WHITE PAPER

Simplifying the Bluetooth® pairing experience using a dynamic NFC transponder

Eddie LaCost
Applications Engineer
Texas Instruments

Introduction

Bluetooth® has become a standard feature in all smart phones and tablets allowing communication with Bluetooth enabled headphones, speakers, cameras, watches and other devices. With so many devices in operation, pairing your phone to the correct device can be troublesome. Pairing the device with your smart phone or tablet consists of setting the Bluetooth enabled device into discoverable mode, generally by pressing and holding a button for an extended period, then inquiring for devices within range on your phone or tablet and selecting your device from a potentially long list.

The Bluetooth Special Interest Group (SIG) defines "secure simple pairing" with four different association models. One of these models uses out of band (OOB) channels such as near field communication (NFC). NFC can be used to simplify and enhance the user experience by removing the need for the inquiry process and also allowing for a specific application to be opened after reading the NFC message. The NFC Forum™ and Bluetooth SIG have collaborated on an application document titled "Bluetooth Secure Simple Pairing using NFC" which provides examples and solidifies the message formatting for out of band pairing using NFC. This paper will discuss how the information in the NFC Forum/ Bluetooth SIG document can be utilized with Texas Instruments (TI) RF430CL330H dynamic NFC transponder to allow for a seamless pairing process.

RF430CL330H dynamic transponder

The RF430CL330H is an NFC type 4B transponder which also implements an I^2C and SPI interface. This allows a connected host microcontroller to update the 3k bytes of SRAM memory available for the NDEF message. All NFC protocol handling is implemented in the ROM code of the device, which means that the developer must only implement configuration of the device and the NDEF message data structure, which speeds their time to market. An interrupt pin is also available to indicate to the host controller when the NDEF memory has been written to or read from. As you can see in the schematic (Figure 1), there are minimal external components required for the dynamic tag functionality. For a *Bluetooth* enabled product utilizing NFC for simplifying the connection and pairing process, the RF430CL330H offers significant benefits from a standard passive tag, which does not implement any wired interface:

- Not required to program tag over the air in manufacturing environment
- I^2C/SPI communication interface offers more interaction with the host controller (wake up host, turn on *Bluetooth* radio, etc.)
- IC and antenna coil can be added directly to existing PCB for lowest cost implementation

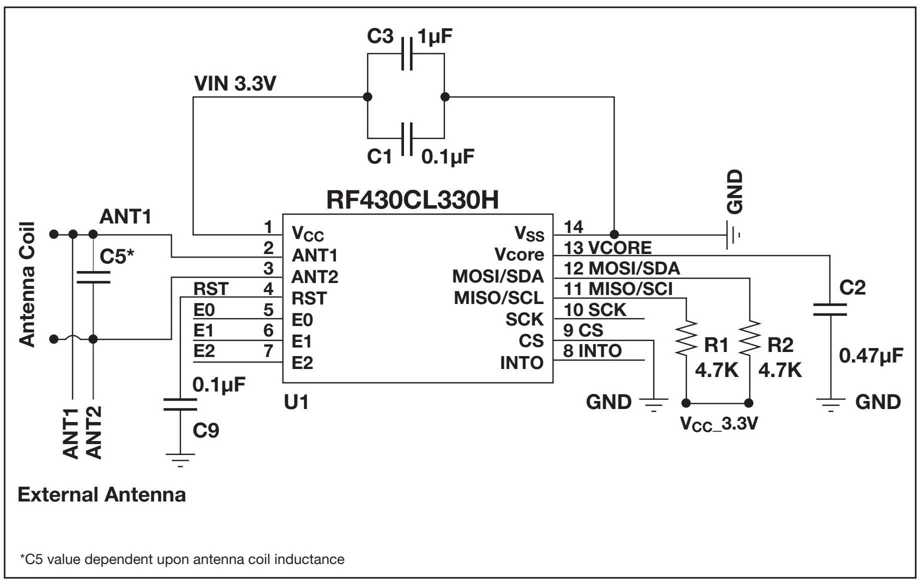

Figure 1: RF430CL330H schematic (I^2C Mode)

I^{2}C communication interface

I^{2}C mode is selected when the CS pin is tied to ground at power up. The 7-bit I^{2}C address contains the four upper bits hard coded to 0x0101, and the lower 3 bits programmed by the state of the E0, E1 and E2 pins.

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	1	E2	E1	E0
MSB						LSB

Figure 2: I^{2}C device address

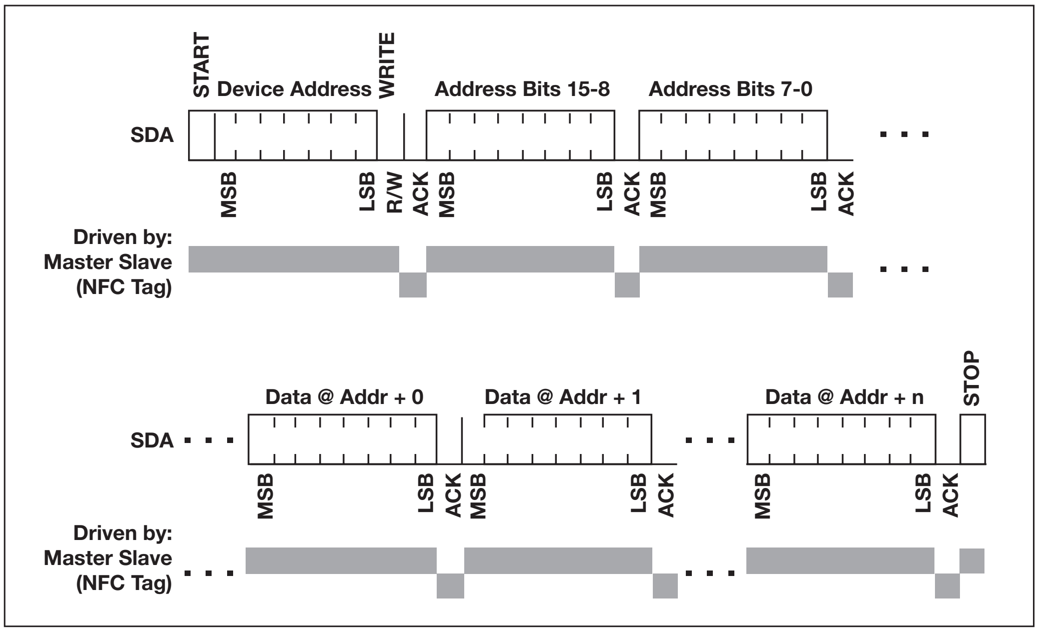

Figure 3: I^{2}C write access

For a data write access, the device is addressed using the specified I^{2}C device address with the R/W = 0 followed by the upper 8 bits of the first address to be written and the lower 8 bits of that address. Immediately after that (without a repeated start), the data to be written starting at the specified address is received. With each data byte received the address is automatically incremented by 1. The write access is terminated by the STOP condition on the I^{2}C bus.

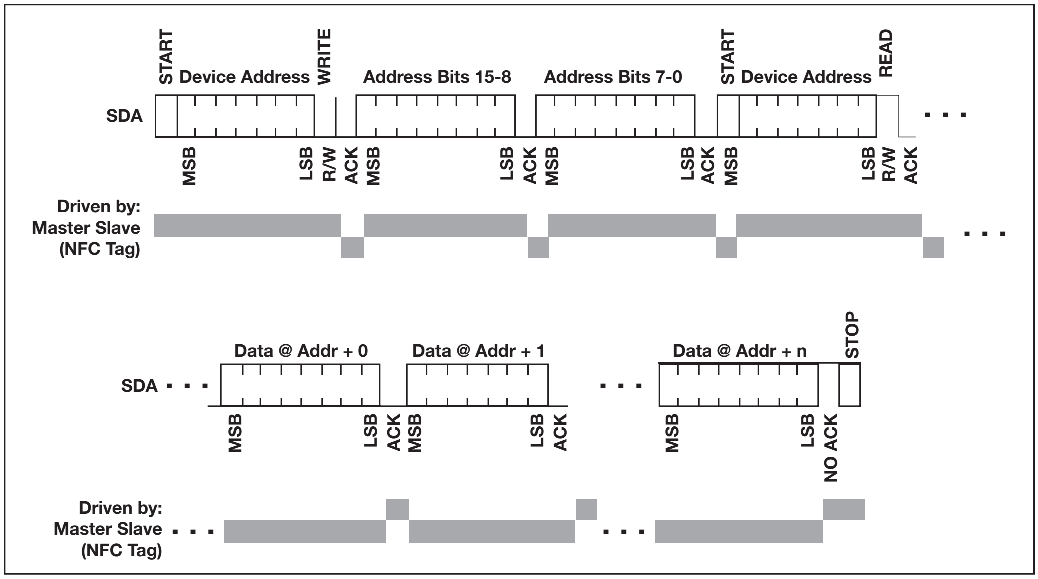

Figure 4: I^2C read access

For a data read access, the device is addressed using the specified device address with the R/W=0 followed by the upper 8 bits of the first address to be written and then the lower 8 bits of that address. Afterwards a repeated start condition is expected with the I^2C device address and R/W=1. The device will then transmit data starting at the specified address until a non acknowledgment and a STOP condition is received.

Typical usage scenario

A typical usage scenario is as follows:

1. Write capability container and message(s) into the NDEF memory (starting from address 0) via the serial interface.
2. Enable interrupts for End of Read and End of Write.
3. Configure the interrupt pin INTO as needed and enable the RF interface.
4. Wait for interrupt signaled via INTO pin.
5. When interrupt is triggered, disable RF interface.
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” loop.

The capability container and NDEF message should be formatted as shown in the code snippet below for an example *Bluetooth* OOB connection handover.

```
unsigned char NDEF_Application_Data[] =
{
   //NDEF Tag Application Name
   0xD2, 0x76, 0x00, 0x00, 0x85, 0x01, 0x01,

   //Capability Container ID
   0xE1, 0x03,

   //Capability Container
   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
   0x0C, 0x02, //Max NDEF size (3072 bytes)
   0x00, //NDEF file read access condition, read access without any security
   0x00, //NDEF file write access condition; write access without 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;
   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, //Record Type Name: application/vnd.blue
   tooth.ep.oob
   0x21, 0x00, //OOB optional data length: 33 octets
   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 octets
   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
```

With the I^{2}C/SPI interface to a host controller, TI's RF430CL330H dynamic NFC transponder allows for a simplified method of programming an NFC transponder for out-of-band pairing for *Bluetooth*. The addition of an interrupt pin provides the host controller with additional information which can be used to trigger different actions such as turning on the *Bluetooth* radio or allowing a specific process to start. While this paper discusses just one of the applications for which the RF430CL330H transponder is well suited, this is a versatile device that offers opportunities for other NFC applications, typically not possible with standard, passive NFC transponders.
For additional details on TI's NFC products, please visit **www.ti.com/nfc**.

Resources

***Bluetooth* Secure Simple Pairing Using NFC:**
http://www.nfc-forum.org/resources/AppDocs/NFCForum_AD_BTSSP_1_0.pdf

RF430CL330H Datasheet: www.ti.com/lit/slas916

E010208

SLOY004