Application Report
SLOA134–March 2009

Firmware Description of the TI TRF796x Evaluation Module (EVM)

ShreHarsha Rao

ABSTRACT

This application note discusses the firmware implemented in the MSP430F2370 (a 16-bit ultra-low power microcontroller from the TI MSP430 family) used with Texas Instruments' TRF796x, a fully integrated 13.56MHz radio frequency identification (RFID) analog front end and data framing reader system.

This document is designed for use by customers who are experienced with Radio Frequency Identification Devices (RFID) and firmware development and want to develop their own application using the TRF796x. This reference guide should be used in conjunction with the ISO15693, ISO 14443A/B standards, which specify the standard protocol, commands and other parameters required for communication between the transponder and the reader.

Contents

List of Figures

List of Tables

1 Abbreviated Terms

The following abbreviations are used:

GUI - Graphical User Interface

NVB - Number of Valid Bits

PCD - Proximity Coupling Device

PICC - Proximity Integrated Circuit Card

SPI - Serial Peripheral Interface

UID - Unique Identifier

UART - Universal Asynchronous Receiver Transmitter

VCD - Vicinity Coupling Device

VICC - Vicinity Integrated Circuit Card

2 Introduction

The TRF796x is an integrated analog front end & 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 TRF796x can be interfaced to a microcontroller 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) interface as shown in Figure 1. The MCU is the master device and initiates all communication with the reader. The anti-collision procedures (as described in the ISO standards 14443A/B, 15693 and Tag-it™) 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) 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 TRF796x.

This document discusses the firmware implemented in the MSP430F2370. The firmware has been developed using the IAR Embedded Workbench V3.41A.

Note: It is recommended that the user initially review the ISO standards 14443A/B 15693.

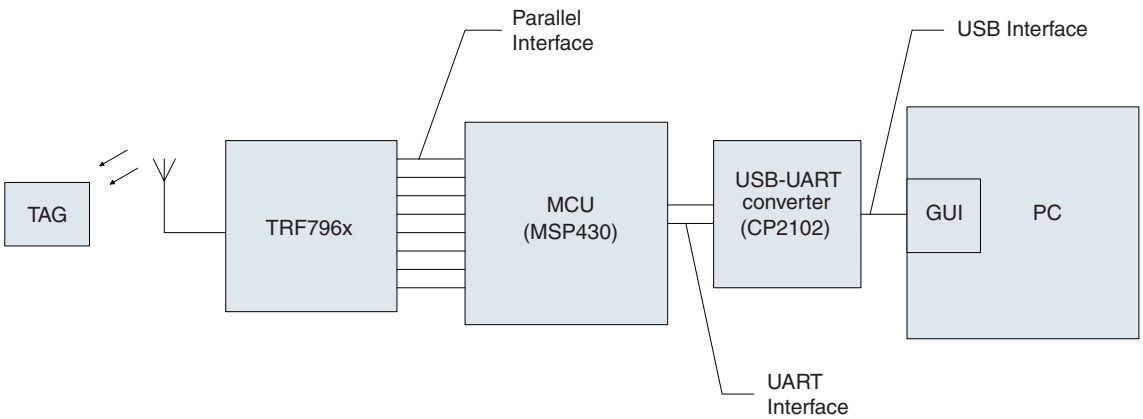

Figure 1. System Block Diagram

3 Basic Program Flow

The MCU clock is provided by the SYS_CLK output of the reader. On power up, an auxiliary clock signal (60 KHz) is made available on the SYS_CLK output. When the main reader enable pin EN is set high, the supply regulators are activated and the 13.56MHz 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 (UART, etc.) are initialized and parallel/SPI interface is chosen (Note: The sample code given uses the parallel interface). At this point, the reader is ready to communicate and perform the required tasks.

The firmware is capable of running in two modes, the stand-alone mode and the GUI mode.

In the stand-alone mode, the firmware automatically detects tags on connecting the EVM to a USB port. The MCU writes appropriate bits to the Chip Status Control Register and the ISO Control Register in the TRF796x to select the operation mode. It then executes the anti-collision sequence (as described in the ISO standards) in order to obtain the UIDs of all the PICCs/VICCs detected. This is done in the FindTags() function (in file main.c) which as the name implies, looks for tags of protocols 15693, 14443A/B and Tag-it™ in a sequence. This loop is executed repeatedly until any data is received from the PC through the UART. Once any data is received in the UART Receive buffer of the MCU, the firmware enters the GUI 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 represented in Figure 3.

3.1 Stack Manipulation

The switch to the GUI mode from the Stand-alone mode is done via a stack manipulation procedure. When the UART RX buffer receives the first SPI data, it raises an interrupt. The interrupt processing begins at this point. The Program Counter (PC), which points to the next instruction, is pushed to the stack by the interrupt logic. Then the interrupt logic loads the address of the UART Interrupt Service Routine (ISR) to the PC and program execution continues from there. In the UART ISR, the portion of the stack that holds the address of the instruction to which the ISR should return is overwritten with the address of the function HostCommands. Thus at the end of the UART ISR, this new address pops from the stack and the program execution automatically starts from the HostCommands function and waits for the rest of the SPI data to be received from the GUI.

For example, consider the following piece of code:

```
#pragma vector = USART0RX_VECTOR
__interrupt void RXhandler (void)
{
.
.
if(FirstSPIdata)
{
asm("mov.w #HostCommands,10(SP)");
}
}
```

For this particular implementation of the ISR, the address to which the ISR should return to is at an offset of 10 from the stack pointer. This offset will vary depending on the implementation of the UART RX ISR. The offset can be calculated by looking at the number of POP instructions preceding the RETI instruction in the Disassembly Window of the IAR Embedded Workbench.

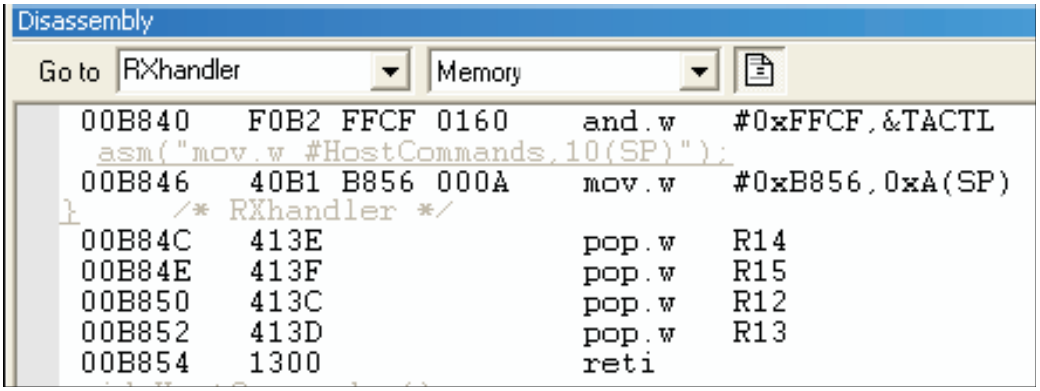

Figure 2. Disassembly Window

In Figure 2, there are four POP WORD instructions before the RETI. This means that 4x2=8 bytes (1 word = 2 bytes) need to be popped from the stack before the PC could be loaded with the return address. Thus it can be seen that the 10th byte from the Stack Pointer needs to be overwritten with the address of the function HostCommands.

The firmware implementation details of the anti-collision procedure for each standard are explained in Section 5.

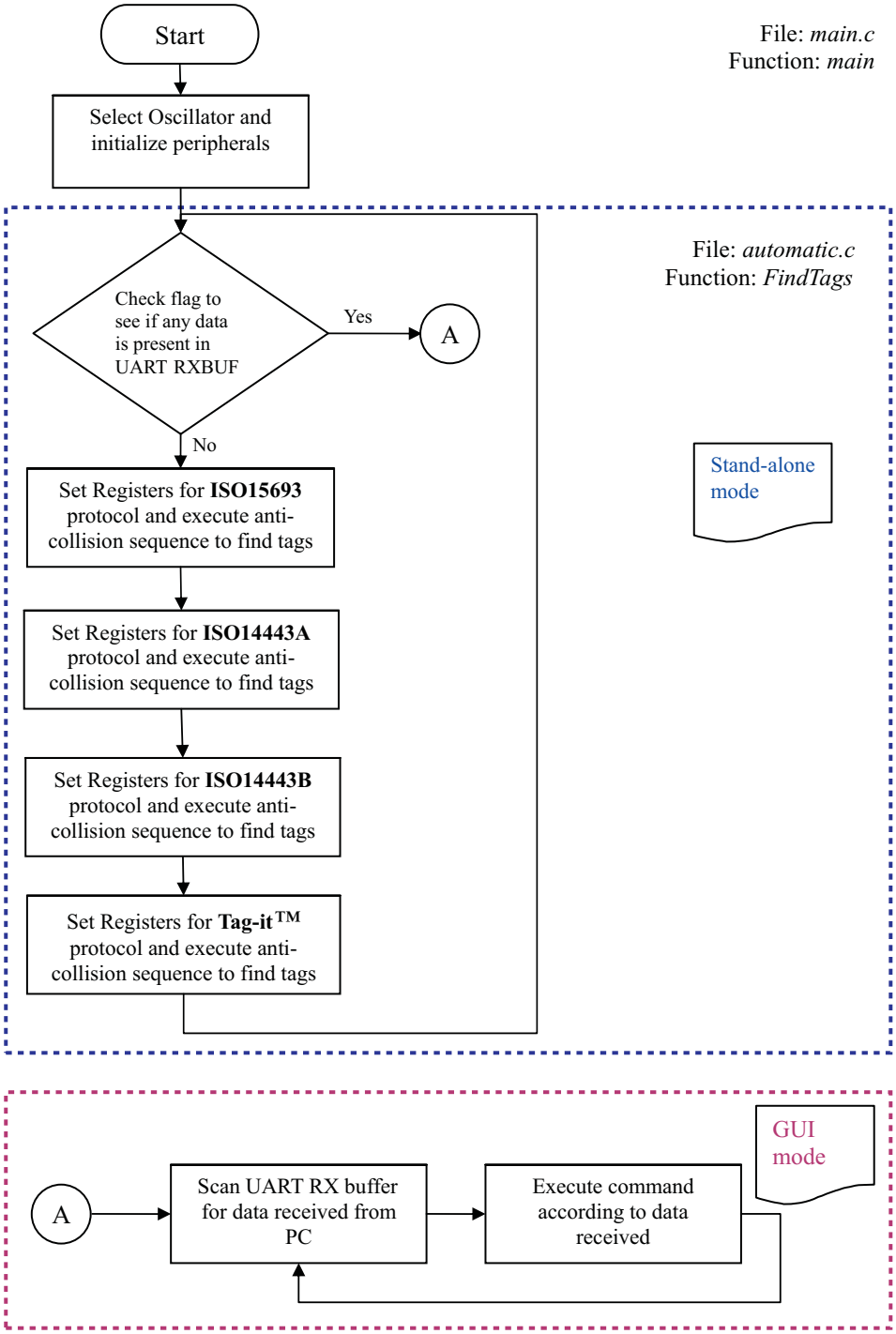

Figure 3. main

4 Interupt Handler Routine

Before delving into the details of each anti-collision sequence, understanding of the Interrupt Service Routine Handler is important.

The reader which is a slave device has an IRQ pin to prompt/flag the MCU for attention in cases when the reader detects a response from the PICC/VICC. The Interrupt Handler Routine described below determines how the IRQ should be handled.

The TRF796x IRQ status register (Table 2) is read to determine the cause of the IRQ. The following conditions (Table 1) are checked and appropriate actions taken:

Table 1. Interrupt Conditions

CONDITION	ACTION
Transmission complete	Reset FIFO
Collision occurred	1. Read Collision Position Register (in the TRF796x). 2. Determine the number of valid bytes and bits. 3. Read the valid received bytes and bits in FIFO and write to local buffer. 4. Reset FIFO.
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 #3.
CRC error	Set error flag
Byte framing error	Set error flag
No-reponse time-out	-
Any other	1. Reset FIFO. 2. Clear interrupt flag.

Table 2. IRQ Status Register

BIT NO.	BIT NAME	FUNCTION	COMMENTS
B7	Irq_tx	IRQ set due to end	Signals in the TX are in progress. The flag is set at the start of TX, but the interrupt request is sent when TX is finished.
B6	Irg_srx	IRQ set due to RX start	Signals that RXZ SOF was received and RX is in progress. The flag is set at the start of RX, but the interrupt request is sent when RX is finished.
B5	Irq_fifo	Signals the FIFO is 1/3 > FIFO > 2/3	Signals FIFO high or low (less than four or more than eight).
B4	Irq_err1	CRC error	Reception CRC
B3	Irq_err2	Parity error	
B2	Irg_err3	Byte framing or EOF error	
B1	Irq_col	Collision error	For ISO14443A and ISO15693 single sub-carrier.
B0	Irq_noresp	No response interupt	Signal to MCU that the next slot command can be sent.

Note: 1. Though registers 0Dh and 0Eh give the collision position, only register 0Eh is used because the Anti-collision command in ISO 14443A is maximum only 7 bytes long. Hence 8 bits (0Dh) are enough to determine the position.

2. The lower nibble of the Collision register (0Eh) has the bit count and the upper nibble has the byte count. For example, if the collision position register holds the value 0x40, it means that the collision happened in the 4th byte on the bit 0.

3. The anti-collision procedure in the ISO14443A standard is done in such a way, that the reader sends at least 2 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.

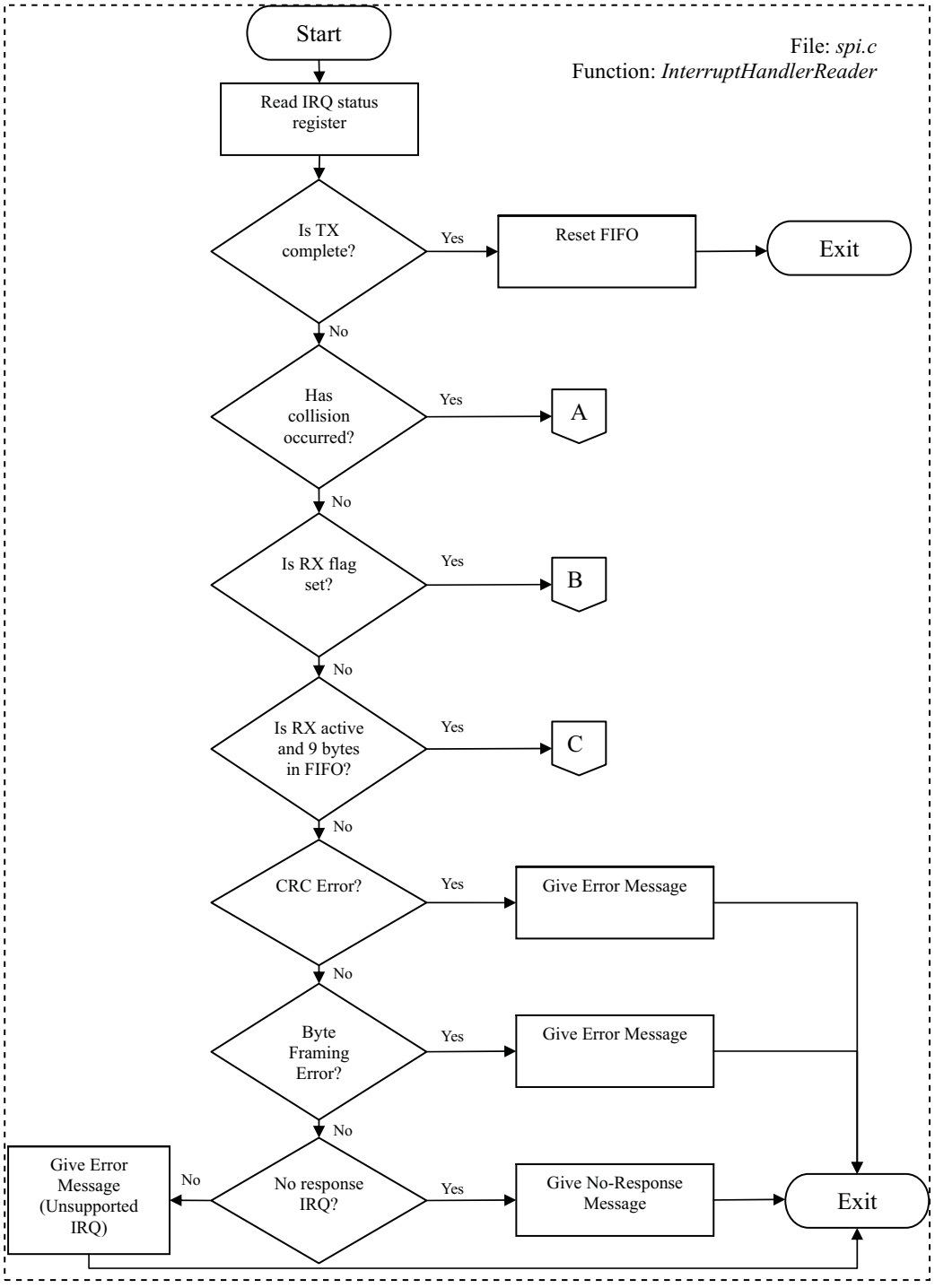

Figure 4. Interrupt Handler Routine (1)

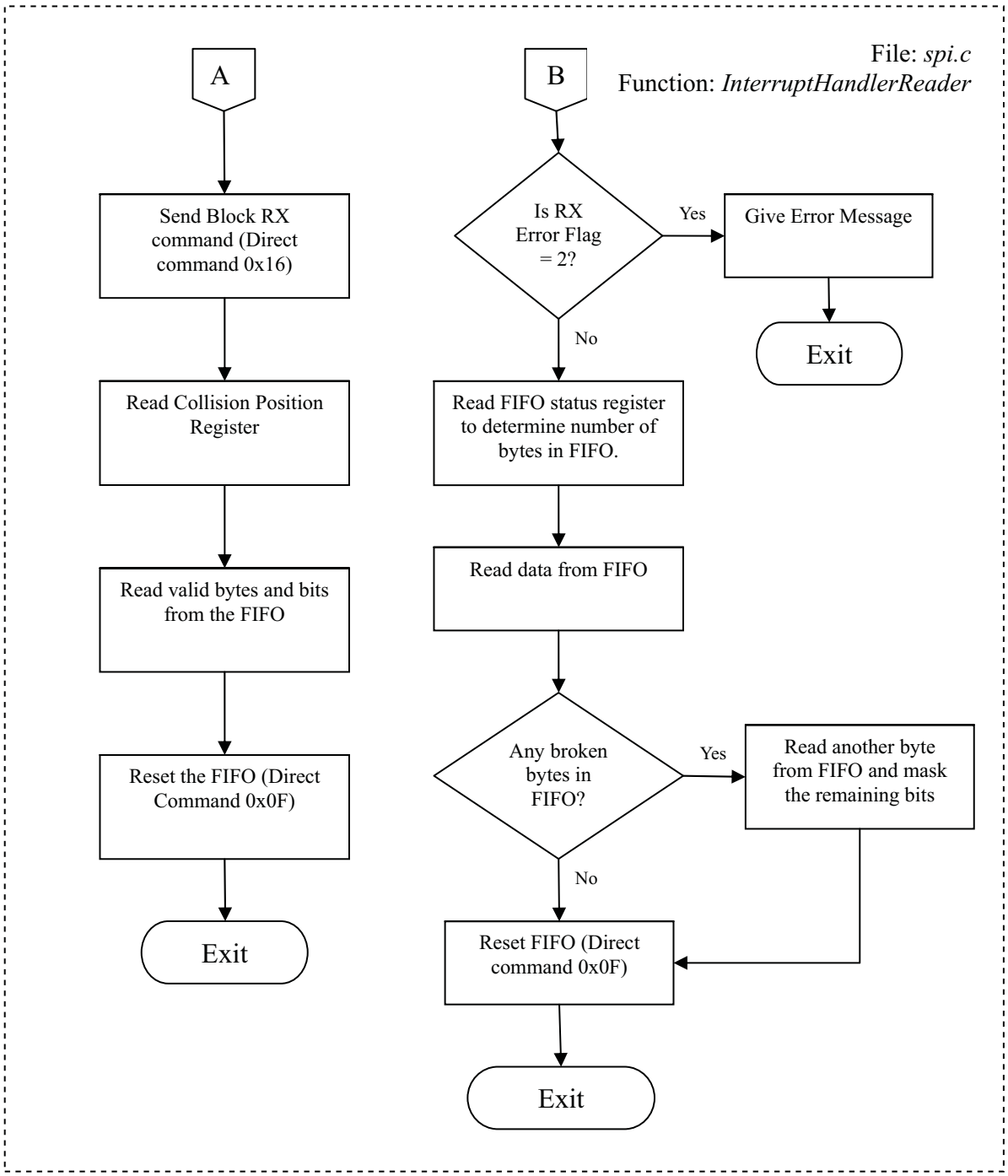

Figure 5. Interrupt Handler Routine (2)

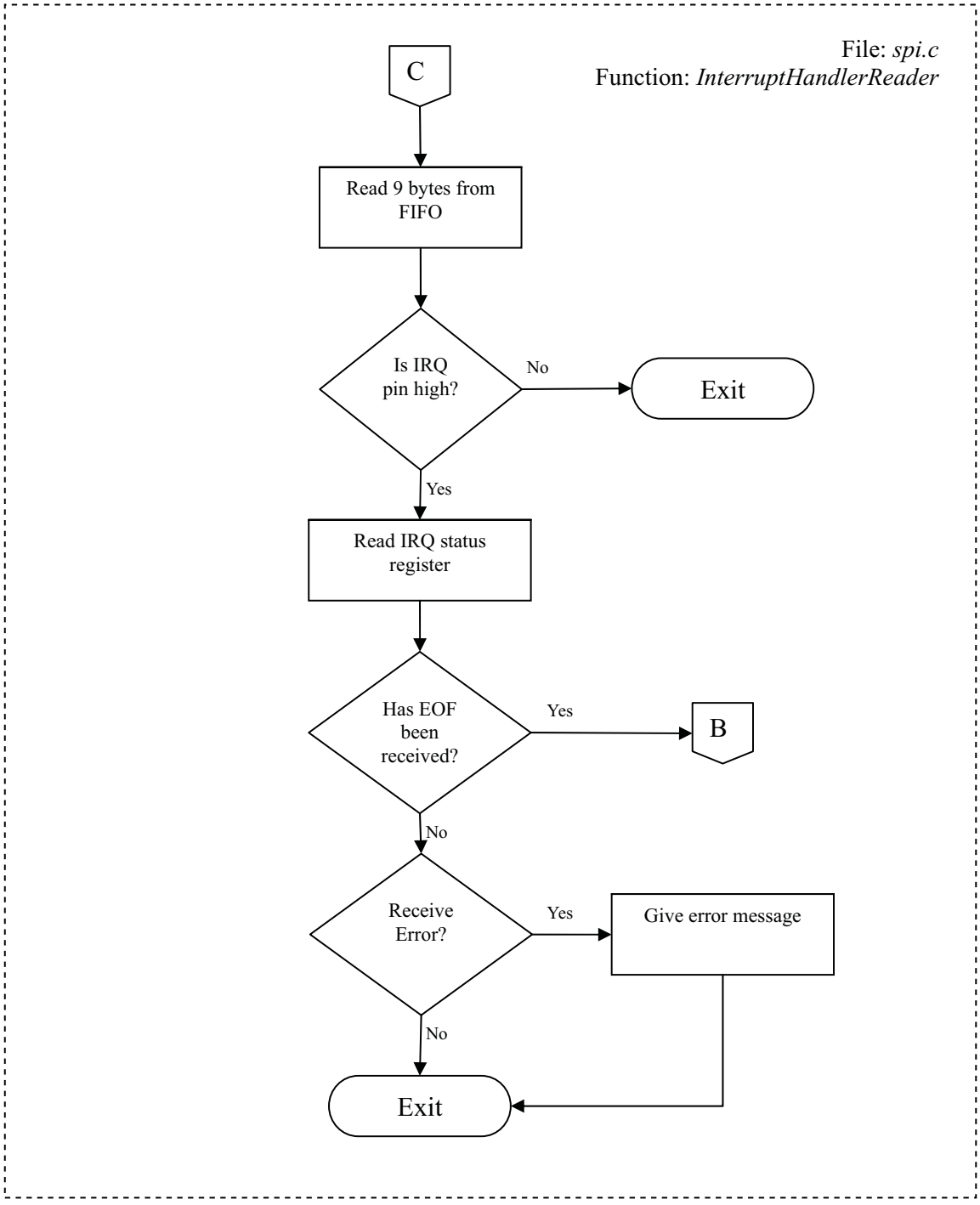

Figure 6. Interrupt Handler Routine (3)

5 Anti-Collision Sequences

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.

Steps 1-4 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 new mask value:

1. Increment the mask length by 4.
2. Calculate New mask = Slot number (in which collision occurred) + old mask.
3. Decrement slot pointer by 1.

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 collision occurs. It is highly recommended that the user implement stack overflow check in the firmware.

A detailed description of the firmware implementation of the anti-collision sequence is given in Figure 7 and Figure 8 in the form of a flowchart:

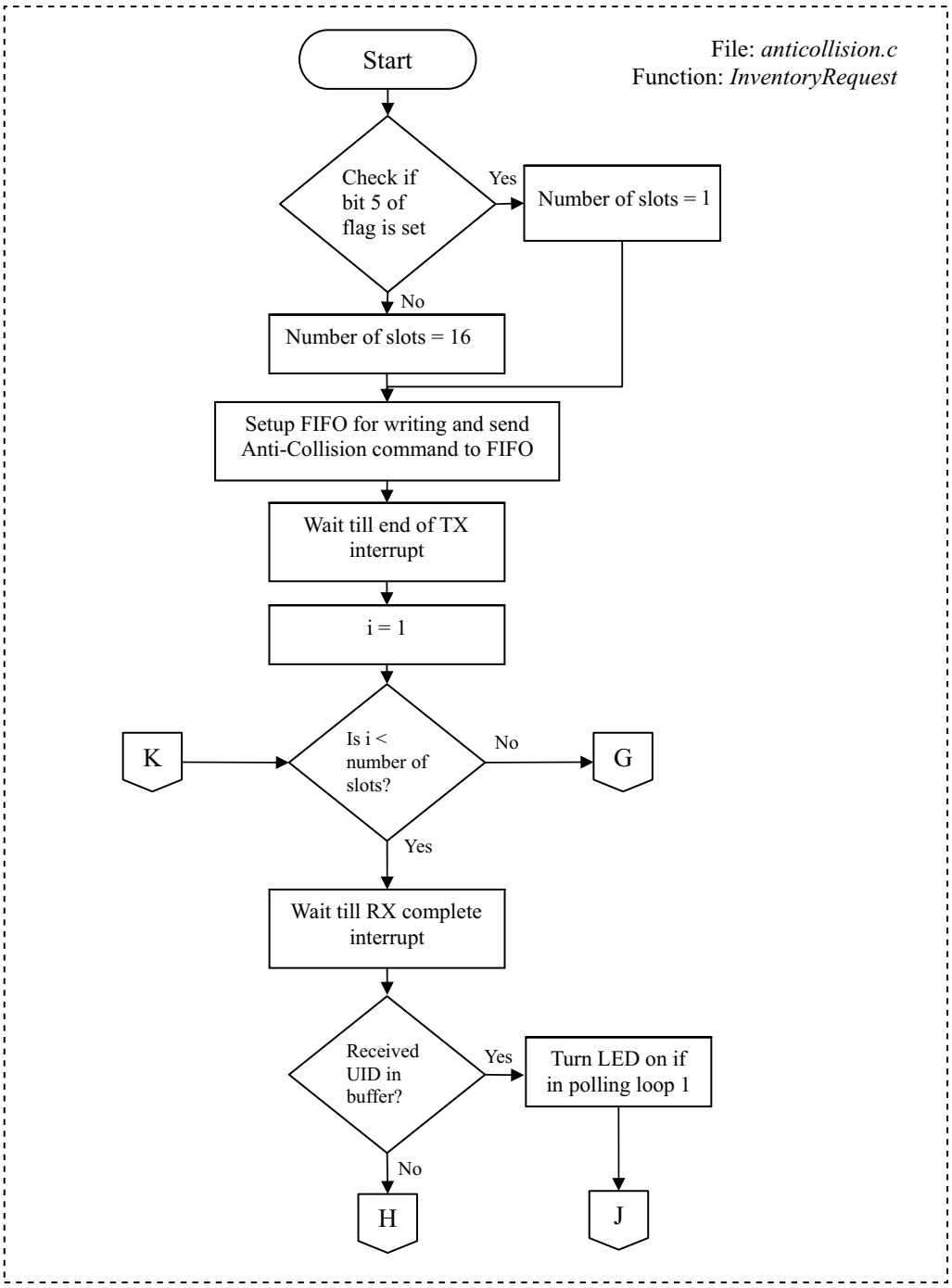

Figure 7. Inventory Request (1)

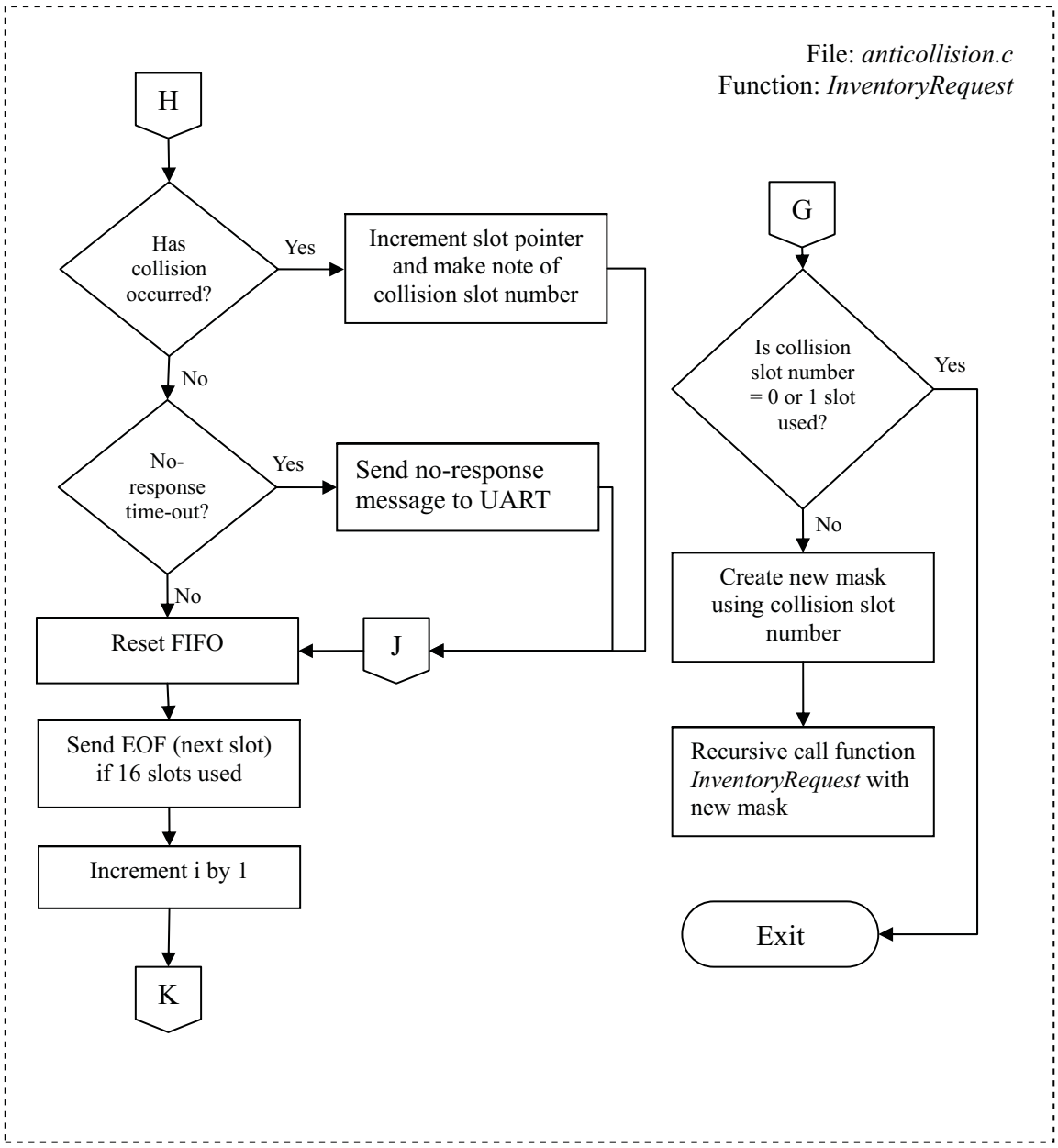

Figure 8. Inventory Request (2)

5.2 Anti-Collision Sequence for ISO14443A

The anti-collision loop for 14443A is as follows:

1. The PCD sends the Anti-collision command with NVB = 0x20.
2. All PICCs will respond with their UIDs.
3. If more than one PICC responds, there will be collision. If there is no collision, steps 4-8 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 of which part of the UID is equal to the valid bits transmit the remaining bits of the UID.
8. If again collision occurs, steps 4-7 are repeated.
9. If no collision occurs, PCD transmits SELECT command with NVB = 0x70 followed by the complete UID.
10. The PICC which UID matches responds with a SAK message.
11. The PCD checks for the cascade bit in the SAK. If set, steps 1-9 are executed with the appropriate SELECT command.

Note: 1. The lower nibble of the Collision register (0Eh) has the bit count and that the upper nibble has the byte count. For example, if the Collision position register holds the value 0x40, it means that the collision happened in the 4th byte on the bit 0.

2. The Anti-collision procedure in the ISO14443A standard is done in such a way that the reader sends at least 2 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.

3. The NVB is similar to the Collision Position Register. The lower nibble of the NVB has the bit count and that the upper nibble has the byte count. For example, if the NVB holds the value 0x52, it means that there are 5 valid bytes and 3 valid bits.

4. The possible values of SELECT command are 0x93, 0x95 and 0x97 corresponding to different cascade levels.

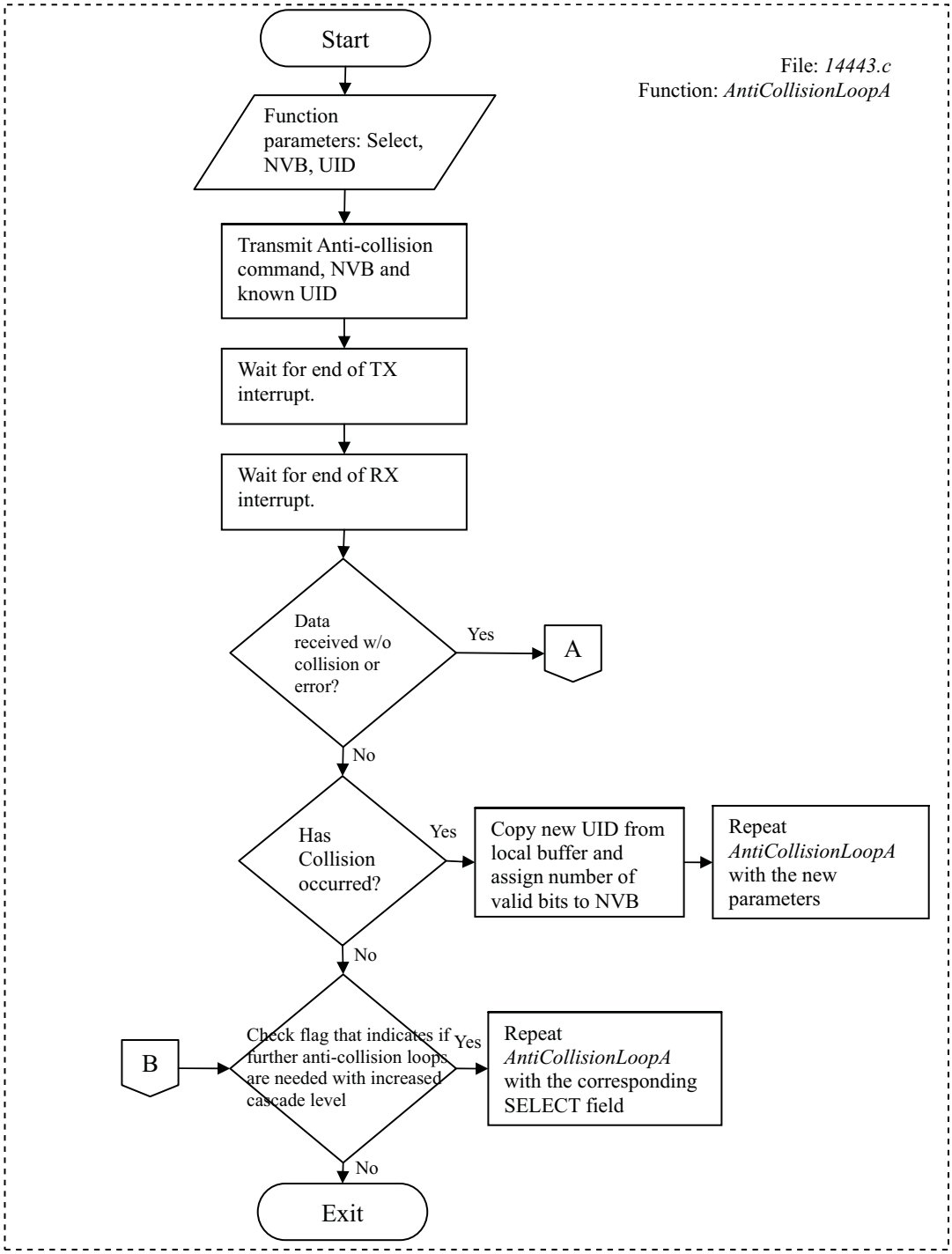

Figure 9. Anti-Collision Loop A (1)

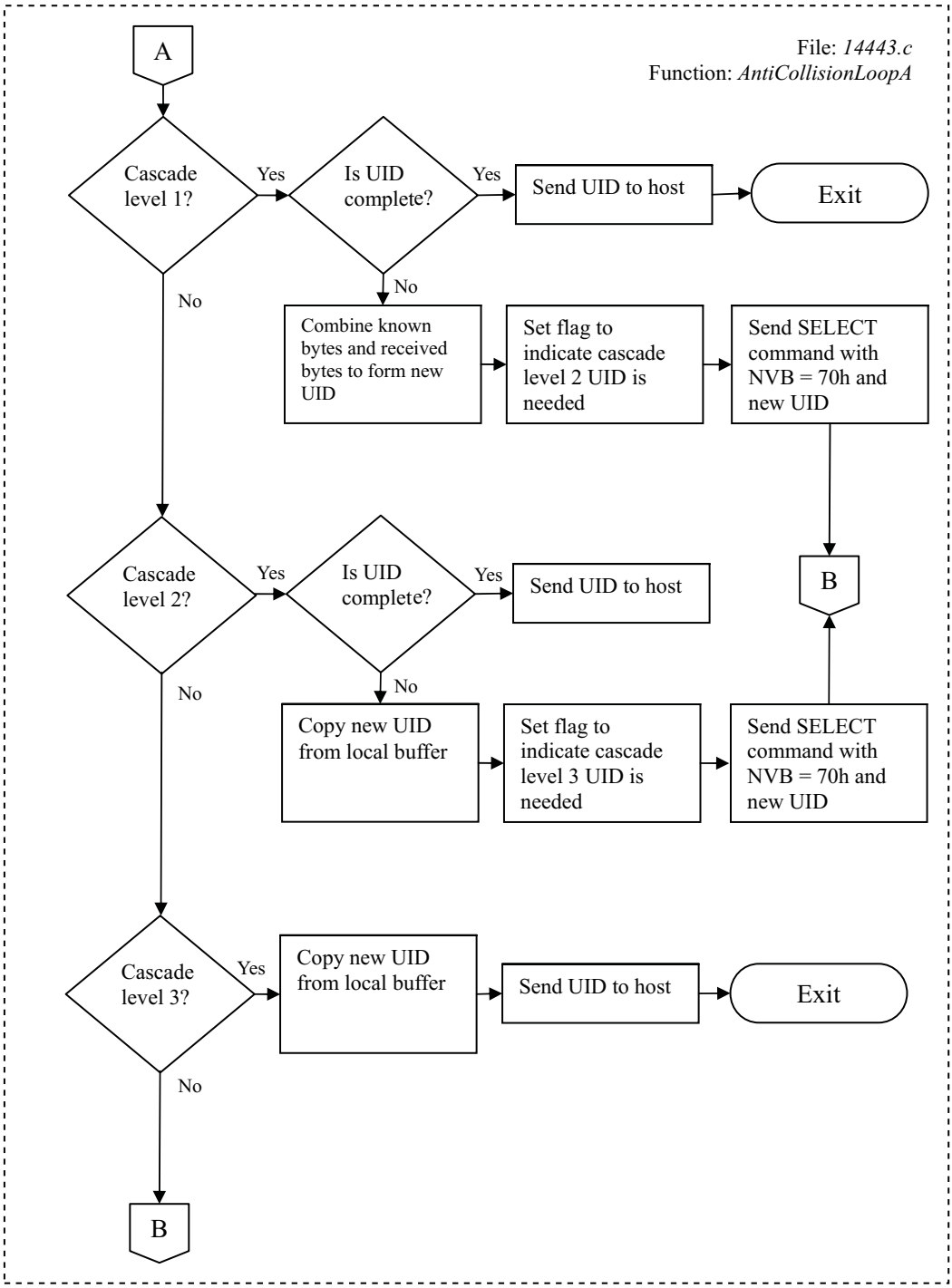

Figure 10. Anti-Collision Loop A (2)

5.3 Anti-Collision Sequence for ISO14443B

The anti-collision sequence for 14443B follows the slotted Aloha approach:

1. The PCD sends 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. When multiple PICCs respond, the PCD makes note of the collision. The PCD generates a new N and steps 1-4 are repeated.

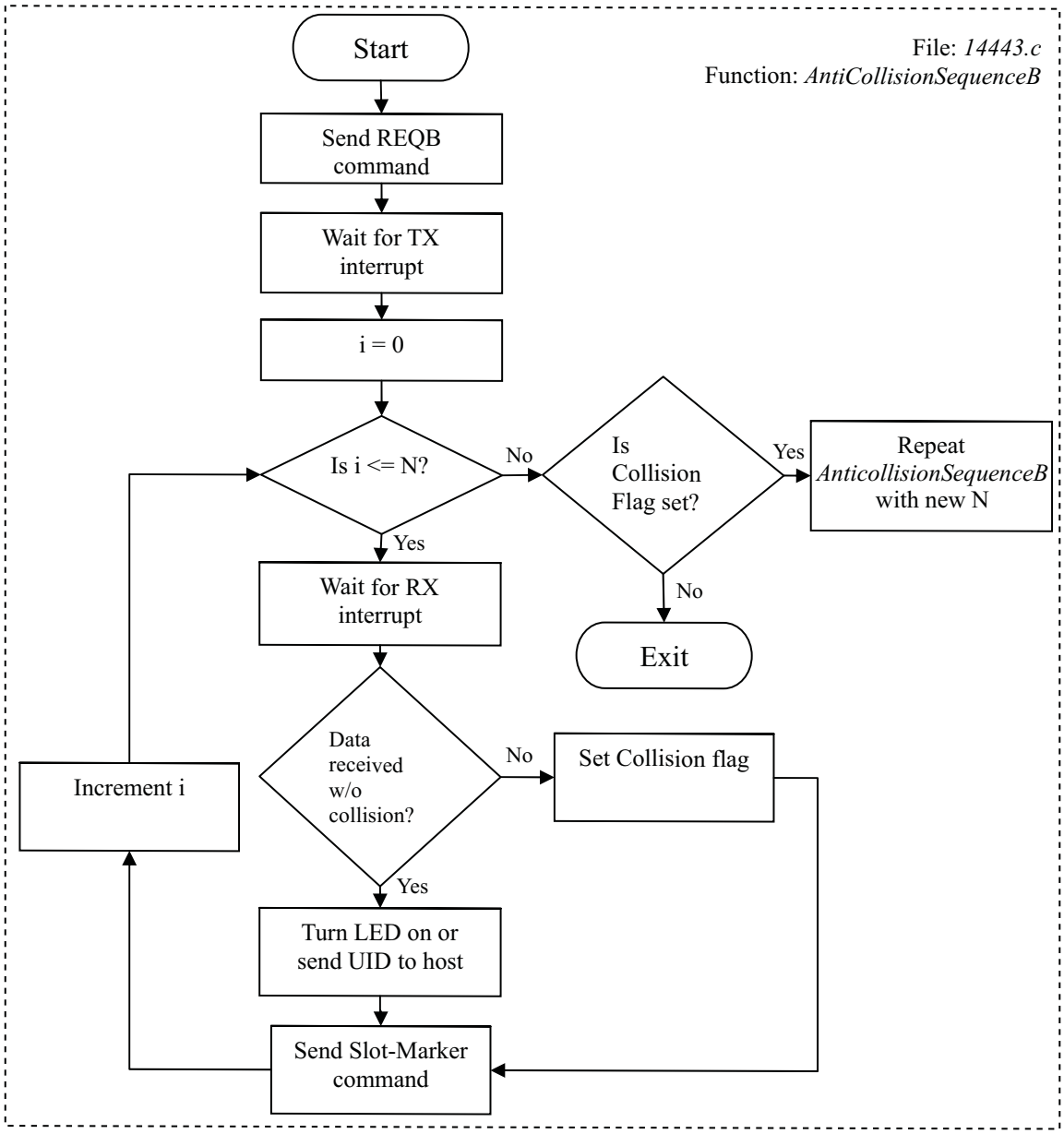

Figure 11. Anti-Collision Loop B

5.4 Anti-collision Sequence for Tag-it™

The anti-collision algorithm for Tag-it™ is the same as that of ISO15693 except that the number of slots is fixed (16).

Anti-collision algorithm:

1. The reader sends a mask value along with the inventory request. The number of slots is always 16.
2. The VICC compares its UID to the slot number + mask value. If it matches, it sends a response.
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.

Steps 1-4 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 new mask value:

1. Increment the mask length by 4.
2. Calculate New mask = Slot number (in which collision occurred) + old mask.
3. Decrement slot pointer by 1.

Repeat from start with the new mask value until slot pointer is zero.

A detailed description of the firmware implementation of the anti-collision sequence is given in Figure 12 and Figure 13.

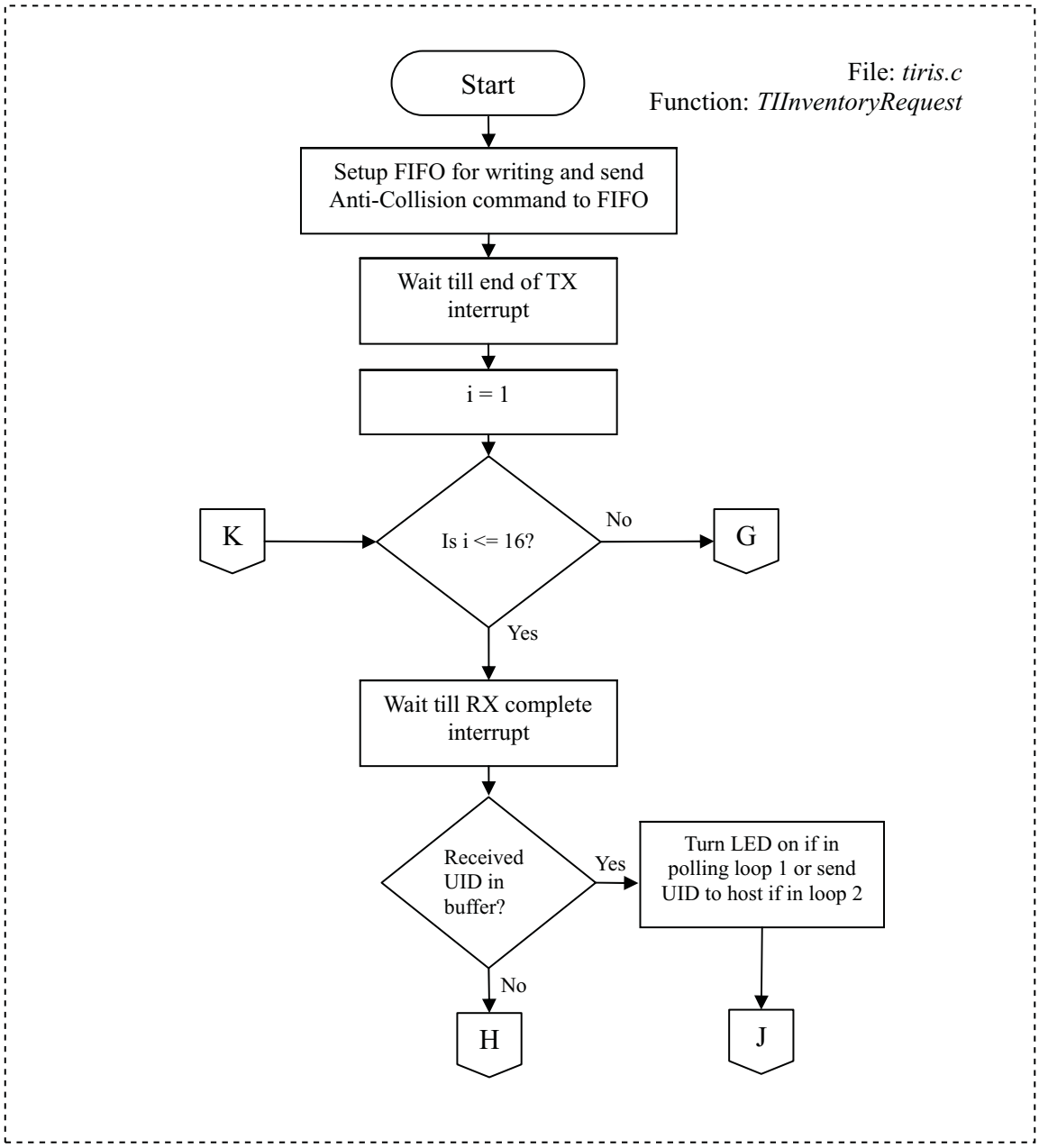

Figure 12. TI Inventory Request (1)

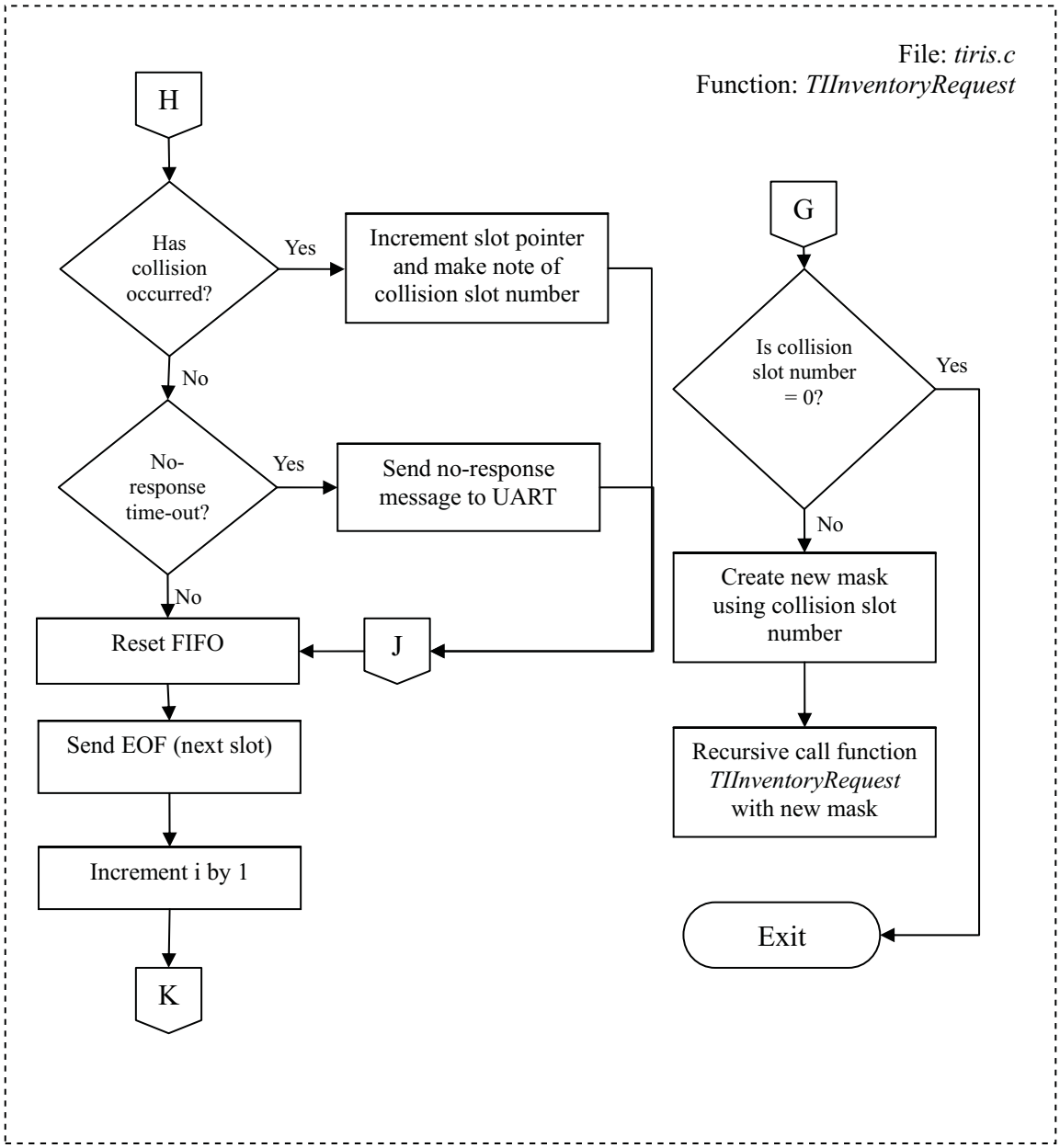

Figure 13. TI Inventory Request (2)

6 Graphical User Interface

The Graphical User Interface (GUI) (which can be used as an API) helps users to communicate with the TRF796x reader through the MCU. The GUI on the host machine (PC) issues commands to the MCU through a USB-UART converter. The MCU receives the commands in the UART receive buffer, interprets the commands and sends suitable data to the registers or FIFO buffer in the TRF796x reader. As shown in Figure 14, the UART receive buffer of the MCU is continuously scanned for data received from the PC.

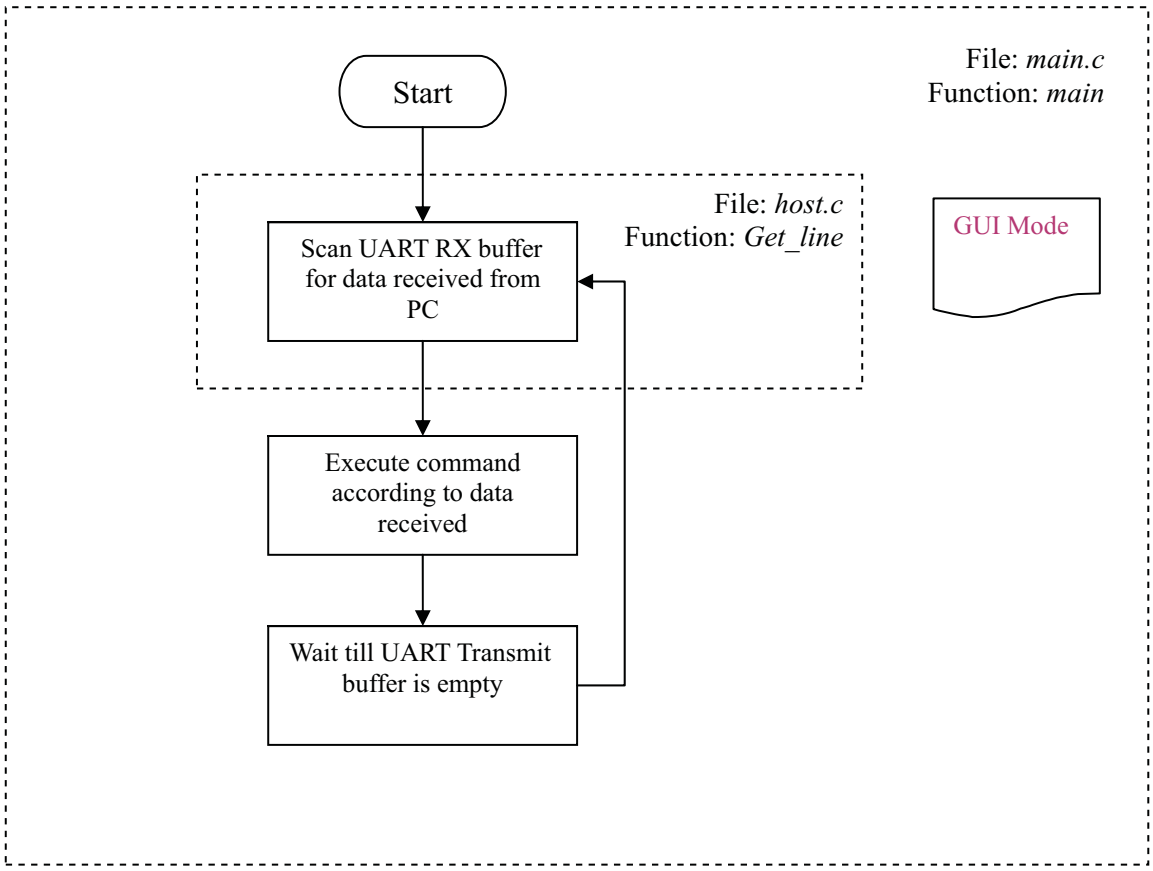

Figure 14. GUI Mode

Communications format from host to reader is organized into data frames of 6 fields.

Table 3. Data Frame

SOF (0x01)	Number of bytes	0x00	0x03-04	Command + parameters	EOF (0x00-00)

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 kept at 0x00, fourth byte at 0x03, & the fifth byte at 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 (2 bytes of 0x00).

Shown in Table 4 is a list of some of the host commands to the TRF796x. For a comprehensive listing of all the available commands, refer to the EVM User's Guide Manual or the HostCommands function in file host.c.

Table 4. Host Commands to TRF796x

COMMAND	PARAMETERS	EXAMPLE
0x10 Write single register	Address, data, address, data...	010A0003041015670000 01-0A-00-03-04-10-15-67-00-00
0x11 Write continuous	Address, data, data...	010C00030411136746A40000 01-0C-00-03-04-11-13-67-46-A4-00-00
0x12 Read single register	Address, address...	010B00030412010A130000 01-0B-00-03-04-12-01-0A-13-00-00
0x13 Read continuous	Number of bytes to read, start address	010A0003041305030000 01-0A-00-03-04-13-05-03-00-00
0x14 Inventory ISO 15693	FIFO data	010B000304140601000000 01-0B-00-03-04-14-06-01-00-00-00
0x15 Direct command	Direct command code	0109000304150F0000 01-09-00-03-04-15-0F-00-00
0x16 Write raw	Data or commands...	011000030416913D0040AABBCCDD0000 01-10-00-03-04-16-913D-00-40-AA-BBCC- DD-00-00
0x18 Request command ISO 15693, Tag-it, 14443B Halt	Flags, command code, data… (as specified in ISO and Tag-it™)	010B000304180620010000 01-0B-00-03-04-18-06-20-01-00-00
0x34 SID poll Tag-it™	Flags, command code, mask (as specified in Tag-it™)	010B000304340050000000 01-0B-00-03-04-34-00-50-00-00-00
0x0F Direct mode	-	01080003040F0000 01-08-00-03-04-0F-00-00
0x0B Configuration for 14443B (configures the ASIC)	-	01080003040B0000 01-08-00-03-04-0B-00-00
0x0C Configuration for Tag-it™ and 15693 (configures the ASIC)	-	01080003040C0000 01-08-00-03-04-0C-00-00
0xB0 14443B REQB	No. of slots (optional) 0x01, 0x02 – probabilistic approach (probability ½ and ¼) 0x04 – 16 timeslots using the Slot Marker command	0108000304B0040000 01-08-00-03-04-B0-04-00-00
0xB1 14443B WUPB	-	0108000304B1000 01-08-00-03-04-B1-0-00

7 FIFO

The TRF 796x reader contains a 12-byte FIFO buffer. The receiver in the TRF reader removes special signals like SOF, EOF, CRC bytes, etc. and places the clean or raw data in the FIFO where it can be read by the external MCU. The digital portion of the transmitter in the reader is very similar to that of the receiver. The MCU loads data into the FIFO and starts the transmit operation. The procedure to initiate transmission is as follows:

1. Start condition
2. Send Reset command 0x0F (command mode – 0x8F) to FIFO
3. Send Transmission command (0x90 - without CRC or 0x91 – with CRC)
4. Continuous write to register 0x1D (0x3D)
5. Data for register 0x1D (Upper and middle nibble of the number of bytes to be transmitted)
6. Data for register 0x1E (Lower nibble of the number of bytes to be transmitted)
7. Data byte(s) for FIFO
8. Stop condition

In this case, transmission starts when the first data byte is written into FIFO.

Note: The FIFO can be filled with data only by the Continuous Address Mode.

The FIFO should be read only after a tag reply. It cannot be read immediately after writing to it.

8 References

[1] TRF7960-61 Data Sheet (Texas Instruments Literature Number SLOU186)

[2] TRF7960 EVM User's Guide

[3] ISO/IEC FDIS 14443-3:2000(E)

[4] ISO/IEC 15693-3:2001(E)

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**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

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