Application Report
SLOA192B–April 2014–Revised March 2019

NFC active and passive peer-to-peer communication using the TRF7970A

Erick Macias, Josh Wyatt

ABSTRACT

Peer-to-peer (P2P) is one of the three modes supported by the TRF7970A. The Near Field Communication (NFC) market is emerging into multiple fields including medical, consumer, retail, industrial, automotive, and smart grid. P2P is very common in these fields, because it allows for a wireless virtual channel to be created between two devices. This application report describes two NFC technologies, the NFC-F and NFC-A protocols, which are used to initiate the communication for peer-to-peer active or passive modes. Furthermore, this application report explains how to implement an application using active or passive P2P communication, supporting baud rates of 106 kbps, 212 kbps and 424 kbps on the TRF7970A transceiver.

Project collateral and source code discussed in this application report can be downloaded from http://www.ti.com/lit/zip/sloa192.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The TRF7970A supports three modes: reader/writer, card emulation, and peer-to-peer. This document describes how to use the TRF7970A in peer-to-peer (P2P) active or passive. P2P requires two NFC-enabled devices to communicate using technologies NFC-A or NFC-F at baud rates of 106 kbps (NFC-A), 212 kbps (NFC-F), or 424 kbps (NFC-F). The transceiver that is initially polling and initiates the communication is the initiator. The transceiver that is initially listening is the target.

The initiator always generates the RF field for both active and passive communication modes. However, the target generates its own RF field only for active mode (while the initiator's RF field is disabled) and load modulates the initiator's RF field in passive mode. After the technology selection for either mode has been completed, the higher layers are the same (as shown in Figure 1). The NDEF Push Protocol (NPP) was commonly used on Gingerbread Android NFC devices as the protocol to push NDEF messages from one NFC device to another; however, the Simple NDEF Exchange Protocol (SNEP) has become the standard on Ice Cream Sandwich Android NFC devices and onward.

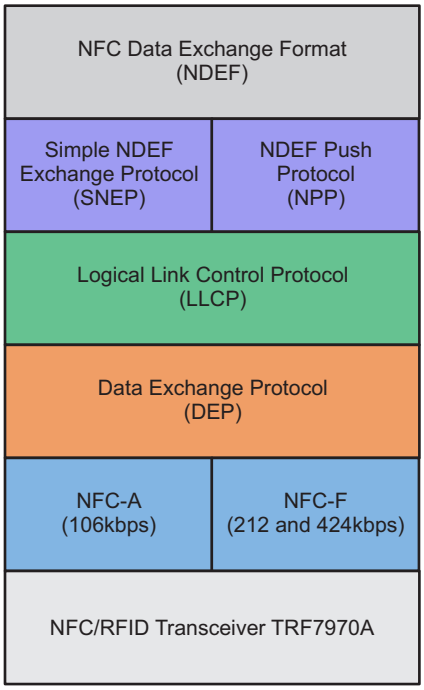

Figure 1. Peer-to-Peer Layers Including the PHY (TRF7970A)

A 16-bit and a 32-bit microcontroller are used to interface with the TRF7970A to demonstrate a reference example of the peer-to-peer mode. The firmware supports flexible functions that let the user enable or disable different peer-to-peer modes and technologies. The firmware supports both initiator and target for active and passive communication at baud rates of 106 kbps, 212 kbps, and 424 kbps. Additionally, the firmware can send and receive NDEF messages from NFC-enabled smart phones with SNEP.

Table 1 lists the NFC-enabled devices used to validate the firmware.

Table 1. NFC Enabled Devices Used to Test Peer-to-Peer

Smartphone Model (Release Date)	Operating System	Kernel Version
Samsung Galaxy Nexus (Nov 2011)	Android 4.3	3.0.72 Jun 7 2013
Samsung Galaxy S3 (T-Mobile) (May 2012)	Android 4.0.4	3.031 Oct 31 2013
Asus Nexus 7 (July 2012)	Android 4.4.2	3.1.10 Nov 20 2013
Samsung Galaxy Note 2 (Sept 2012)	Android 4.3	3.0.31 Nov 19 2013
AU Arrows Fujitsu FJL21 (Oct 2012)	Android 4.0.4	3.0.21 Oct 16 2012
Nokia Lumia 820 (Oct 2012)	Windows 8	8.0.10211.204
HP Elite Tablet (Nov 2012)	Windows 8	Windows 8 Pro
Samsung Nexus 10 (Nov 2012)	Android 4.4.2	3.4.39 Nov 20 2013
Google Nexus 4 (Nov 2012)	Android 4.3	3.4.0 Nov 20 2013
Samsung Galaxy S4 (April 2013)	Android 4.3	3.4.0 Nov 16 2013
Hisense Sero 7 Pro (June 2013)	Android 4.2.1	3.1.10
Asus Nexus 7 (July 2013)	Android 4.4	3.4.0 Dec 11 2013
Google Nexus 5 (Oct 2013)	Android 4.4	3.4.0 Nov 20 2013

2 Initial RF Collision

Various implementations of peer-to-peer exist today; these implementations may poll through commands at different time intervals (500 ms, 300 ms, and so forth). To ensure that two NFC 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, which measures the strength of the demodulated subcarrier signal, before enabling its own RF field. If the RSSI value is greater than 0x00, the transceiver does 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 3 through Figure 5 provide the correlation of the free space distance between two unmodified TRF7970ATB modules and the 3-bit external RSSI value in three directions. For more details on each orientation, see Figure 2. One TRF7970A has its RF field at full power (+23 dBm), and the second TRF7970A is used to take RSSI measurements across different distances.

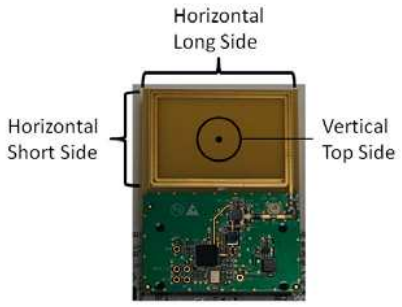

Figure 2. RSSI Level Measurement Orientations

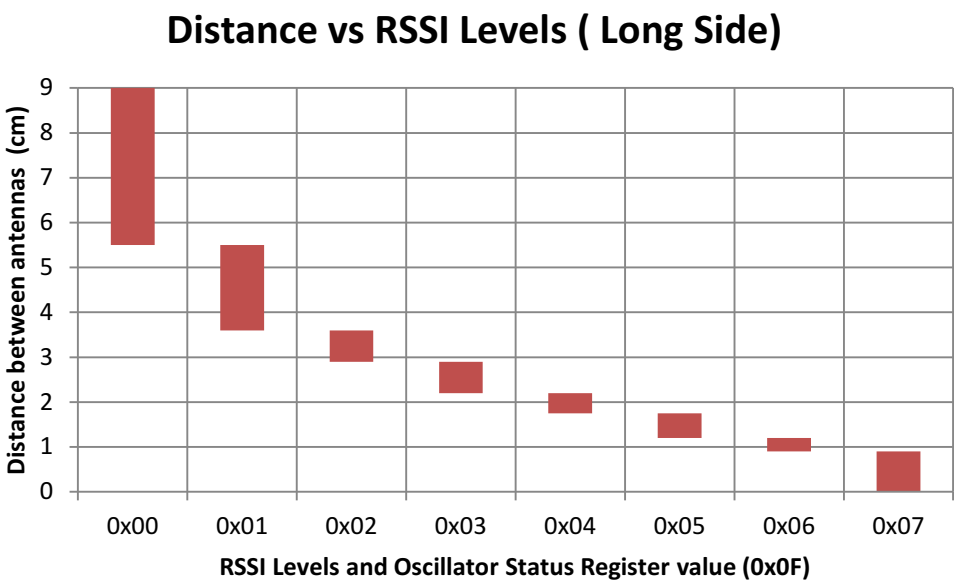

Figure 3. Long Side RSSI Level Measurement

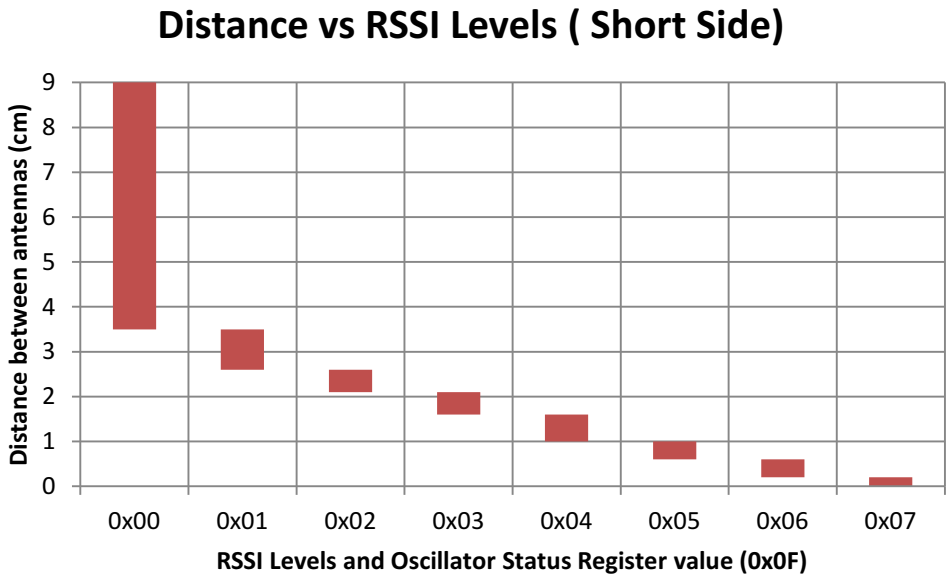

Figure 4. Short Side RSSI Level Measurement

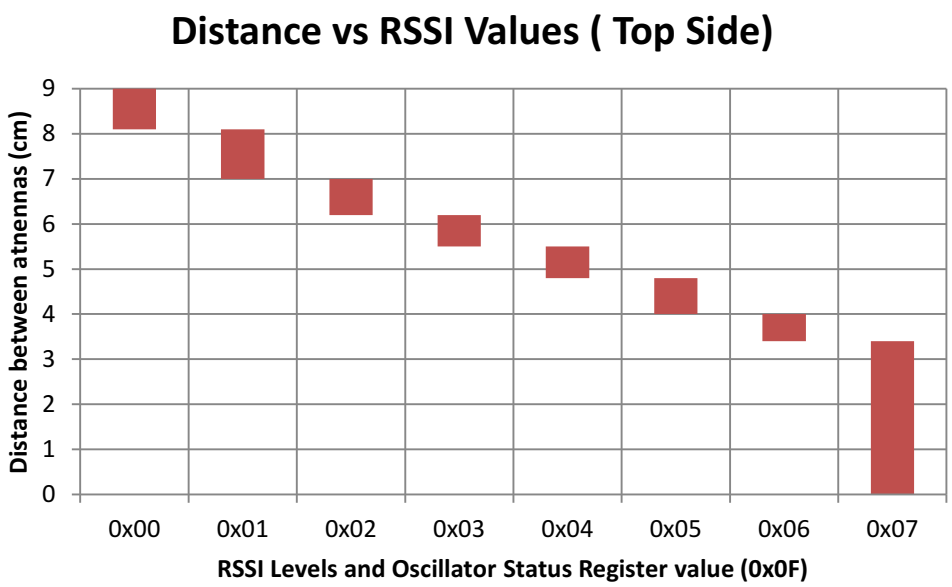

Figure 5. Top-Side RSSI Level Measurement

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

1. Write 0x02 and 0x03 (3-VDC and 5-VDC 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* ms.
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.

3 TRF7970A Register Settings

After powering up the TRF7970A, the MCU must send SOFT_INIT (0x03) and IDLE (0x00) direct commands to enable the passive target mode at 106 kbps. Table 2 shows the default value of registers 0x00 through 0x16 and 0x18 through 0x1C after the commands are issued. The table also shows the registers that must be modified for both target and initiator modes.

The ISO Control (0x01) register is modified whenever the peer-to-peer 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 Modulator and SYS_CLK Control (0x09), RX Special Settings (0x0A), and Regulator and I/O Control (0x0B) registers need to be modified only once, right after initialization. The NFC Low Field Detection Level (0x16) register needs to be modified only for Target mode operation. The NFC Target Detection Level (0x18) register must be modified after initialization for Target mode operation, and also for Initiator mode operation based on the *TRF7970A silicon errata*.

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

Address	Register	Value	Notes	Requires Modification	
				Initiator	Target
0x00	Chip status control	0x01	5-V operation	Yes	Yes
0x01	ISO control	0x21	Passive target	Yes	Yes
0x02	ISO14443B TX options	0x00	Automatic single device detection (SDD) disabled	No	No
0x03	ISO14443A high bit-rate options	0x00	See TRF7970A data sheet	No	No
0x04	TX timer setting, H-byte	0xC1	See TRF7970A data sheet	No	No
0x05	TX timer setting, L-byte	0xC1	See TRF7970A data sheet	No	No
0x06	TX pulse-length control	0x00	See TRF7970A data sheet	No	No
0x07	RX no response wait	0x0E	See TRF7970A data sheet	No	No
0x08	RX wait time	0x07	See TRF7970A data sheet	No	No
0x09	Modulator and SYS_CLK control	0x91	27-MHz crystal enabled, SYS_CLK enabled, OOK (100%)	Yes	Yes
0x0A	RX special setting	0x10	Band-pass 100 kHz to 1.5 MHz	Yes	Yes
0x0B	Regulator and I/O control	0x87	Automatic VDD RF	Yes	Yes
0x0C	IRQ status	0x00	See TRF7970A data sheet	No	No
0x0D	Collision position and interrupt mask	0x3E	See TRF7970A data sheet	No	No
0x0E	Collision position	0x00	See TRF7970A data sheet	No	No
0x0F	RSSI levels and oscillator status	0x40	See TRF7970A data sheet	No	No
0x10	Special function	0x00	See TRF7970A data sheet	No	No
0x11	Special function	0x00	See TRF7970A data sheet	No	No
0x12	RAM	0x00	See TRF7970A data sheet	No	No
0x13	RAM	0x00	See TRF7970A data sheet	No	No
0x14	Adjustable FIFO IRQ levels	0x00	See TRF7970A data sheet	Optional	Optional
0x15	Reserved	0x00	N/A	No	No
0x16	NFC low field detection level	0x00	See TRF7970A data sheet	No	Yes
0x18	NFC target detection level	0x00	See TRF7970A data sheet	Yes	Yes
0x19	NFC target protocol	0x00	See TRF7970A data sheet	No	No
0x1A	Test	0x00	See TRF7970A data sheet	No	No
0x1B	Test	0x00	See TRF7970A data sheet	No	No
0x1C	FIFO status	0x00	See TRF7970A data sheet	No	No

4 Peer-to-Peer at 106 kbps

The TRF7970A supports initiator and target P2P mode at 106 kbps (fc/128). When the transceiver is in default mode [ISO mode (for more information, see the *Direct Mode* section of [1])] only the decoded data is available to the MCU through the FIFO. The frame format for Data Exchange Protocol (DEP) packets at 106 kbps (shown in Figure 6) are based on the NFC-A technology specified in the NFCForum-TS-DigitalProtocol-1.0. The format does not include the Start Byte for the passive anticollision of 106 kbps. This section covers the register settings and anticollision sequence for both the active and passive peer-to-peer modes at 106 kbps.

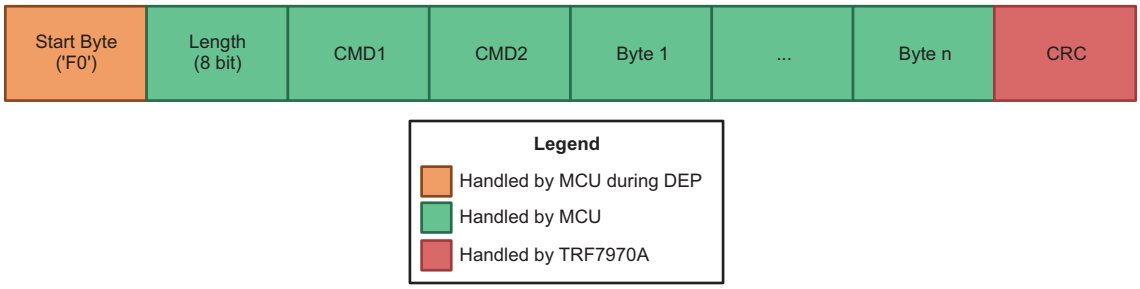

Figure 6. Frame Format for 106 kbps

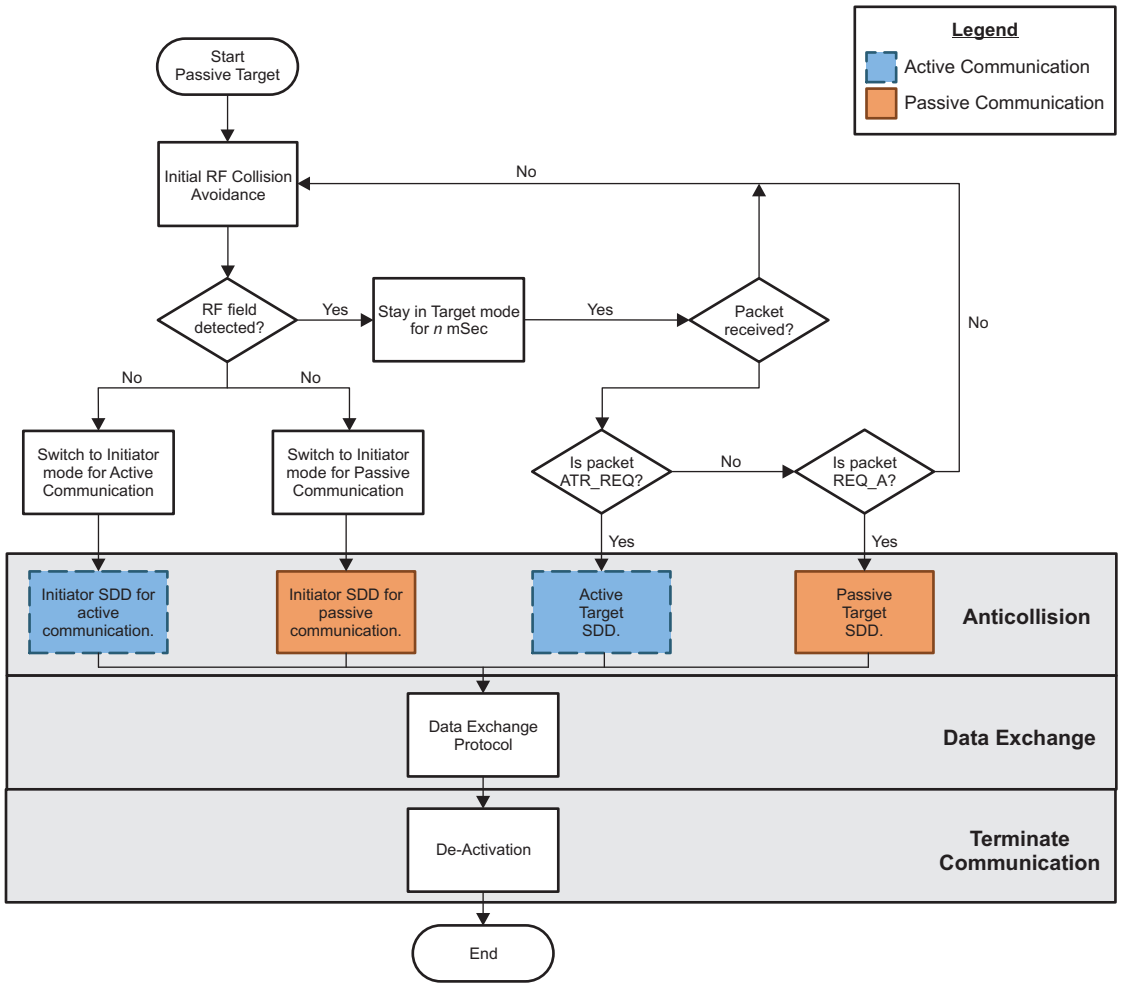

Figure 7. Peer-to-Peer 106-kbps Flow Diagram

4.1 Active Communication

The TRF7970A ISO Control register (0x01) sets the modulation of the transceiver's RF field when is in initiator mode or active NFC target. The NFC initiator and active target for baud rate of 106 kbps can be achieved by initializing the transceiver as a NFC-A RFID reader (0x08). Each time the transceiver finishes sending a command while being an initiator and active Target it must:

1. Turn off the field by modifying the ISO Control register → 0x21 (passive target).
2. Write to the RX Special Setting register (0x0A) → 0x30 (band-pass 450 kHz to 1.5 MHz and band-pass 100 kHz to 1.5 MHz).
3. Write to the Adjustable FIFO IRQ Levels register (0x14) → 0x0F (IRQ triggered when there are 96 bytes in FIFO during RX or 32 bytes during TX).

4.1.1 Initiator

Once the Initial RF Collision avoidance is completed and no RF field has been detected (see Figure 7), the following registers must be modified each time before a DEP_REQ is sent to the active target (see Figure 8). For more information, see Section 2.

1. Delay time specified in the technology specification (56 µs to 188 µs).
2. ISO Control register (0x01) → 0x08 (ISO1443A at 106 kbps, receive with CRC).
3. Delay 1 ms.
4. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with (0x11) or without (0x10) CRC direct command.
   c. TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
   d. Write the command to the FIFO.

4.1.2 Target

After receiving commands from the initiator, the transceiver must follow the next steps each time before responding to the initiator:

1. Delay time specified in the technology specification (56 µs to 188 µs).
2. Perform Response RF Collision avoidance, and ensure no RF field is detected.
3. ISO Control register (0x01) → 0x08 (ISO14443A at 106 kbps, receive with CRC).
4. Delay 1 ms
5. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with (0x11) or without (0x10) CRC direct command.
   c. TX Length Byte 1 and 2 (0x1D and 0x1E) registers.
   d. Write the response to the FIFO.

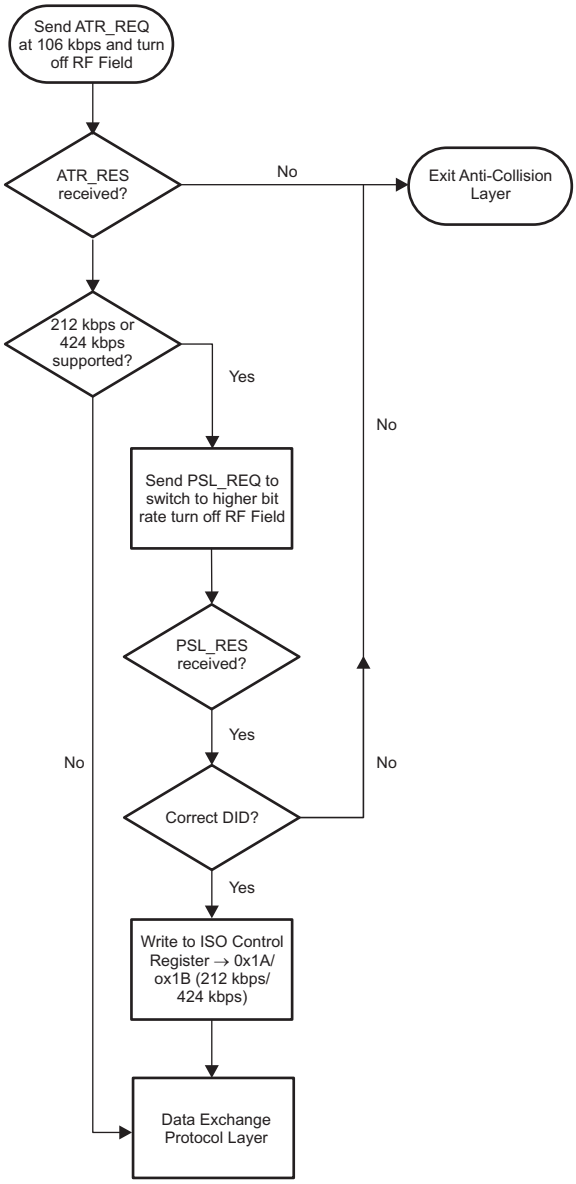

Figure 8. Peer-to-Peer 106-kbps Initiator Anticollision for Active Communication

4.2 Passive Communication

The TRF7970A ISO Control register (0x01) sets the modulation of the transceiver RF field when is in initiator mode and the bit rate at which to load modulate when it is in target mode. Unlike active communication, the target load modulates the initiator RF field instead of modulating its own field. Furthermore, the anticollision procedure is different for passive communication at 106 kbps.

4.2.1 Initiator

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 as shown in Figure 9. For more information, see Section 2.

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

Unlike the active communication, no further RF collisions are required. The ISO control register needs to be modified for the anticollision state to receive without CRC for the required commands. For more information, see the ISO14443-3 specification. Once the anticollision is completed the ISO Control register needs to be modified to receive with CRC. Step 2 must be used to send commands to the passive target.

4.2.2 Target

Initially the TRF7970A must be receiving without CRC. After receiving commands from the initiator, the following registers must be modified before and after the anticollision is completed:

1. ISO Control register (0x01) → 0xA4 (ISO14443A 106 kbps, receive without CRC during anticollision, before Select command)
   or
   ISO Control register (0x01) → 0x24 (ISO14443A 106 kbps, receive with CRC, after anticollision is completed).
2. 2. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission without (0x10, anticollision before Select command) or with (0x11, after anticollision is completed) CRC direct command.
   c. c. TX Length Byte 1 and 2 (0x1D and 0x1E) registers.
   d. Write the command to the FIFO.

The ISO control register needs to be modified for the anticollision state to receive without CRC for the required commands. For more information, see the ISO14443-3 specification. Once the anticollision is completed the ISO Control register needs to be modified to receive with CRC. Step 2 must be used to send commands to the initiator.

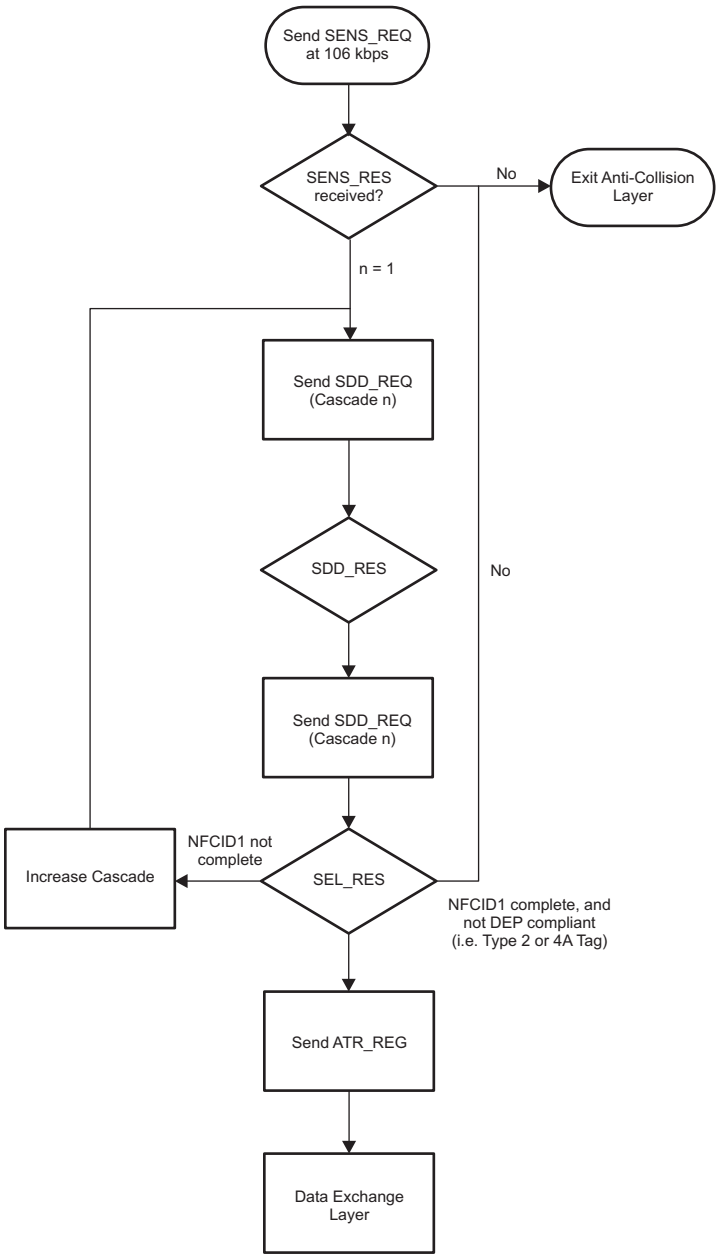

Figure 9. Peer-to-Peer 106-kbps Initiator Anticollision for Passive Communication

5 Peer-to-Peer at 212 kbps and 424 kbps

The TRF7970A supports initiator and target for NFC-F FeliCa™ at 212 kbps (fc/64) and 424 kbps (fc/32). When the transceiver is in default mode [ISO mode (for more information, see the *Direct Mode* section of [1])] the decoded data is available to the MCU through the FIFO. The frame format for 212 kbps and 424 kbps is shown in Figure 10. This section covers the register settings and anticollision sequence for both the active and passive NFC-F peer-to-peer modes.

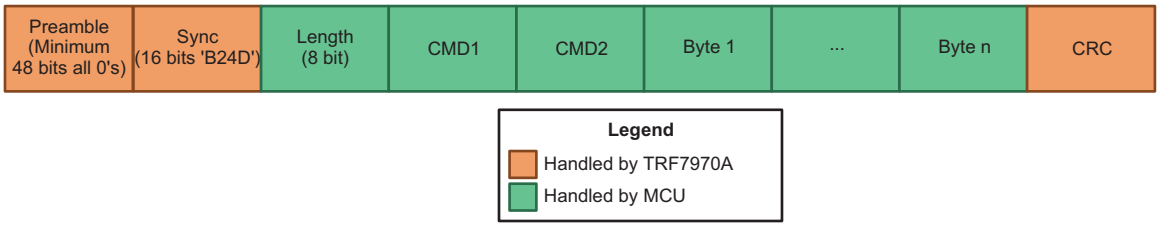

Figure 10. Frame Format for 212 kbps and 424 kbps

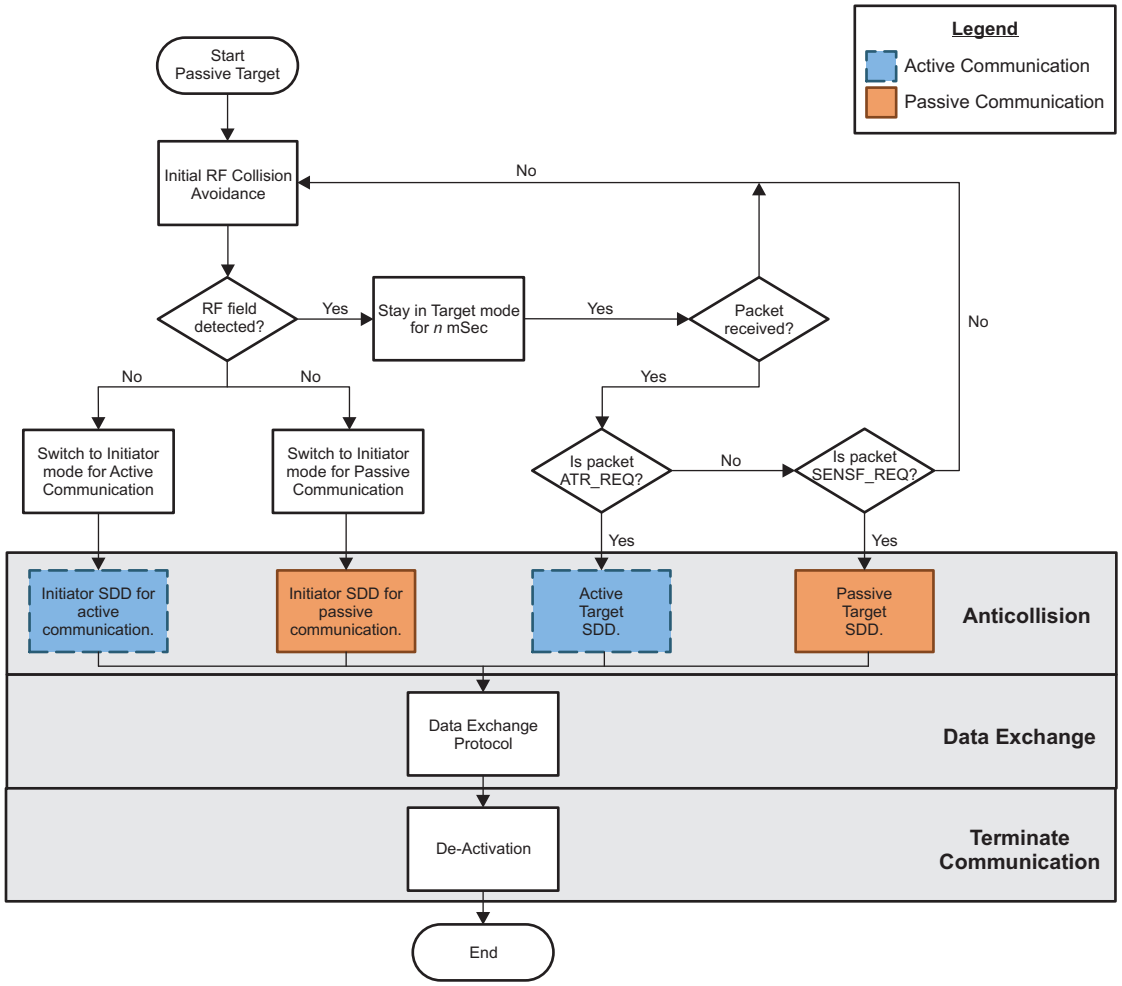

Figure 11. Peer-to-Peer 212-kbps or 424-kbps Initiator Anticollision for Active Communication

5.1 Active Communication

The TRF7970A ISO Control register (0x01) sets the modulation of the transceiver RF field when is in initiator mode and the bit rate to load modulate when it is in target mode. The NFC active and passive initiator (bit 5 = 1 and bit 2 = 1) for baud rates of 212 kbps and 424 kbps, can be achieved by initializing the transceiver as a NFC-F reader (Bit 5 = 1 and bits 4-0 = 0x12 (212 kbps) or 0x13 (424 kbps)). Each time the transceiver finishes sending a command while being an initiator or target, the transceiver must:

1. Turn off the field by modifying the ISO Control register → 0x21 (passive target).
2. Write to RX Special Setting register (0x0A) → 0x80 (band-pass filter set for 110 kHz to 570 kHz)
3. Write to Adjustable FIFO IRQ Levels register (0x14) → 0x0F (IRQ triggered when there are 96 bytes in FIFO during RX or 32 bytes during TX)

5.1.1 Initiator

Once the Initial RF Collision is completed and no RF field has been detected (as shown in Figure 11), the following registers must be modified to initialize the anticollision (as shown in Figure 12). For more information, see Section 2.

1. Delay time specified in the technology specification (56 µs to 188 µs).
2. ISO Control register (0x01) → 0x32 (NFC-F at 212 kbps, receive with CRC).
3. Delay 1 ms
4. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with CRC direct command (0x11).
   c. TX Length Byte 1 and 2 (0x1D and 0x1E) registers.
   d. Write the response to the FIFO.

5.1.2 Target

After receiving commands from the initiator, the transceiver must follow the next steps each time before responding to the initiator:

1. Delay time specified in the technology specification (56 µs to 188 µs).
2. Perform Response RF Collision avoidance, and ensure no RF field is detected.
3. ISO Control register (0x01) → 0x32 (NFC-F at 212 kbps).
4. Delay 1 ms
5. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with CRC direct command (0x11).
   c. TX Length Byte 1 and 2 (0x1D and 0x1E) registers.
   d. Write the response to the FIFO.

NOTE: For more details on the NFC-F protocol, see [2], [3], [6], [7] and [8].

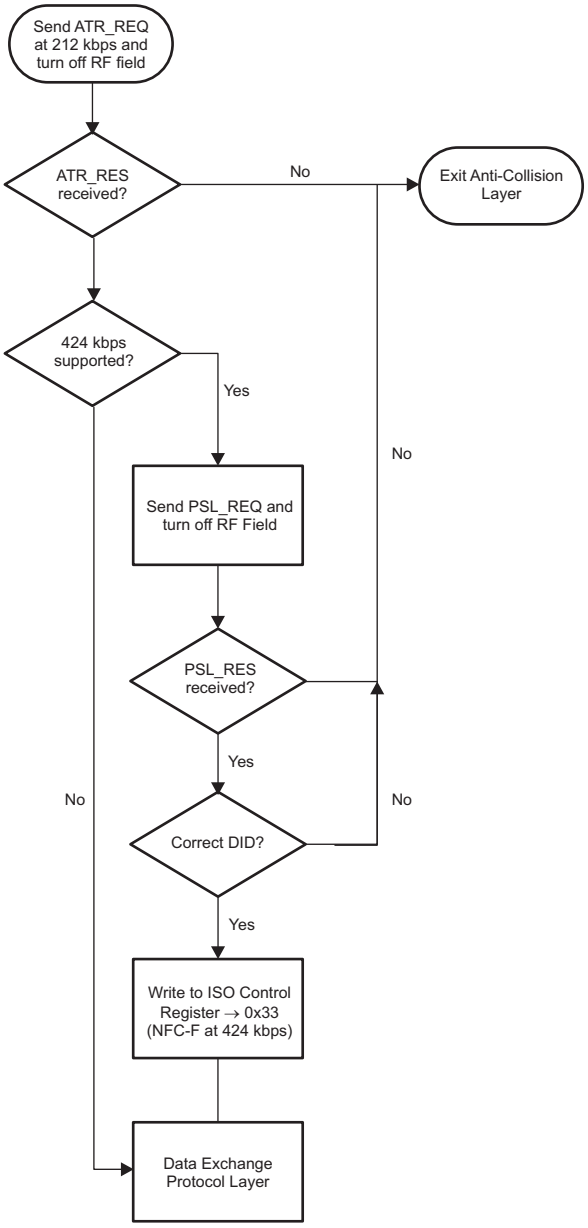

Figure 12. Peer-to-Peer 212-kbps Active Initiator Anticollision

5.2 Passive Communication

The TRF7970A ISO Control register (0x01) sets the modulation of the transceiver RF field when it is in initiator mode and the bit rate at which to load modulate the initiator's field when it is in target mode. Unlike active communication, the target load modulates the initiator RF field instead of modulating its own field. Furthermore, the anticollision procedure is different for passive communication and active communication.

5.2.1 Initiator

Once the Initial RF Collision is completed and no RF field has been detected (as shown in Figure 11), the following registers must be modified before the anticollision. For more information, see Section 2).

1. ISO Control register (0x01) → 0x32 (NFC-F at 212 kbps, receive with CRC).
2. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with CRC direct command (0x11).
   c. TX Length Byte 1 (0x1D) and TX Length Byte 2 (0x1E) registers.
   d. Write the response to the FIFO.

Step 2 must be used to send commands to the passive target.

5.2.2 Target

After receiving commands from the initiator, the transceiver must be modified when the anticollision process is started:

1. ISO Control register (0x01) → 0x22 (NFC-F at 212 kbps, receive with CRC).
2. Send packet:
   a. Reset FIFO (0x0F) direct command.
   b. Transmission with CRC direct command (0x11).
   c. TX Length Byte 1 and 2 (0x1D and 0x1E) registers.
   d. Write the response to the FIFO.

Step 2 must be used to send commands to the passive target.

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 13 and Figure 14).

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 13).

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

Figure 13. 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 14).

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

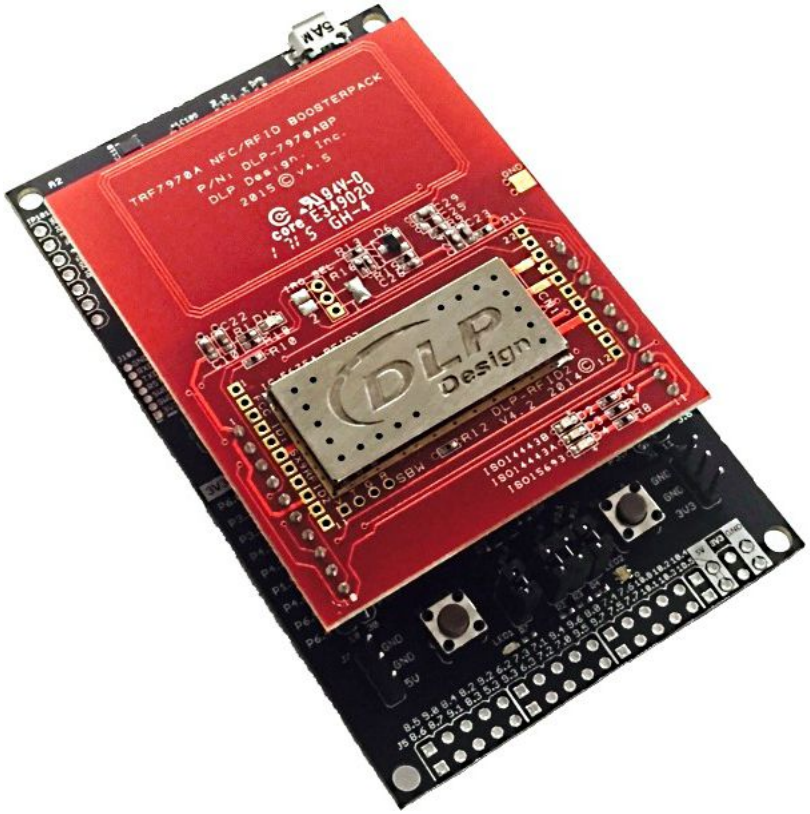

Figure 14. MSP432P401R LaunchPad Development Kit and DLP-7970ABP BoosterPack Plug-in Module

6.2 Bundle Available for Purchase

The TI store offers this bundle:

MSP-EXP430F5529LP and DLP-7970ABP

7 Passive and Active Peer-to-Peer Firmware Example

This section explains which APIs are used by the NFC/RFID layer (see Figure 15) to initialize and handle the peer-to-peer communication. Furthermore, it covers how to implement a sample peer-to-peer application, which can send and receive an NDEF message to and from an NFC-enabled device.

The firmware example that contains the peer-to-peer APIs discussed in this document can be downloaded from http://www.ti.com/lit/zip/sloa192.

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.

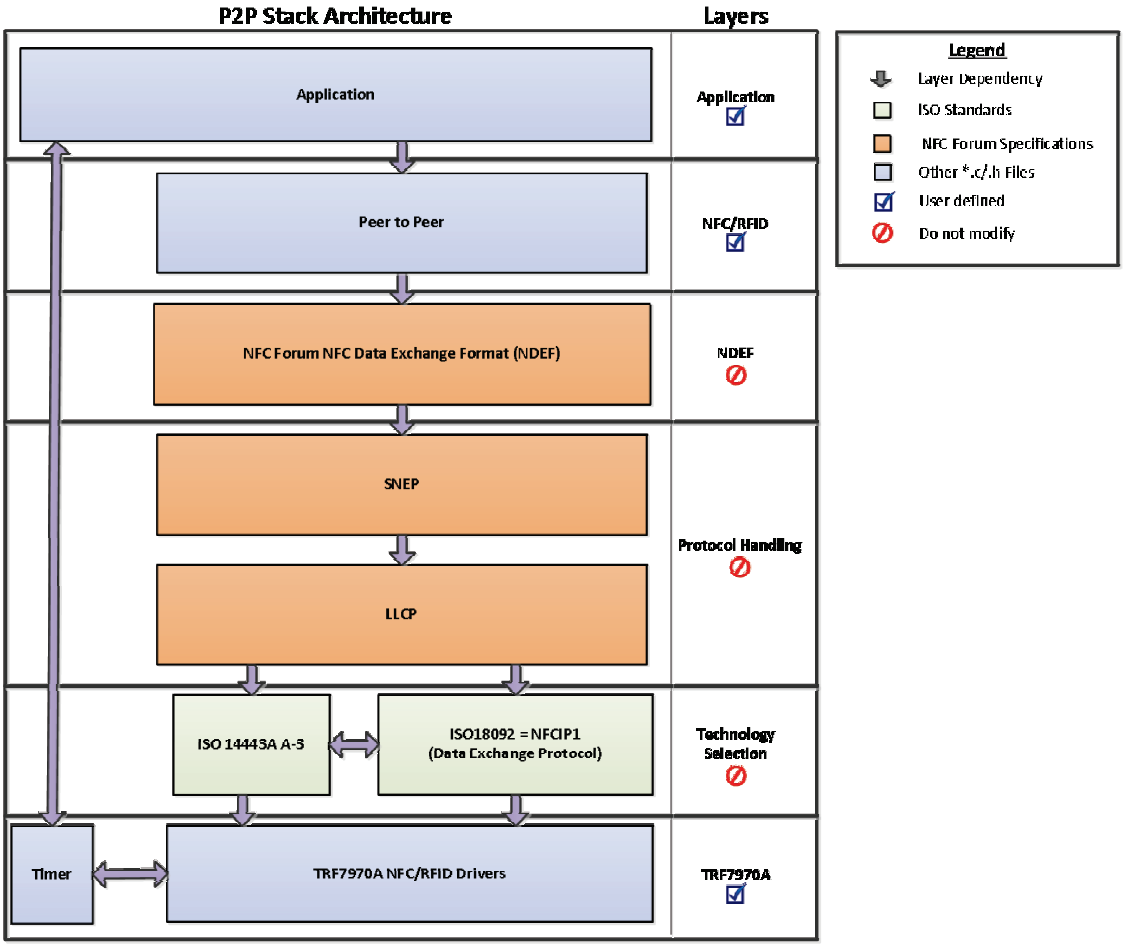

Figure 15. Peer-to-Peer NFC Stack Architecture

7.1 Peer-to-Peer APIs

For details on all available APIs used in the provided example firmware for NFC peer-to-peer 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 peer-to-peer functionality, and each function to help users with developing custom NFC peer-to-peer applications.

7.2 Implementing a Peer-to-Peer Sample Application

This section explains how to implement a peer-to-peer 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 list 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 MSP432P401R LaunchPad development kit.

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 defaults to P2.2 for DLP-7970ABP v4.5 and newer (see the *DLP-7970ABP hardware update overview*).
(2) Pin is needed for only Special Direct mode.

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 and MSP-EXP432P401R LaunchPad Kit Hardware Connections

DLP-7970ABP Pins	MSP432P401R LaunchPad Development Kit Pins
TRF7970A EN1	P6.2
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 only Special Direct mode.

7.2.1 Low-Level Initialization

For the low-level initialization, the MCU is initialized in *MCU_init()* by setting the MSP430F5529 main clock frequency 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()*. The local variables are needed for transmit and receive and the peer-to-peer stack initialization. The *Buttons_init* function set the GPIO direction to inputs and the *Buttons_interruptEnable()* enable the interrupt for buttons S1 and S2.

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

uint16_t g_ui16ListenTime;
t_sNfcP2PMode g_sP2PSupportedModes;
t_sNfcP2PCommBitrate g_sP2PSupportedTargetBitrates;
t_sNfcP2PCommBitrate g_sP2PSupportedInitiatorBitrates;
t_sNfcDEP_P2PSetup g_sP2PSetupOptions;
uint8_t g_ui8NfcDepInitiatorDID;

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

     uint32_t ui32PacketRemaining;
     uint8_t ui8TXBytes;
     uint16_t ui16TxIndex;
     uint32_t ui32PacketLength;
     uint8_t * pui8NdefPointer;
     uint8_t ui8FragmentSize;

     // Peer to peer RX Status
     tNfcP2PRxStatus sP2PRxStatus;
     t_sNfcP2PMode sP2PMode;
     t_sNfcP2PCommBitrate sP2PBitrate;

     // Bytes Received from Peer to Peer
     uint16_t ui16BytesReceived = 0x00;

     // Initialize MCU
     MCU_init();

     //Enable interrupts globally
```

```
__enable_interrupt();

// Initialize TRF7970
TRF79x0_init();

// Initialize S1 and S2 buttons
Buttons_init(BUTTON_ALL);
Buttons_interruptEnable(BUTTON_ALL);

// Initialize TRF7970A Idle Mode
TRF79x0_idleMode();

// Initialize the NFC Controller
NFC_init();

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

// Initialize IDs for NFC-A, NFC-B and NFC-F
NFC_initIDs();
```

7.2.2 Peer-to-Peer NFC Stack Setup

The peer-to-peer NFC stack is initialized by setting the bInitiatorEnabled or bTargetEnabled bits inside the sP2PSupportedModes variable. If both bInitiatorEnabled and bTargetEnabled are set, the stack operates in a switching mechanism similar to how NFC enabled handsets operate when NFC is enabled. For this demo, all passive target and initiator bitrates are enabled.

```
// Enable Peer 2 Peer Supported Modes
g_sP2PSupportedModes.bits.bTargetEnabled = 1;
g_sP2PSupportedModes.bits.bInitiatorEnabled = 1;

// Set P2P Supported Bit Rates - Target mode
g_sP2PSupportedTargetBitrates.bits.bPassive106kbps = 1;
g_sP2PSupportedTargetBitrates.bits.bPassive212kbps = 1;
g_sP2PSupportedTargetBitrates.bits.bPassive424kbps = 1;
g_sP2PSupportedTargetBitrates.bits.bActive106kbps = 0;
g_sP2PSupportedTargetBitrates.bits.bActive212kbps = 0;
g_sP2PSupportedTargetBitrates.bits.bActive424kbps = 0;

// Set P2P Supported Bit Rates - Initiator mode
g_sP2PSupportedInitiatorBitrates.bits.bPassive106kbps = 1;
g_sP2PSupportedInitiatorBitrates.bits.bPassive212kbps = 1;
g_sP2PSupportedInitiatorBitrates.bits.bPassive424kbps = 1;
g_sP2PSupportedInitiatorBitrates.bits.bActive106kbps = 0;
g_sP2PSupportedInitiatorBitrates.bits.bActive212kbps = 0;
g_sP2PSupportedInitiatorBitrates.bits.bActive424kbps = 0;

// Configure Peer 2 Peer functions for the correct modes and communication bitrates
NFC_P2P_configure(g_sP2PSupportedModes,g_sP2PSupportedTargetBitrates,g_sP2PSupportedInitiatorBitra
tes);
```

7.2.3 Sending NDEF Packets

The NFC_run switches between target and initiator until an NFC-enabled device is presented to the TRF7970A RF field. Once the peer-to-peer communication is established and eTempNFCState is NFC_DATA_EXCHANGE_PROTOCOL, the firmware checks whether button S1 or button S2 has been pressed. When button S1 is pressed, a text RTD is queued to the SNEP TX buffer. When button S2 is pressed, a MIME RTD is queued to the SNEP TX buffer. When the size of the NDEF packet is smaller than the SNEP queue, the *NFC_P2P_sendNdefPacket()* is only executed once because the number of remaining bytes return by the function would be 0. When the size of the NDEF packet is larger than the SNEP queue, the *NFC_P2P_sendNdefPacket()* continues to queue fragments of the complete packet until the number of remaining bytes is 0.

```
eTempNFCState = NFC_run();
if(eTempNFCState == NFC_DATA_EXCHANGE_PROTOCOL)
{
   else if(NFC_P2P_getModeStatus(&sP2PMode,&sP2PBitrate))
   {
    // Check for button input
    if ((buttonsPressed & BUTTON_S1) && (buttonDebounce == 2))
    {
        ui16TxIndex = 0x00;
        buttonDebounce = 0x00;
        // Total Length of the packet.
        ui32PacketLength = 46;
        ui32PacketRemaining = ui32PacketLength;
        // Send Text String
        pui8NdefPointer = (uint8_t *) (pui8NfcPoweredByTexasInstruments+2);
        if(ui32PacketRemaining < LLCP_MIU)
        {
            ui8FragmentSize = (uint8_t) ui32PacketRemaining;
        }
        else
        {
            ui8FragmentSize = LLCP_MIU;
        }
        ui8TXBytes =
NFC_P2P_sendNdefPacket(pui8NdefPointer,true,ui8FragmentSize,ui32PacketLength);

        if(ui8TXBytes)
        {
            ui32PacketRemaining = ui32PacketRemaining - (uint16_t) (ui8TXBytes);
            ui16TxIndex = ui16TxIndex + (uint16_t) ui8TXBytes;

            // Toggle TX LED
            NFC_TX_LED_POUT ^= NFC_TX_LED_BIT;
        }
    }
    else if((buttonsPressed & BUTTON_S2) && (buttonDebounce == 2))
    {
        ui16TxIndex = 0x00;
        buttonDebounce = 0x00;
        // Total Length of the packet.
        ui32PacketLength = 3597;
        ui32PacketRemaining = ui32PacketLength;
        // Send TI Logo
        pui8NdefPointer = (uint8_t *) (pui8TiLogo + 2);
        if(ui32PacketRemaining < LLCP_MIU)
        {
            ui8FragmentSize = (uint8_t) ui32PacketRemaining;
        }
        else
        {
            ui8FragmentSize = LLCP_MIU;
        }
        ui8TXBytes =
NFC_P2P_sendNdefPacket(pui8NdefPointer,true,ui8FragmentSize,ui32PacketLength);
```

```
            if(ui8TXBytes)
            {
                ui32PacketRemaining = ui32PacketRemaining - (uint16_t) (ui8TXBytes);
                ui16TxIndex = ui16TxIndex + (uint16_t) ui8TXBytes;

                // Toggle TX LED
                NFC_TX_LED_POUT ^= NFC_TX_LED_BIT;
            }
        }
        else if(ui32PacketRemaining > 0)
        {
            if(ui32PacketRemaining < LLCP_MIU)
            {
                ui8FragmentSize = (uint8_t) ui32PacketRemaining;
            }
            else
            {
                ui8FragmentSize = LLCP_MIU;
            }
            ui8TXBytes = NFC_P2P_sendNdefPacket((uint8_t *)
    (pui8NdefPointer+ui16TxIndex),false,ui8FragmentSize,(uint32_t) ui32PacketLength);

            if(ui8TXBytes)
            {
                ui32PacketRemaining = ui32PacketRemaining - (uint16_t) (ui8TXBytes);
                ui16TxIndex = ui16TxIndex + (uint16_t) ui8TXBytes;

                // Toggle TX LED
                NFC_TX_LED_POUT ^= NFC_TX_LED_BIT;
            }
        }
        else if(buttonDebounce == 0x00)
        {
            // Enable the button debounce.
            buttonDebounce = 0x01;
        }
        else if (ui32PacketRemaining == 0)
        {
            // Clear TX LED
            NFC_TX_LED_POUT &= ~NFC_TX_LED_BIT;
        }
       }
    }
```

7.2.4 Receiving NDEF Packets

The *NFC_run* function switches between target and initiator until an NFC-enabled device is presented to the TRF7970A RF field. Once the peer-to-peer communication is established and eTempNFCState is NFC_DATA_EXCHANGE_PROTOCOL, the firmware checks the RX status with *NFC_P2P_getReceiveState()* . When a packet is received, the sP2PRxStatus.sDataReceivedStatus will be RECEIVED_FIRST_FRAGMENT, or RECEIVED_N_FRAGMENT, or RECEIVED_FRAGMENT_COMPLETED. When the first fragment is received from another NFC device if the packet is incomplete, the sP2PRxStatus.sDataReceivedStatus will be RECEIVED_FIRST_FRAGMENT. When the first fragment is received from another NFC device if the packet is complete, the sP2PRxStatus.sDataReceivedStatus will be RECEIVED_FRAGMENT_COMPLETED. When the packet is broken into multiple fragments due to the total size being larger than the SNEP queue the sP2PRxStatus.sDataReceivedStatus will be RECEIVED_N_FRAGMENT until the last fragment is received. When the last fragment is received, the sP2PRxStatus.sDataReceivedStatus will be RECEIVED_FRAGMENT_COMPLETED.

```
eTempNFCState = NFC_run();
if(eTempNFCState == NFC_DATA_EXCHANGE_PROTOCOL)
{
     else if(NFC_P2P_getModeStatus(&sP2PMode,&sP2PBitrate))
     {
          //
          // Read the receive status structure -
 check if there is a received packet from the Target
          //
          sP2PRxStatus = NFC_P2P_getReceiveState();
          if(sP2PRxStatus.sDataReceivedStatus != RECEIVED_NO_FRAGMENT)
          {
               ui16BytesReceived = sP2PRxStatus.ui16DataReceivedLength + ui16BytesReceived;

               // Check if the last packet was received completely
               if((uint16_t) sP2PRxStatus.ui32PacketSize == ui16BytesReceived)
               {
                    // Reset Bytes received
                    ui16BytesReceived = 0;
               }
          }
     }
}
```

8 Quick Start Guide

The *NFCLink standalone getting started guide* provides complete details of how to get started with the provided example firmware and the MSP-EXP430F5529LP and DLP-7970ABP.

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 16), which is installed with the firmware package.

The TI NFC Tool allows for quick configuration of the different NFC modes and provides an interface to send and receive data with NFC-enabled devices.

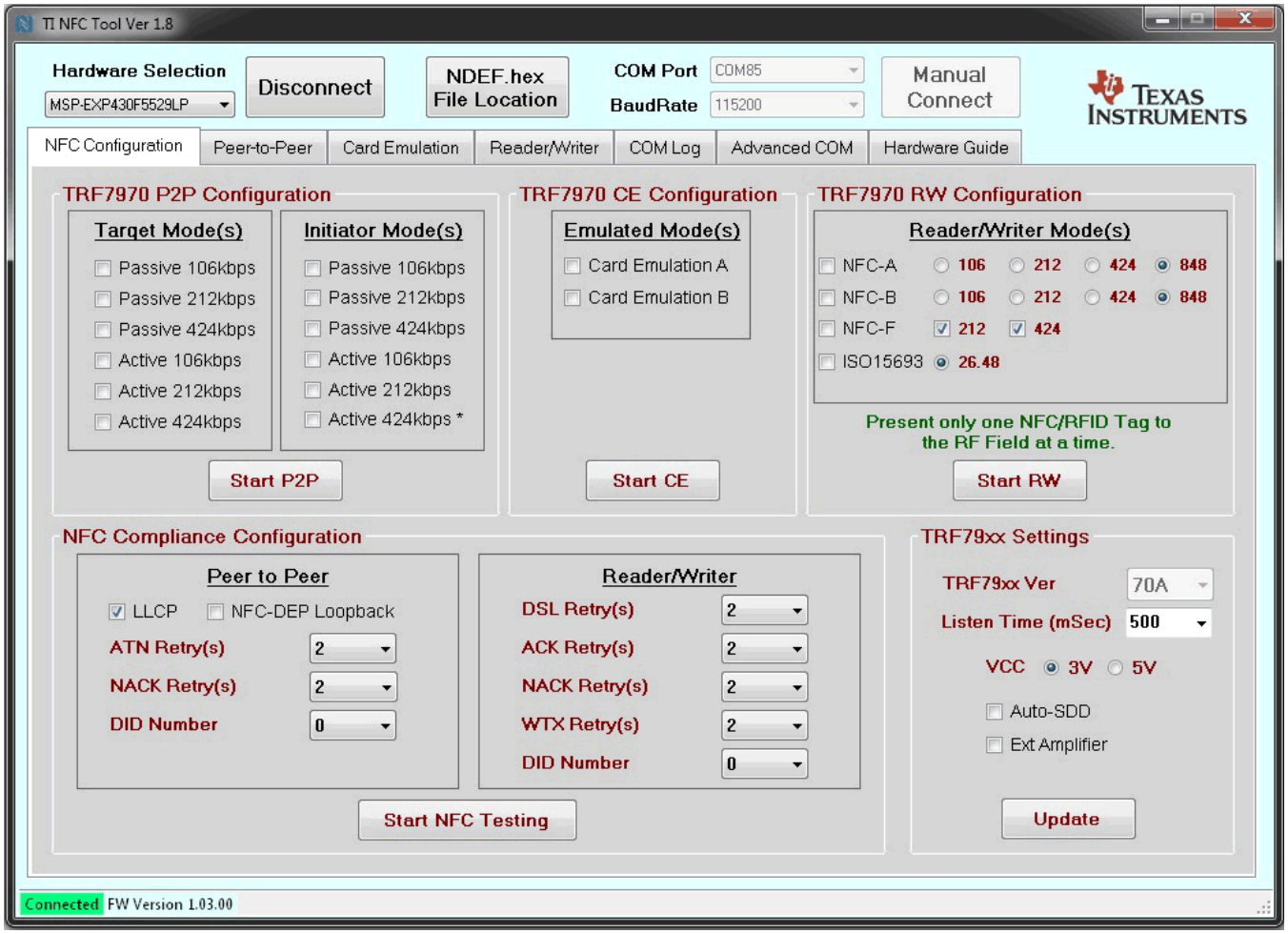

Figure 16. TI NFC Tool GUI

9 Operational Overview

The peer-to-peer demo on the MSP430F5529 has two modes. The first mode is a stand-alone mode where both initiator and target are enabled for passive supporting 106 kbps, 212 kbps and 424 kbps. The firmware sends polling commands for NFC-A Passive at 106 kbps and NFC-F passive at 212 kbps each time the initiator mode is enabled. If all the polling commands timeout, the peer-to-peer stack switches the mode to target mode. In target mode, the firmware waits for a technology to be activated for 500 ms. If no technology is activated during target mode, the firmware switches the mode to initiator mode. When a connection is established with an NFC-enabled device, press the button S1 to send a Text RTD or press the button S2 to send a MIME RTD (TI Logo).

The second mode requires a host (PC) to run the TI NFC Tool and connect to the MSP430F5529 through the USB CDC. Figure 17 shows the system block diagram. When the GUI is connected to the MSP430F5529, pressing the buttons S1 or S2 will not send an NDEF as it previously did in stand-alone mode. The GUI allows you to select which modes to enable and disable. Once a connection is established with an NFC-enabled device, the GUI switches to the peer-to-peer tab, which allows you to send a modifiable Text or URI RTD.

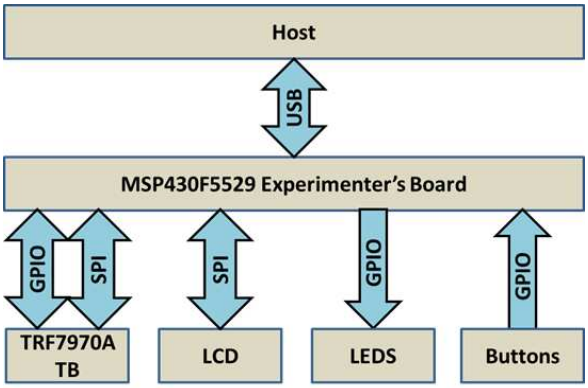

Figure 17. Peer-to-Peer Demo System Block Diagram

10 Peer-to-Peer Interoperability Results

This section covers the results of the interoperability between the existing TRF7970A peer-to-peer stack and the list of NFC-enabled devices mentioned earlier in the application report (see Table 1). Use the legend shown in Table 6 for the Target results in Table 7 and for the Initiator results in Table 8.

Based on the results of the interoperability tests, TI recommends using passive peer-to-peer modes as they have proven to be more consistent, reliable, and robust than active peer-to-peer modes.

Table 6. Legend for the Result of the NFC Enabled Devices Tests

Legend	
✓	Success
✓1	The P2P communication only works if the TRF7970A is polling for both NFC-A and NFC-F. The field cannot be disabled between passive commands.
✓2	The P2P communication only works when the initiator sends a PSL_REQ from 106 to 212 or 424 kbps.
✓3	Requires both Initiator Active Polling commands for 106 kbps and 212 kbps to work properly.
✓4	Difficult to establish an active P2P link.
☒1	The device does not support NFC DEP
☒NRP	No Reply to Polling command - The target does not reply to the polling command.
☒NRT	No Reply to Target response - The initiator does not reply to the polling response command.
⌘	Not Applicable

Table 7 includes the interoperability results for test cases where the TRF7970A is a passive and active target. The results show that legacy NFC-enabled devices did not support sending a PSL REQ, instead they would poll for 106 kbps, 212 kbps, and 424 kbps separately. The newer devices go through the anticollision loop and immediately send a PSL_REQ up to 424 kbps for both active and passive communication at 106 kbps or 212 kbps up to 424 kbps. The smartphone devices are listed with model and release date to illustrate that the interoperability across all target modes is better for newer NFC-enabled devices. The TRF7970A peer-to-peer stack is replying to incoming commands SENS_REQ (polling at 106 kbps), and ATR_REQ (active polling at 106 kbps, or 212 kbps, or 424 kbps); however, no further commands are received from the initiator.

Table 7. TRF7970A and Smart Phone Interoperability Results (Target Mode)

Test With Firmware 1.00.35	TRF7970A Modes and Bitrates (kbps)						
Smartphone Model (Release Date)	Target (Passive Communication)			Target (Active Communication)			TRF7970A Received PSL REQ
	106	212	424	106	212	424	
Samsung Galaxy Nexus (Nov 2011)	✓	✓	✓	⌘	⌘	✓	No
Blackberry Curve 9360 (August 2011)	☒NRT	✓	✓	⌘	⌘	⌘	Yes
Samsung Galaxy S3 (T-Mobile) (May 2012)	✓	✓	✓	⌘	⌘	✓	No
Asus Nexus 7 (July 2012)	✓	✓	✓	⌘	⌘	✓	No
Samsung Galaxy Note 2 (Sept 2012)	✓	✓	✓	⌘	⌘	✓	No
AU Arrows Fujitsu FJL21 (Oct 2012)	✓	✓	✓	⌘	⌘	⌘	No
Samsung S3 Mini (October 2012)	☒NRT	✓	✓	☒NRT	✓	✓	No
Nokia Lumia 820 (Oct 2012)	✓	✓	✓	✓	⌘	⌘	No
HP Elite Tablet (Nov 2012)	☒NRT	✓	✓	✓	⌘	⌘	No
Samsung Nexus 10 (Nov 2012)	✓	✓	✓	✓	✓	✓	Yes
Google Nexus 4 (Nov 2012)	✓	✓	✓	✓	✓	✓	Yes
Samsung Galaxy S4 T-Mobile (April 2013)	✓	✓	⌘	✓	✓	⌘	No
Samsung Galaxy S4 (April 2013)	✓	✓	✓	✓	✓	✓	Yes
Hisense Sero 7 Pro (June 2013)	✓	✓	✓	✓	✓	✓	Yes
Asus Nexus 7 (July 2013)	☒NRT	✓	✓	✓	✓	✓	Yes
Samsung Galaxy Note 3 (Sept. 2013)	☒NRT	✓	✓	⌘	⌘	⌘	No
Google Nexus 5 (Oct 2013)	✓	✓	✓	✓	✓	✓	Yes

Table 8 includes the interoperability results for test cases where the TRF7970A is a passive and active initiator. The smartphone devices are listed with model and release date included to illustrate that the interoperability across all initiator modes is better for newer NFC-enabled devices. The TRF7970A peer-to-peer stack does not get a response from the devices that are failing for the SENS_REQ (polling at 106 kbps), and ATR_REQ (active polling at 106 kbps or 212 kbps). The peer-to-peer stack does not support incrementing from 106 kbps → 212 kbps or 424 kbps for both active and passive communication.

Table 8. TRF7970A and Smart Phone Interoperability Results (Initiator Mode)

Test With Firmware 1.00.35	TRF7970A Modes and Bitrates (kbps)					
Smartphone Model (Release Date)	Initiator (Passive Communication)			Initiator (Active Communication)		
	106	212	424	106	212	424
Samsung Galaxy Nexus (Nov 2011)	☒1	✓	✓	✓	✓	✓
Blackberry Curve 9360 (August 2011)	✓	✓	✓	☒NRP	☒NRP	☒NRP
Samsung Galaxy S3 (T-Mobile) (May 2012)	☒NRP	✓1	✓1	☒NRP	☒NRP	☒NRP
Asus Nexus 7 (July 2012)	☒NRP	✓1	✓1	✓	✓2	✓2
Samsung Galaxy Note 2 (Sept 2012)	☒NRP	✓1	✓1	☒NRP	☒NRP	☒NRP
AU Arrows Fujitsu FJL21 (Oct 2012)	✓	✓2	✓2	☒NRP	☒NRP	☒NRP
Samsung S3 Mini (October 2012)	☒NRP	✓	✓	☒NRP	✓	✓
Nokia Lumia 820 (Oct 2012)	☒NRP	✓1	✓1	✓	✓2	✓2
HP Elite Tablet (Nov 2012)	☒NRP	✓	✓	✓	✓	✓
Samsung Nexus 10 (Nov 2012)	✓	✓	✓	✓	✓2	✓2
Google Nexus 4 (Nov 2012)	✓	✓	✓	☒NRP	✓3	✓3
Samsung Galaxy S4 T-Mobile (April 2013)	☒NRP	✓	✓	☒NRP	✓	✓
Samsung Galaxy S4 ATT (April 2013)	☒NRP	✓	✓	☒NRP	✓3	✓3
Hisense Sero 7 Pro (June 2013)	☒NRP	✓	✓	✓4	✓4	✓4
Asus Nexus 7 (July 2013)	✓	✓	✓	✓	✓	✓
Samsung Galaxy Note 3 (Sept. 2013)	✓	✓2	✓2	☒NRP	☒NRP	☒NRP
Google Nexus 5 (Oct 2013)	✓	✓	✓	✓	✓	✓

Table 9 shows the results for the time it took to send a 3.6kB file from the TRF7970A to the NFC-enabled devices listed. The NFC-enabled devices were the initiators (passive communication) at 424 kbps. The start edge was measured from the Transmit Complete Interrupt of the LLCP CONNECT PDU (sent from the TRF7970A to the NFC-enabled device). The end edge was measured at the RX Complete interrupt of the SNEP SUCCESS response. The results show that the throughput is highly related to the performance of the processor in the NFC-enabled device. The highest throughput was measured with the Nexus 10, which has the fastest processor.

Table 9. Data Throughput at 424 kbps for a 3.6kBytes NDEF

Test With Firmware 1.00.23		
Smartphone Model (Release Date)	**Send 3.6kB File (seconds)**	**Throughput at 424 kbps (kBps)**
Samsung Galaxy Nexus (Nov 2011)	0.7569	4.76
Samsung Galaxy S3 (T-Mobile) (May 2012)	1.8567	1.94
Asus Nexus 7 (July 2012)	1.339	2.69
Samsung Galaxy Note 2 (Sept 2012)	0.8087	4.45
AU Arrows Fujitsu FJL21 (Oct 2012)	0.3547	10.15
Nokia Lumia 820 (Oct 2012)	0.5099	7.06
HP Elite Tablet (Nov 2012)	0.5541	6.5
Samsung Nexus 10 (Nov 2012)	0.163	22.09
Google Nexus 4 (Nov 2012)	0.2557	14.08
Samsung Galaxy S4 (April 2013)	0.887	4.06
Hisense Sero 7 Pro (June 2013)	0.2981	12.08
Asus Nexus 7 (July 2013)	0.2964	12.15
Google Nexus 5 (Oct 2013)	0.5295	6.8

11 Conclusion

Peer-to-peer is one of three modes supported by the TRF7970A transceiver. The existing P2P NFC stack supports active and passive communication for both initiator and target at 106 kbps, 212 kbps, and 424 kbps. The transceiver that is initially polling and initiates the communication is the initiator. The transceiver that is initially listening is the target. The initiator always generates the RF field for both active and passive communication modes. However, the target generates its own RF field for active communication and load modulates the initiator's RF field for passive communication. To avoid two devices enabling their RF field at the same time, one must perform an initial RF collision (see Section 2) before enabling its own RF field.

The P2P demo has two modes: a stand-alone mode and a TI NFC Tool GUI mode. The stand-alone mode is the only way to test sending a large NDEF packet (3.6kB), however, using the TI NFC Tool GUI allows you to modify the default peer-to-peer modes enabled at power up. Furthermore, the GUI can be used to test for throughput rates when NDEF packets (larger than 1000 bytes) are sent from an NFC-enabled device to the TRF7970A transceiver.

Based on the tests executed with NFC-enabled devices in Table 1, the newer NFC-enabled devices interoperability with the TRF7970A is better. For legacy devices, the active communication may not be supported at all for either target or initiator at all bit rates. For newer devices, active communication has mixed results in terms of support, reliability and robust connections. Therefore, it is recommended to use passive communication over active communication.

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 card emulation operation, see *NFC card emulation using the TRF7970A*.

For more information about NFC reader/writer, see *NFC/HF RFID reader/writer using the TRF7970A*.

12 References

1. *TRF7970A multiprotocol fully integrated 13.56-MHz RFID and Near Field Communication (NFC) transceiver IC*
2. ISO/IEC18092/ECMA-340 (NFCIP – 1) (http://www.ecma-international.org)
3. ISO/IEC21481/ECMA-352 (NFCIP – 2) (http://www.ecma-international.org)
4. ISO/IEC14443-3:2009(E) (http://www.ansi.org)
5. ISO/IEC14443-4:2008(E) (http://www.ansi.org)
6. NFCForum-TS-DigitalProtocol-1.0 (Digital Protocol) (http://www.nfc-forum.org)
7. NFCForum-TS_LLCP_1.1 (Logical Link Control Protocol) (http://www.nfc-forum.org)
8. NFCForum-TS-SNEP_1.0 (Simple NDEF Exchange Protocol) (http://www.nfc-forum.org)
9. NFC Research Lab Hagenberg (http://www.nfc-research.at)

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*	18
• Updated the available bundle and link in Section 6.2, *Bundle Available for Purchase*	18

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