Application Report
SLOA227B–October 2015–Revised March 2019

NFC/HF RFID reader/writer using the TRF7970A

Erick Macias, Ralph Jacobi, Josh Wyatt *NFC/RFID Applications*

ABSTRACT

The Near Field Communication (NFC) market is emerging into multiple fields including medical, consumer, retail, industrial, automotive, and smart grid. Reader/Writer is one of the three operational modes supported by the TRF7970A. When using reader/writer mode the user can configure the TRF7970A to read Type 2, Type 3, Type 4A, Type 4B, and Type 5 tag platforms, also called transponders. The tags can store NFC Data Exchange Format (NDEF) messages or proprietary defined data.

This application report describes the fundamental concepts of reader/writer mode and how to properly configure the TRF7907A transceiver for each supported technology. Furthermore, it describes how to successfully activate, select, and read or write supported tags.

This application report and example firmware focuses on examples where the user presents only one tag at a time. However, it is possible from both a hardware and firmware standpoint to read multiple tags at the same time.

For applications that use only reader/writer mode, the TRF7964A can be used as a drop-in alternative to the TRF7970A.

Sample code described in this document can downloaded from http://www.ti.com/lit/zip/sloa227.

Contents

1 Terms, Definitions, and Symbols 4
2 Introduction 7
3 Initial RF Collision 9
4 TRF7970A Register Settings 12
5 Reader/Writer Mode 13
5.1 Technology Activation Using the TRF7970A 15
5.2 Tag Memory Format With NDEF Examples 20
6 Hardware Description 27
6.1 LaunchPad™ Development Kit and BoosterPack™ Plug-in Module Setup 27
6.2 Bundle Available for Purchase 28
7 Reader/Writer Firmware Example 29
7.1 Reader/Writer APIs 30
7.2 Implementing a Reader/Writer Sample Application 30
8 Quick Start Guide 34
9 Operational Overview 35
10 Reader/Writer Interoperability Results 36
11 Conclusion 37
12 References 37

List of Figures

1 NFC Transceiver Polling Sequence Example 7
2 Reader/Writer Layers Including the PHY (TRF7970A) 8
3 RSSI Level Measurement Orientations 9
4 Long Side RSSI Level Measurement 10
5 Short Side RSSI Level Measurement 10
6 Top Side RSSI Level Measurement 10
7 TRF7970A Reader/Writer Flow Diagram 14
8 ISO14443A Frame Format 15
9 Reader/Writer Anticollision for ISO14443A 16
10 Tag Activation Process for Type 4A Tags 17
11 ISO14443B Frame Format 18
12 ISO18092 Frame Format 18
13 ISO15693 Frame Format 19
14 Flow Chart for Determining Type 5 Tag Memory Size 20
15 Text RTD Example for Type 2 Tags With a Static Memory Structure 21
16 Text RTD Example for Type 2 Tags With a Dynamic Memory Structure 22
17 Text RTD Example for Type 3 Tags 22
18 Type 3 Tag Attribute Information Block Format 23
19 Type 4 Tag NDEF Structure 24
20 Type 4 NDEF File Example for Text RTD 24
21 Example of Type 4 Capability Container Contents 24
22 Text RTD Example for Type 5 Tags 25
23 MSP430F5529 LaunchPad Development Kit and DLP-7970ABP BoosterPack Plug-in Module 27
24 MSP432P401R LaunchPad Development Kit and DLP-7970ABP BoosterPack Plug-in Module 28
25 Reader/Writer NFC Stack Architecture 29
26 TI NFC Tool GUI 34
27 Reader/Writer Tag Polling Mechanism 35
28 Reader/Writer Demo System Block Diagram for MSP430F5529 35
29 Reader/Writer Demo System Block Diagram for MSP432P401 36

List of Tables

1 NFC/RFID Tags Supported by Reader/Writer Mode 8
2 TRF7970A Default Register Values After SOFT_INIT and IDLE Direct Commands 12
3 DLP-7970ABP BoosterPack Module and MSP-EXP430F5529LP LaunchPad Kit Hardware Connections 30
4 TRF7970ATB and MSP-EXP430F5529 Experimenter Board Hardware Connections 30
5 DLP-7970ABP BoosterPack Module and MSP-EXP432P401R LaunchPad Kit Hardware Connections 31
6 Legend for NFC/RFID Tag Support 36
7 NFC/RFID Tags Supported by Reader/Writer Mode 36

Trademarks

1 Terms, Definitions, and Symbols

ALL_REQ
NFC-A Polling command, equivalent to ISO14443-3 short frame command WupA (0x52)

ALLB_REQ
NFC-B Polling command, equivalent to ISO14443-3 command WupB

APDU
Application **P**rotocol **D**ata **U**nit, used in command-response pairs to exchange I (information), R (receive ready) or S (supervisory) blocks.

ATTRIB
PICC Selection Command, Type B

ATQA
Answer **T**o Re**Q**uest **A,** ISO14443-3 term, equivalent to NFC term SENS_RES

ATQB
Answer **T**o Re**Q**uest **B,** ISO14443-3 term, equivalent to NFC term SENSB_RES

CC
Capability **C**ontainer, contains management data and is stored inside a read-only EF. This EF is located inside the NDEF tag application. Default (or basic) file ID for this file (for NFC) is 0xE103. See NFC Type 4 tag Operation Specification and ISO/IEC 7816-4 for more information.

CE
Card **E**mulation, one of the three modes offered by NFC devices. In this optional mode of NFC operation, an NFC Forum device is considered to be a card emulation platform only when it is emulating a Type 3, Type 4A or Type 4B tag platform. Emulation of any other cards or tags is outside the scope of the NFC Forum Brand Promise.

DEP
Data **E**xchange **P**rotocol, an abstracted operational layer that is using either ISO or NFC protocols to exchange data. Thus, two terms can be created with this acronym: ISO-DEP and NFC-DEP. In the context of this document, after activation and selection of the emulated card and before deactivation, ISO-DEP is used exclusively to exchange data between reader/writer (PCD) and tag platform (PICC).

EF
Elementary **F**ile, a set of data objects, records or units sharing the same file identifier and the same security attributes

File Identifier
Two byte data element used to address a file

fc
Carrier frequency, in the context of NFC/HF RFID, is 13.56 MHz ±7 kHz. This is the fundamental transmit frequency of the reader/writer (also called PCD)

Initiator
Generator of the RF field and source of the beginning of the NFCIP-1 communication

MIME
Multipurpose **I**nternet **M**ail **E**xtensions

Modulation Index, *m*
$\frac{\text{peak} - \text{minimum}}{\text{peak} + \text{minimum}}$ or $\frac{1 - b}{1 + b}$, where ***b*** is the ratio between the modulated amplitude and the initial signal amplitude. The index ***m*** is defined per protocol type for both downlink and uplink and is generally expressed as a percentage (for example, Type A uses ***m*** = 100%, and Type B uses ***m*** = 8 to 14%).

NDEF
NFC **D**ata **E**xchange **F**ormat

NFC
Near **F**ield **C**ommunication

PCB
Protocol **C**ontrol **B**yte, used for conveying information required to control data transmission of blocks during the exchange of command-response APDU pairs. Bit coding of this byte and use rules are found in ISO/IEC FDIS 14443-4.

PCD
Proximity **C**oupling **D**evice (also commonly referred to as reader/writer)

PICC
Proximity **I**ntegrated **C**ircuit **C**ard (also commonly referred to as tag, tag platform or transponder)

PUPI
Pseudo **U**nique **P**ICC **I**dentifier (randomly generated or static number returned by PICC as part of the response to REQB, WupB, ALLB_REQ or SENSB_REQ)

RTD
Record **T**ype **D**efinition

SAK
Select **A**c**K**nowledge (from ISO14443-3), in NFC terms this is also called SEL_RES

SDD_RES
Equivalent to ISO14443-3 response to SDD_REQ, and is complete NFCID1 CLn + BCC (if cascade level 1 (single size UID) or indicates NFCID1 is incomplete in the response and further cascade levels must be completed to obtain complete NFCID1+BCC.

SDD_REQ
Equivalent to ISO14443-3 Type A anticollision sequence. Comprised of SEL_CMD, SEL_PAR and n data bits coded based on cascade level specified by SEL_CMD and calculated by value of SEL_PAR

SEL_RES
Equivalent to ISO14443-3 SAK response

SENS_REQ
NFC-A Polling command, equivalent to ISO14443-3 short frame command REQ_A (0x26)

SENS_RES
NFC-A Polling Response, equivalent to ISO14443-3 ATQA

SENSB_REQ
NFC-B Polling command, equivalent to ISO14443-3 command REQ_B

SENSB_RES
NFC-B Polling command response, equivalent to ISO14443-3 ATQB

SENSF_REQ
NFC-F Polling command

SENSF_RES
NFC-F Polling response

Single Device Detection (SDD)
Algorithm used by the reader/writer to detect one out of several targets in its RF field (Type A anticollision, from ISO/IEC 14443-3)

T2T
Type **2** **T**ag

T3T
Type **3** **T**ag

T4T
Type **4** **T**ag

T5T
Type **5** **T**ag

Tag
See Tag Platform

Tag Platform
Responds to reader/writer (PCD) commands by using load modulation scheme (passive operation)

Throughout this document, the terms *tag*, *tag platform*, and *transponder* all have the same meaning.

TLV
Type **L**ength **V**alue

Transponder
See Tag Platform

UID
Unique **ID**entifier

VICC
Vicinity **I**ntegrated **C**ircuit **C**ard

2 Introduction

The TRF7970A device supports three modes: reader/writer, card emulation, and peer-to-peer. This document describes how to use the TRF7970A in reader/writer mode. This mode allows an NFC enabled system to activate and read existing RFID and NFC tags. In the case of the TRF7970A, it is possible to read Type 2 (T2T), Type 3 (T3T), Type 4A or B (T4T), and ISO15693 (T5T) tag platforms. NFC transceivers supporting reader/writer mode typically poll to check if a tag or NFC peer is present every 500 ms. This time is known as the period time, which includes the time the transceiver is polling (searching for a tag) plus the time the transceiver is listening to be activated (in the case it is emulating a tag or is in peer-to-peer target mode). In Figure 1, for example, the transceiver is first polling for Active A and F technology (active peer-to-peer). Next, it polls for passive A technology (ISO14443A-3), passive B (ISO14443B-3), passive F (ISO18092), and then passive V (ISO15693).

For more information on active peer-to-peer communication and how passive A and F technology can be used to activate peer-to-peer communication, see *NFC active and passive peer-to-peer communication using the TRF7970A*.

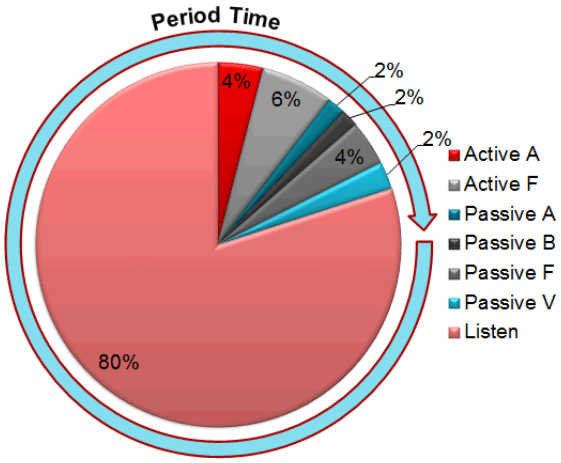

Figure 1. NFC Transceiver Polling Sequence Example

The reader always generates the RF field and the tag load modulates the RF field to send their response. Figure 2 shows the NFC/RFID layers used to read and write tags. Both T2T and T4TA platforms use technology ISO14443-3 (Type A) at an initial bit rate of 106 kbps for tag activation. T4TB platforms use technology ISO14443-3 (Type B) at an initial bit rate of 106 kbps for tag activation. After the activation process is completed, both T4TA and T4TB platforms use ISO7816-4 and ISODEP layers to read or write the tag content. After the T2T platform is activated it uses T2T commands to read or write the tag content. T3T platforms use technology ISO18092 at an initial bit rate of 212 kbps or 424 kbps for tag activation. When the T3T is activated it uses T3T commands to read or write the tag content. T5T platforms use technology ISO15693 for activation and to read the tag content.

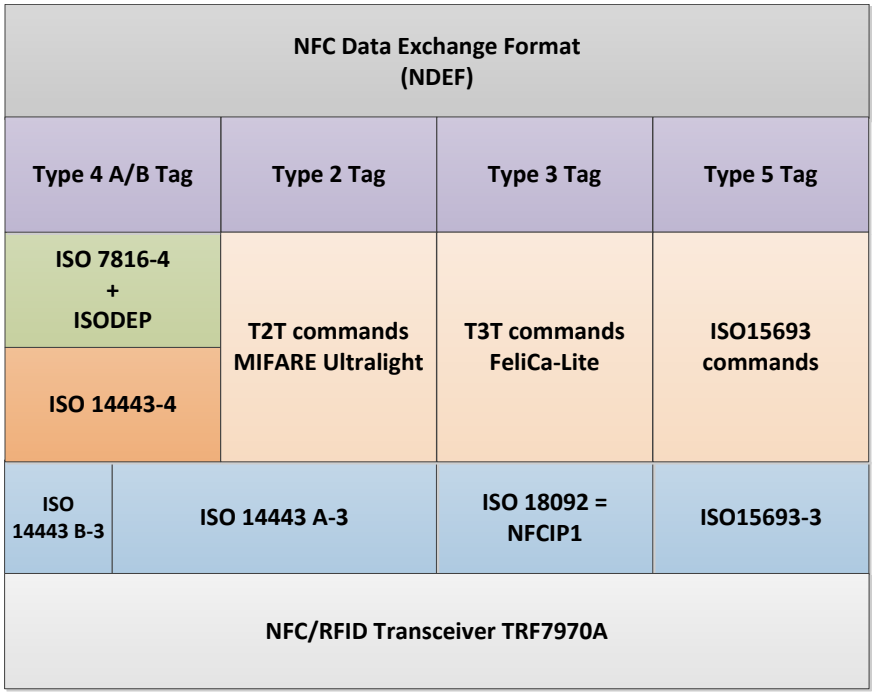

Figure 2. Reader/Writer Layers Including the PHY (TRF7970A)

A 16-bit and a 32-bit microcontroller were used to interface with the TRF7970A to demonstrate a reference example of the reader/writer mode. The firmware supports flexible functions that allow the user to enable or disable reading T2T, T3T, T4TA/B, and T5T platforms. Additionally, the firmware provides support for NFC-enabled smart phones that can emulate tag platforms. The firmware can activate and select NFC-enabled smart phones, allowing users to read card data by creating their own custom applications using existing APIs.

Table 1 lists the NFC/RFID technologies that the provided example firmware supports.

NOTE: Type 1 tags are not ISO compliant and require the use of Direct Mode 0 (DM0) to receive raw RF packets. Due to this, the firmware example does not support these tags.

Table 1. NFC/RFID Tags Supported by Reader/Writer Mode

Tag Technology	Support
Type 1	✗
Type 2	✓
Type 3	✓
Type 4A	✓
Type 4B	✓
Type 5	✓

3 Initial RF Collision

NFC enabled systems can support both reader/writer mode and listen mode. To ensure that two NFC reader devices do not send commands at the same time, an initial RF collision detection is required. The transceiver must check the external received signal strength indicator (RSSI) value that measures the strength of the demodulated subcarrier signal before enabling its own RF field. If the RSSI value is greater than 0x00, it will not enable its RF field.

The relation between the 3-bit code and the external RF field strength (A/m) sensed by the antenna must be determined by calculation or by experiments for each antenna design. The antenna Q-factor, coupling factor between the two antennas and connection to the RF input influence the result. Figure 4 through Figure 6 provide the correlation of the free space distance between two unmodified TRF7970ATB modules and the 3-bit external RSSI value in three directions (see Figure 3 for more details on each orientation). One TRF7970A has its RF field at full power (+5 V), and the second TRF7970A is used to take RSSI measurements across different distances.

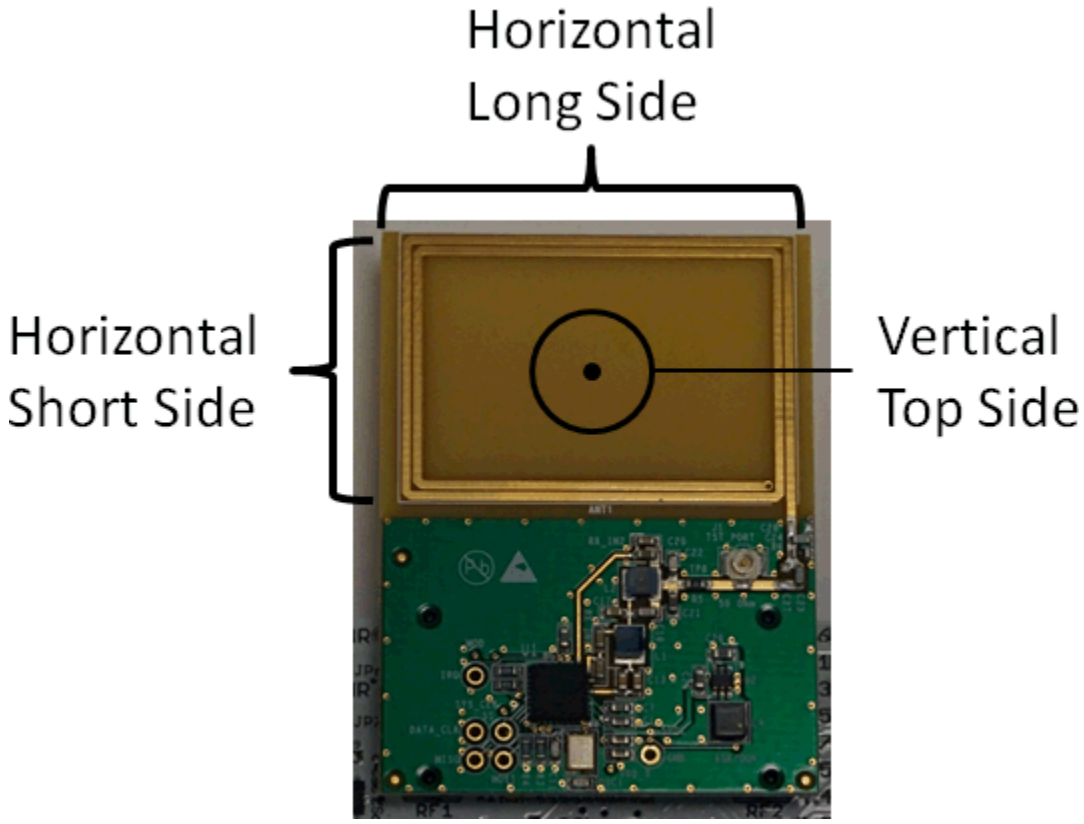

Figure 3. RSSI Level Measurement Orientations

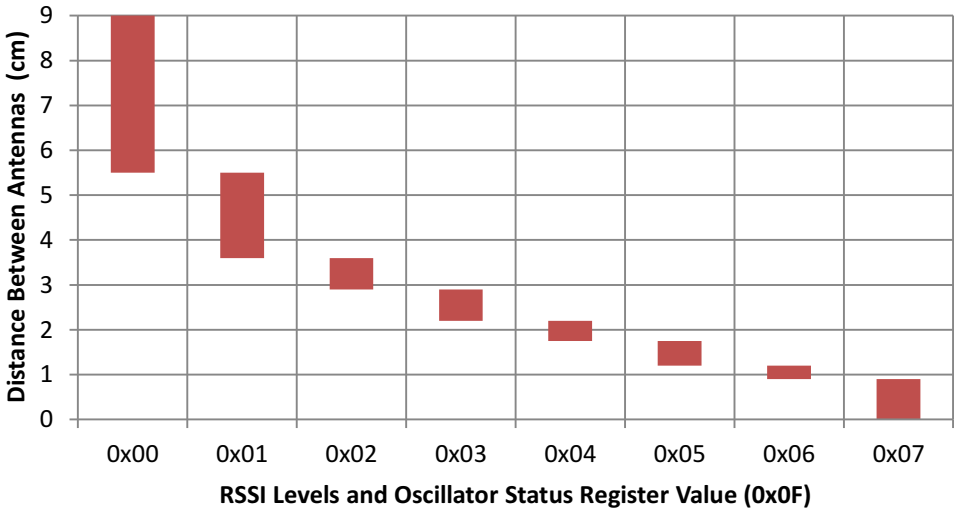

Figure 4. Long Side RSSI Level Measurement

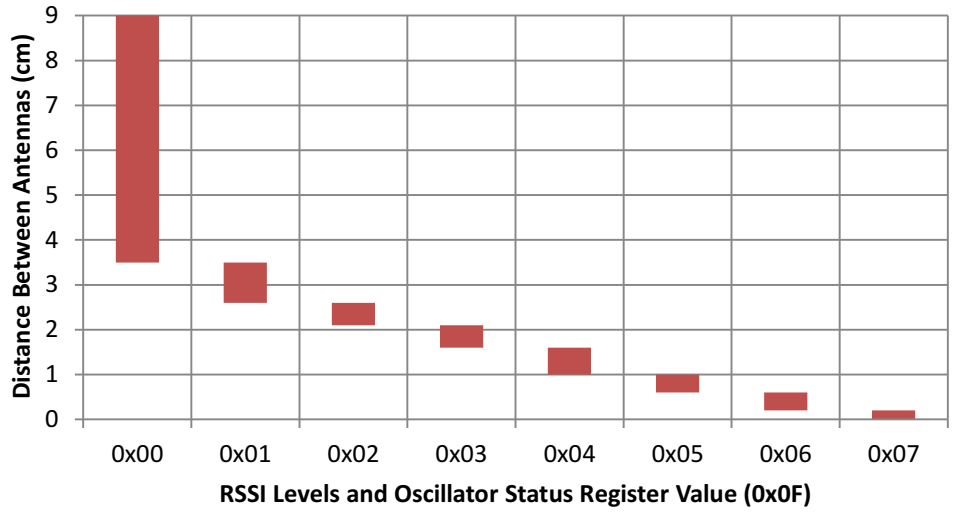

Figure 5. Short Side RSSI Level Measurement

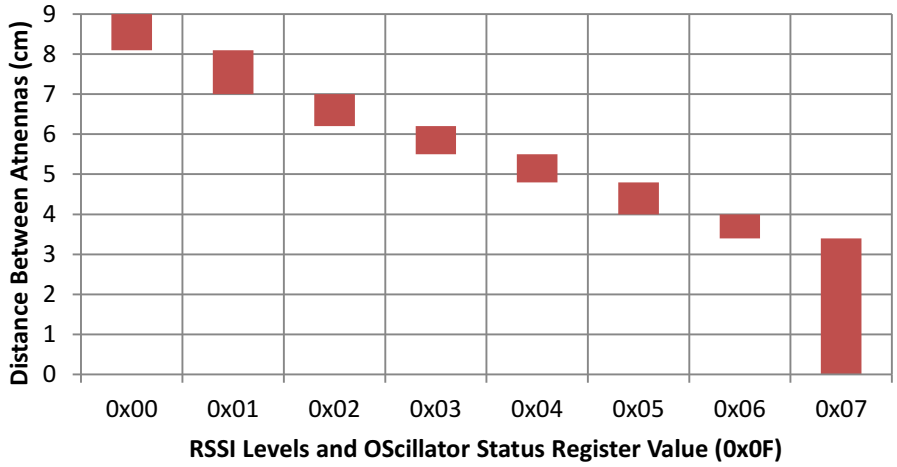

Figure 6. Top Side RSSI Level Measurement

The initial RF collision can be accomplished by performing the following steps:

1. Write a 0x02 or 0x03 (3-V or 5-V operation) to the Chip Status Control register (0x00), disabling the transmitter and enabling the receiver.
2. Send a Test External RF direct command (0x19).
3. Delay 50 µs to allow the transceiver to measure the field strength and latch the value into the RSSI register.
4. Read the RSSI Levels and Oscillator Status register (0x0F).
5. If the active channel RSSI value (bits 2-0) is greater than 0, remain in target mode for a predetermined *n* milliseconds.
6. If the active channel RSSI value (bits 2-0) is equal to 0, go into initiator or target mode for active or passive communication.

4 TRF7970A Register Settings

After powering up the TRF7970A, sending SOFT_INIT (0x03) and IDLE (0x00) direct commands enables the passive target mode at 106 kbps. Table 2 lists the default value of registers 0x00 to 0x16 and 0x18h to 0x1C after the commands are issued. Furthermore, it shows the registers that must be modified from their default values for reader/writer mode.

The ISO Control (0x01) register is modified whenever the reader/writer technology or bit rate changes. The Chip Status Control (0x00) register is modified after initialization, and whenever the RF field is enabled or disabled. The Special Function (0x10) register requires modification for writing to Type 2 tags. The Modulator and SYS_CLK Control (0x09), RX Special Settings (0x0A), and Regulator and I/O Control (0x0B) registers need to be modified only once, after initialization. The NFC Target Detection Level register must be modified after initialization based on the *TRF7970A silicon errata*.

Table 2. TRF7970A Default Register Values After SOFT_INIT and IDLE Direct Commands

Address	Register	Value	Requires Modification
0x00	Chip Status Control	0x01	Yes
0x01	ISO Control	0x21	Yes
0x02	ISO14443B TX options	0x00	No
0x03	ISO14443A high bit rate options	0x00	No
0x04	TX timer setting, H-byte	0xC1	No
0x05	TX timer setting, L-byte	0xC1	No
0x06	TX pulse-length control	0x00	No
0x07	RX no response wait	0x0E	No
0x08	RX wait time	0x07	No
0x09	Modulator and SYS_CLK control	0x91	Yes
0x0A	RX Special Setting	0x10	Yes
0x0B	Regulator and I/O control	0x87	Yes
0x0C	IRQ status	0x00	No
0x0D	Collision position and interrupt mask	0x3E	No
0x0E	Collision position	0x00	No
0x0F	RSSI levels and oscillator status	0x40	No
0x10	Special Function	0x00	Yes
0x11	Special Function	0x00	No
0x12	RAM	0x00	No
0x13	RAM	0x00	No
0x14	Adjustable FIFO IRQ Levels	0x00	Optional
0x15	Reserved	0x00	No
0x16	NFC Low Field Detection Level	0x00	No
0x18	NFC Target Detection Level	0x00	Yes
0x19	NFC Target Protocol	0x00	No
0x1A	Test	0x00	No
0x1B	Test	0x00	No
0x1C	FIFO status	0x00	No

5 Reader/Writer Mode

The TRF7970A supports reader/writer mode for NFC-A (Type 2/4A), NFC-B (Type 4B), NFC-F (Type 3), and NFC-V (Type 5) tag platforms. Only one technology can be activated at a time, requiring the reader/writer mode to poll through each technology individually to detect NFC tags. The way this is handled in the example firmware is illustrated by the flow diagram presented in Figure 7.

After the initial RF collision is completed and the RF field is turned on, there is a guard time before polling for the first technology. The guard time is a specified period of time where the RF field is on, but is unmodulated. The guard time allows for a tag platform to have enough time to be ready to reply to commands that are issued. The guard time requirements vary with each technology, and therefore the ISO specifications for an individual technology should be referenced to find out more information about the guard time for that technology.

For the TRF7970A, there must be a 0x20 or 0x21 written to the Chip Status Control (0x00) register, and the appropriate technology setup for the ISO Control (0x01) register. The example firmware provided implements longer guard times than the minimum required by the specifications for NFC-A and NFC-B. This is done to improve the performance of systems that rely on an RF field to power up additional devices.

The flow diagram for example firmware in Figure 7 does not explicitly show the guard times, but they occur before sending the initial polling command for each technology. For example, the guard time before sending an ALL_REQ or SENS_REQ is set to be 5 ms in the firmware, so if the firmware confirms that NFC-A is enabled, the firmware then turns on the RF field and waits for 5 ms before issuing the ALL_REQ or SENS_REQ command.

The firmware has been structured based on the NFC Forum Activity 1.0 specification, and does not include any anticollision processes for detecting and selecting multiple tags of the same technology. This is equivalent to having the device connection limit set to zero.

It is possible to implement such processes with the TRF7970A by modifying the provided example firmware to support anticollision processes for the different technologies. The firmware has been designed to promote flexibility for modifications as well as custom applications, and the existing APIs can be leveraged to add anticollision functionality for each technology.

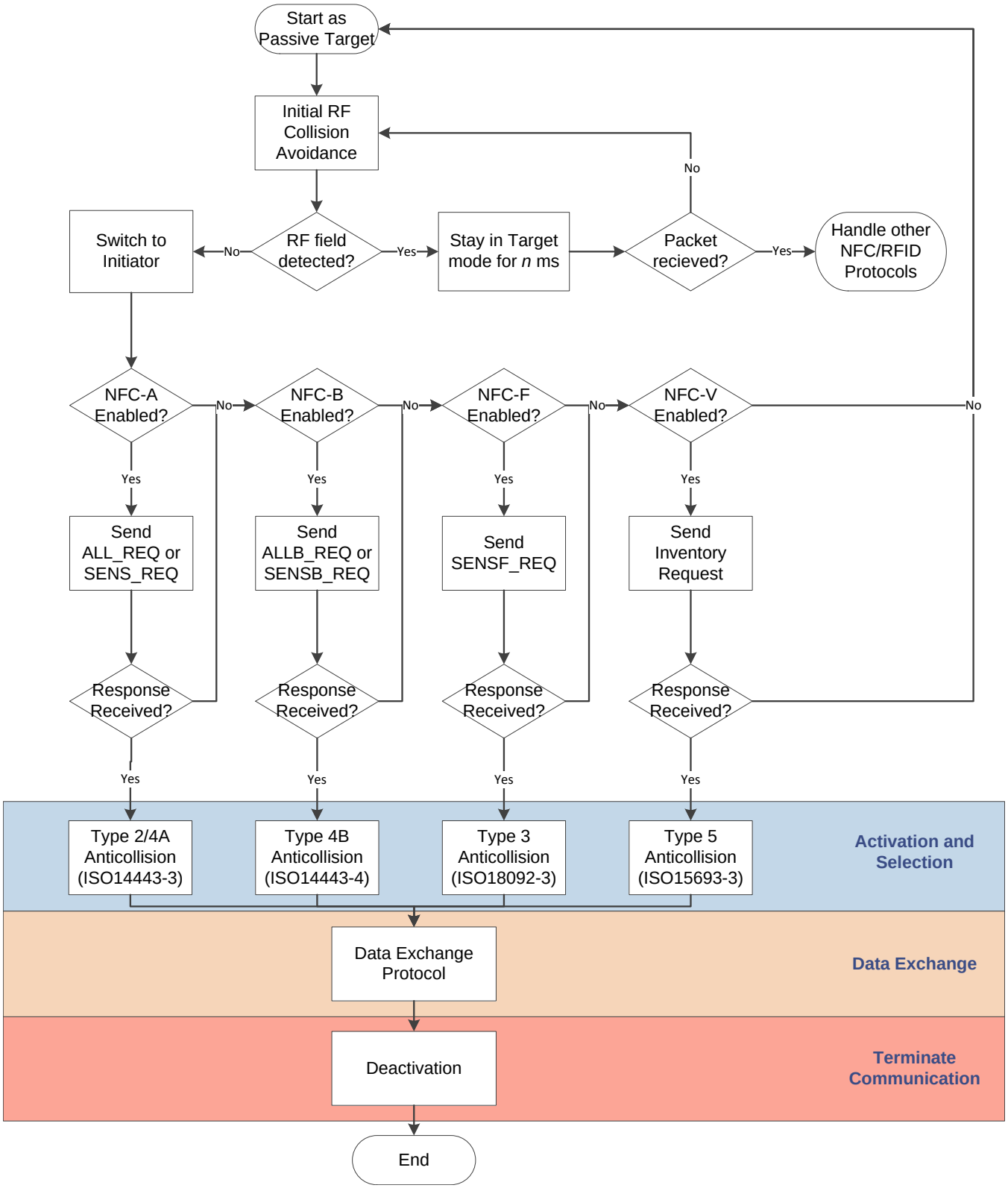

Figure 7. TRF7970A Reader/Writer Flow Diagram

5.1 Technology Activation Using the TRF7970A

The TRF7970A defaults as a passive target and therefore requires the ISO Control register (0x01) to be modified to enable the correct NFC technologies for reader/writer mode. Therefore, the following sections describe how to set up the TRF7970A and what the frame format is for each supported technology.

5.1.1 ISO14443-3 Type A (Type 2 and Type 4A Tags)

The frame format for ISO14443A packets at 106 kbps during anticollision (shown in Figure 8) is based on NFC-A technology specifications in NFCForum-TS-DigitalProtocol-1.0 and the ISO 14443-3 Specifications. The Type A frame format does not include CRC bytes when sending anticollision commands, but does include the CRC bytes when sending tag selection and data exchange command.

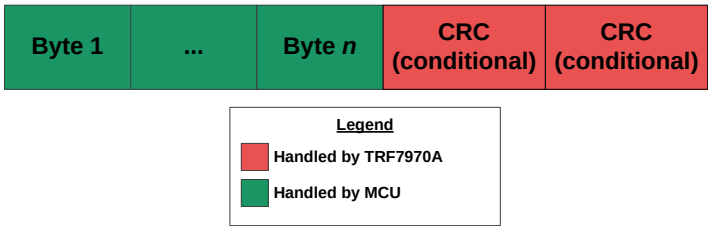

Figure 8. ISO14443A Frame Format

When the initial RF collision is completed and no RF field has been detected (as shown in Figure 7) the following registers must be modified before and after the anticollision is completed (see Figure 9). For more information, see Section 3.

- ISO Control register (0x01) = 0x88 (ISO14443A at 106 kbps, receive without CRC during anticollision, before Select command)
  or
  ISO Control register (0x01) = 0x08 (ISO14443A at 106 kbps, receive with CRC after anticollision is completed).
- Send packet:
  - Reset FIFO (0x0F) direct command.
  - Transmission without (0x10, anticollision before Select command) or with (0x11, after anticollision is completed) CRC direct command.
  - TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
  - Write the command to the FIFO.

In the context of Type A, anticollision refers to the mandatory process for the retrieval of the UID, and does not imply support for detecting multiple Type A tags presented at the same time.

The ISO control register must be modified for the anticollision state to receive without CRC for the required commands (see the ISO14443-3 specification for more information). When the anticollision is completed, the ISO Control register must be modified to receive with CRC. Step 2 must be used to send commands to the passive target.

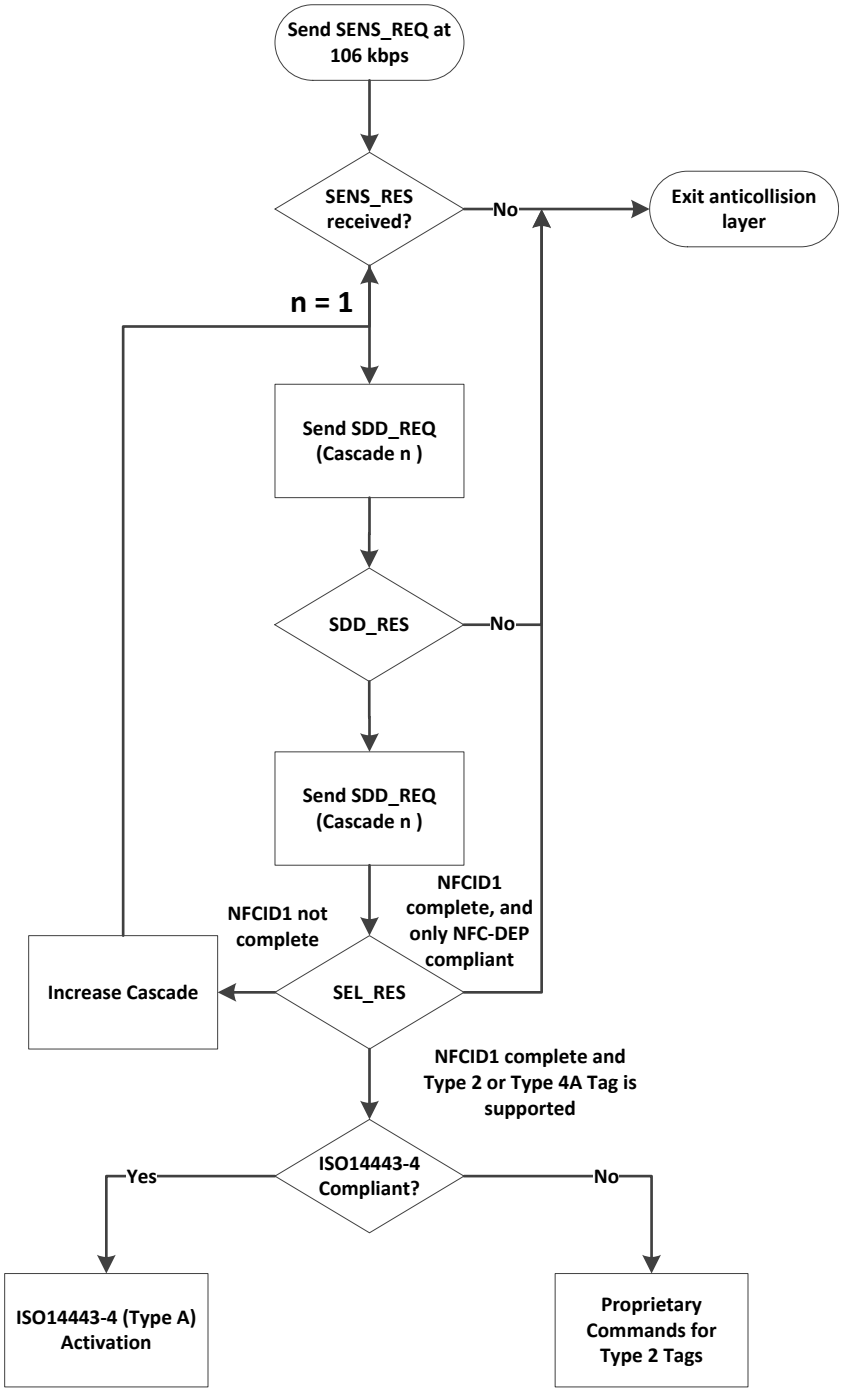

Figure 9. Reader/Writer Anticollision for ISO14443A

5.1.1.1 Additional Tag Activation Commands for Type 4A Tags

Tags that communicate with NFC-A technology and are ISO14443-4 compliant are considered Type 4A tags and must be sent an additional command, RATS, to finish the activation process and allow the exchange of data. There is also an optional command, PPS Request, that can be sent prior to entering data exchange. Because the example firmware provided sends both of these commands (see Figure 10), this section describes how both of them work.

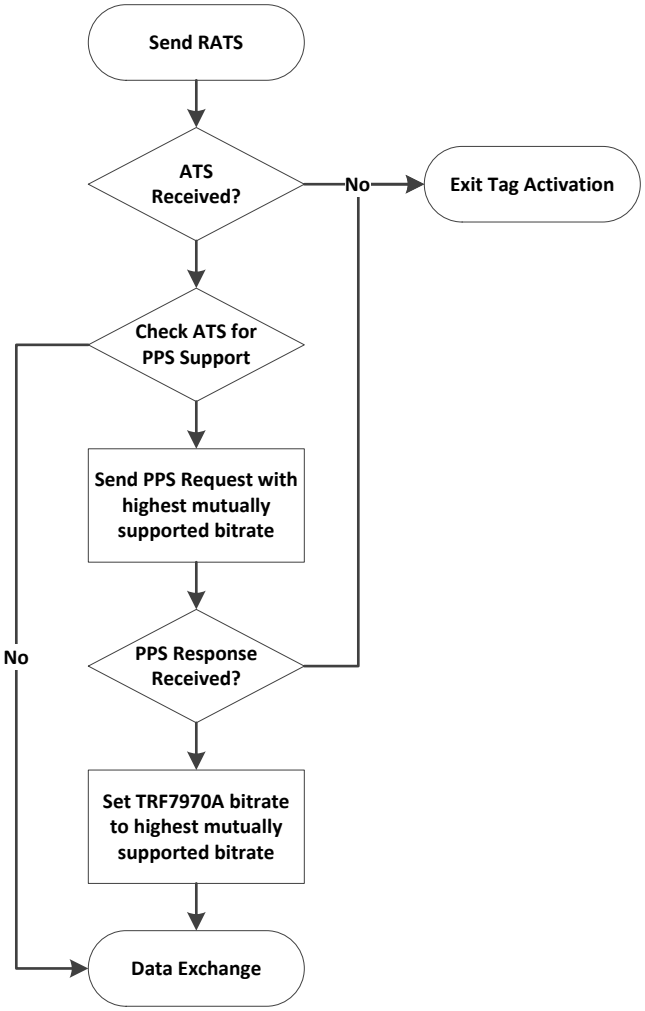

Figure 10. Tag Activation Process for Type 4A Tags

1. Request Answer To Select (RATS)

   The RATS command is used to select a Type 4A tag that has indicated support for ISO14443-4. All Type 4A tags must respond to the RATS command by sending back an Answer to Select (ATS) response. The ATS response can optionally include information about supported bit rates, timing requirements, and optional fields for over the air commands. For further detail on the format of the ATS response, refer to the ISO14443-4 specification.

2. Protocol and Parameter Selection Request (PPS Request)

   The PPS Request command is an optional command that can be issued by the reader to increase the communication bit rate with a Type 4A tag. The ATS response contains the information about the supported bit rates: 106 kbps, 212 kbps, 424 kbps, and 848 kbps. With this information, the firmware determines the highest mutually supported bit rate by comparing the ATS response to the enabled reader/writer bit-rate settings, and send that bit rate to the tag in the PPS Request to ensure that the tag will operate at that data rate.

   Upon receiving a PPS Response, the firmware will set the ISO Control register (0x01) with the correct settings to transmit data at the previously determined bit rate. At this point both the reader/writer and the Type 4A tag are ready to begin exchanging data.

5.1.2 ISO14443B-3 (Type 4B Tags)

The frame format for ISO14443B packets at 106 kbps (see Figure 11) is based on NFC-B technology specifications in NFCForum-TS-DigitalProtocol-1.0 and the ISO 14443-3 Specifications. The Type B frame format must always include the two CRC bytes.

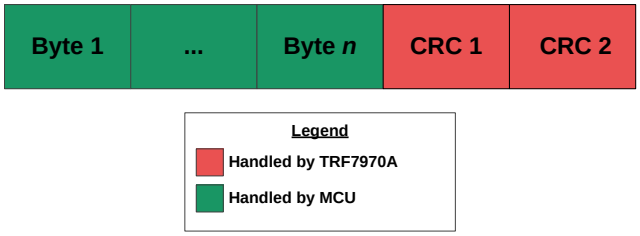

Figure 11. ISO14443B Frame Format

When the initial RF collision is completed and no RF field has been detected (see Figure 7), the ISO control register must be configured as shown in Step 1 below. Step 2 must be used to send commands to the NFC-B/ISO14443B tag. For more information on the initial RF collision, see Section 3.

1. ISO Control Register (0x01) → 0x0C (ISO 14443B at 106 kbps, receive with CRC)
2. Send packet
    1. Reset FIFO (0x0F) direct command.
    2. Transmission (0x11) with CRC direct command.
    3. TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
    4. Write the command to the FIFO.

5.1.2.1 Selection for Type 4B Tags

After activation is finished for a Type 4B tag the ATTRIB command must be issued to finish the selection of the tag to exchange data with it. The ATTRIB command sends the tag information on timing requirements, framing specifications for RF communication and for data packets, supported communication bit rates, and the CID value assigned to the tag.

To determine the highest mutually supported bit rate between the tag and the reader/writer the tenth byte of the SENSB_RES is compared with the enabled reader/writer bit-rate setting. One of the bytes in the ATTRIB command is then set to tell the tag to operate at that bit rate. The possible bit rates are: 106 kbps, 212 kbps, 424 kbps, and 848 kbps.

The response to the ATTRIB (Answer to ATTRIB) is an acknowledgment that it received the ATTRIB command. Upon receiving the ATTRIB response, the firmware sets the ISO Control Register (0x01) with the correct settings to transmit data at the previously determined bit rate. Now the reader/writer and Type 4B tag can enter data exchange. For further detail on the format of the ATTRIB command, refer to the ISO14443-3 specification.

5.1.3 ISO18092 (Type 3 Tags)

The frame format for ISO18092 packets at 212 kbps and 424 kbps (see Figure 12) is based on NFC-F technology specifications in NFCForum-TS-DigitalProtocol-1.0 and the ISO 18092/JIS X 6319-4 specifications. The Type F frame format must always include the two CRC bytes.

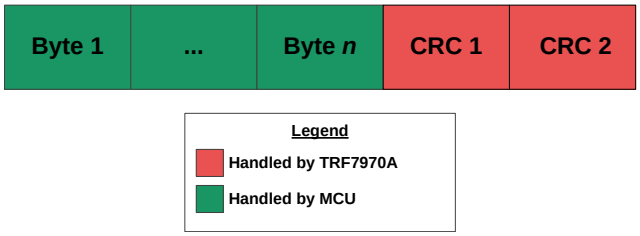

Figure 12. ISO18092 Frame Format

When the initial RF collision is completed and no RF field has been detected (see Figure 7), the ISO control register must be configured as shown in Step 1 below. Step 2 must be used to send commands to the Type 3 tag. For more information on the initial RF collision, see Section 3.

1. ISO Control Register (0x01) → 0x1A (NFC-F at 212 kbps, receive with CRC)
   or
   ISO Control Register (0x01) → 0x1B (NFC-F at 424 kbps, receive with CRC).
2. Send packet
   1. Reset FIFO (0x0F) direct command.
   2. Transmission with CRC direct command (0x11).
   3. TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
   4. Write the command to the FIFO.

The example firmware selects the first Type 3 tag that replies within an allotted time slot. Refer to the ISO 18092/JIS X 6319-4 or NFCForum-TS-DigitalProtocol-1.0 document for further details on the tag selection process and time slots.

5.1.4 ISO15693 (Type 5 Tags)

The frame format for ISO15693 packets at 26.48 kbps (see Figure 13) is based on the ISO 15693-3 Specifications. The Type 5 frame format must always include the two CRC bytes.

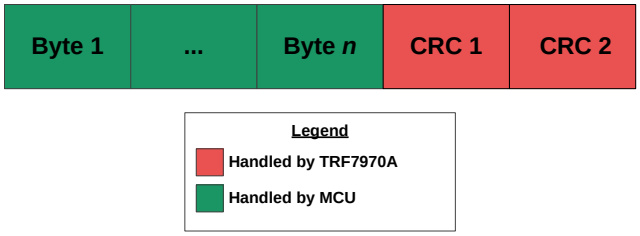

Figure 13. ISO15693 Frame Format

When the initial RF collision is completed and no RF field has been detected (see Figure 7), the ISO control register must be configured as shown in Step 1 below. Step 2 must be used to send commands to the ISO15693 tag. For more information on the initial RF collision, see Section 3.

1. ISO Control Register (0x01) → 0x02 (ISO15693 at 26.48 kbps, one subcarrier, 1 out of 4, receive with CRC).
2. Send packet
   1. Reset FIFO (0x0F) direct command.
   2. Transmission with CRC direct command (0x11).
   3. TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
   4. Write the command to the FIFO.

The provided example firmware does not contain anticollision for NFC-V/ISO15693 tags, so only one tag can be read or written to at one time.

Upon reading the basic information of tag, the firmware determines the available memory size and if any special request flags must be used (such as extended protocol or option flags) based on the response to Get System Information commands (see Figure 14 for further details).

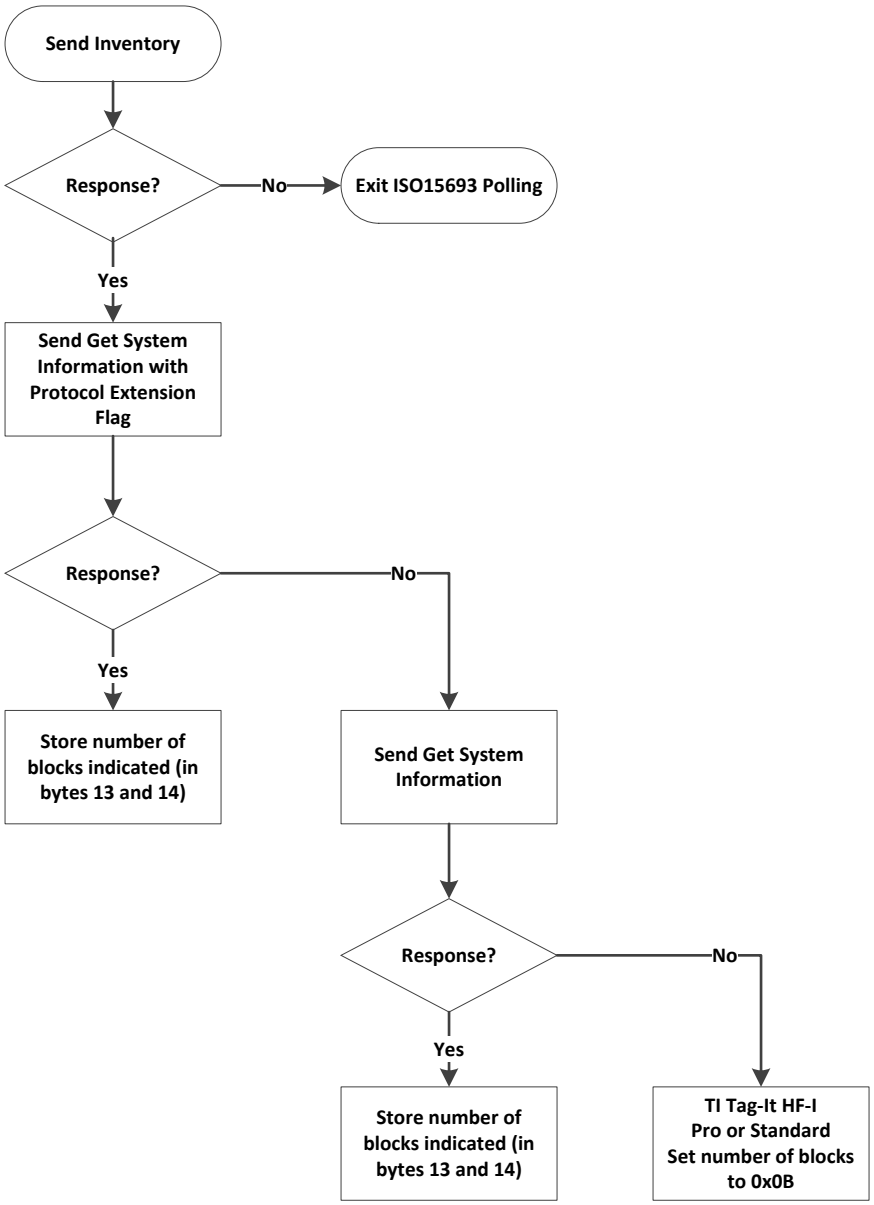

Figure 14. Flow Chart for Determining Type 5 Tag Memory Size

5.2 Tag Memory Format With NDEF Examples

The NFC Data Exchange Format (NDEF) specification is a data format standard for storing information on NFC tag platforms in messages. The composition of NDEF messages are determined by the various NFC Forum Record Type Definition (RTD) formats. Among the supported RTD formats are Text, Uniform Resource Identifier (URI), Smart Poster, V-Card, and MIME. The examples for the following sections focus on the NFC Forum Text RTD structure.

The method to detect and read/write NDEF contents vary between tag types, but one common factor is the requirement for basic information needed to read the contents of an NDEF formatted message. In most tags, this is called the Capability Container (CC) file. The exception to that is with Type 3 tags, which instead have an 'Attribute Information Block'. The information stored in these bytes is used to determine the mapping specification, data sizes, read and write access, and other parameters. According to the current specifications of each tag type, only after the CC (or Attribute Block) is read can the NDEF contents of a tag be read.

For most tag types, with Type 3 being the exception once again, information about NDEF messages are stored with Tag, Length, Value (TLV) bytes. The Tag field provides information on the format and the Length field indicates the length of the NDEF message. For Type 2 and Type 5 tags, the Value field is where the NDEF message itself is stored. For Type 4 tags, the Value field contains additional information about the NDEF message. Based on each NFC Forum tag specification, the TLV bytes may not be separate and must follow one another sequentially.

The following sections describe the basics of the memory layout, NDEF formatting, and Capability Container (or Attribute Block) for each tag type. For further details, review the NFC Forum specifications for each tag type. References for each specification can be found in Section 12.

5.2.1 Type 2 Tags

5.2.1.1 Layout

Type 2 tags use a block memory format where each block contains four bytes of data. The read command for Type 2 tags read out four blocks at a time, so if a block number of 0x03 is given, then blocks 0x03 through 0x06 are read.

Type 2 tags have two possible memory structures depending on the size of the tag. Static memory structures apply for Type 2 tags that have a memory size equal to 64 bytes, whereas dynamic memory structures are applied to Type 2 tags with memory sizes larger than 64 bytes.

5.2.1.2 NDEF Format

For all Type 2 tags, the Capability Container (CC) is located on Block 3. Therefore, it is possible to determine if a tag is NDEF formatted just by checking the values read from Block 3. Only if the first byte is equal to a 0xE1 does the tag have NDEF data stored in the data area.

5.2.1.2.1 Static Memory Structure

For tags with a static memory structure, the CC is immediately followed by the NDEF contents starting in Block 4 starting with the Tag field of the TLV bytes. The end of the NDEF message is indicated by a TLV that has a Tag field of 0xFE. This is known as the terminator TLV. See Figure 15 for an example of a Text RTD stored in a Type 2 tag that has a static memory structure.

	Byte 0	Byte 1	Byte 2	Byte 3
Block 0x03 (CC)	0xE1	0x10	0x06	0x00
Block 0x04	0x03 (NDEF TLV Tag Field)	0x19 (NDEF TLV Length Field)	0xD1 (Short Record, Well Known RTD)	0x01 (Record Type Length)
Block 0x05	0x15 (NDEF Payload Length)	0x54 (Text RTD)	0x02 (Language Length)	0x65 ('e')
Block 0x06	0x6E ('n')	0x4E ('N')	0x46 ('F')	0x43 ('C')
Block 0x07	0x20 (' ')	0x50 ('P')	0x6F ('o')	0x77 ('w')
Block 0x08	0x65 ('e')	0x72 ('r')	0x65 ('e')	0x64 ('d')
Block 0x09	0x20 (' ')	0x42 ('B')	0x79 ('y')	0x20 (' ')
Block 0x0A	0x54 ('T')	0x49 ('I')	0x21 ('!')	0xFE (Terminator)

Figure 15. Text RTD Example for Type 2 Tags With a Static Memory Structure

5.2.1.2.2 Dynamic Memory Structure

For tags with a dynamic memory structure, the CC is followed by a series of TLV bytes for Lock Control and Memory Control. The NDEF TLV bytes are found starting at Byte 2 of Block 6. The end of the NDEF message is indicated by a TLV that has a Tag field of 0xFE. This is known as the terminator TLV. See Figure 16 for an example of a Text RTD stored in a Type 2 tag that has a dynamic memory structure.

	Byte 0	Byte 1	Byte 2	Byte 3
Block 0x03 (CC)	0xE1	0x10	0x06	0x00
…	…	…	…	…
Block 0x06	Memory Control TLV	Memory Control TLV	0x03 (NDEF TLV Tag Field)	0x19 (NDEF TLV Length Field)
Block 0x07	0xD1 (Short Record, Well Known RTD)	0x01 (Record Type Length)	0x15 (NDEF Payload Length)	0x54 (Text RTD)
Block 0x08	0x02 (Language Length)	0x65 ('e')	0x6E ('n')	0x4E ('N')
Block 0x09	0x46 ('F')	0x43 ('C')	0x20 (' ')	0x50 ('P')
Block 0x0A	0x6F ('o')	0x77 ('w')	0x65 ('e')	0x72 ('r')
Block 0x0B	0x65 ('e')	0x64 ('d')	0x20 (' ')	0x42 ('B')
Block 0x0C	0x79 ('y')	0x20 (' ')	0x54 ('T')	0x49 ('I')
Block 0x0D	0x21 ('!')	0xFE (Terminator)		

Figure 16. Text RTD Example for Type 2 Tags With a Dynamic Memory Structure

5.2.1.3 Capability Container

The format for the CC for Type 2 tags is specified in the NFC Forum Type 2 Tag Operation Specification. Version 1.2 was used to write this application report.

- **Byte 0** – This value is the 'magic number' and must always be equal to 0xE1 to indicate that NDEF data is stored.
- **Byte 1** – This value indicates the reader which mapping specification is being used by the tag.
- **Byte 2** – This value indicates the memory size of the data sections for the Type 2 tag. This value must be multiplied by 8 to get the total memory size of the data section in Bytes.
- **Byte 3** – This value indicates the Read/Write access capabilities of the Type 2 tag. The most significant nibble is for Read access. A value of 0x0 means the tag can be read from without any security. The least significant nibble is for Write access. A value of 0x0 means the tag can be written to without any security, and a value of 0xF means that the tag is Read Only.

5.2.2 Type 3 Tags

5.2.2.1 Layout

Type 3 tags have block memory format where each block contains 16 bytes of data. The blocks are numbered starting at Block 0. It is possible to read multiple blocks at once, and the maximum number of blocks that can be read is defined the Attribute Information Block (see Section 4.2.2.3 below for more details).

5.2.2.2 NDEF Format

For Type 3 tags, all the information about the NDEF content is stored within the Attribute Information Block, and therefore Type 3 tags do not use a Capability Container. The Attribute Information Block is always located at Block 0, and contains all the required information to read and write NDEF messages. Unlike the other tag types, the specifications for Type 3 tags have not defined TLVs and therefore they are not used. See Figure 17 for an example of a Text RTD stored in a Type 3 tag.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
Block 0	10h	04h	01h	00h	0Dh	00h	00h	00h	00h	00h	01h	00h	00h	19h	00h	3Ch
Block 1	D1h	01h	15h	54h	02h	65h	6Eh	4Eh	46h	43h	20h	50h	6Fh	77h	65h	72h
Block 2	65h	64h	20h	42h	79h	20h	54h	49h	21h	--	--	--	--	--	--	--

Figure 17. Text RTD Example for Type 3 Tags

5.2.2.3 Attribute Information Block

The format for the Attribute Information Block (see Figure 18) is specified in the NFC Forum Type 3 Tag Operation Specification. Version 1.2 was used to write this application report.

Mapping Version	Max Blocks to Read	Max Blocks to Write	Blocks for NDEF Storage		Unused				Write Flag	NDEF Access Read/Write Flag	Current NDEF Message Length			Checksum	
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15

Figure 18. Type 3 Tag Attribute Information Block Format

- **Mapping Version (Byte 0):** For NDEF format it is required that this is set to a valid version number.
- **Max Blocks to Read (Byte 1):** This value indicates how many blocks of data can be read from the tag at one time.
- **Max Blocks to Write (Byte 2):** This value indicates how many blocks of data can be written to the tag at one time.
- **Blocks for NDEF Storage (Bytes 3 and 4):** These bytes indicate the number of blocks available for NDEF content. Since each block is 16 bytes, this value should be multiplied by 16 to know how big the tag is in bytes.
- **Unused (Bytes 5-8):** These bytes are not used, and typically are set to 0x00.
- **Write Flag (Byte 9):** This flag is toggled when data is being written to the tag to indicate that the tag is currently in a write cycle.
- **NDEF Access Read/Write Flag (Byte 10):** This flag is used to indicate the Read/Write privileges of the tag. A value of 0x00 means that the tag is Read Only, and a value of 0x01 means that the tag is Read/Write available.
- **Current NDEF Message Length (Bytes 11-13):** These bytes indicate the size of the NDEF message that is stored in bytes. Since this value is stored in bytes, it must be divided by 16 to determine how many blocks must be read to receive the full NDEF message.
- **Checksum (Bytes 14-15):** This is used to verify the validity of the data in the Attribute block, and it is calculated by adding together the values of Bytes 0 through 13. (modified when written to)

5.2.3 Type 4 Tags

5.2.3.1 Layout

Type 4 tags have an object oriented memory format that uses files and not blocks for data storage. Data is read or written by providing how many data bytes should be read or written out by using an index via an offset value of the selected file. The command for reading data from a Type 4 tag is known as a Read Binary, and the command for writing data to a Type 4 tag is known as an Update Binary.

5.2.3.2 NDEF Format

To read an NDEF message from a Type 4 tag, a series of selections and reads must be performed to properly access the data. The first step is to select the NDEF Application, which is signified by an Application ID of 0xD2760000850101. The second step is to select the Capability Container file. NDEF applications must contain the Capability Container file with a File ID of 0xE103.

To read the contents of the CC, the first two bytes must be read first to determine the length of the CC, and then the rest of the CC must be read. The CC contains the File ID information for the NDEF data, and that must be used to read the correct File IDs to read all of the NDEF data from the tag. The CC also contains the TLV fields for each NDEF message stored.

There may be multiple files present in a single tag, so the information for each individual file must be stored within the CC. See Figure 19 and Figure 20 for an example of how the tag memory of a properly formatted NDEF message looks.

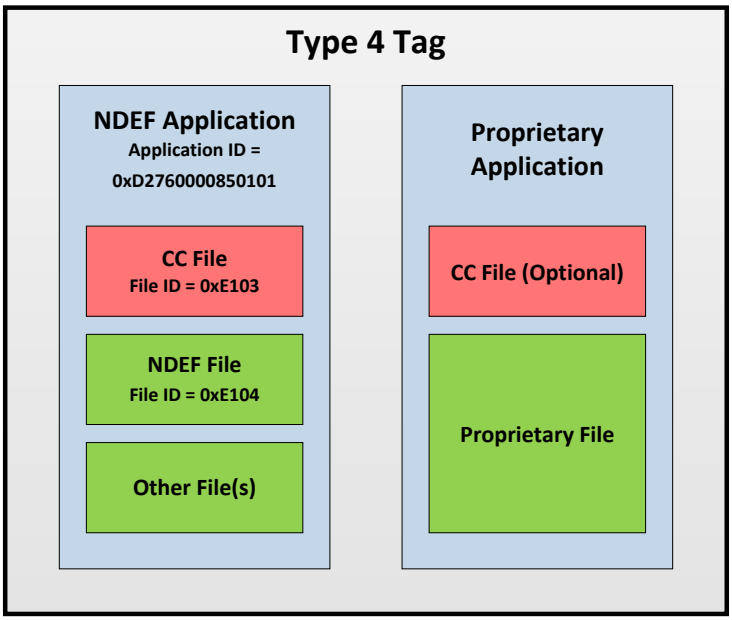

Figure 19. Type 4 Tag NDEF Structure

NDEF File Contents for Text RTD				
0x00 (NDEF Message Length)	0x19 (NDEF Message Length)	0xD1 (Short Record, Well Known RTD)	0x01 (Record Type Length)	0x15 (NDEF Payload Length)
0x54 (Text RTD)	0x02 (Language Length)	0x65 ('e')	0x6E ('n')	0x4E ('N')
0x46 ('F')	0x43 ('C')	0x20 (' ')	0x50 ('P')	0x6F ('o')
0x77 ('w')	0x65 ('e')	0x72 ('r')	0x65 ('e')	0x64 ('d')
0x20 (' ')	0x42 ('B')	0x79 ('y')	0x20 (' ')	0x54 ('T')
0x49 ('I')	0x21 ('!')	--	--	--

Figure 20. Type 4 NDEF File Example for Text RTD

5.2.3.3 Capability Container

The Capability Container is stored inside of a file that has a file identifier of 0xE103. The format for the Capability Container is specified in the NFC Forum Type 4 Tag Operation Specification 3.0. See Figure 21 for an example of a properly formatted CC.

Capability Container (File Number = 0xE103)		
CC Length	0x00	0x0F
Mapping Version	0x20	--
MLe	0x00	0x3B
MLc	0x00	0x34
Type	0x04	--
Length	0x06	--
File ID	0xE1	0x04
Max NDEF Size	0x0B	0xDF
Read Capability	0x00	--
Write Capability	0x00	--

Figure 21. Example of Type 4 Capability Container Contents

- **CC Length** – This 2 byte value is determined by the number of different files contained within the tag. The formula to calculate the CC Length is: [7 + (*number of files* × 8)].
- **Mapping Version** – This value tells the reader the mapping specification used by the tag.

- **MLe** – These 2 bytes indicate the maximum number of bytes that can be read from the tag by a single Read Binary command.
- **MLc** – These 2 bytes indicate the maximum number of bytes that can be written to the tag by a single Write Binary command.
- **File Control TLV** – The TLV blocks provide the reader with information about the tag data.
- **Tag Field (T)** – This field contains the file type information of the TLV block. The following values may be used:
  - 0x04 – NDEF File
  - 0x05 – Proprietary File
  - 0x06 – Extended NDEF file
- **Length Field (L)** – This field contains the length of the value field of the TLV block
- **Value Field (V)** – This field contains the file ID, the maximum file size, and the read/write access conditions of the file.
  - File ID – Each file within the tag should have its own unique file ID number.
  - Max File Size – This value is not the size of the data itself, but the size of the maximum possible data that can be stored within the file.
  - Read Access (R) – Determines the read access properties of the file. A value of 0x00 means that the file has full read access. All other values are for limited read access.
  - Write Access (W) – Determines the write access properties of the file. The value 0x00 allows for full write privileges. The value 0xFF disables all write privileges and makes the file read-only. All other values are for limited write access.

There must be a File Control TLV for each different file contained within the tag. Therefore, if there are three different files, there must be three TLVs that correspond to the settings of each file.

5.2.4 Type 5 Tags

5.2.4.1 Layout

Type 5 tags have a block memory format where each block can contain either four or eight bytes of data. The blocks are numbered starting at Block 0. Blocks can be read with the Read Single Block command or with the Read Multiple Blocks command if the Type 5 tag supports that command.

5.2.4.2 NDEF Format

For an NDEF Formatted tag, the Capability Container (CC) must be placed in Block 0. The CC is immediately followed by the NDEF contents that should contain the Type and Length (TLV) information for the tag. The NDEF data is stored directly after the TLV length byte, and the end of the NDEF data is indicated by a TLV that starts with 0xFE, which is known as the terminator TLV. The example in Figure 22 is for a Type 5 tag that has 4-byte memory blocks.

	Byte 0	Byte 1	Byte 2	Byte 3
Block 0x00 (CC)	0xE1	0x10	0x06	0x00
Block 0x01	0x03 (NDEF TLV Tag Field)	0x19 (NDEF TLV Length Field)	0xD1 (Short Record, Well Known RTD)	0x01 (Record Type Length)
Block 0x02	0x15 (NDEF Payload Length)	0x54 (Text RTD)	0x02 (Language Length)	0x65 ('e')
Block 0x03	0x6E ('n')	0x4E ('N')	0x46 ('F')	0x43 ('C')
Block 0x04	0x20 (' ')	0x50 ('P')	0x6F ('o')	0x77 ('w')
Block 0x05	0x65 ('e')	0x72 ('r')	0x65 ('e')	0x64 ('d')
Block 0x06	0x20 (' ')	0x42 ('B')	0x79 ('y')	0x20 (' ')
Block 0x07	0x54 ('T')	0x49 ('I')	0x21 ('!')	0xFE (Terminator)

Figure 22. Text RTD Example for Type 5 Tags

5.2.4.3 Capability Container

The format of the Capability Container for Type 5 tags presented below is based on the current conventions that are used in the field with existing Type 5 tags.

- **Byte 0** – This value is the 'magic number' and must always be equal to 0xE1 to indicate that NDEF data is stored.
- **Byte 1** – This value indicates the reader which mapping specification is being used by the tag.
- **Byte 2** – This value indicates the memory size of the data sections for the Type 5 tag. Multiply this value by 8 to get the total memory size of the data section in bytes.
- **Byte 3** – This value indicates the Read/Write access capabilities of the Type 5 tag. The most significant nibble is for Read access. A value of 0x0 means the tag can be read from without any security. The least significant nibble is for Write access. A value of 0x0 means the tag can be written to without any security, and a value of 0xF means that the tag is read only.

6 Hardware Description

6.1 LaunchPad™ Development Kit and BoosterPack™ Plug-in Module Setup

6.1.1 BoosterPack Plug-in Module: DLP-7970ABP

The third party provider DLP Design NFC/RFID BoosterPack™ plug-in module (DLP-7970ABP) is an add-on board designed to fit all of TI's MCU LaunchPad™ development kits. This BoosterPack plug-in module allows the software application developer to get familiar with the functionalities of TRF7970A multi-protocol fully integrated 13.56-MHz NFC/HF RFID IC on their TI embedded microcontroller platform of choice without having to worry about designing the RF section (see Figure 23 and Figure 24).

The TRF7970A device is an integrated analog front end and data-framing device for a 13.56-MHz NFC/HF RFID system. Built-in programming options make the device suitable for a wide range of applications for proximity and vicinity identification systems. The device can perform in one of three modes: reader/writer, peer-to-peer, or card emulation mode. Built-in user-configurable programming registers allows fine tuning of various reader parameters as needed.

Link for purchase: https://store.ti.com/dlp-7970abp.aspx

6.1.2 LaunchPad Development Kit: MSP-EXP430F5529LP

The MSP-EXP430F5529LP LaunchPad development kit is an easy-to-use evaluation module for the MSP430F5529 USB microcontroller. It contains everything needed to start developing, including on-board emulation for programming and debugging, as well as on-board buttons and LEDs for quickly adding a simple user interface. Rapid prototyping is a snap, thanks to 40-pin access headers and a wide range of BoosterPack plug-in modules. This enables technologies such as wireless, display drivers, temperature sensing, and much more (see Figure 23).

Link for purchase: https://store.ti.com/msp-exp430f5529lp.aspx

Figure 23. MSP430F5529 LaunchPad Development Kit and DLP-7970ABP BoosterPack Plug-in Module

6.1.3 LaunchPad Development Kit: MSP-EXP432P401R

The MSP432P401R LaunchPad development kit enables you to develop high-performance applications that benefit from low-power operation. It features the MSP432P401R – which includes a 48-MHz Arm® Cortex®-M4F, 95-µA/MHz active power, and 850-nA RTC operation, a 14-bit 1-Msps differential SAR ADC, and an AES256 accelerator.

This LaunchPad development kit includes an on-board emulator with EnergyTrace+ technology, which means you can program and debug your projects without the need for additional tools while also measuring total system energy consumption (see Figure 24).

Link for purchase: https://store.ti.com/msp-exp432p401r.aspx

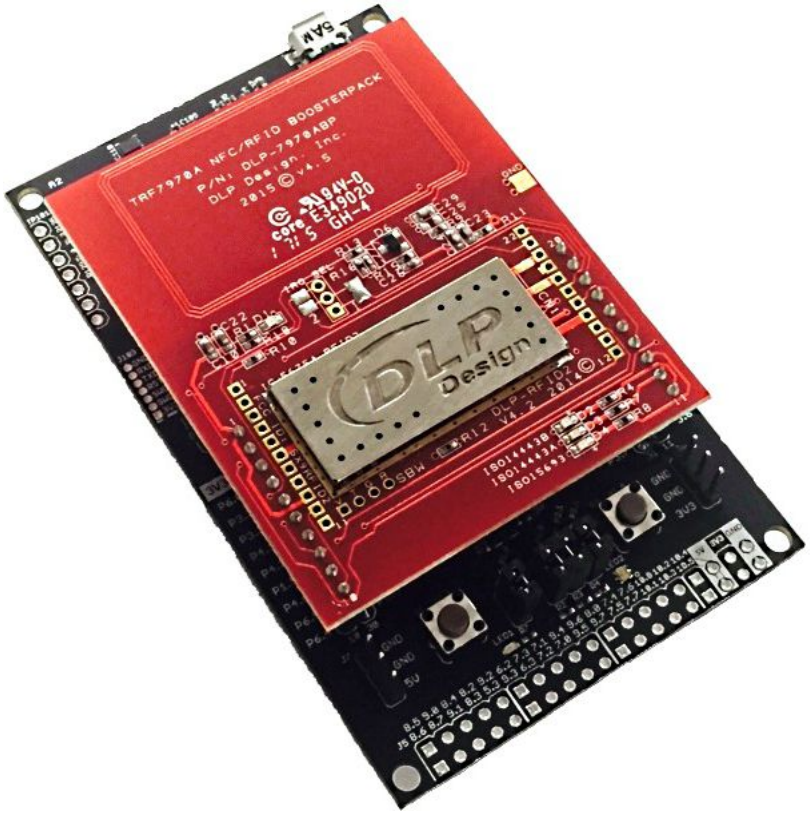

Figure 24. MSP432P401R LaunchPad Development Kit and DLP-7970ABP BoosterPack Plug-in Module

6.2 Bundle Available for Purchase

The TI store offers this bundle:

MSP-EXP430F5529LP and DLP-7970ABP

7 Reader/Writer Firmware Example

This section explains which APIs in the NFC/RFID layer (see Figure 25) are used to initialize and handle the reader/writer communication. Furthermore, it describes how to implement a reader/writer application that can send and receive NDEF message to and from an NFC-enabled device.

The firmware example that contains the reader/writer APIs discussed in this document can be downloaded from http://www.ti.com/lit/zip/sloa227.

As downloaded, the firmware example includes the full TI NFC stack which supports peer-to-peer, card emulation, and reader/writer modes. For applications that do not require all NFC operating modes, there are configuration options available to reduce the NFC stack memory footprint (only compiling required operating modes). These configurations can be made by modifying the #define statements within the *nfc_config.h* file, located at [Install Path]\nfclink\Source\headers.

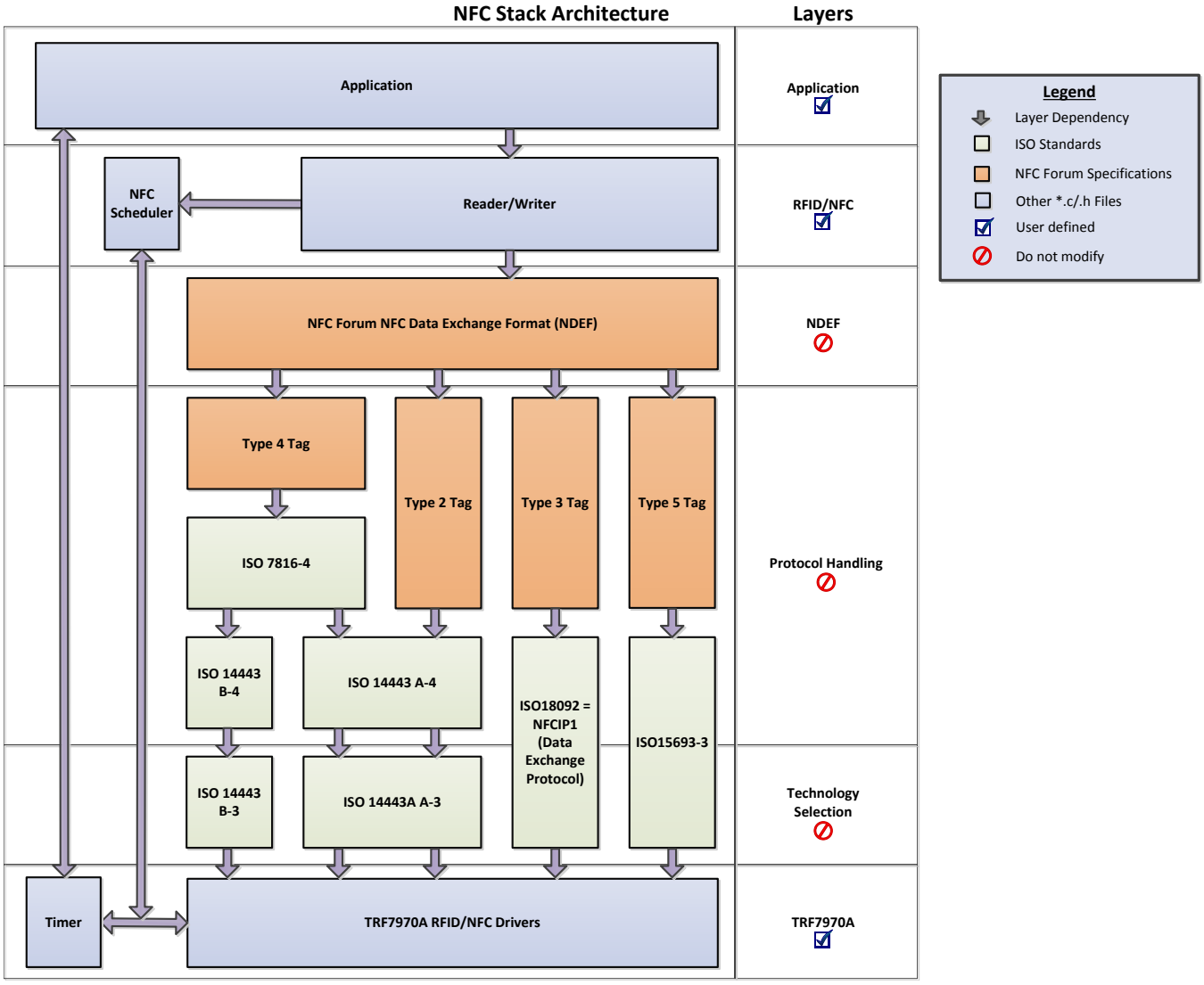

Figure 25. Reader/Writer NFC Stack Architecture

7.1 Reader/Writer APIs

For details on all available APIs used in the provided example firmware for NFC reader/writer mode, see the *NFCLink Standalone Software Library API Guide* included in the install package. The guide is located in [Install Path]\doc.

The *NFCLink Standalone Software Library API Guide* describes the flow of the software stack, all APIs that are available for NFC reader/writer functionality, and each function to help users with developing custom NFC reader/writer applications.

7.2 Implementing a Reader/Writer Sample Application

This section explains how to implement a reader/writer sample application that uses buttons S1 and S2 on the MSP430F5529 LaunchPad development kit to send different NDEF messages to an NFC-enabled device. Table 3 and Table 4 show the connections between the MSP430F5529 and the TRF7970A for the different MSP430F5529 evaluation platforms. Table 5 lists the connections between the MSP432P401R and the TRF7970A for the evaluation platforms.

Table 3. DLP-7970ABP BoosterPack Module and MSP-EXP430F5529LP LaunchPad Kit Hardware Connections

DLP-7970ABP Pins	MSP430F5529 LaunchPad Development Kit Pins
TRF7970A EN1	P4.1
TRF7970A IRQ	P2.2(1)
MOSI	P3.0
MISO	P3.1
CLK	P3.2
Slave Select	P4.2
I/O_2	P6.6(2)
I/O_3	P2.0(2)
I/O_5	P1.6(2)

(1) IRQ is defaulted to P3.0 for DLP-7970ABP v4.5 and newer (see the *DLP-7970ABP hardware update overview*).

(2) Pin is needed for Special Direct mode only.

Table 4. TRF7970ATB and MSP-EXP430F5529 Experimenter Board Hardware Connections

TRF7970ATB Pins	MSP430F5529 Experimenter Board Pins
TRF7970A EN1	P2.3
TRF7970A IRQ	P2.0(1)
MOSI	P3.0
MISO	P3.1
CLK	P3.2
Slave Select	P2.6
MOD	P2.1
ASK/OOK	P4.7

(1) Requires a jumper to be placed between P2.0 and P4.0 on the experimenter board.

Table 5. DLP-7970ABP BoosterPack Module and MSP-EXP432P401R LaunchPad Kit Hardware Connections

DLP-7970ABP Pins	MSP432P401R LaunchPad Development Kit Pins
TRF7970A EN1	P6.4
TRF7970A IRQ	P3.0[(1)]
MOSI	P1.6
MISO	P1.7
CLK	P1.5
Slave Select	P6.5
I/O_2	P4.3[(2)]
I/O_3	P2.5[(2)]
I/O_5	P4.1[(2)]

(1) IRQ defaults to P3.0 for DLP-7970ABP v4.5 and newer (see the *DLP-7970ABP hardware update overview*).

(2) Pin is needed for Special Direct mode only.

7.2.1 Low-Level Initialization

For the low-level initialization the MCU is initialized in MCU_init() by setting the main clock frequency of the MSP430F5529 to 25 MHz. The TRF7970A hardware connections and the MSP430F5529 SPI module (SPI clock running at 4 MHz; minimum recommended is 2 MHz) is initialized in TRF79x0_init(). After this, the NFC_init must be called to properly initialize all variables within the NFC stack. When the NFC stack is initialized, it can be configured by calling the NFC_configuration function, which sets up the NFC stack to enable the TRF7970A for the desired modes of communication. Users can customize the enabled NFC modes either by modifying the function within the firmware or by using the PC GUI interface. See Example 1 for an example of how to properly initialize the TRF7970A in the main(void) program of a C project.

Example 1. MCU and TRF7970A Initialization Code Snippet

```
#include "msp430.h"
#include "nfc_controller.h"
#include "tag_header.h"
#include "lp_buttons.h"

t_sNfcRWMode g_sRWSupportedModes;
t_sNfcRWCommBitrate g_sRWSupportedBitrates;
t_sIsoDEP_RWSetup g_sRWSetupOptions;
uint8_t g_ui8IsoDepInitiatorDID;

uint8_t g_ui8TxBuffer[256];
uint8_t g_ui8TxLength;

void main(void)
{
     tNfcState eTempNFCState;
     tNfcState eCurrentNFCState;

     // Reader/Writer Variables
     t_sNfcRWMode sRWMode;
     t_sNfcRWCommBitrate sRWBitrate;

     // Initialize MCU
     MCU_init();

     // Enable interrupts globally
     __enable_interrupt();

     // Initialize USB Communication
```

Example 1. MCU and TRF7970A Initialization Code Snippet (continued)

```
     Serial_init();

     // Initialize TRF7970
     TRF79x0_init();

#ifdef MSP430F5529_EXP_BOARD_ENABLED
     Buttons_init(BUTTON_ALL);
     Buttons_interruptEnable(BUTTON_ALL);
#endif

     TRF79x0_idleMode();

     // Initialize the NFC Controller
     NFC_init();

     // This function will configure all the settings for each protocol
     NFC_configuration();

     // Initialize the RW T2T, T3T, T4T and T5 state machines
     T2T_init(g_ui8TxBuffer,256);
     T3T_init(g_ui8TxBuffer,256);
     T4T_init(g_ui8TxBuffer,256);
     T5T_init(g_ui8TxBuffer,256);
```

7.2.2 Reader/Writer NFC Stack Configuration

The reader/writer NFC stack is initialized by setting the bits for each desired reader/writer mode with the g_sRWSupportedModes variable inside of the NFC_configuration function. Then for each NFC mode, it is possible to customize which bit rates are supported by the firmware. In Example 2, ISO14443A (NFC-A) is enabled with bit rates of 106 kbps and 848 kbps selected, ISO14443B (NFC-B) is enabled with bit rates of 106 kbps and 848 kbps selected, FeliCa (NFC-F) is enabled with a bit rates of 212 kbps selected, and ISO15693 is enabled with a bit rate of 26.48 kbps selected.

For NFC-A and NFC-B, the 106-kbps bit rate must be used even if a higher bit rate is desired, because the tags can only be selected with 106-kbps communication. When the tag selection is complete, the firmware increases the bit rate if the higher bit rate is supported by the tag.

Example 2. Reader/Writer Stack Configuration Code Snippet

```
// Enable Reader Writer Supported Modes
g_sRWSupportedModes.bits.bNfcA = 1;
g_sRWSupportedModes.bits.bNfcB = 1;
g_sRWSupportedModes.bits.bNfcF = 1;
g_sRWSupportedModes.bits.bISO15693 = 1;

// NFC-A Bit rates
g_sRWSupportedBitrates.bits.bNfcA_106kbps = 1;   // Must be enabled if bNfcA is set
g_sRWSupportedBitrates.bits.bNfcA_212kbps = 0;
g_sRWSupportedBitrates.bits.bNfcA_424kbps = 0;
g_sRWSupportedBitrates.bits.bNfcA_848kbps = 1;

// NFC-B Bit rates
g_sRWSupportedBitrates.bits.bNfcB_106kbps = 1;   // Must be enabled if bNfcB is set
g_sRWSupportedBitrates.bits.bNfcB_212kbps = 0;
g_sRWSupportedBitrates.bits.bNfcB_424kbps = 0;
g_sRWSupportedBitrates.bits.bNfcB_848kbps = 1;

// NFC-F Bit rates
g_sRWSupportedBitrates.bits.bNfcF_212kbps = 1;
g_sRWSupportedBitrates.bits.bNfcF_424kbps = 0;
```

Example 2. Reader/Writer Stack Configuration Code Snippet (continued)

```
// ISO15693 Bit rates
g_sRWSupportedBitrates.bits.bISO15693_26_48kbps = 1;
```

7.2.3 Activation

After configuration is completed, the NFC_run function is called to run the NFC stack that polls for the enabled technologies, and then goes through activation and selection for the first tag that it receives a reply from. When the tag is properly activated and selected, then the application calls the state machines needed to read the data from the tag.

7.2.4 Reading and Writing Tags

There are four different state machines available for reading and writing tags: T2T_stateMachine, T3T_stateMachine, T4T_stateMachine, and T5T_stateMachine. Since Type 4A and Type 4B tags use the same memory structure, after activation and selection the same process can be used to read and write either T4TA or T4TB platforms.

When a state machine is called, it automatically attempts to read a tag of that technology by default. It first checks for NDEF content on the tag and, if an NDEF message is found, then reads it. If no NDEF content is found, then the state machine proceeds to read the raw data from the tag. The firmware sends the received data to the USB interface to display it on the TI NFC Tool GUI (see Section 8).

If a user needs to access the read data for an application specific purpose, it is possible to copy the data read from the tag by finding the correct read within the state machine. Each state machine is designed to output received tag data to the USB interface, so the most recently received data can be found at any Serial_printBuffer function call.

To keep the memory sizes at reasonable levels, the amount of data that is stored from reading tags at any time is limited. At the top of each state machine is a declaration for a buffer labeled as g_pui8T*X*TRxBuffer, where the *X* represents the tag type number.

When a tag has been fully read, then the state machine enters an IDLE state. During this state, the state machine can enter a write state to write new data onto the tag. However, this behavior could be modified if desired.

When the tag state machine has finished reading and, if applicable, writing the tag, it exits and returns back to the main application code. No further polling, reads, or writes take place until the tag is removed from the RF field. When the tag is removed, then the reader reinitializes the state machines to reset them and resumes polling for each enabled technology (see Section 7.2.3).

8 Quick Start Guide

The *NFCLink Standalone Getting Started Guide* provides complete details of how to get started with the provided example firmware and TI hardware.

This guide describes how to load the example firmware to TI evaluation boards and explains the features of the TI NFC Tool GUI (see Figure 26), which is installed with the firmware package.

The TI NFC Tool allows for quick configuration of the different NFC modes and provides an inferface to read and write data with the supported NFC tag platforms.

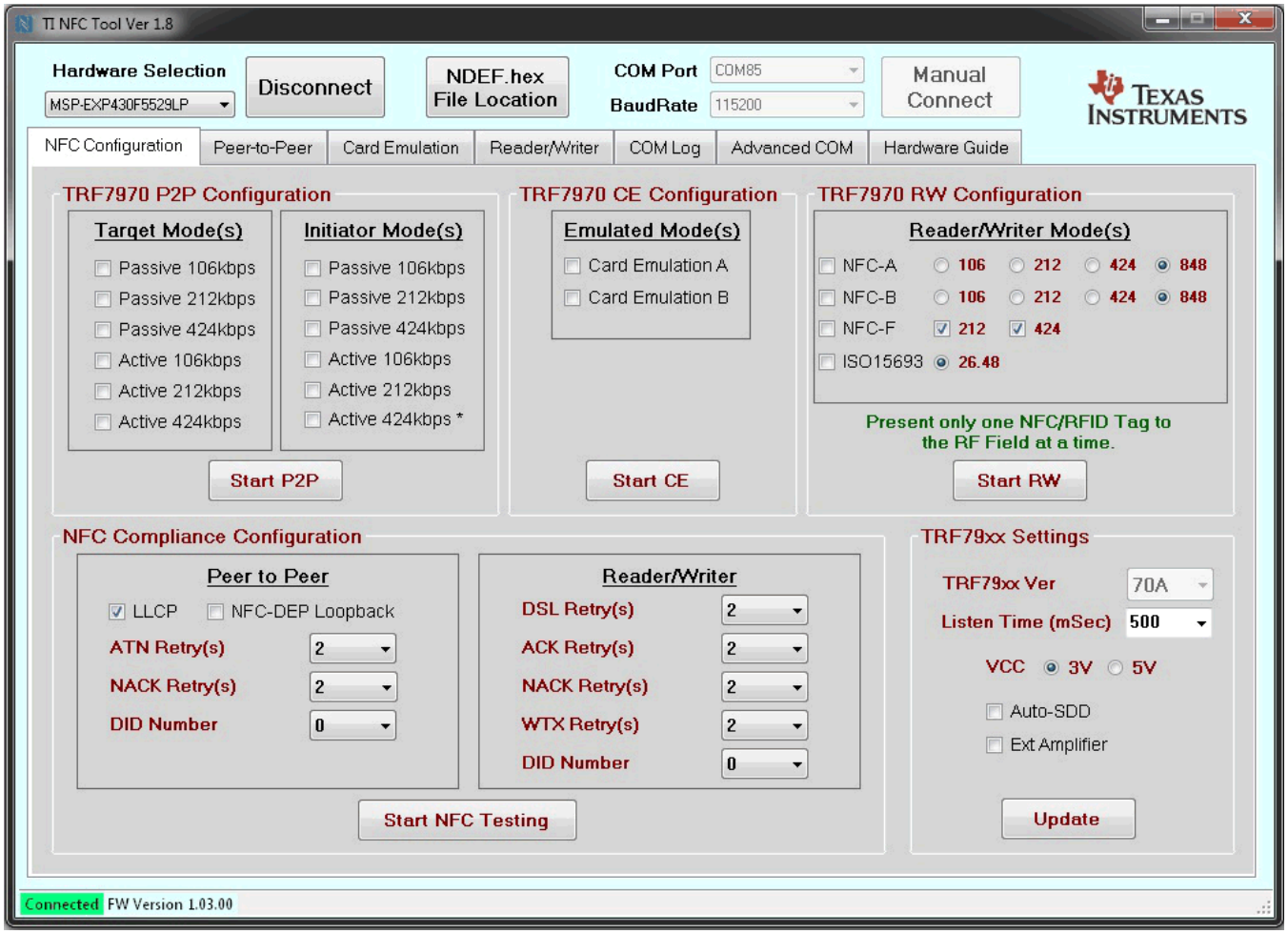

Figure 26. TI NFC Tool GUI

9 Operational Overview

The reader/writer demo on the MSP430F5529 has two modes: a stand-alone mode and a host mode that requires a system (PC typically) to run the TI NFC Tool GUI. The stand-alone mode can be configured for specific bit rates for each supported tag technology: Type 2 and Type 4A at 106, 212, 424, or 848 kbps, Type 3 at 212 or 424 kbps, Type 4B at 106, 212, 424, or 848 kbps, and Type 5 at 26.48 kbps.

The order in which the firmware polls for the different tag technologies is NFC-A (Type 2 or Type 4A), NFC-B (Type 4B), NFC-F (Type 3), and NFC-V (Type 5). If no tags are detected, the polling returns to NFC-A (Type 2 or Type 4A) (see Figure 27 for the switching mechanism while in stand-alone mode). Polling commands for NFC-A and NFC-B must be sent at 106 kbps, and then the bit rate can be increased after activation is completed.

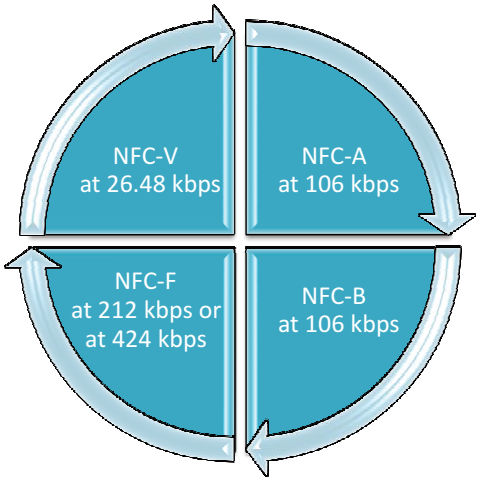

Figure 27. Reader/Writer Tag Polling Mechanism

The second mode requires a host (PC) to run the TI NFC Tool GUI and then connect to either the MSP430F5529 Experimenter Board or LaunchPad development kit through the USB CDC (see Figure 28), or the MSP432P401R LaunchPad development kit through the back channel UART interface (see Figure 29). The GUI allows the user to select which modes to enable/disable. When a connection is established with an NFC-enabled device, the GUI switches to the reader/writer tab, which allows the user to view the tag data and send a customized text or URI RTD for NDEF formatted tags.

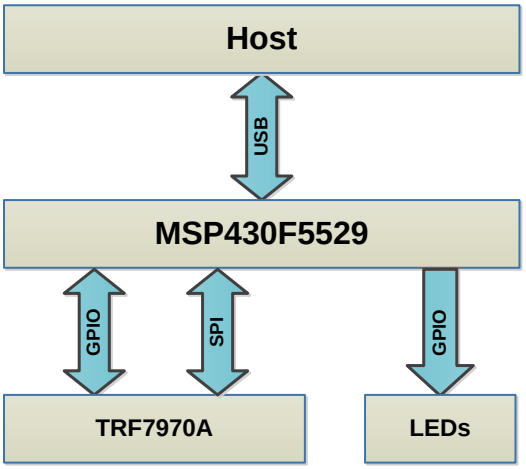

Figure 28. Reader/Writer Demo System Block Diagram for MSP430F5529

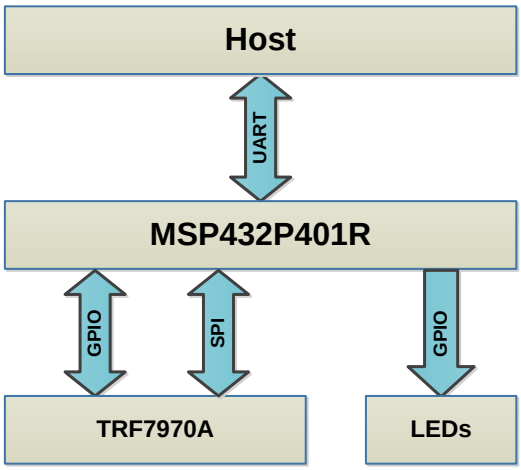

Figure 29. Reader/Writer Demo System Block Diagram for MSP432P401

10 Reader/Writer Interoperability Results

The firmware example provided has been tested for interoperability between the TRF7970A and a multitude of NFC tags existing in the market to ensure that each supported tag type can be activated, selected, and have NDEF data read and written. Use the legend shown in Table 6 for the results in Table 7.

Table 6. Legend for NFC/RFID Tag Support

Legend	
Supported	✓
Not Supported	✗

Table 7 lists the results from the tests that were run to validate the interoperability of the reader/writer firmware stack with various existing tag technology types and NFC devices that are currently in the market. NFC Type 1 tags are not supported as they are not ISO compliant and therefore require the use of Direct Mode 0 to communicate with them. While it is possible from a hardware standpoint to support these tags with the TRF7970A, the reader/writer firmware as provided does not support Type 1 tags.

Table 7. NFC/RFID Tags Supported by Reader/Writer Mode

NFC/RFID Tags	Read	Write
Type 1	✗	✗
Type 2	✓	✓
Type 3	✓	✓
Type 4A	✓	✓
Type 4B	✓	✓
RF430CL33xH	✓	✓
Type 5	✓	✓
RF430FRL15xH	✓	✓
RF37S114	✓	✓
Tag-It HF-I Standard	✓	✓
Tag-It HF-I Pro	✓	✓
Tag-It HF-I Plus	✓	✓

11 Conclusion

Reader/Writer is one of the three modes supported by the TRF7970A transceiver. The existing reader/writer NFC stack supports reading and writing for Type 2, Type 3, Type 4A, Type 4B, and Type 5 tag platforms. The stack has been designed to read a single tag at a time, and does not have anticollision procedures for when multiple tags of the same technology are presented in the field at the same time. If multiple tags of different technologies are presented, only one of those tags is activated.

The reader/writer demo has two modes, a stand-alone mode and a GUI mode. In stand-alone mode the reader/writer must be preconfigured to read tags and indicates successful communication with NFC tag platforms by lighting an LED. The GUI mode enables users to see the contents that are read from tags, including NDEF Text and URI RTDs. It is also possible to write custom NDEF Text or URI RTDs to an NDEF formatted tag.

Based on the test results shown in Table 7, the reader/writer firmware can support all NFC Forum compliant tag platforms except for Type 1.

For applications that use only reader/writer mode, the TRF7964A can be used as a drop-in alternative to the TRF7970A.

As downloaded, the firmware example includes the full TI NFC stack which supports peer-to-peer, card emulation, and reader/writer modes. For applications that do not require all NFC operating modes, there are configuration options available to reduce the NFC stack memory footprint (only compiling required operating modes). These configurations can be made by modifying the #define statements within the *nfc_config.h* file, located at [Install Path]\nfclink\Source\headers.

For more information about NFC peer-to-peer operation, see *NFC active and passive peer-to-peer communication using the TRF7970A*.

For more information about NFC Card Emulation, see *NFC card emulation using the TRF7970A*.

12 References

1. *TRF7970A Multiprotocol Fully Integrated 13.56-MHz RFID and NFC Transceiver IC*
2. *MSP430F551x, MSP430F552x mixed-signal microcontrollers*
3. *MSP432P401R, MSP432P401M mixed-signal microcontrollers*
4. ISO/IEC 7816-4:2005(E) (http://www.ansi.org)
5. ISO/IEC 18092/ECMA-340 (NFCIP – 1) (http://www.ecma-international.org)
6. ISO/IEC 21481/ECMA-352 (NFCIP – 2) (http://www.ecma-international.org)
7. JIS X 6319-4 (http://www.webstore.jsa.or.jp/)
8. ISO/IEC 14443-3:2009(E) (http://www.ansi.org)
9. ISO/IEC 14443-4:2008(E) (http://www.ansi.org)
10. ISO/IEC 15693-2:2006(E) (http://www.ansi.org)
11. ISO/IEC 15693-3:2009(E) (http://www.ansi.org)
12. NFCForum-TS-DigitalProtocol-1.0 (Digital Protocol) (http://www.nfc-forum.org)
13. NFCForum-TS-Activity-1.0 (Activity Protocol) (http://www.nfc-forum.org)
14. NFCForum-TS-RTD_1.0 (NFC Record Type Definition (RTD)) (http://www.nfc-forum.org)
15. NFCForum-TS-RTD_Text_1.0 (Text Record Type Definition) (http://www.nfc-forum.org)
16. NFCForum-TS-RTD_URI_1.0 (URI Record Type Definition) (http://www.nfc-forum.org)
17. NFCForum-TS-NDEF_1.0 (NFC Data Exchange Format (NDEF)) (http://www.nfc-forum.org)
18. NFC Forum T2TOP (Type 2 Tag Operation) (http://www.nfc-forum.org)
19. NFC Forum T3TOP (Type 3 Tag Operation) (http://www.nfc-forum.org)
20. NFC Forum T4TOP (Type 4 Tag Operation) (http://www.nfc-forum.org)
21. NFC Forum Logo (http://nfc-forum.org/our-work/nfc-branding/n-mark/the-n-mark-license/)

Revision History

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

Changes from December 10, 2016 to March 13, 2019	Page
• Removed former Section 6.2 *Experimenter Board Setup* and Section 6.3 *TRF7970ATB Module*	28
• Updated available bundle and link in Section 6.2, *Bundle Available for Purchase*	28

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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