Application Report
SLAA460–May 2010

# RF1A Library for CC430F613x/F513x Sub-1GHz SoCs

Miguel Morales

MSP430

**ABSTRACT**

The RF1A radio module consists of the CC1101-based radio core and the RF1A interface, which interfaces the MSP430F5xx and the radio core within the CC430 devices.. The RF1A library provides functions that implement the most common operations when using the RF1A interface, such as sending command strobes, reading and writing the RF configuration registers, and writing to the PA Table. This application report documents the definition and proper use of the available library calls.

The related code files can be downloaded from http://www.ti.com/lit/zip/slaa460.

**Contents**

**List of Figures**

## 1 Glossary of Acronyms

| | |
|---|---|
| FIFO | First In First Out |
| MSB | Most Significant Bit |
| RX | Receive |
| TX | Transmit |
| PA | Power Amplifier |

SmartRF is a registered trademark of Texas Instruments.

# 2 RF1A Library

## 2.1 Introduction to the RF1A Interface

The RF1A interface is the mailbox module between the MSP430 and the CC1101-based radio core, whose registers sit within the linear memory map of the MSP430 and can be accessed as with any other MSP430 peripheral. The radio interface registers, logical channels, and their relation to the radio core, are depicted in Figure 1.

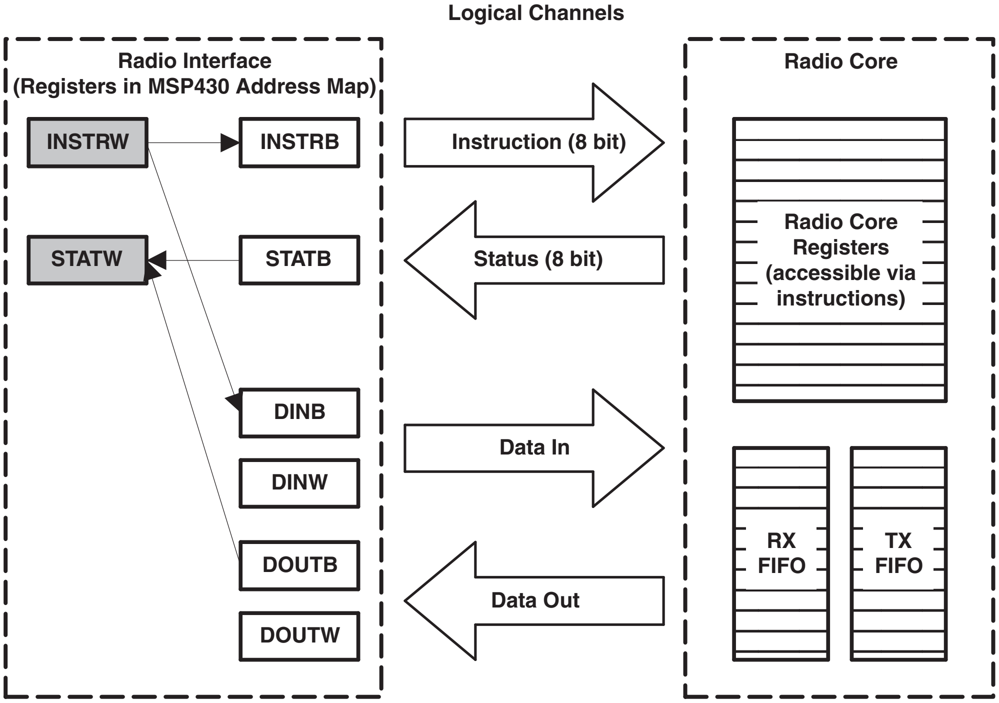


**Figure 1. RF1A Interface Usage**

There are essentially three types of instruction-based (i.e., non-interrupt) interactions that one can make with the CC1101-based core through the RF1A interface:

1. Changing the state of the Radio Core through command strobes (e.g., IDLE to RX mode)
2. Reading from or writing to the radio configuration, PA table, or status registers
3. Read from or writing to the RX and TX FIFOs

The RF1A library provides functions for handling all three scenarios while observing the recommended protocol for interfacing the Instruction, Data, and Status Registers, per the *CC430 Family User's Guide*. For a detailed description of the RF1A interface, register description, and recommended protocol, see the *CC430 Family User's Guide* (SLAU259).[1]

## 2.2 Using the RF1A.h/.c

The RF1A.h file declares the RF1A functions as well as a structure type, RF_SETTINGS, to hold the register values output by SmartRF® Studio. The functions are described in the following sections.

### 2.2.1 void ResetRadioCore (void);

Resets the radio, putting the radio state machine into SLEEP mode.

### 2.2.2 unsigned char Strobe(unsigned char strobe);

Used to send a command strobe to the radio. Includes the workaround RF1A7.[2] Note that this workaround includes an 810-µs delay in software (using the __delay_cycles() intrinsic) that is dependent on the frequency of operation. As the frequency changes from the default MCLK of 1.05 MHz, the value in the __delay_cycles() intrinsic should be adjusted accordingly to result in a delay of at least 810 µs.

**Parameters**

strobe

The RF_xxx command strobe that changes the state of the radio.

**Returns**

A status byte indicating the current state of the radio and the number of bytes available in the RX or TX FIFO, depending on the MSB of the input strobe parameter.

**Examples**

```
Strobe( RF_NOP );              // Retrieve status byte, including bytes in TX FIFO
Strobe( RF_NOP | RF_RXSTAT );  // Retrieve status byte, including bytes in RX FIFO
Strobe( RF_SIDLE );            // Enter IDLE mode
```

### 2.2.3 void WriteRfSettings(RF_SETTINGS *pRfSettings);

Copies the full set of SmartRF Studio settings from the RF_SETTINGS structure pointed to by pRfSettings to the configuration register set of the CC1101-based core.

**Parameters**

pRfSettings

A pointer to the RF_SETTINGS structure that has been generated using SmartRF Studio – see Section 2.2.10 of this document for how to generate the appropriate structure using SmartRF Studio.

### 2.2.4 void WriteSingleReg(unsigned char addr, unsigned char value);

Write a single CC1101 configuration register.

**Parameters**

addr

The address of the configuration register to be written. The value could also be the address of the TX FIFO.

value

The intended value of the configuration register

**Examples**

```
WriteSingleReg(FSCTRL1, pRfSettings->fsctrl1);  // Write FSCTRL1 config register
```

### 2.2.5 void WriteBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);

Write multiple radio core configuration registers.

**Parameters**

addr

The address of the configuration register to be written. The value could also be the address of the TX FIFO

buffer

A pointer to the array holding the values to be written consecutively into the radio register(s)

count

The total number of values to be written

**Examples**

```
WriteBurstReg(TXFIFO, buffer, length);  // Write the 'buffer' array to the TX FIFO
```

### 2.2.6 unsigned char ReadSingleReg(unsigned char addr);

Read a single radio core configuration register.

**Parameters**

addr

The address of the configuration register to be read. The value could also be the address of the RX FIFO.

**Returns**

value

The value of the register at address

**Examples**

```
ReadSingleReg(RSSI);  // Read the current RSSI value from the RSSI status register
```

### 2.2.7 void ReadBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);

Read multiple radio core configuration registers.

**Parameters**

addr

The address of the configuration register to be read. The value could also be the address of the RX FIFO.

buffer

A pointer to the array that will hold the values that are read consecutively from the radio register(s)

count

The total number of values to be read

**Examples**

```
// Read RxBufferLength bytes from the RX FIFO
ReadBurstReg(RXFIFO, RxBuffer, RxBufferLength);
```

### 2.2.8 void WriteSinglePATable(unsigned char value);

Write a single byte to the PA Table.

**Parameters**

value

The intended value of the PA setting

### 2.2.9 void WriteBurstPATable(unsigned char *buffer, unsigned char count);

Write multiple radio core configuration registers (i.e., for ASK modulation).

**Parameters**

buffer

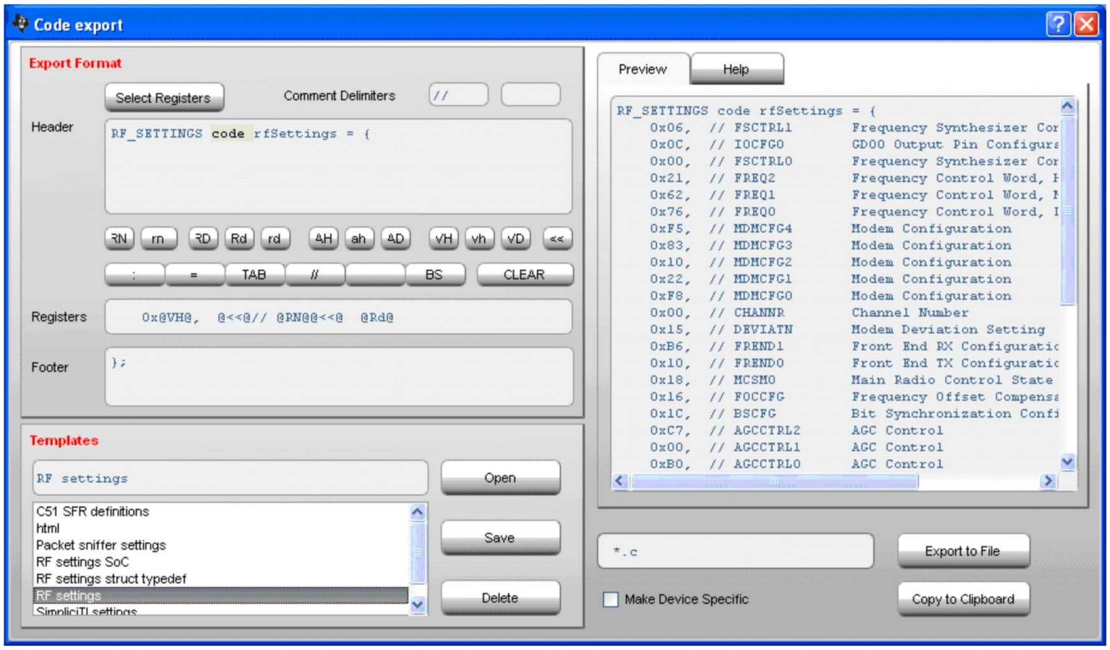
A pointer to the array holding the values to be written consecutively into the PA Table

count

The total number of values to be written

### 2.2.10 Code Export Using SmartRF® Studio

The SmartRF Studio can output a file that defines the RF_SETTINGS structure referred to in the WriteRfSettings() description. In the Code Export window of the SmartRF Studio tool, select RF settings as the Template to apply, and then delete code from the Header entry prior to generating the file. The generated .c file can then be used by the application, by including the RF1A.h file, to easily initialize the radio to the settings defined using SmartRF studio (see Figure 2).



Comment Delimiters: //

Header:
```
RF_SETTINGS code rfSettings = {
```

Registers:
```
    0x@VH@,  @<<@// @RN@@<<@  @Rd@
```

Footer:
```
};
```

Templates: RF settings

- C51 SFR definitions
- html
- Packet sniffer settings
- RF settings SoC
- RF settings struct typedef
- RF settings
- SimpliciTI settings

Preview:
```
RF_SETTINGS code rfSettings = {
    0x06,  // FSCTRL1      Frequency Synthesizer Cor
    0x0C,  // IOCFG0       GDO0 Output Pin Configura
    0x00,  // FSCTRL0      Frequency Synthesizer Cor
    0x21,  // FREQ2        Frequency Control Word, H
    0x62,  // FREQ1        Frequency Control Word, M
    0x76,  // FREQ0        Frequency Control Word, I
    0xF5,  // MDMCFG4      Modem Configuration
    0x83,  // MDMCFG3      Modem Configuration
    0x10,  // MDMCFG2      Modem Configuration
    0x22,  // MDMCFG1      Modem Configuration
    0xF8,  // MDMCFG0      Modem Configuration
    0x00,  // CHANNR       Channel Number
    0x15,  // DEVIATN      Modem Deviation Setting
    0xB6,  // FREND1       Front End RX Configuratic
    0x10,  // FREND0       Front End TX Configuratic
    0x18,  // MCSM0        Main Radio Control State
    0x16,  // FOCCFG       Frequency Offset Compensa
    0x1C,  // BSCFG        Bit Synchronization Confi
    0xC7,  // AGCCTRL2     AGC Control
    0x00,  // AGCCTRL1     AGC Control
    0xB0,  // AGCCTRL0     AGC Control
```

*.c

□ Make Device Specific

**Figure 2. Defining the RF_SETTINGS struct Using SmartRF Studio**

For a detailed description on how to use SmartRF Studio, see the *SmartRF® Studio User Manual* (SWRU070).[3] For an example on how to use the CC430 RF1A.h/.c files, see the CC430 RF Code Examples on the CC430F6137 product page.

## 3 References

1. *CC430 Family User's Guide* (SLAU259)

2. *CC430F613x Device Erratasheet* (SLAZ052)

3. *SmartRF™ Studio User Manual* (SWRU070)

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DLP® Products | www.dlp.com | Communications and Telecom | www.ti.com/communications |
| DSP | dsp.ti.com | Computers and Peripherals | www.ti.com/computers |
| Clocks and Timers | www.ti.com/clocks | Consumer Electronics | www.ti.com/consumer-apps |
| Interface | interface.ti.com | Energy | www.ti.com/energy |
| Logic | logic.ti.com | Industrial | www.ti.com/industrial |
| Power Mgmt | power.ti.com | Medical | www.ti.com/medical |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| RFID | www.ti-rfid.com | Space, Avionics & Defense | www.ti.com/space-avionics-defense |
| RF/IF and ZigBee® Solutions | www.ti.com/lprf | Video and Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless-apps |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265