# Frequently asked questions for RF430FRL15xH devices

*Alexander Kozitsky, Ralph Jacobi* *NFC/RFID Applications*

**ABSTRACT**

This guide contains a compilation of frequently asked questions concerning the RF430FRL15xH series of devices. It serves as a quick reference resource for evaluating or designing with the RF430FRL15xH devices. This guide includes links to recommended evaluation modules, TI Designs, firmware examples, and other collateral that is relevant for the RF430FRL15xH family.

**Contents**

## Trademarks

MSP430, Code Composer Studio, E2E are trademarks of Texas Instruments.
Android is a trademark of Google LLC.
IAR Embedded Workbench is a registered trademark of IAR Systems.

# 1 General Questions

## 1.1 What is the RF430FRL15xH?

The RF430FRL15xH devices are highly integrated, fully programmable, RFID / NFC sensor transponders. This system-on-chip (SoC) family combines an ISO 15693-compliant RFID / NFC interface with a programmable ultra-low-power microcontroller (MCU), nonvolatile FRAM, an analog-to-digital converter (ADC), integrated temperature sensor, and a SPI or $I^2C$ interface.

The dual-interface RF430FRL15xH NFC sensor transponder is optimized for use in fully passive (battery-less) operation for on-the-spot measurements using RF energy harvested from an NFC-enabled reader or smartphone, or semi-active modes for battery-supported data logging in FRAM to achieve extended battery life in a wide range of applications.

Features and benefits of the RF430FRL15xH NFC sensor transponders:

- Supports wireless communication through the ISO/IEC 15693 and ISO/IEC 18000-3 compliant RFID interface.
- Optimized for 1.5-V single-cell-battery-powered designs or battery-less designs that harvest energy from the RF field generated from an NFC reader. Intelligent power management includes a battery switch to ensure long battery life.
- 14-bit sigma-delta ADC with ultra-low input current, low noise, and ultra-low offset enables developers to connect up to three additional external sensors in addition to the integrated temperature sensor.
- SPI or $I^2C$ interface can support digital sensors or connect the device to a host system.
- Application code embedded in ROM manages RF communication and sensor readings to provide the ultimate flexibility in configuring the device. Developers can configure sampling rates, measurement thresholds, and alarms.
- Universal nonvolatile memory (FRAM) allows data storage as well as extension and adjustment of application code.
- Integrates a 16-bit ultra-low-power programmable MSP430™ CPU core that is supported by a robust ecosystem of development tools.
- Fully integrated into the TI Code Composer Studio™ (CCS) and IAR Embedded Workbench® integrated development environments (IDEs).

## 1.2 What are the variations of the RF430FRL15xH?

Table 1 lists the features available by part number.

**Table 1. Device Variants**

| Device | $I^2C$ or SPI (eUSCI_B) | SD14 (ADC) |
|---|---|---|
| RF430FRL152H | Yes | Yes |
| RF430FRL153H | No | Yes |
| RF430FRL154H | Yes | No |

### 1.3 What is the recommended hardware and software to evaluate the RF430FRL15xH?

The following tools allow for full evaluation of the RF430FRL152H:

- RF430FRL152HEVM
  - This EVM allows evaluation of all the capabilities of the RF430FRL15xH.
- DLP-7970ABP BoosterPack plug-in module and MSP-EXP430G2ET LaunchPad development kit
  - The DLP-7970ABP uses TI's TRF7970A NFC transceiver to communicate with the RF430FRL15xH. This bundle is needed to operate the GUI for the RF430FRL152HEVM.
- SLOC346
  - Software binary to allow the TRF7970A-BNDL to interface with the RF430FRL152HEVM Windows GUI.
- MSP-FET
  - Offers firmware download and debug capability to support development for the RF430FRL15xH.
- RF430FRL152HEVM Windows GUI
  - PC application interface to easily test the device features.

Connect the TRF7970A-BNDL (or TRF7970AEVM if using legacy hardware) to the PC, and place the RF430FRL152HEVM above the antenna of the DLP-7970ABP BoosterPack plug-in module so it can communicate using RF. Then use the GUI to send commands to the TRF7970A-BNDL, which transmits commands and receives the responses from the device.

For further information and instructions on how to use the RF430FRL152HEVM and the GUI, see the RF430FRL152HEVM user's guide.

### 1.4 I cannot purchase a TRF7970AEVM. How do I use the RF430FRL152HEVM GUI?

The TRF7970AEVM is no longer sold by Texas Instruments. For all RF430FRL152H topics that refer to the use of the TRF7970AEVM, use the DLP-7970ABP BoosterPack plug-in module and MSP-EXP430G2ET LaunchPad development kit instead.

The software binary can be found in the Debug folder of MSP430G2 LaunchPad TRF7970A BoosterPack Host Control Binary. The file name is TRF7970A_BoosterPack_MSP430G2.out. Use the latest version of Uniflash (www.ti.com/tool/uniflash) to load this file onto an MSP-EXP430G2 LaunchPad development kit.

**NOTE:** To enable the firmware to run correctly on the MSP-EXP430G2 LaunchPad development kit, remove the jumper for P1.6. In the default configuration, P1.6 is connected to an LED, but the firmware requires this pin for SPI communication. When you remove the jumper, the firmware can use P1.6 for SPI communication as required.

For applications that require the Test Tab functionality, the TRF7970AEVM GUI is available for download.

### 1.5 Does the RF430FRL15xH support NDEF messages?

Using the NFC example provided in the RF430FRL152H firmware, you can use user-defined NDEF messages that will be recognized by NFC handsets. Some older NFC mobile handsets do not support NFC Tag Type 5 configured tags but still support ISO/IEC 15693 RFID with raw transceiver commands, like read and write single block.

When NDEF support is used, much of the ROM functionality must be deactivated, because the NDEF message and the virtual registers share the same portion of memory. Only the RF stack is supported with this feature. While the ROM in this mode cannot take SD14 measurements and there is no host controller support, the SD14 and $I^2C$ or SPI modules are still functional and can be controlled by custom user code in FRAM. For the example projects provided (see Section 2.4 ) only the NFC example has NDEF support. The "Default" and "SensorHub" examples only have ISO/IEC 15693 RFID support.

**NOTE:** The term *virtual registers* refers to all the registers that exist in FRAM memory space that are programmable over RF (or a host controller) and control ROM operation.

### 1.6 What RF communication range can be expected with the RF430FRL15xH?

ISO/IEC 15693 is the best choice for NFC if range is a key factor in the application. However, the range depends on the reader used and the size of the antenna connected to the RF430FRL152H. For example, using a TRF7970AEVM and a RF430FRL152HEVM (passively powered) and outputting a constant RF field with the TRF7970AEVM, the range is approximately 9 cm. As mobile handsets generally do not leave the RF field on all the time, there will be observed reductions in read range when interfacing with them. Additionally, if the RF430FRL15xH is used to source power to other devices, the read range is expected to decrease. For further information on RF communication ranges between handsets and RF430FRL152H, see the reading performance section in RF430FRL15xH NFC and ISO/IEC 15693 sensor transponder practical antenna design.

# 2 Hardware and Design Questions

## 2.1 What TI Designs are available for the RF430FRL152H?

These designs are useful to find schematics, layout, and BOM files that can be referenced to build a custom design.

RF430FRL152H NFC Temperature and Light Sensor reference design

Battery-less NFC/RFID Temperature Sensing Patch reference design

## 2.2 How do I design and tune an antenna to 13.56 MHz for my application?

The proper antenna design is critical to the RF communication performance. See RF430FRL15xH NFC ISO 15693 sensor transponder practical antenna design for details.

## 2.3 Why are the various external components required?

Figure 1 and Table 2 show the external components that are needed for the RF430FRL152H to operate.

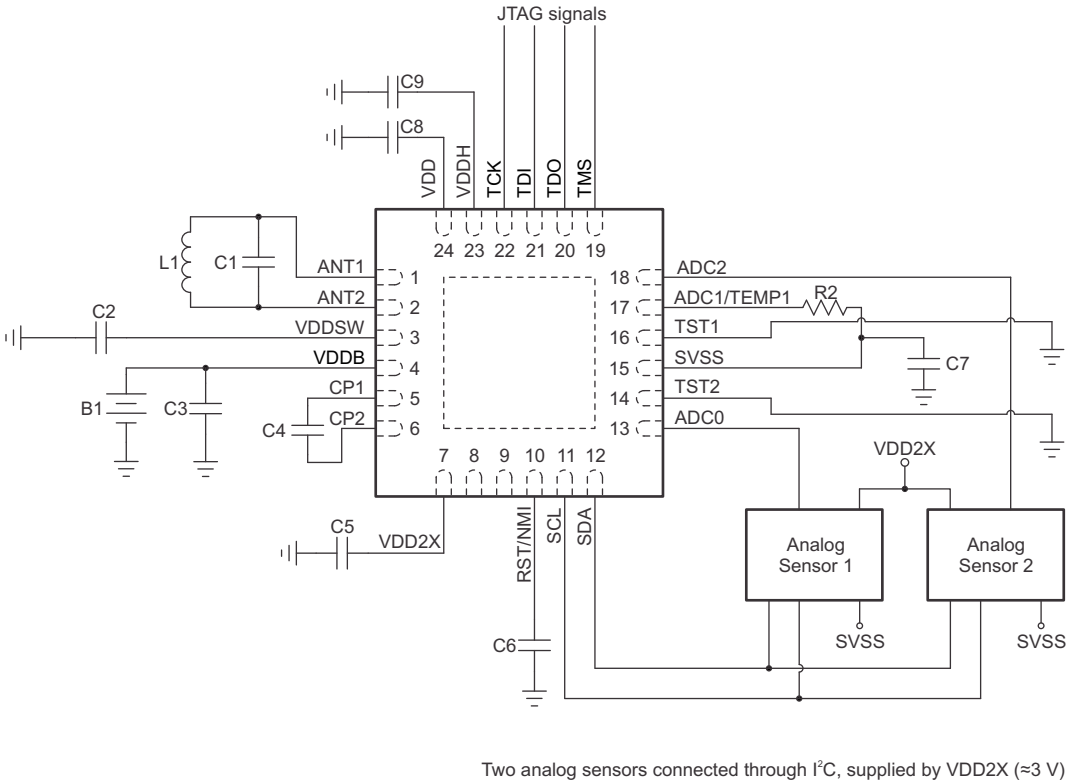


**Figure 1. RF430FRL15xH Application Circuit**

**Table 2. External Capacitors**

| Name | Value | Description | Comment |
|---|---|---|---|
| C4 | 10 nF | Charge pump capacitor | Needed for proper device operation |
| C5 | 100 nF | Decoupling cap at VDD2X | Needed for proper device operation |
| C7 | 1 µF | Bypass capacitor between SVSS and VSS | For proper ADC operation |
| C6 | 10 nF | Decoupling cap at RST | To prevent noise and unintentional resets |
| C2 | 2.2 µF | Decoupling cap at VDDSW | Power supply decoupling |
| C8 | 100 nF | Decoupling cap at VDD | Digital logic decoupling |
| C9 | 100 nF | Decoupling cap at VDDH | Power supply decoupling |
| C3 | 100 nF | Decoupling cap at VDDB (if a battery is being used) | To reduce ESR on a battery |

When using the ROM library, as with the "Default" and "SensorHub" examples, some external resistors may be necessary. See Table 3 for specifics. If using the "NFC" example, this section does not apply.

**Table 3. External Resistors**

| Pin | Value | Description | Comment |
|---|---|---|---|
| 19 (P1.7) | 10 kΩ to 100 kΩ | Pullup: External host controller<br>Pulldown: Digital sensor used | If neither host controller nor digital sensor is used, then use a pullup.<br>Also this pin can be reused for JTAG. |
| 20 (P1.6) | 10 kΩ to 100 kΩ | $I^2C$ address set bit 1 of $I^2C$ address<br>SPI mode: sets the SPI Mode (1-4) | Necessary only if host controller mode is being used. Is a don't care when digital sensor mode is used.<br>Also this pin can be reused for JTAG. |
| 21 (P1.5) | 10 kΩ to 100 kΩ | $I^2C$ mode: set bit 0 of $I^2C$ address<br>SPI mode: sets the SPI Mode (1-4) | Necessary only if host controller mode is being used. Is a don't care when digital sensor mode is used.<br>Also this pin can be reused for JTAG. |
| 8 (P1.3) | Tie to GND or float | Tie low to enable $I^2C$ slave functionality for host controller.<br>High or floating on start-up enables SPI functionality for host controller. | Necessary only if host controller mode is being used. Is a don't care when digital sensor mode is used. |

For more information, see the section on interfacing a host controller in the RF430FRL15xH firmware user's guide.

## 2.4 How much power can be sourced from the RF430FRL15xH?

Three sources are available for RF energy harvesting: VDDSW/VDDR, VDDH, and VDD2X.

Table 4 lists test values that were measured using the TRF7970AEVM with a 5-V power supply.

The current that can be harvested from the RF430FRL15xH depends on the distance from the reader to the antenna.

**Table 4. Energy Sources**

| Source | Pin | Voltage (V) | Current (µA) |
|---|---|---|---|
| VDDSW/VDDR | 3 | 1.5 | 450 (1) |
| VDDH (2) | 23 | 1.86 | 600 (1) |
| VDD2X | 7 | 2.9 | 150 (3) |
| VDD2X | 7 | 2.7 | 90 (1) |

(1) Device is operational and can respond to RF commands.
(2) VDDH is available only when there is an active RF field and not when operating from a battery.
(3) This value is with the RF430FRL152H in low-power mode. Because the MCU uses the VDD2X rail during active mode, less energy is available externally. If too much energy is harvested, the device will brownout.

The voltage doubler is automatically turned on during FRAM access. The doubler can be set to always on by setting RFPMMCTL0 -> EN_V_DOUB.

## 2.5 What battery can be used with the RF430FRL15xH?

Generally a battery that supplies no more than 1.65 V. If using a battery with a higher voltage, an LDO or buck converter is necessary. The minimum device starting voltage is 1.45 V. As the battery is discharged, the device functions down to 1.35 V before it resets. Compatible batteries include silver-oxide types (for example, SR377 and SR626).

## 2.6 How should the switches be set on the RF430FRL152HEVM?

The switches allow configurations such as passive, debugging, using a BoosterPack plug-in module, host controller, or using a battery. For details on how to set them, see the hardware overview section of the RF430FRL152HEVM user's guide.

## 2.7 How can an external sensor be connected on the RF430FRL152HEVM?

The following sensors can be connected to this EVM.

- I²C or SPI sensor: see Section 3.2.2 for an example
- Thermistor: see Table 5 for connections
- Analog sensor: see Table 5 for connections

All three of these signals are routed to the not-populated SV16 header. To use any of these ADC inputs for an external sensor, one of the components in Table 5 must be removed.

**Table 5. Sensor Connections**

| ADC Input | SV16 Pin (Connect Sensor Here) | Remove |
|---|---|---|
| ADC0 | 3 | R16 |
| ADC1/TEMP1 | 1 | R4 |
| ADC2/TEMP2 | 2 | R3 |
| Ground | 4 | – |

Sensors should be grounded to SVSS, especially if virtual ground is chosen (SD14CTL0: VIRTGND = 0). For more information on virtual ground, see the SVSS generator section in the RF430FRL15xH family technical reference manual.

## 2.8 What kinds of analog sensors can be used?

The sensor should meet these characteristics:

- Analog voltage output range: 0 to 0.9 V
- Signal bandwidth: <1 Hz

If the sensor maximum output is less than 0.45 V, using the internal PGA helps improve the resolution of the measurements. For example if the maximum output voltage of the sensor is 0.35 V, using a PGA gain of 2 improves the resolution of the ADC result.

Also, this device includes hardware support for thermistors. The device can support up to two thermistors on TEMP1 and TEMP2.

The ADC (sigma-delta 14-bit) on this device is a slow acquisition one. Up to 1 second is required for a conversion to complete with full accuracy. Shorter conversion times are possible if 14-bit accuracy is not needed.

## 2.9 How can a digital sensor be used?

Digital sensors that can be used with this device must meet these criteria:

- I²C or SPI communication
- Maximum digital output of 1.65 V
- 1.5-V digital input signal level

If a digital sensor meets these conditions, the next step is to determine how it will be used by the RF430FRL15xH. A digital sensor can be used with the ROM support or without it.

The best example of how to develop a digital sensor into the application is to reference the SensorHub example project that leverages the ROM support. The example projects can be found in the RF430FRL152H firmware. This starting point allows maximum use of the ROM firmware and therefore limits the use of the available FRAM memory for application code.

Without ROM support means that the firmware is developed from the "NFC only" example project which only uses ROM support for the NFC/RFID communication. Any other functionality must be custom developed applications that run in the available FRAM code space.

### 2.10 How can the internal temperature sensor be used?

The internal temperature sensor must be calibrated on each device before it can effectively be used.

Two calibrations methods can be used: 1 point or 2 point.

At least a 1-point calibration is needed. Without this calibration, the offset of the internal temperature sensor is not known. After the offset is known, the coefficient (35.7) can be applied to determine the temperature. The coefficient is basically a sloped line on a graph, and the offset places the line somewhere vertically.

A 2-point calibration determines not only the offset but also the slope. This calibration can be done if higher accuracy is required over a greater temperature range.

### 2.11 What are the settings to program the RF430FRL152HEVM with an MSP-FET tool?

Make sure that switch S6 is set to *Supply* mode.

For powering the RF430FRL152H through an RF field or battery, use the *Battery* selection.

See the hardware overview section of the RF430FRL152HEVM user's guide for more information.

### 2.12 Is there a low-voltage programmer suitable for the RF430FRL15xH?

A 14-pin target development board for low-voltage MCUs is available as the MSP-TS430L092. Only the low-voltage active cable is needed. The active cable is compatible with the RF430FRL15xH.

# 3 Software Questions

## 3.1 How is the RF430FRL15xH ROM library used?

- With full ROM library support

  In this case, settings are written using the NFC/RFID ISO/IEC 15693 interface to the virtual registers in FRAM. The ROM library acts on the settings and can execute sampling processes and log the result back to FRAM. The results can then be read out over RF.

  Using this method, minimal FRAM usage is required for the application, and more space is available for logging the samples. The main function in this setup is not available to the user. The user can still add ISO/IEC 15693 custom commands to do certain operations as needed. For example source code for this use case, see Section 3.2 and read about the "default" and "SensorHub" projects.

- With ROM library support for NFC/RFID stack only

  In this case, the user has a main function to develop a custom application with and access is available to all modules of the device. The ROM library support using virtual registers is not functional. ISO/IEC 15693 custom commands are still possible. For example source code for this use case, see Section 3.2 and read about the "NFC" project.

**NOTE:** The term *virtual registers* refers to all the registers that exist in FRAM memory space that are programmable over RF (or a host controller) and control ROM operation.

## 3.2 What example firmware is available for the RF430FRL15xH?

Example firmware is provided for several device variants in the RF430FRL152H firmware. See the following sections for more information.

### 3.2.1 Default Project

This small footprint example uses the built-in ROM library. This example includes the required patches for the ROM code. It also defines a custom command example. With this example, a thermistor can be used for temperature measurements over RF. Other sensors can be used.

This example cannot have a user main function as it is built into the ROM code. Because the ROM uses the SD14, eUSCI, RF module, GPIO, watchdog, and timer modules, these modules cannot be used for custom code. Any required functionality needed is likely available as part of the ROM library functionality. However, if any desired functionality is not available, then using the NFC example project may be the only option, because it does not have these restrictions.

### 3.2.2 SensorHub Project

This example is similar to the default project but it also includes functionality to use the SensorHub BoosterPack plug-in module. This is an example of how to develop support for custom digital sensors using the $I^2C$/SPI interface and with ROM library support. As in the default project, the ROM uses the SD14, RF module, GPIO, watchdog, and timer modules, so these cannot be used for custom code. eUSCI ($I^2C$ or SPI) control is available to the user.

### 3.2.3 NFC Project

This example only uses the ROM NFC stack, and all other ROM functionality is disabled. It enables NFC NDEF support on this device and allows the user to develop their own main function for a custom application.

The SD14, GPIO, watchdog, timer, and eUSCI functionality can be controlled by the user firmware in FRAM.

This example cannot be used with the RF430FRL152HEVM PC GUI.

### 3.3 Is there an Android app available to interface with the RF430FRL15xH?

An example Android™ app is available with source code to read temperature data from the RF430-TMPSNS-EVM. See this post on the TI E2E™ Community.

### 3.4 What is the procedure to change between 4- and 8-byte ISO/IEC 15693 blocks?

This change can be made in the *System* tab of the RF430FRL152HEVM GUI. For more information, see the section on changing firmware system settings in the RF430FRL152HEVM user's guide.

1. Read the current settings by clicking the *Read* button in the *System* tab.
2. Select the option for 4 or 8 bytes.
3. Click the *Write* button.

   The GUI requires the 8 block option to work. If the 4 block option is configured, the GUI prompts to change to the supported option.

### 3.5 How can custom ISO/IEC 15693 commands be sent and received?

While TRF7970A EVM Control GUI provides many of the standard commands in the *15693* tab, more control may be necessary.

Low-level control of ISO/IEC 15693 commands is possible using the TRF7970A EVM Control GUI (to download, see Section 1.4). For test purposes, this is possible using the *Test* tab with this tool.

This example explains how to send a custom command that turns on or off an LED on the RF430FRL152HEVM. To test this feature, the EVM must be loaded with the *Default* project available in the RF430FRL152H firmware. The Default project has support for a custom command ID of AAh. The function where this is defined is called userCustomCommand in the main.c file. For the red *Alarm* LED to work, configure the EVM to be USB powered and set S6 to *Supply*.

To send a custom command of AAh with data of 0x10, this string is used: 18 02 AA 07 10

18: TRF7970AEVM Host command (omit for other readers; not sent out over RF)

02: Flags - high speed mode selection (start of actual RF packet)

AA: The actual custom command

07: TI Manufacturer ID (needed by the RF stack)

01: Turn on the Alarm LED (0x00 to turn off)

Before entering anything into the *Test* tab, it is important to set initial settings. Using the *15693* tab, it is important to set the TRF7970AEVM to the proper settings. This must be done only once.

1. Make sure that the only flag that is selected in the *Tag Flags* options is *High Data Rate*.

2. Click *Set Protocol*.

3. Go to the *Test* tab.

To issue the actual command to activate the red LED, enter the custom command string that was detailed previously into the *String to send* text box as: 1802AA0710.

To turn the LED off, the last byte of the command should be 00. Therefore to turn the LED off, enter this string: 1802AA0700.

For further information on the host control protocol for the TRF79xx, see TRF79xx ISO 15693 host commands.

More information on custom commands is provided in the section on adding custom commands in the RF430FRL15xH firmware user's guide.

### 3.6 How can a simple SD14 ADC conversion be set up and executed, and the result read back?

The following example configuration samples a light sensor on the RF430FRL152HEVM. This example is meant to be run on the TRF7970AEVM PC Control GUI. To run on another ISO/IEC 15693 reader/writer, remove the 18 on the start of the packet. The rest of the packet can be used on that reader. For guidance on how to send a write and read single block command using the TRF7970AVEM Control GUI, see Section 3.5.

1. Write Block 2 First

   1 => 0x00 - Reference-ADC1 Configuration Register (not used, so don't care here)

   2 => 0x00 - Thermistor-ADC2 Sensor Configuration Register (not used, so don't care here)

   3 => 0xYY - ADC0 Sensor Configuration Register (the light sensor)

   8 bits (LSB first):

   0 to 1 => 00 (PGA gain of 1) (other options are 2, 4, and 8)

   2 => 1 (CIC filter) (the other option is Moving Average filter)

   3 to 5 => 101 (1024 CIC decimation ratio with 14-bit accuracy)

   6 => 0 (Use virtual ground to improve accuracy; this setting must be consistent for all sensors)

   7 => 0 (Reserved)

   Final value: 0010 1100 or 0x2C

   4 => 0x00 [Internal Sensor Configuration register (not used, so don't care here)]

   5 => 0x00 [Initial Delay Period Setup register (not used, so don't care here)

   6 => 0x00 [JTAG Enable Password Register (JTAG disabled)]

   7 to 8 => 0x00 0x00 [Initial Delay Period register (not used, so don't care here)]

   Final value: 00 00 2C 00 00 00 00 00

   Final packet to send with the Test tab: 1802210200002C0000000000

2. Write Block 0 Last: (after writing, this will initiate the sampling process)

   1 => 0x01 (Start bit is set, after this is written this starts the sampling process)

   2 => 0xYY (Status byte)

   3 => 0x04 (ADC0 sensor selected)

   4 => 0x00 (Frequency register, this is don't care since only one sample or pass is done)

   5 => 0x01 (Only one pass is needed)

   6 => 0x01 (No averaging selected)

   7 => 0x00 (No options selected)

   8 => 0x00 (No options selected)

   Final value: 01 00 04 00 01 01 00 00

   Final packet to send with the Test tab: 1802210001000400010100 00

3. Result

   Because one sensor was selected and at 1024 decimation ratio, sampling requires approximately 1 second to complete. Reading block 0 and checking the status register to see if the result completed is another way to determine if the sampling process has completed. This example simply reads block 0x09 to see the results:

   Reading block 0x09:

   Enter in Test tab Read Single Block for Block 0x09: 18022009

   Response: [009024FFFFFFFFFFFF] (Light sample result = 2490h, the rest are don't care)

   To repeat the measurement, enter block 0 string again (covering the green light sensor):

   1802210001000400010100 00

   Result (wait 1 second):

   Enter in Test tab Read Single Block for Block 0x09: 18022009

   Response: [009013FFFFFFFFFFFF] (Light sensor result = 1390h, the rest are don't care)

### 3.7 Where is the software for the TIDM-RF430-TEMPSENSE EVM?

There is no software download for the TIDM-RF430-TEMPSENSE EVM. This EVM runs from the ROM code that is part of the RF430FRL152H. The EVM does not have any FRAM programmed at production.

The RF430FRL152H ROM RF Stack plus Sensor Stack enable temperature measurements. The RF stack is used to receive configuration commands for the SD14 ADC through NFC/RFID communication. The configuration commands are written to the device virtual registers starting at the Block 0x00 for RFID commands, which corresponds to Address 0xF868 of FRAM. The virtual registers are used by the Sensor Stack to manage the ADC sampling process and read the raw ADC result from the thermistor.

The conversion from the ADC result to a temperature measurement is not done on the RF430FRL152H device. Instead, the raw ADC result is stored in FRAM and transmitted over RF. The application that receives the ADC result must convert the data to temperature. Section 4.1 describes the calculations for the conversion.

**NOTE:** Details about device memory addressing are described in Section 4.4 of the RF430FRL15xH firmware user's guide and details about available registers and virtual register address mapping are described throughout Section 7 of the firmware user's guide.

## 4 Miscellaneous Questions

### 4.1 How can the SD14 ADC result be converted to temperature?

The ROM library in the RF430FRL15xH does not calculate out the temperature. The device only provides the raw reference resistor and thermistor results as they come from the SD14 ADC. These actual results of the raw values are expected to be calculated on a device with more processing power and program memory space available, or with a look-up table.

The following example shows how the temperature is calculated for the RF430FRL152HEVM by an external reader. The equation in step 3 came from the thermistor data sheet. This equation converts from raw ADC result to temperature.

As part of the calculation of the temperature, the formula to determine the thermistor's resistance is required. This formula is provided in the thermistor resistance calculation section in the RF430FRL15xH firmware user's guide.

Definitions:

- refValue is the reference resistor ADC result.
- thermValue is the thermistor ADC result.
- refResistance is the resistance value of the reference resistor, which for the RF430FRL152HEVM is 100 kΩ.

1. tempConv = (thermValue / refValue) × refResistance
2. Using the resistance from the previous equation, the temperature in Celsius is calculated.
3. Temp in C = (1.0 / ((((Math.Log10(tempConv / 100000.0) / Math.Log(2.718))) / 4330.0) + (1.0 / 298.15))) – 273.15
4. Temperature in °F = (Temperature in °C × 9 / 5 + 32)

Notice that the reference resistor result is also used in the calculation. This is needed to properly measure the current through the thermistor and reference resistor. When the current source is known, using the thermistor ADC result, the temperature can be calculated.

### 4.2 Is there a way to program RF430FRL15xH over-the-air?

The RF430FRL15xH PC GUI supports this feature. The functionality is available in the *RF Programming* tab. See the section on over-the-air programming in the RF430FRL152HEVM user's guide for more information on this feature.

### 4.3 How can the battery-less NFC/RFID temperature sensing patch be used?

This patch can be used with:

- The MSP-EXP430FR4133 and DLP-7970ABP. See RF430FRL152H NFC Sensor Tag Application Example for more detailed information.
- An Android handset that supports NFC. See Section 3.3 for more information.

### 4.4 Is the RF430FRL15xH available as a bare die?

The device is not commercially available in this form, and no samples are available for the device as a bare die.

## 5 References

1. RF430FRL15xH NFC ISO 15693 sensor transponder
2. RF430FRL15xH family technical reference manual
3. RF430FRL15xH firmware user's guide
4. RF430FRL15xH NFC ISO 15693 sensor transponder practical antenna design
5. RF430FRL152HEVM user's guide

## Revision History

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

| Changes from July 6, 2018 to March 13, 2019 | Page |
|---|---|
| • Updated links to purchase hardware throughout document | 2 |

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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