Application Report
SLOA154–May 2011

TRF7960A Reference Firmware Description

Josh Wyatt, Kostas Aslanidis, Andre Frantzke, Peter Reiser *Texas Instruments Embedded RF*

ABSTRACT

This application report describes the firmware implemented in the MSP430F2370 for use with the Texas Instruments TRF7960A evaluation module (EVM). The TRF7960AEVM is a multiple-standard fully integrated 13.56-MHz radio frequency identification (RFID) analog front end and data framing reader system. This reference firmware was developed using the Code Composer Studio™ IDE v4.2.1.00004 and can be also used with IAR Embedded Workbench® IDE for MSP430.

This document is designed for readers who may or may not be experienced with firmware development for RFID and want to understand the reference firmware and/or develop their own firmware for the TRF7960A. This application report should be used in conjunction with the relevant ISO or device specific standard/specification (for example, ISO15693 or ISO14443A/B), which specifies the protocol, specific commands, and other parameters required for communication between the transponder and the reader.

Contents

List of Figures

List of Tables

Code Composer Studio is a trademark of Texas Instruments.
IAR Embedded Workbench is a registered trademark of IAR Systems AB.
my-d is a trademark of Infineon.

1 Glossary

API	Application Programming Interface
EVM	Evaluation Module
GUI	Graphical User Interface
ID	Integrated Circuit
MCU	Microcontroller (for example, an MSP430 device)
NVB	Number of Valid Bits
PCD	Proximity Coupling Device (Reader/Writer, ISO14443)
PICC	Proximity Integrated Circuit Card (Transponder, ISO14443)
PUPI	Pseudo Unique PICC Identifier (ISO14443B)
SPI	Serial Peripheral Interface
UID	Unique Identifier (ISO15693, ISO14443A)
UART	Universal Asynchronous Receiver Transmitter
VCD	Vicinity Coupling Device (Reader/Writer, ISO15693)
VICC	Vicinity Integrated Circuit Card (Transponder, ISO15693)

2 Introduction

The TRF7960A is an integrated analog front end and data framing system for a 13.56-MHz RFID reader system. Built-in programming options make it suitable for a wide range of applications both in proximity and vicinity RFID systems. The reader is configured by selecting the desired protocol in the control registers. Direct access to all control registers allows fine tuning of various reader parameters as needed.

The TRF7960A can be interfaced to an MCU such as the MSP430F2370 through a parallel 10-pin interface (I/O-0 to I/O-7, IRQ, and Data Clock) or a 4-wire SPI (serial) as shown in Figure 1. The MCU is the master device and initiates all communication with the TRF7960A. The anti-collision procedures (as described in the ISO standards 14443A/B and 15693) are implemented in the MCU firmware to help the reader detect and communicate with one PICC/VICC among several PICCs/VICCs. The MCU is also used for communication (through a UART IC) to a higher-level host station, which is normally a personal computer. The user can send the desired commands to the MCU through the GUI. The MCU interprets the data received and sends appropriate commands to the TRF7960A.

NOTE: It is recommended that firmware developers review ISO14443A/B and ISO15693 if possible.

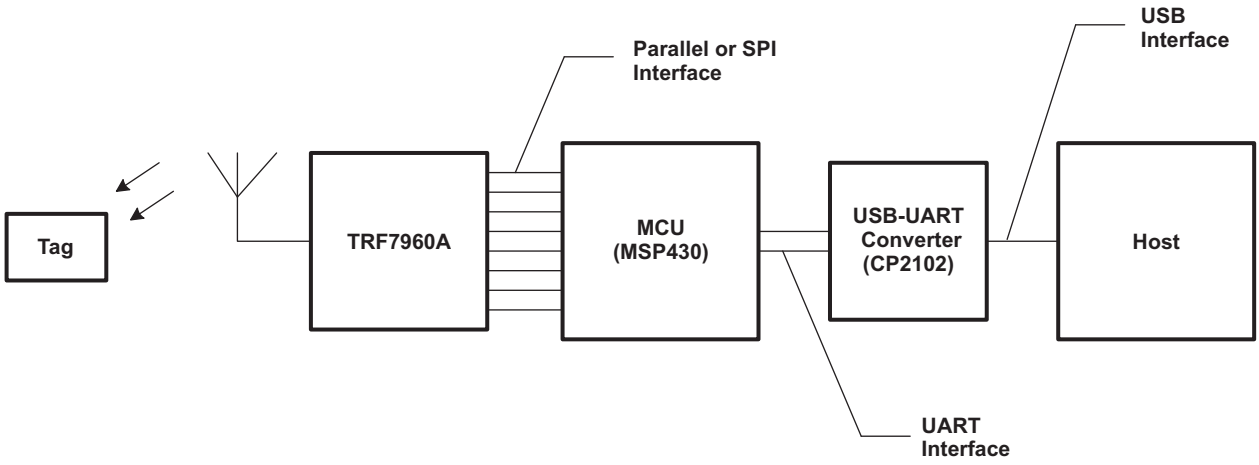

Figure 1. TRF7960A EVM Application Block Diagram

3 Basic Program Flow

In the reference firmware, the MCU clock is provided by the SYS_CLK output of the reader. Upon power up, an auxiliary clock signal (60 kHz) is made available to the MCU. When the main reader enable pin (EN) is set high, the supply regulators are activated and the 13.56-MHz oscillator is started. When the supplies are settled and the oscillator frequency is stable, the SYS_CLK output is switched from the auxiliary frequency of 60 kHz to the selected frequency derived from the crystal oscillator. All peripherals (for example, UART) are initialized and parallel or SPI interface is chosen. At this time, the reader (see Figure 2) is ready to communicate and perform the required tasks.

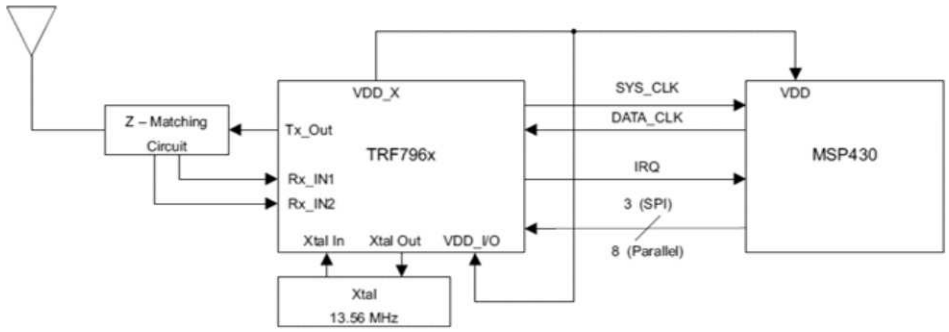

Figure 2. TRF7960A Embedded System Block Diagram

The firmware is capable of running in two operating modes:

- Stand-alone (demo)
- Host (terminal) control

In stand-alone mode, when power is applied to the EVM and the initialization sequence has completed, the firmware automatically detects tags and illuminates a protocol-related LED on the EVM.

During the initialization sequence, the MCU writes appropriate bits to the Chip Status Control register (0x00) and the ISO Control register (0x01) in the TRF7960A to select the operation mode. It then polls for transponders in the field by executing the anti-collision sequences (as described in the ISO standards) to obtain the UIDs/PUPIs or UIDs of PICCs or VICCs in range of the EVM antenna. This is done in the Iso15693FindTag(), Iso14443aFindTag(), and Iso14443bFindTag() functions (in files iso15693.c, iso14443A.c, and iso14443B.c).

The stand-alone loop is executed repeatedly until any data is received from the PC through the UART, at which point the EVM enters the host control mode. Program execution jumps to the second loop and depending on the data received in the UART buffer, the MCU sends commands to the 12-byte FIFO buffer in the TRF796x. The two modes are shown in Figure 3. The switch to the host control mode from the stand-alone mode is done via the host_control_flag.

Standards that are not needed can be disabled in the respective header file, and the source files can be excluded from the firmware build.

NOTE: The my-d™ move functions can be enabled in the my-d.h file and then the my-d.c file can be included in the build. The my-d move functions are used with Infineon my-d move transponders.

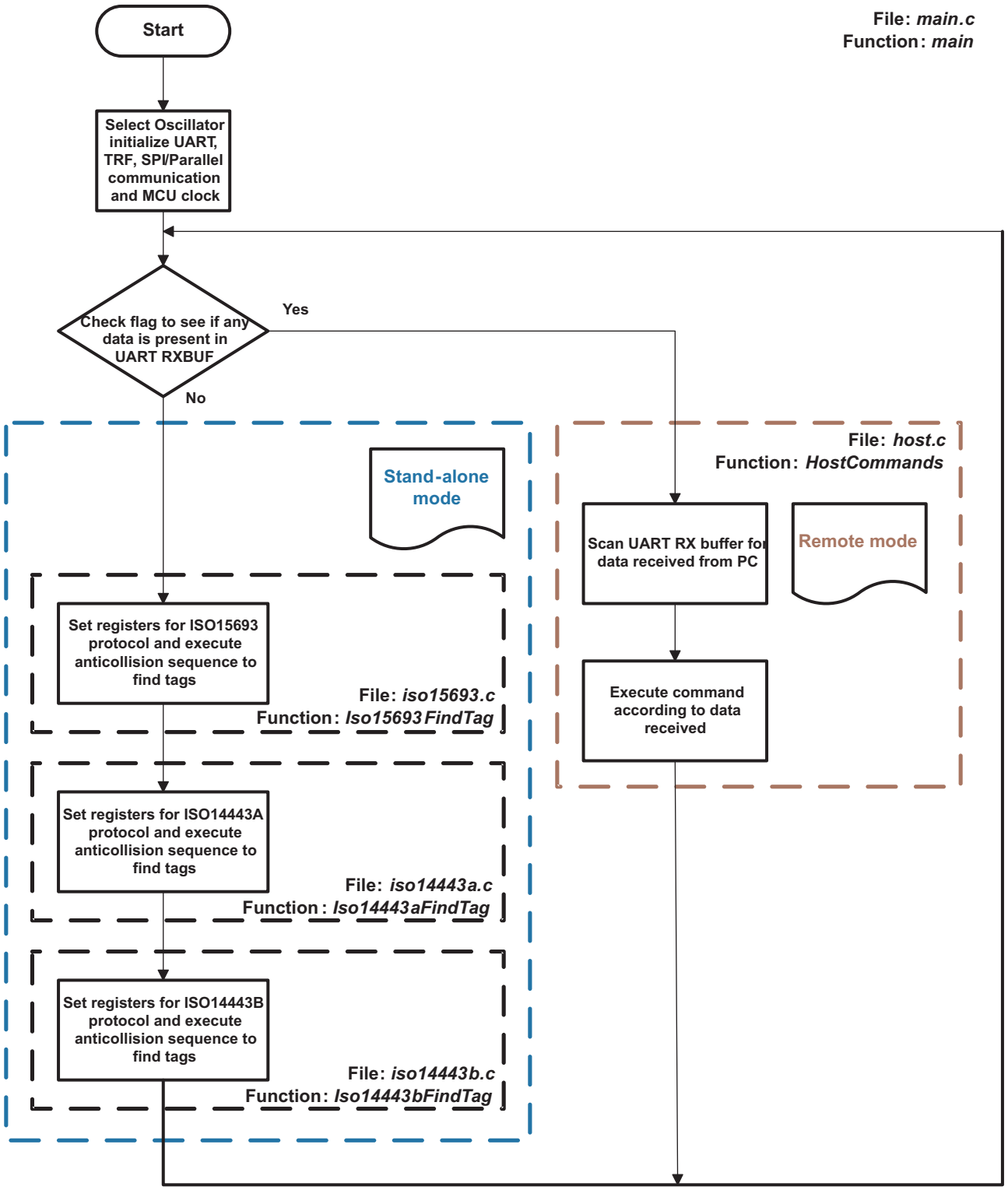

Figure 3. main.c Flow Chart

4 Interrupt Service Routine (ISR)

The TRF7960A use its IRQ (pin 13) to prompt the MCU for attention. As there are multiple reasons for interrupt condition to occur, the TRF7960A IRQ Status register (0x0C) (see Table 1) is present and is read when interrupt occurs to determine the cause and action to be taken. The interrupt service routines (see Figure 4, Figure 5, and Figure 6) show the logical flow of what has been implemented in the reference firmware.

Table 1. IRQ Status Register (0x0C)

Bit	Name	Function	Comments
B7	irq_tx	IRQ set due to end of TX	The flag is set at the start of TX, and the IRQ is sent when TX is complete (IRQ = 1)
B6	irq_rx	IRQ set due to end of RX	The flag is set at start of RX, and the IRQ is sent when RX is complete (IRQ = 1)
B5	irq_fifo	FIFO is high or low	Signals when the FIFO is high or low (more than 8 bits during RX or less than 4 bits during TX).
B4	irq_err1	CRC error	RX CRC error (valid only when B7 of ISO Control register (0x01) is set to 0)
B3	irq_err2	Parity error	RX parity error (ISO14443A)
B2	irq_err3	Byte framing or EOF error	RX framing or EOF error
B1	irq_col	Collision error	Valid only for ISO14443A or ISO15693 (single subcarrier). Bit is set as defined by register 0x10.
B0	irq_noresp	No response interrupt	Trigger for MCU to send the next EOF/slot marker as defined by No Response Wait Time register (0x07) (for ISO15693).

Table 2. Interrupt Conditions

Interrupt Condition	Action to Take
Transmission Complete	Reset FIFO
Collision Occurred (indicated by Bit 1 in register 0x0C)	1. Read Collision Position register (in the TRF796x). 2. Determine the number of valid bytes and bits. 3 . Read valid received bytes and bits in FIFO and write to local buffer.
RX Flag Set	1. Read FIFO Status register (in the TRF796x) to determine the number of unread bytes and bits in the FIFO. 2. Read the data in FIFO and write to local buffer. 3. Reset FIFO.
RX Active and 9 bytes in FIFO	1. Read 9 bytes from FIFO. 2. Check if IRQ pin is still high. If yes, go to condition C.
CRC Error	Set error flag If my-d move functions are enabled check for 4-bit receive
Byte Framing Error	Set error flag
No Response Time Out	—
Any Other Interrupt Condition	1. Reset FIFO. 2. Clear interrupt flag.

NOTE: Although registers 0x0D and 0x0E give the collision position, only register 0x0E is used, because the anti-collision command in ISO 14443A is maximum of 7 bytes long. Therefore, 8 bits (0x0D) are enough to determine the position.

The lower nibble of the Collision register (0x0E) contains the bit count and the upper nibble contains the byte count. For example, if the collision position register holds the value 0x43 (0100 0011b), then the collision occurred in the fourth byte at bit position 3.

The anti-collision procedure in the ISO14443A standard is done in such a way that the reader sends at least two bytes (cascade level and length information) in the anti-collision command. The collision position is counted from this reader command on. Therefore, to know the number of valid bytes and bits, subtract 0x20 from the Collision Position register value.

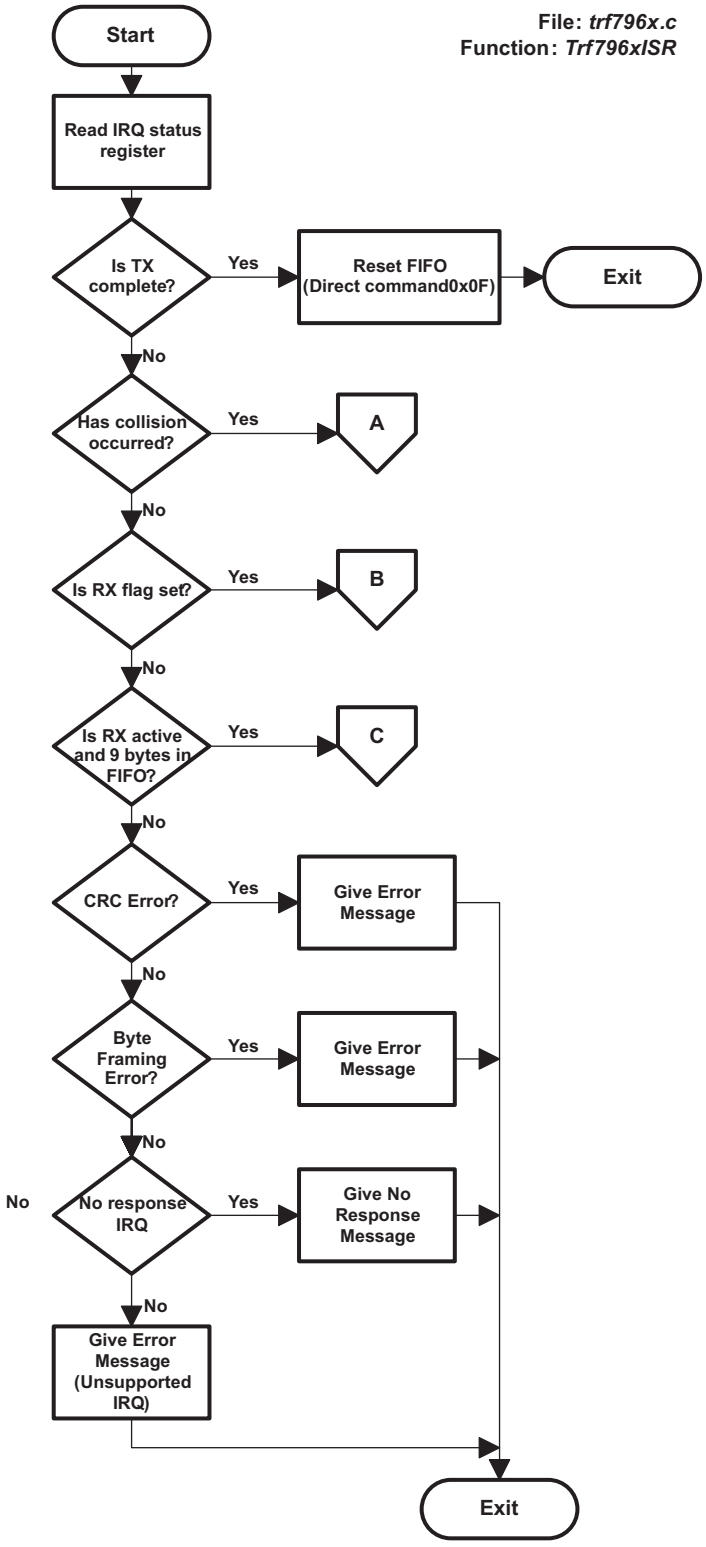

Figure 4. Interrupt Service Routine (1)

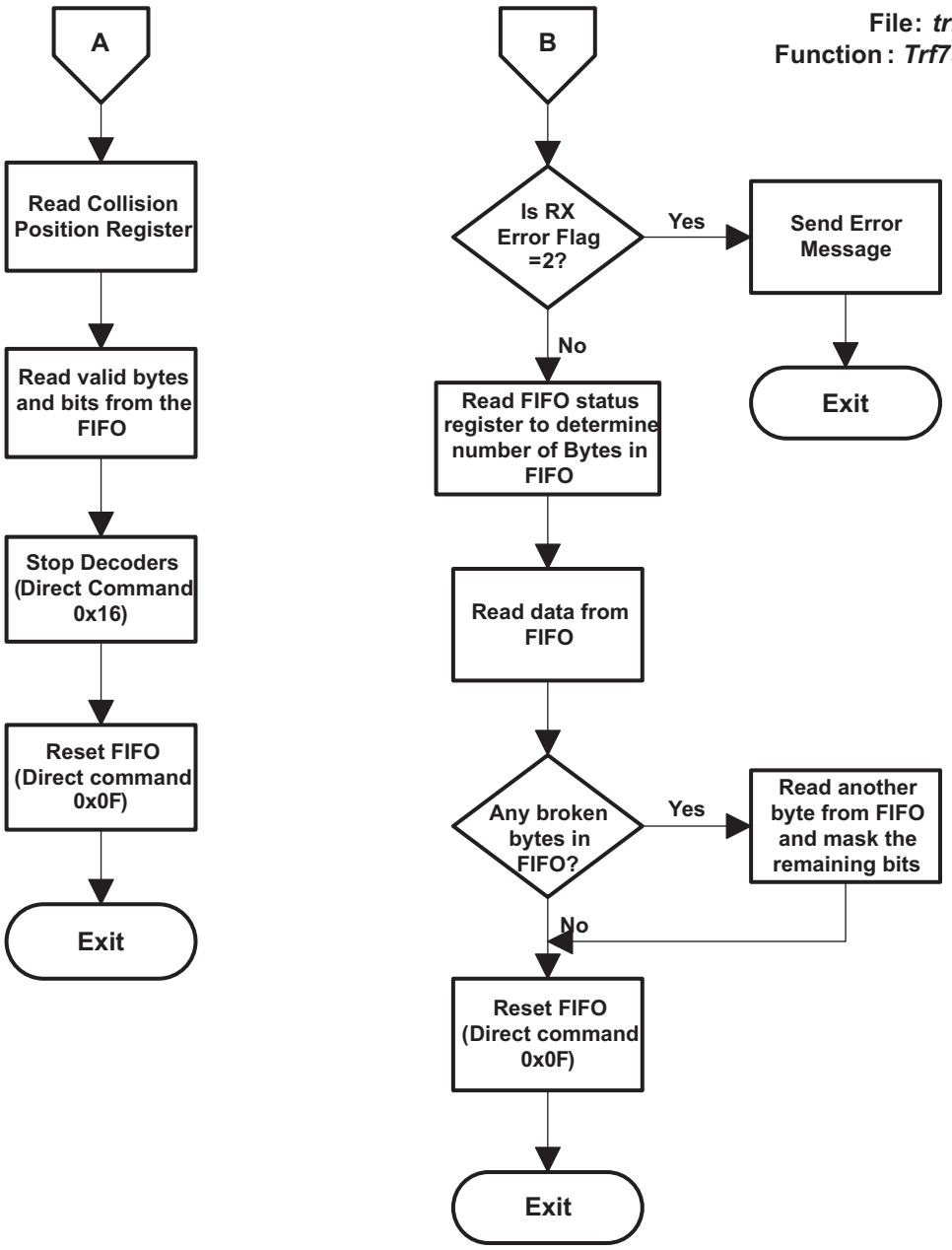

Figure 5. Interrupt Service Routine (2)

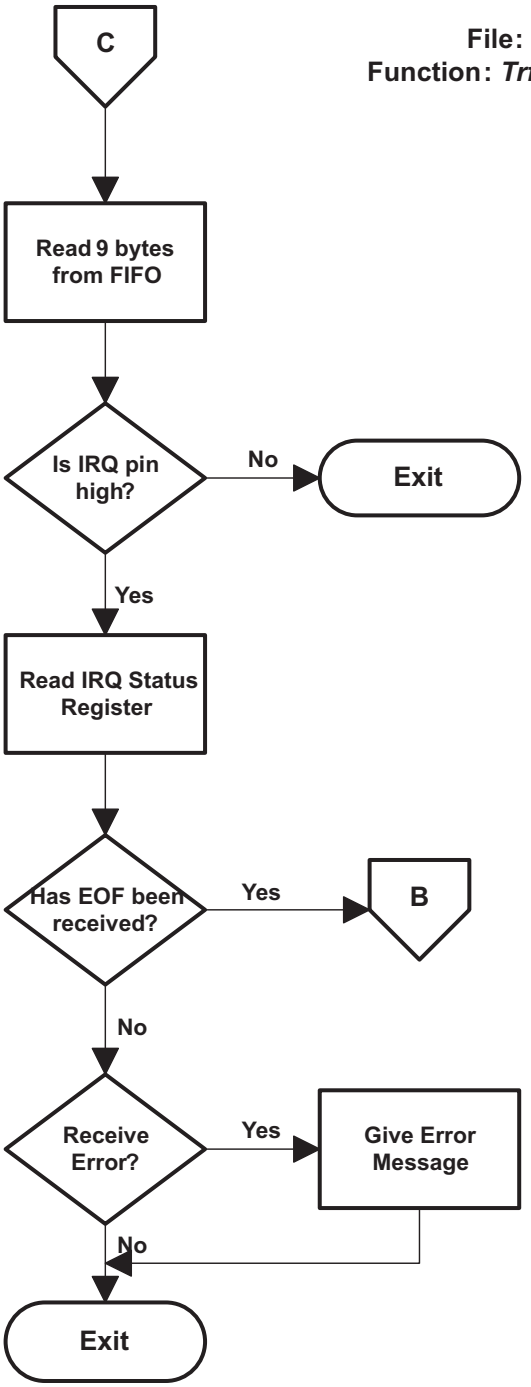

Figure 6. Interrupt Service Routine (3)

5 Anti-Collision Sequences (Stand-Alone and Host Control)

The following sections describe the anti-collision sequences that are to be executed for the corresponding standards.

5.1 Anti-Collision Sequence for ISO15693

Anti-collision algorithm:

1. The reader sends a mask value and number of slots along with the inventory request. The number of slots can be 1 or 16.
2. The VICC compares the least significant bits of its UID to the slot number + mask value. If it matches, it sends a response. If number of slots is 1, comparison is made on mask value only.
3. If only one VICC responds, then there is no collision and the VCD receives the UID.
4. If the reader detects a collision, it increments the slot pointer and makes note of the slot number in which collision occurred.
5. The reader sends an EOF to switch to the next slot. The VICC increments its slot counter on reception of EOF.

If the number of slots is 16, steps 1 to 4 are repeated for all 16 slots.

At the end of 16 slots, the reader examines the slot pointer contents. If it is not zero, it means that collision has occurred in one or more slots.

To determine the new mask value:

1. Increment the mask length by 4.
2. Calculate the new mask, which equals the slot number (in which the collision occurred) plus the old mask.
3. Decrement slot pointer by 1.
4. Repeat from start with the new mask value until slot pointer is zero.

NOTE: Due to the recursive nature of the algorithm, there is a risk of stack overflow when a collision occurs. It is highly recommended that the user implement a stack (RAM) overflow check in the firmware.

The flow charts in Figure 7 and Figure 8 describe the firmware implementation of the anti-collision sequence.

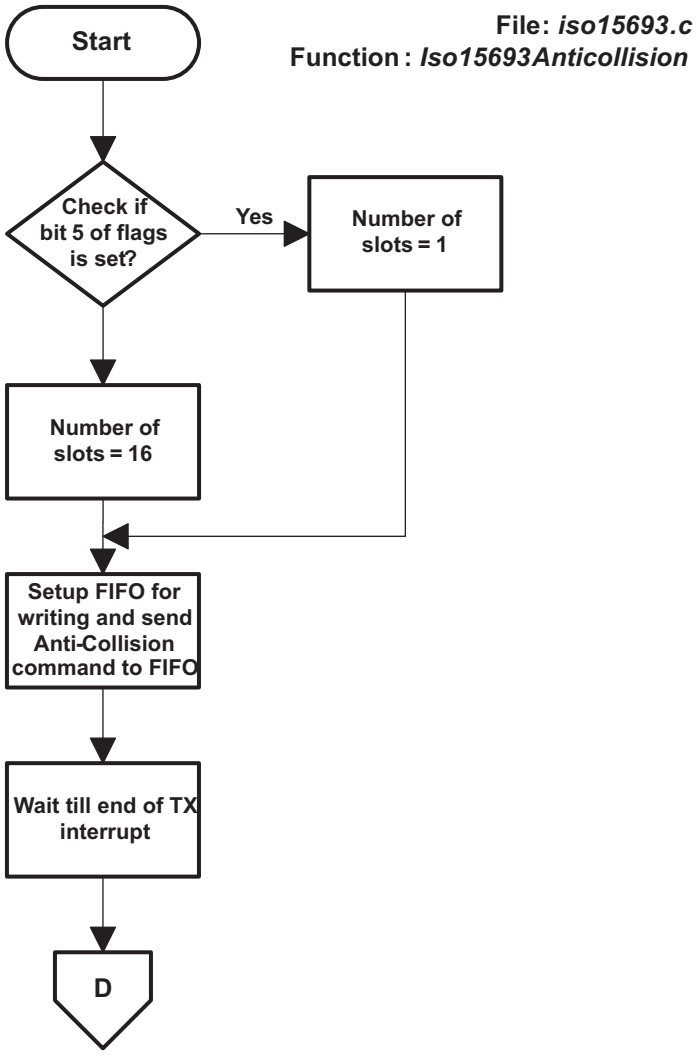

Figure 7. ISO15693 Anti-Collision Method Flow Chart (1)

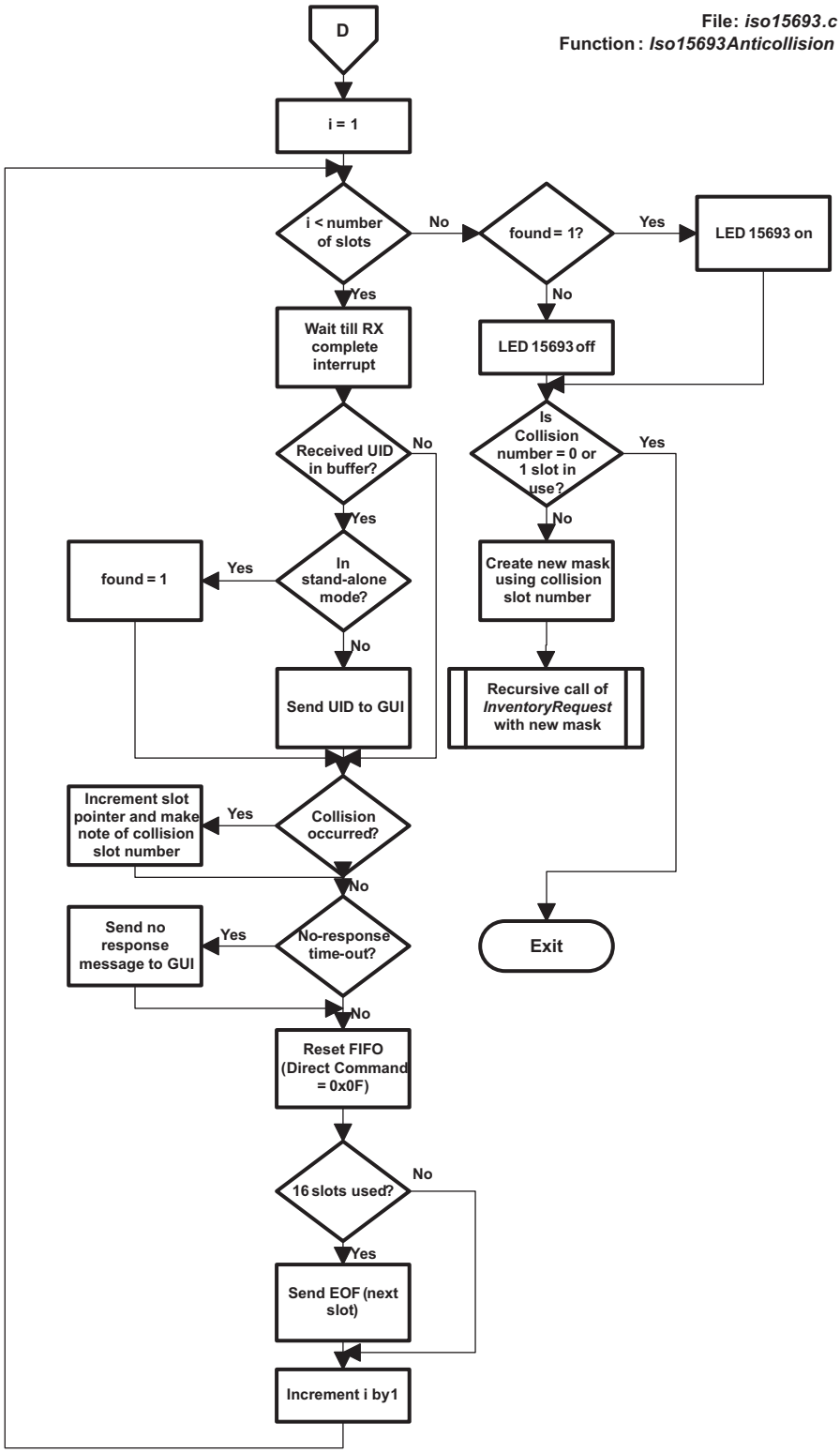

Figure 8. ISO15693 Anti-Collision Method Flow Chart (2)

5.2 Anti-Collision Sequence for ISO14443A

The anti-collision loop for ISO14443A PICCs is as follows:

1. The PCD sends the anti-collision command with NVB = 0x20.
2. All PICCs respond with their UIDs.
3. If more than one PICC responds, there is a collision. If there is no collision, steps 4 to 8 should be skipped.
4. The PCD then reads the Collision Position register to determine the number of valid bytes and bits and reads the valid data from the FIFO.
5. The PCD assigns the value of the Collision Position register to NVB.
6. The PCD transmits the anti-collision command with the new NVB followed by the valid bits.
7. Now only the PICCs for which part of the UID is equal to the valid bits transmit the remaining bits of the UID.
8. If a collision occurs again, steps 4 to 7 are repeated.
9. If no collision occurs, the PCD transmits the SELECT command with NVB = 0x70, followed by the complete UID.
10. The PICC that matches the UID responds with a SAK message.
11. The PCD checks for the cascade bit in the SAK. If it is set, steps 1 to 9 are executed with the appropriate SELECT command (host command 0xA2).

- The lower nibble of the Collision register (0x0E) contains the bit count and the upper nibble contains the byte count. For example, if the collision position register holds the value 0x43 (0100 0011b), then the collision occurred in the fourth byte at bit position 3.
- The anti-collision procedure in the ISO14443A standard is done in such a way that the reader sends at least to bytes (cascade level and length information) in the anti-collision command. The collision position is counted from this reader command on. Therefore, to know the number of valid bytes and bits, subtract 0x20 from the Collision Position register and NVB.
- The NVB is similar to the Collision Position register. The lower nibble of the NVB contains the bit count, and the upper nibble contains the byte count. For example, if the NVB holds the value 0x52, it means that there are 5 valid bytes and 2 valid bits.
- The possible values of SELECT command are 0x93, 0x95, and 0x97, which correspond to the different cascade levels (1 through 3), as defined by ISO14443A SEL coding.

The flow chart in Figure 9 describes the firmware implementation of the anti-collision sequence.

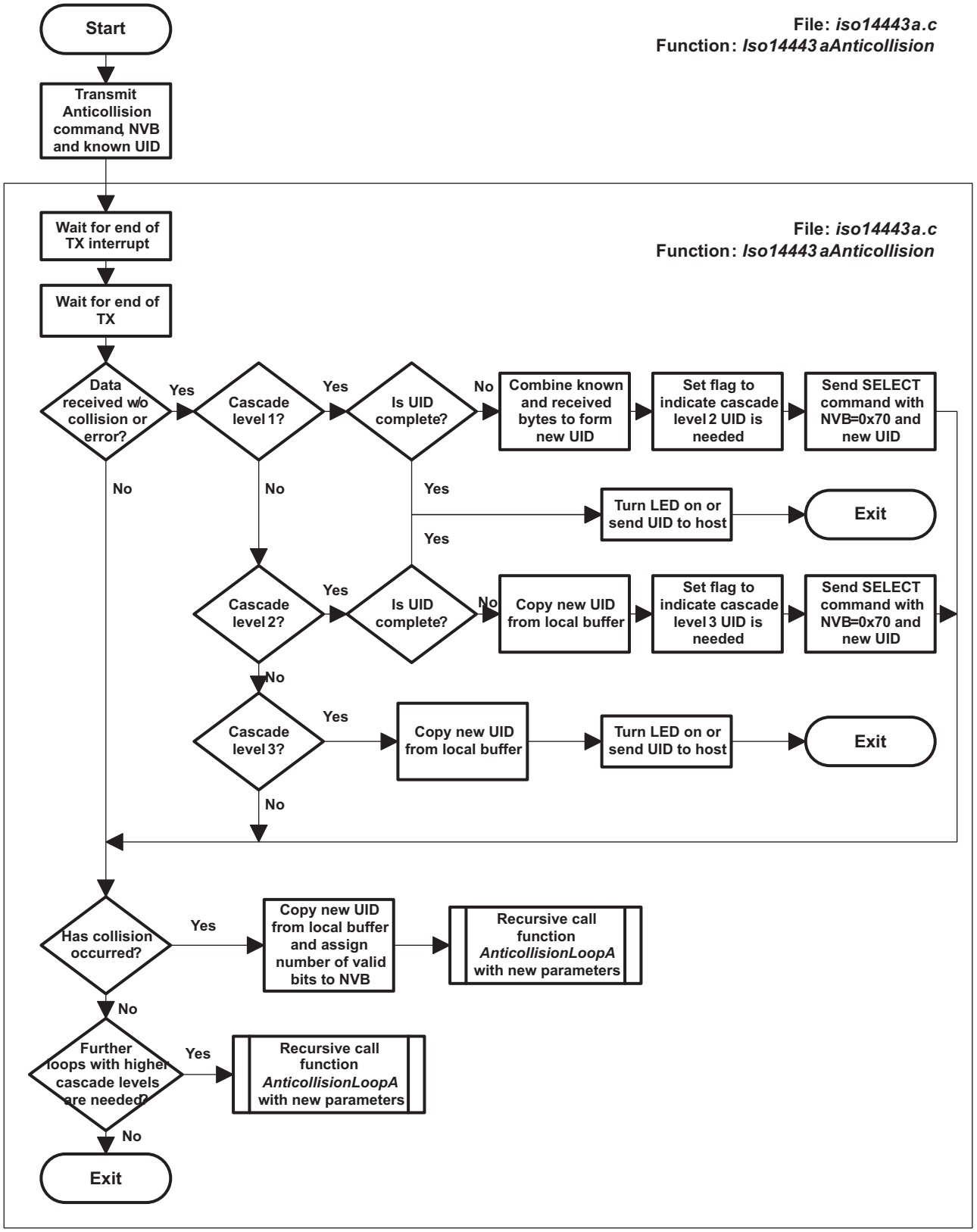

Figure 9. ISO14443A Anti-Collision Method Flow Chart

5.3 Anti-Collision Sequence for ISO14443B

The anti-collision sequence for ISO14443B follows the slotted Aloha approach:

1. The PCD sends the REQB command with parameter N, which specifies the number of slots.
2. Each PICC generates a random number R in the range from 1 to N.
3. The PCD sends a Slot-Marker command during every time slot.
4. The PICC responds only if R matches the slot number. Otherwise, it sends no response.
5. If multiple PICCs respond, the PCD makes note of the collision. The PCD generates a new N, and steps 1 to 4 are repeated.

The flow chart in Figure 10 describes the firmware implementation of the anti-collision sequence.

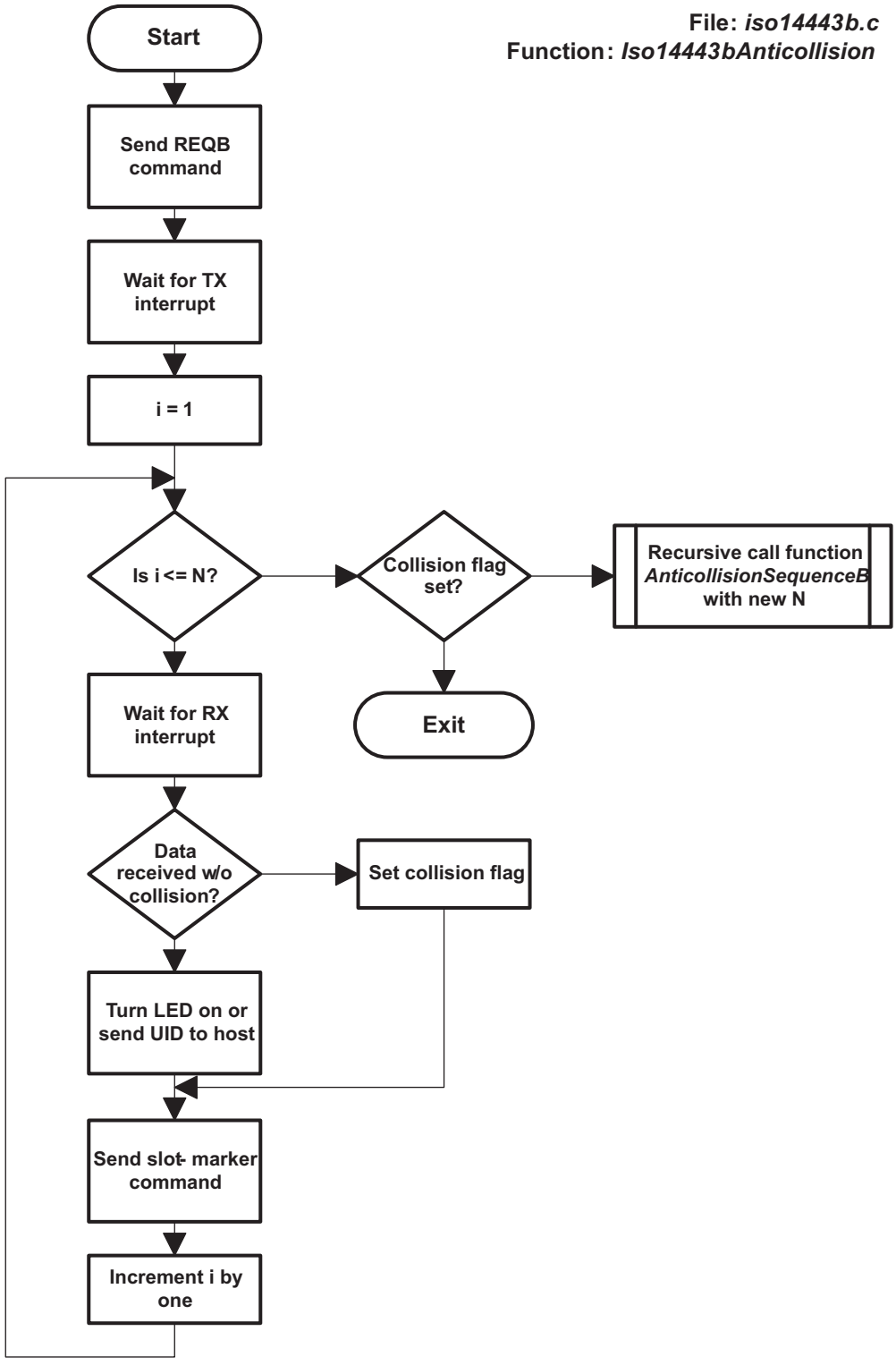

Figure 10. ISO14443B Anti-Collision Method Flow Chart

6 Host Control Mode

The reader can be host control controlled by a higher-level host such as a personal computer. A graphical user interface (GUI), which can be used as an API, helps users to communicate with the TRF7960A reader through the MCU. The GUI on the host machine issues commands to the EVM MCU through a USB to UART converter. The MCU receives the commands in the UART receive buffer, interprets them, and sends suitable data to the register or FIFO buffer in the reader. As shown in Figure 11, the UART receive buffer of the MCU is continuously scanned for data received from the host in UartGetLine().

To send a response to the host, the functions UartPutChar(), UartPutCrlf(), UartPutByte(), and UartSendCString() in uart.c are used.

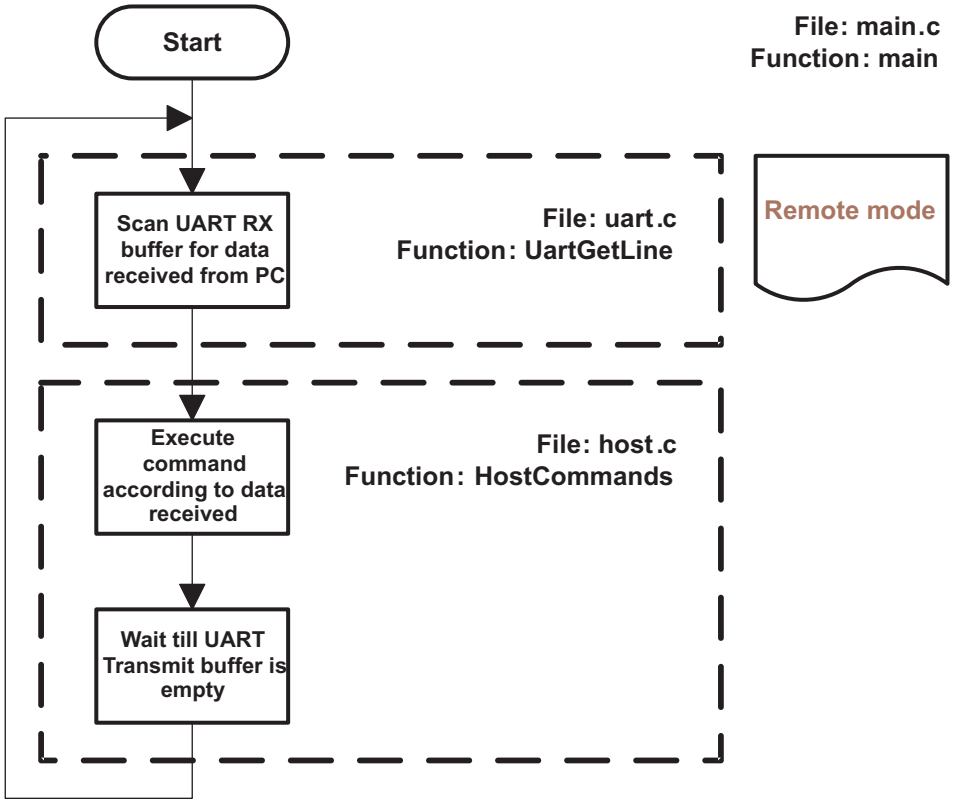

Figure 11. Host Control Flow Chart

The communication format from host to reader is organized into data frames of six fields.

Table 3. Data Frame Format from Host to Reader

SOF (0x01)	Number of bytes	0x00	0x03, 0x04	Command + Parameters	EOF (0x00, 0x00)

The data frame starts with SOF (0x01). The second byte defines the number of bytes in the frame including SOF and EOF. The third byte should be 0x00, the fourth byte should be 0x03, and the fifth byte should be 0x04. The sixth byte is the command code, which is followed by command parameters or data (bytes 7 and 8). The communications ends with EOF (two bytes of 0x00).

6.1 Host Commands

Using the host the following commands can be executed:

Table 4. Host (PC GUI to MCU) Commands

Command (hex)	Command Function	Parameters	Example
0x10	Write Single	Address, data, address, data,…	01|0A|00|0304|**100108**|0000 Write 0x08 to register 0x01
0x11	Write Continuous	Address, data, data, data,…	01|0B|00|0304|**11002108**|0000 Write 21, 08 to register 00, 01
0x12	Read Single	Address, address, address,…	01|0A|00|0304|**120100**|0000 Read register 01, 00
0x13	Read Continuous	Number of bytes to read, start address	01|0A|00|0304|**130502**|0000 Read registers 0x02 to 0x06
0x14	ISO15693 Anti-collision	Flags, command, mask length…	01|0B|00|0304|**14060100**|0000 Flags = 0x06, command 0x01, mask length = 0
0x15	Direct Command	Direct command code	01|09|00|0304|**151F**|0000 Command 0x1F (reset FIFO)
0x16	RAW write	Data or commands	01|09|00|0304|**168F**|0000 Send 0x8F to TRF (command 0x1F)
0x18	ISO15693 Request Command ISO14443B Halt	Flags, command code, data… (as specified in ISO standard)	01|0B|00|0304|**18022033**|0000 ISO15693 Read Single Block 0x33
0x0F	Direct Mode	-	01|08|00|0304|**0F**|0000
0x72	NFC Type 2 Command	Command code, address, data… (as specified in ISO standard)	01|0A|00|0304|**723011**|0000 Read 4 Blocks from 0x11
0xA0	ISO14443A Anti-collision REQA	-	01|08|00|0304|**A0**|0000
0xA1	ISO14443A Anti-collision WUPA	-	01|08|00|0304|**A1**|0000
0xA2	ISO14443A Select	SEL, UID	01|0D|00|0304|**A2DE655D5ABC**|0000 UID = DE655D5A, CRC = 0xBC
0xB0	ISO14443B Anti-collision REQB	Slots	01|09|00|0304|**B004**|0000 2^4 = 16 slots
0xB1	ISO14443B Anti-collision WUPB	Slots	01|09|00|0304|**B104**|0000 2^4 = 16 slots
0xF3 – 0xFC	Select GPIO output levels	-	01|08|00|0304|**F7**|0000 (i.e. switch LED 4 on, Table 5)
0xFE	Get Firmware Version	-	01|08|00|0304|**FE**|0000

6.2 Request Command (0x18)

To execute ISO15693 commands and ISO14443B HALT command after setting the protocol and execute anticollision the function HostRequestCommand() is used. Flags, command and the data, which must be sent, are given by the GUI.

6.3 GPIO Control

The commands in Table 5 can be used to control GPIO output levels and switch the LEDs on the Board on or off using the GUI. In actual customer applications, these could either be used for illustrated purpose or for driving switches, relays, etc.

Table 5. GPIO Output Levels controlled from PC GUI Host Commands

Host Command (hex)	GPIO Level	Function
0xF3	P1.2 High	LED 6 On
0xF4	P1.2 Low	LED 6 Off
0xF5	P1.3 High	LED 5 On
0xF6	P1.3 Low	LED 5 Off
0xF7	P1.4 High	LED 4 On
0xF8	P1.4 Low	LED 4 Off
0xF9	P1.5 High	LED 3 On
0xFA	P1.5 Low	LED 3 Off
0xFB	P1.6 High	LED 2 On
0xFC	P1.6 Low	LED 2 Off

6.4 NFC Type 2 Command (0x72)

The NFC Type 2 commands in Table 6 are implemented in the firmware.

Table 6. NFC Type 2 Commands Implemented in TRF7960A Firmware for MSP430F2370

Function	Command (hex)	Parameters
Read (4 Blocks)	0x30	Address of first block
Write (1 Block)	0xA2	Address, Data (4 bytes)
Read 2 Blocks (my-d move)	0x31	Address of first block
Write 2 Blocks (my-d move)	0xA1	Address, data (16 bytes)

7 MCU to TRF7960A Communication

The interface to the microcontroller is selected by a jumper. For SPI mode, the macro SPIMODE is set to 1. For parallel mode, the macro is set to 0. If the same communication interface is always used, SPIMODE can be set to a constant value. To communicate with the TRF7960A, one of the functions in trf796x.c is called. After checking the selected interface either the according function in spi.c or parallel.c is called.

7.1 Direct Command (0x15)

Trf796xDirectCommand() is used to execute a direct command. The parameter to this function is the address of an 8-bit variable that contains the 5-bit command in bit 4 through bit 0. To set the required Address/Command Word, the command control bit (bit 7) is set to 1 (command). The Address/Command Word Bit is sent to the reader IC, which executes the command.

7.2 Read Single (0x12)

Trf796xReadSingle() is used to read the contents of specified reader IC registers. The parameters to this function are the address of an array that contains the addresses of the registers to read, and the number of registers. The 5-bit addresses (0x00 to 0x1F) are stored in bit 4 through bit 0 of the 8-bit array elements. To set the required Address/Command Words, the Read/Write bit (BIT6) is set to 1 (read). The function sends the Address/Commands Word to the reader IC and stores the received register values in the array element that contained the register address as many times as required.

7.3 Read Continuous (0x13)

Trf796xReadCont() is used to read a specified number of reader IC registers starting from a given address. The parameters are an array address and the number of registers to read out. The fist of the 8-bit array elements contains the 5-bit address of the first register. To set the required Address/Command Word, the Read/Write bit (bit 6) and the continuous address mode bit (bit 5) are set to 1 (write, continuous address mode). The function sends the Address/Command Word and receives the required register values, which are stored in the array.

7.4 Write Single (0x10)

Trf796xWriteSingle() is used to write to specified reader IC registers. The parameter for this function is the address of an array that contains the register addresses and the values to write, and the number of used array elements, which is twice the number of registers. To get the required Address/Command Words, bit 7 to bit 5 of the 8-bit array elements containing an address are left at 0. They are sent to the reader IC followed by the value to write.

7.5 Write Continuous (0x11)

Trf796xWriteCont() is used to write to a specified number of reader IC registers starting at a given address. The parameters are an array address and the number of used array elements, which is one more than the specified number of registers. The address of the first register is stored in the first 8-bit array element and the values to write are stored in the following elements. To set the rewired Address/Command Word, the continuous address mode bit (bit 5) is set to 1. The Address/Command Word is sent to the reader IC followed by all the values to write.

7.6 RAW Write (0x16)

Trf796xRawWrite() is used to send a raw string to the reader chip. The function gets the address of an array, which contains the data, which shall be sent, and the number of bytes, which shall be sent. The Address/Command Word is not handled by the function and must be given in the right way. This allows for example to send a direct command followed by a write request using only one function call.

NOTE: To read from or write to the FIFO, continuous mode should be used here, because only the first FIFO register address can be addressed.

Table 7. Address/Command Word Distribution

Bit	Description	Bit Function	Address	Command
B7	Command Control Bit	0 = address 1 = command	0	1
B6	Read/Write	0 = read 1 = write	R/W	0
B5	Continuous Address Mode		Cont. Mode	Not Used
B4	Address/Command Bit 4		Address 4	Command 4
B3	Address/Command Bit 3		Address 3	Command 3
B2	Address/Command Bit 2		Address 2	Command 2
B1	Address/Command Bit 1		Address 1	Command 1
B0	Address/Command Bit 0		Address 0	Command 0

8 Debugging

The debug and trigger features have been implemented in case the firmware developer does not have ready access to logic analyzer but does have oscilloscope.

Alternatively, if the developer does not have access to an oscilloscope, relatively low-cost logic analyzers are available from http://www.saleae.com/logic/ and http://www.pctestinstruments.com/.

8.1 Macro DBG Use

If the macro DBG in trf796x.h is set to 1, interrupts are displayed in host control mode. The contents of the IRQ Status register (0x0C) is displayed as hex value in the Log Window and special events are represented by a character shown in Table 8. This way it is possible to check, whether the expected interrupt events occurred or not.

Table 8. Displayed Interrupt Events in Debug Mode

Character	Event Represented
T	End of TX
E	End of RX
F	FIFO Level High
x	End of RX and Error Condition
N	No Response

8.2 Use of the Trigger Function

For debugging of the firmware, the communication must be observed. For this reason, a protocol trigger on LED 5 can be activated by setting the macro TRIGGER in msp430f2370.h to 1. This setting works for all commands and in stand-alone mode but does not work for anti-collision sequences in host control mode. The trigger can be used by the scope to help to capture the RF signal and validate the communication signals and timings. Figure 12 shows an example setup, and Figure 13 shows oscilloscope screen capture.

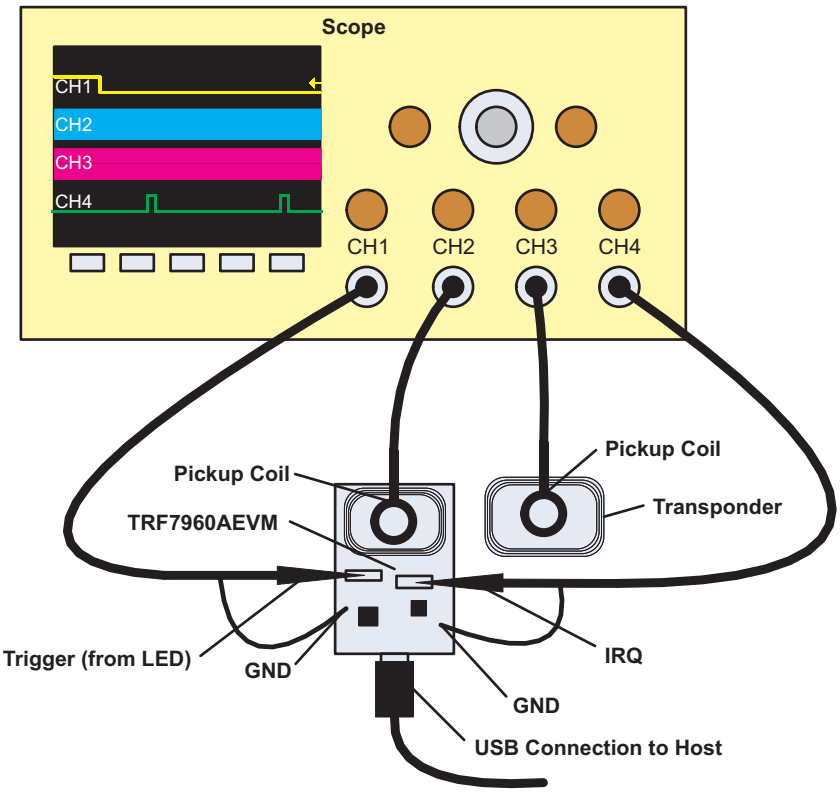

Figure 12. Measurement Setup for Using Trigger Feature

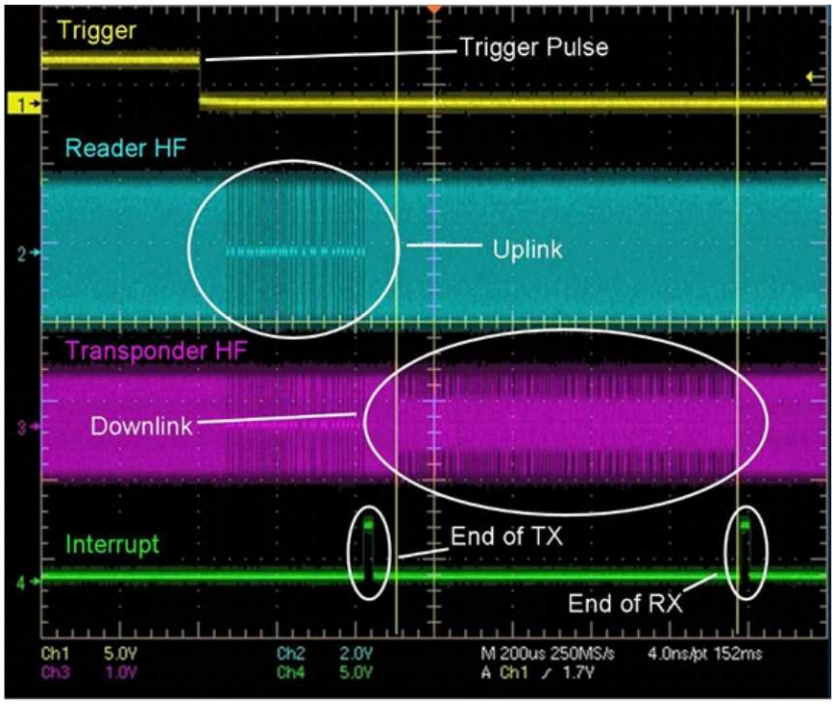

Figure 13. Oscilloscope Screen Example Using Trigger Feature

9 References

- TRF7960A firmware project
- TRF7960A data sheet (SLOS732)
- Comparison of TRF7960 and TRF7960A (SLOA156)
- ISO15693-3
- ISO14443-3
- Infineon my-d™ move (Infineon ISO14443A Products)

IMPORTANT NOTICE

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

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

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

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

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

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

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

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