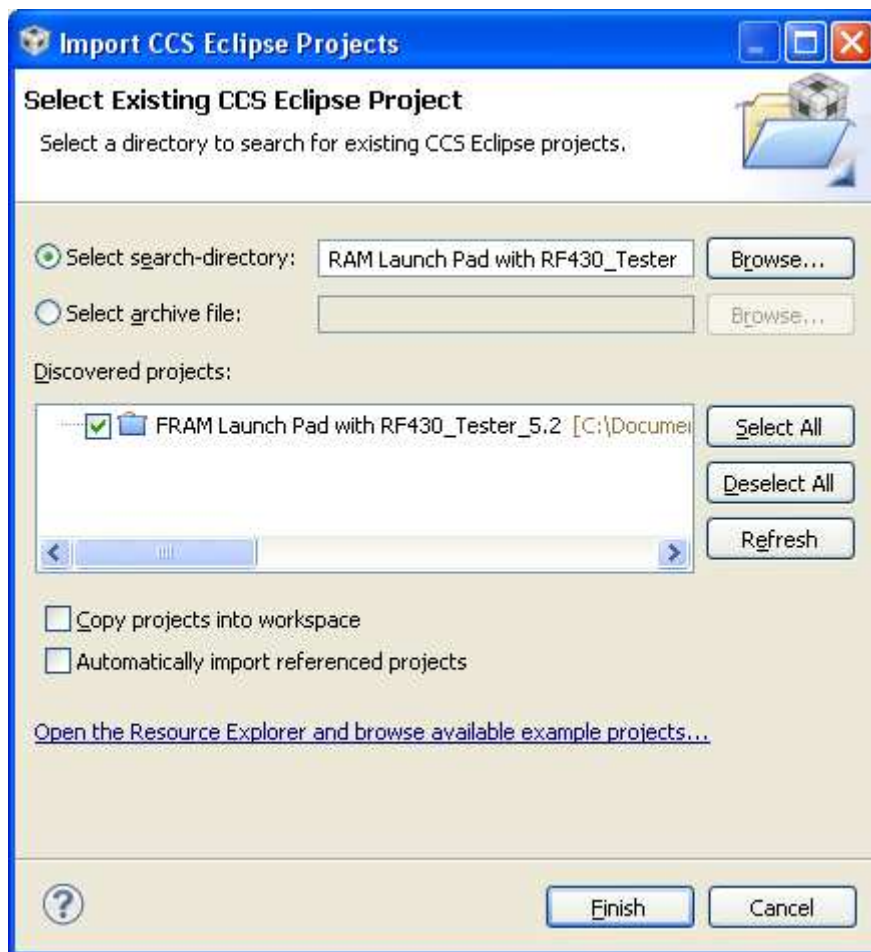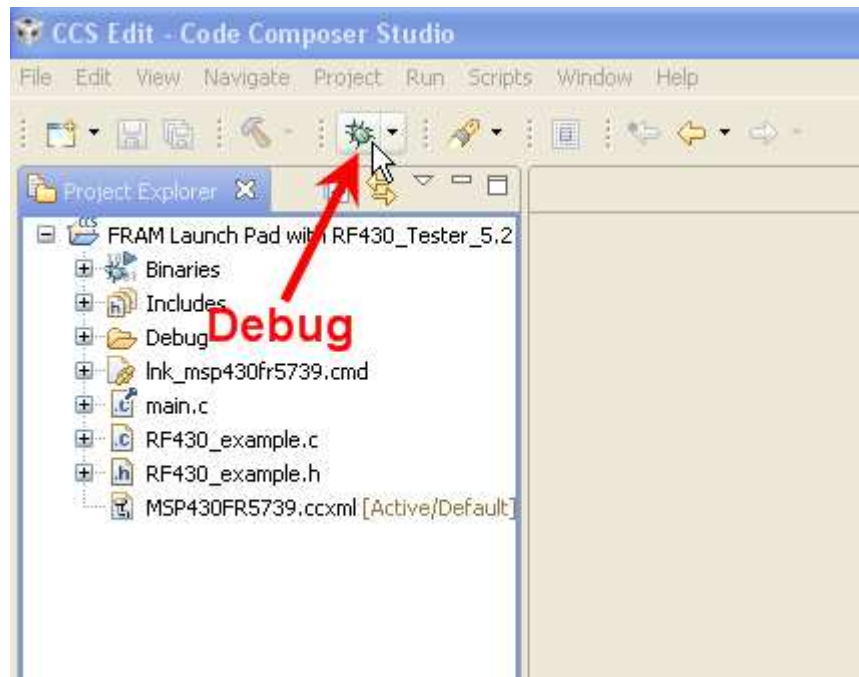

Figure 6. Import Existing CCS Eclipse Project

The project is now imported into the workspace and the project .c and .h files can be seen.
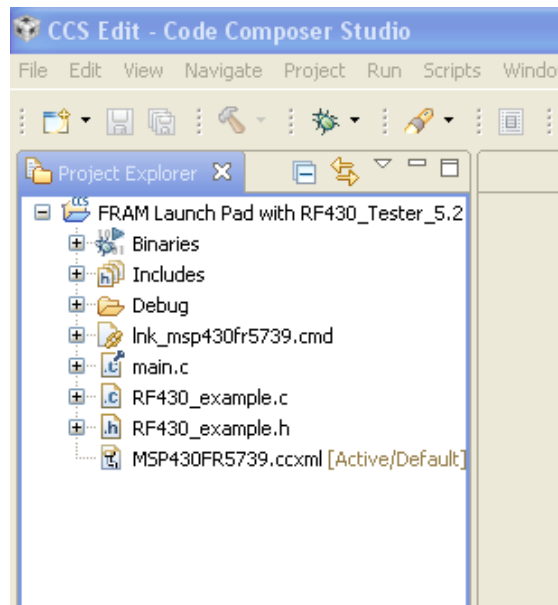
Figure 7. Project Shown in Workspace

Now, plug in the MSP430 experimenters board into the USB port. To compile and download the project to the MSP430, click the debug button.

Figure 8. Debug

After compiling is complete, the project can be run and debugged. Or, simply stop the project and reset the board to run without the development environment, as shown in Figure 9.
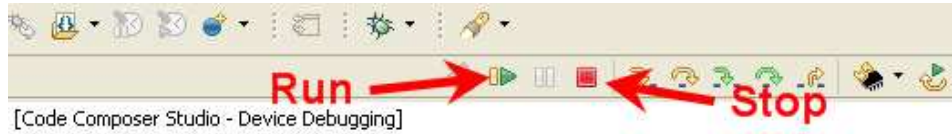
Figure 9. Debug Options

When the code runs, the "Hello, World!" NDEF information will be written to the SRAM, interrupts for end of read/write enabled, INTO pin configured for active low and RF enabled. The code snippet below shows these operations.

```
//write NDEF memory with Capability Container + NDEF message
Write_Continuous(0, NDEF_Application_Data, 104);
//Enable interrupts for End of Read and End of Write
Write_Register(INT_ENABLE_REG, EOW_INT_ENABLE + EOR_INT_ENABLE);
//Configure INTO pin for active low and enable RF
Write_Register(CONTROL_REG, INT_ENABLE + INTO_DRIVE + RF_ENABLE);
```

After RF is enabled, the RF430CL330H is in a state waiting to be read or written by an NFC reader/writer device. An NFC enabled phone or tablet can be presented to the antenna of the RF430CL330H target board and the NDEF message will be read and displayed (see Figure 10). After the read operation is completed, the INTO pin will trigger and subsequently, LED 7 will illuminate for ~1 second.

Figure 10. "Hello, World!" NDEF Message Displayed on Smart Phone

An NFC writer application can also be used to write NDEF data to the RF430CL330H when RF is enabled. Similar to the INTO pin handling of a read operation, a write operation will trigger LED 8 to illuminate for ~1 second. The code snippet below shows the interrupt handling routine.

```
if(into_fired)
        {

//before we read/write to RF430CL330H we should disable RF
Write_Register(CONTROL_REG, INT_ENABLE + INTO_DRIVE); //clear control reg to disable RF
__delay_cycles;(750000);
flags = Read_Register(INT_FLAG_REG); //read the flag register to check if a read or write occurred
Write_Register(INT_FLAG_REG, EOW_INT_FLAG + EOR_INT_FLAG); //ACK the flags to clear
PORT_LED_OUT &= ~(LED6 + LED7);         //clear LEDs

if(flags & EOW_INT_FLAG)                //check if the tag was written
      {
//tag was updated, so we should read out the new data
//read out the data
Read_Continuous(0, read_data, 200);
__no_operation();                                       //breakpoint here to examine the data

//show that tag was written with LEDs
PORT_LED_OUT |= LED6;
__delay_cycles(8000000);
PORT_LED_OUT &= ~(LED6 + LED7);            //clear LEDs
}
else if(flags & EOR_INT_FLAG)         //check if the tag was read
{
//show that tag was read with LEDs
PORT_LED_OUT |= LED7;
__delay_cycles(8000000);
PORT_LED_OUT &= ~(LED6 + LED7);       //clear LEDs
}
flags = 0;
into_fired = 0;                       //we have serviced INTO
```

7 References

- [RF430CL330H Product Folder](#)
- [RF430CL330H Example Code](#)
- [MSP-EXP430FR5739 Tool Folder](#)
- Samtec Header Information:
  - [SFM Series Overview](#)
  - [TFM Series Overview](#)

Revision History

Changes from Original (June 2013) to A Revision   **Page**

- Changed Figure 3 7

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		Applications	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		