AN3955
Application note

CR95HF library for ISO/IEC 14443-3 and SRIX contactless tag ICs

1 Introduction

This document describes the ISO/IEC 14443-3 libraries used by a microcontroller to drive the CR95HF 13.56 MHz multiprotocol contactless transceiver using an SPI or UART interface in order to perform wireless communications with ISO/IEC 14443 Type A or B contactless tag.

The library was developed to speed up the development of applications using the CR95HF.

The CR95HF library is composed of three layers:

- CR95HF low level layer
- Standard ISO/IEC 14443-3 protocol layer (type A and type B)
- SRIX4K product specific layer

The library code has been developed in ANSI C language, and validated on an STM32 evaluation board.

1.1 Reference documents

- CR95HF datasheet
- ISO/IEC 14443-3 specification
- SRIX4K datasheet

Contents

List of figures

List of tables

2 Acronyms and notational conventions

2.1 List of terms and acronyms

Table 1. Terms and acronyms

Terms	Definition
AFI	Application Family Identifier
ATQA	Answer to Request Type A
ATQB	Answer to Request Type B
ATS	Answer To Select
CRC	Cyclic Redundancy Check
EOF	End Of Frame
FWT	Frame Waiting Time
HLTA	Halt command Type A
HLTB	Halt command Type B
IC	Integrated Circuit
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MCU	Microcontroller Unit
NACK	Negative Acknowledge
NFC	Near Field Communication
RATS	Request for Answer To Select
REQA	Request command Type A
REQB	Request command Type B
RF	Radio Frequency
RFID	Radio Frequency Identification
RFU	Reserved for Future Use
SAK	Select AcKnowledge
SOF	Start Of Frame
SRI	Short Range Interface
SRIX	Short Range Interface with anti clone function
UID	Unique Identifier
WUPA	Wake-Up A command Type A
WUPB	Wake-Up A command Type B
XOR	eXclusif OR

2.2 Notational conventions

The following conventions and notations apply in this document unless otherwise stated.

2.2.1 Binary number representation

Binary numbers are represented by strings of digits 0 and 1 shown with the Most Significant Bit (MSB) on the left, the Least Signifcant Bit (LSB) on the right, and “0b” added at the beginning.

For example: 0b11110101

2.2.2 Hexadecimal number representation

Hexadecimal numbers are represented by using the numbers 0 to 9, the characters A - F, and a “0x” added at the beginning. The Most Significant Byte (MSB) is shown on the left and the Least Significant Byte (LSB) on the right.

For example: 0xF5

2.2.3 Decimal number representation

Decimal numbers are represented as is, without any trailing character.

For example: 245

3 Overview

3.1 CR95HF overview

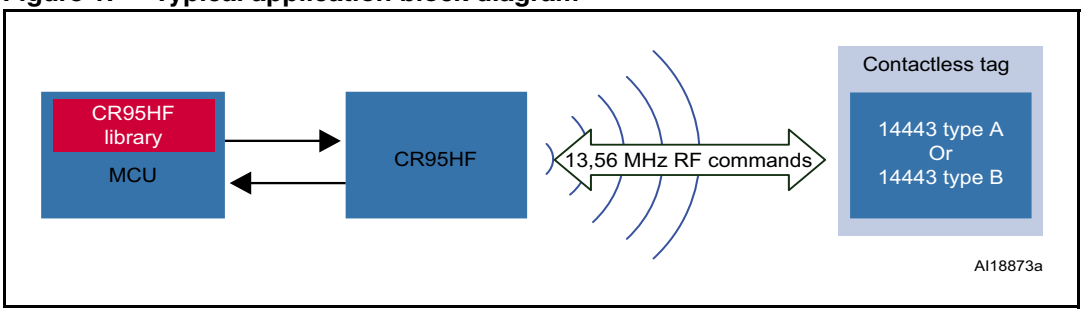
The CR95HF is a Radio Frenquency (RF) transceiver Integrated Circuit (IC) for 13.56 MHz contactless tags, which includes ISO/IEC 14443, ISO/IEC 15693 and ISO/IEC 18092 protocols. It manages the RF communication with Radio Frequency Identification (RFID) or Near Field Communication (NFC) tags. It includes trame coding, RF modulation and contactless tag response decoding.

The CR95HF is a slave device and must be controlled by a host (Microcontroller Unit). This library is an interface between user application function and standard peripheral driver.

The library was written in compliance with ANSI C standards.

Figure 1 describes a typical application block diagram.

Figure 1. Typical application block diagram



For more details concerning the CR95HF, please refer to CR95HF datasheet.

3.2 ISO/IEC 14443-3 overview

The ISO/IEC 14443-3 is a standard, which specifies two different RF protocols (Type A and Type B). It describes low-level functions allowing to handle one ore more contactless tags. As an example, the ISO/IEC 14443-3 commands can be used to read the Unique Identifier (UID) or to perform anti-collision.

The higher level commands to read or write the user memory of contactless tags are not described in ISO/IEC 14443-3 specifications. These commands are defined in the contactless datasheet. For instance, the SRIX4K datasheet defines Read, Write and other specific commands.

For more information about Read, Write or other commands, refer to the SRIX4K datasheet.

Since the frames coding specified by ISO/IEC 14443 Type A differ from Type B, the CR95HF cannot communicate with an ISO/IEC 14443 Type A and an ISO/IEC 14443 Type B contactless tag at the same time.

3.3 SRIX4K product overview

The SRIX4K is an STMicroelectronics contactless tag powered by an RF field. It contains a 4-kbit EEPROM memory that can be accessed by using RF commands.

For more information about low-level commands to handle a tag and an upper level to read, write or lock memory, refer to the SRIX4K datasheeet.

SRIX4K is compliant with ISO/IEC 14443-3 Type B frame format specification.

3.4 Library overview

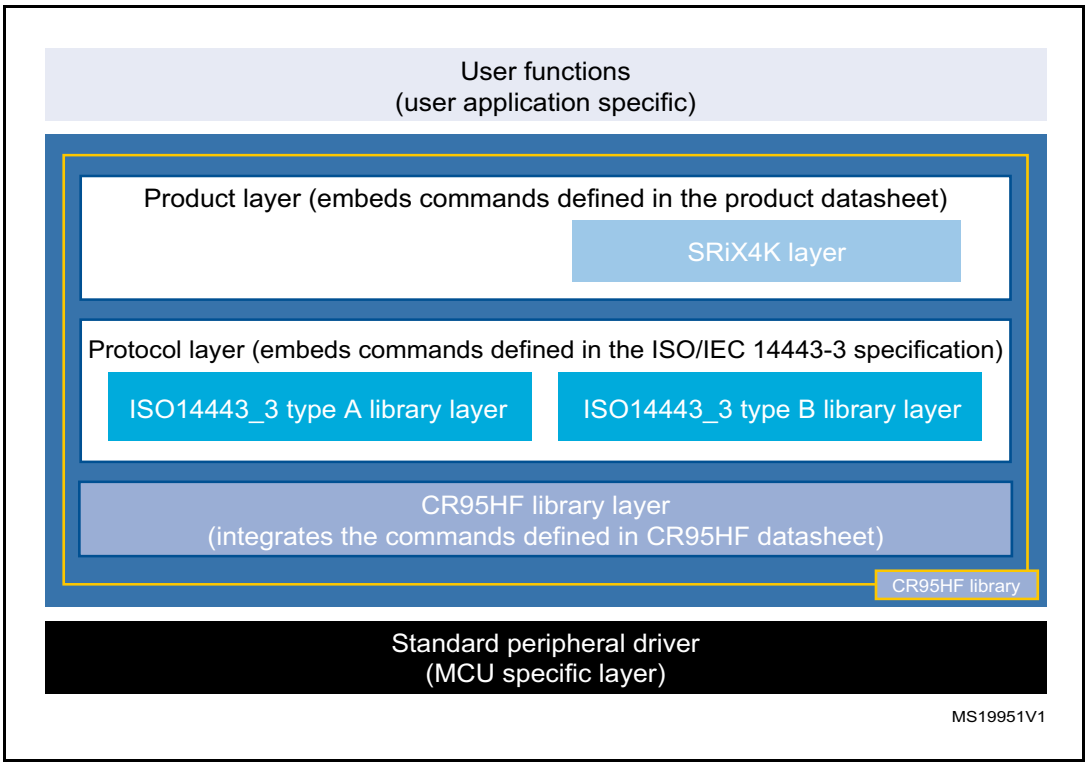
The library described in this application note is composed of three layers:

- A low-level layer supporting the commands described in the CR95HF datasheet. This level is fully supported by the CR95HF library.
- An intermediate layer based on the ISO/IEC 14443-3 protocol. This layer is divided in two files, one for Type A and another for Type B.
- An upper layer supporting the commands described in the SRIX4K datasheet.

The library can be downloaded from STMicroelectronics internet site at http://www.st.com.

Figure 2 shows an example of application architecture and describes the interaction between a typical user application and the CR95HF library layers.

Figure 2. Interaction between typical user application and CR95HF library layers



4 CR95HF low level layer

For a brief description of CR95HF, refer to *Section 3.1: CR95HF overview*.

This layer is composed of:

- the `lib_CR95HF.c` source file
- the `lib_CR95HF.h` include file

4.1 Types

The CR95HF library functions use the following ANSI C compliant types:

```
typedef       unsigned char      uint8_t;
typedef       signed char        int8_t;
typedef       const uint8_t      uc8;
typedef       signed short int   int16_t;
```

4.2 Definition

The layer supports all the commands defined in the CR95HF datasheet. Each CR95HF command corresponds to a specific function. As an example, the function calling the `ECHO` command is `int8_t CR95HF_Echo (uint8_t *pResponse)`.

Additional functions are described in *Table 3: CR95HF layer additional functions*.

Application developers can use the functions described in *Table 2: CR95HF layer functions based on CR95HF commands* to create their own higher level functions.

For more details about commands, refer to the CR95HF datasheet.

4.3 CR95HF layer functions

The tables below introduce the functions available in the CR95HF layer.

Table 2. CR95HF layer functions based on CR95HF commands

Function name	Brief description
`CR95HF_IDN`	Sends an IDN command to CR95HF device. It returns its version number. For more details about `CR95HF_IDN` function, refer to *Section 4.4: IDN function*.
`CR95HF_Echo`	Sends an Echo command to CR95HF device which returns an Echo response. The Echo function checks if communications can be started between the MCU and the CR95HF. For more details about `CR95HF_Echo` function, refer to *Section 4.5: ECHO function*.
`CR95HF_ProtocolSelect`	Sends a ProtocolSelect command to CR95HF device. It selects the RF communication protocol, configures RF parameters and switches the RF field on. For more details about `CR95HF_ProtocolSelect` function, refer to *Section 4.6: ProtocolSelect function*.
`CR95HF_SendRecv`	Sends a SendRecv command. The parameter sent to the command is the frame which is coded according to the protocol previously selected by issuing a ProtocolSelect command. For more details about `CR95HF_SendRecv` function, refer to *Section 4.7: SendRecv function*.
`CR95HF_Idle`	Sends an Idle command to the CR95HF device to switch it into low consumption mode. For more details about `CR95HF_Idle` function, refer to *Section 4.8: Idle function*.
`CR95HF_RdReg`	Sends a ReadRegister command to the CR95HF device to read the CR95HF internal register. For more details about `CR95HF_RdReg` function, refer to *Section 4.9: RdReg function*.
`CR95HF_BaudRate`	Sends a Baudrate command to the CR95HF device. It changes the CR95HF UART baudrate. For more details about `CR95HF_BaudRate`function, refer to *Section 4.10: BaudRate function*.

Table 3. CR95HF layer additional functions

Function name	Brief description
CR95HF_SendEOF	Sends an RF pulse (EOF). For more details about CR95HF_SendEOF function, refer to *Section 4.11: SendEOF function*.
CR95HF_FieldOff	Switches off the RF field. For more details about CR95HF_FieldOff function, refer to *Section 4.12: FieldOff function*.
CR95HF_HexCommandToStringCommand	Translates hexadecimal command code to command name. It returns the ASCII code of the command. For more details about CR95HF_HexCommandToStringCommand function, refer to *Section 4.13: HexCommandToStringCommand function*.
CR95HF_GetReaderErrorCode	Translates hexadecimal error code to error code description. It returns a description of the error contained in the ReaderReply parameter. For more details about CR95HF_GetReaderErrorCode, refer to *Section 4.17: GetReaderErrorCode function*.
CR95HF_IsReaderResultCodeOk	Checks if returned code is a successful code. It returns CR95HF_SUCCESS_CODE if the CR95HF command is successful. For more details about CR95HF_IsReaderResultCodeOk function, refer to *Section 4.14: IsReaderResultCodeOk function*.
CR95HF_IsReaderErrorCode	Checks if returned code is an error code. It returns CR95HF_SUCCESS_CODE value if the CR95HF command has returned an error code. For more details about CR95HF_IsReaderErrorCode function, refer to *Section 4.15: IsReaderErrorCode function*.
CR95HF_IsCommandExists	Checks if command is available. It returns CR95HF_SUCCESS_CODE if the CmdCode value exists. For more details about CR95HF_IsCommandExists function, refer to *Section 4.16: IsCommandExists function*.

4.4 IDN function

The `CR95HF_IDN` function sends an IDN command to the CR95HF device.

Table 4. CR95HF_IDN function description

Prototype	`result = int8_t CR95HF_IDN (uint8_t *pResponse)`
Input parameter	None
Output parameter	**pResponse**: Pointer to CR95HF response
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful.

4.5 ECHO function

The `CR95HF_Echo` function sends an ECHO command to the CR95HF device.

Table 5. CR95HF_Echo function description

Prototype	`result = int8_t CR95HF_Echo (uint8_t *pResponse)`
Input parameter	None
Output parameter	**pResponse**: Pointer to CR95HF response
Return parameter	**CR95HF_SUCCESS_CODE:** Function is successful.

4.6 ProtocolSelect function

The `CR95HF_ProtocolSelect` function sends a ProtocolSelect command to the CR95HF device.

To set up communications with a contactless tag, the ProtocolSelect command must be sent to the CR95HF before the SendRecv commands.

Table 6. CR95HF_ProtocolSelect function description

Prototype	`result = int8_t CR95HF_ProtocolSelect(uc8 Length,uc8 Protocol,uc8 *Data,uint8_t *pResponse)`
Input parameter	**Length**: Number of data bytes **Protocol**: Type of protocol(1) **Data** input: Pointer to data
Output parameter	**pResponse**: Pointer to CR95HF response
Return parameter	**CR95HF_ERRORCODE_PARAMETER**: Function failed **CR95HF_SUCCESS_CODE**: Function is successful. **CR95HF_ERRORCODE_PARAMETERLENGTH**: Parameter length is incorrect.

1. ISO/IEC 15693, ISO/IEC 18092, ISO/IEC 14443-3 Type A or Type B RF protocols.

The Data input parameter depends on the selected protocol. This application note applies to ISO/IEC 14443-3 Type A and ISO/IEC 14443-3 Type B products.

Table 7 describes the ProtocolSelect command in ISO/IEC 14443-3 Type A case.

Table 7. CR95HF_ProtocolSelect ISO/IEC 14443-3 Type A protocol parameter

Parameter name	Byte	Bit	Value	Example
Command Code	0	7:0	0x02	0x02020200 ISO/IEC 14443-3 Type A with reception and transmission data rates at 106 kbps
Length	1	7:0	0x02	
Protocol	2	7:0	0x02: ISO/IEC 14443-3 Type A protocol	
Parameter	3	7:6	Transmission data rate	
		5:4	Reception data rate	
		3:0	RFU	
	5:4	7:0	FWT= (256*16/fc)*(2*PP)*(MM+1)(1) (2) where “PP” represents byte 4 and “MM” byte 5.	

1. Optional. If not specify default FWT~86 µs
2. PP <= 0x0E and 0x01 <= MM <= 0xFE

Table 8 describes the ProtocolSelect command in ISO/IEC 14443-3 Type B case.

Table 8. CR95HF_ProtocolSelect ISO14443-3 Type B protocol parameter

Parameter name	Byte	Bit	Value	Example
Command Code	0	7:0	0x02	0x02020301 ISO/IEC 14443-3 Type B with reception and transmission data rates at 106 kbps The CR95HF manages the CRC.
Length	1	7:0	0x02	
Protocol	2	7:0	0x03: ISO/IEC 14443-3 Type B protocol	
Parameter	3	7:6	Transmission data rate	
		5:4	Reception data rate	
		3:0	RFU	
		0	Append CRC 1: CR95HF appends CRC 0: CR95HF does not append CRC	
	5:4	7:0	FWT= (256*16/fc)*(2*PP)*(MM+1)(1) (2) where “PP” represents byte 4 and “MM” byte 5.	

1. Optional. If not specify default FWT~300µs.
2. PP <= 0x0E and 0x01 <= MM <= 0xFE

4.7 SendRecv function

The `CR95HF_SendRecv` function sends a SendRecv command. The CR95HF encodes the frame, transmits it at 13.56 MHz, and decodes the contactless tag response.

The contactless tag response is then sent back in the `pResponse` parameter.

Table 9. CR95HF_SendRecv function description

Prototype	`result = int8_t CR95HF_SendRecv (uc8 Length,uc8 *Parameters,uint8_t *pResponse)`
Input parameter	**Length**: Number of bytes in Parameters input **Parameters** input: Pointer to data
Output parameter	**pResponse** : Pointer to CR95HF response
Return parameter	**CR95HF_ERRORCODE_PARAMETER**: Function failed **CR95HF_SUCCESS_CODE**: Function is successful. **CR95HF_ERRORCODE_PARAMETERLENGTH**: Parameter length is incorrect.

The `Parameters` input depends on the selected protocol. This application note applies to ISO/IEC 14443-3 Type A and ISO/IEC 14443-3 Type B products.

Table 10 describes the SendRecv command in ISO/IEC 14443-3 Type A case.

Table 10. SendRecv command in ISO/IEC 14443-3 Type A case

Parameter name	Byte	Bit	Value	Comments	Example
Command Code	0	7:0	0x04	-	REQA command: 0x04022607 with 26: Data (REQA command code) 07: number of significant bits (REQA is coded on 7 bits)
Length	1	7:0	0xXX	Number of Data bytes and transmission flag	
Data	2:X	7:0	0xXX	Data	
Transmission flag	X+1	7	-	RFU	
		6	0b0	Split frame	
		5	1: CR95HF appends CRC 0: CR95HF does not append CRC	Append CRC	
		4	0b0	RFU	
		3:0	0xX	Number of significant bits	

Table 11 describes the SendRecv command in ISO/IEC 14443-3 Type B case.

Table 11. SendRecv command in ISO/IEC 14443-3 Type B case

Parameter name	Byte	Bit	Value	Comments	Example
Command Code	0	7:0	0x04	-	REQB command: 0x040305000
Length	1	7:0	0xXX	Length of entire data	
Data	2:X	7:0	0xXX	Data	

For more information, refer to CR95HF datasheet. *Table 12* describes the CR95HF success response value for ISO/IEC 14443-3 Type A.

Table 12. CR95HF success response value for ISO/IEC 14443-3 Type A

Parameter name	Byte	Bit	Value	Comments	Example
Result Code	0	7:0	0x04	Success code	0X80054400280000 where: 80: success 05: Number of bytes 4400: ATQA 28: CRC error(1) & Number of significant bits: 8 00 00: No collision detected
Length	1	7:0	0xXX	Number of bytes of Data + control bytes	
Data	2:X	7:0	0xXX	Contactless tag response	
Control bytes	X+1	7	1: CR95HF has detected a collision 0: No collision detected	Collision between two contactless tag responses	
		6	0	RFU	
		5	1: CRC error 0: No CRC error	CRC error on contactless response	
		4	1: Parity error 0: No Parity error	Parity error on contactless response	
		3:0		Number of significant bits in the first byte(2)	
	X+2	7:0	0xXX	Index of the first byte where the collision is detected(3)	
	X+3	7:4	0b000	RFU	
		3:0	0bXXXX	Index of the first bit where the collision is detected	

1. CRC error bit is set but there is no CRC on ATQA response. So this bit is not significant.
2. Since the ISO/IEC 14443-3 Type A is a bit-oriented protocol, CR95HF can receive a non integer amount of bytes.
3. To calculate the point of collision between two contactless tag responses, the user application must refer to the control bytes. The byte where the collision is detected is indicated by Control Byte X+2 (Index of first byte), and the specific bit in the byte where the collision is detected is indicated by Control Byte X+3 (Index of first bit). Both indices start from 0 and the bit index can be 8, meaning that the collision affected parity. This information is only valid if the Collision Dectected bit (Bit 7 of control Byte X+1) is set.

CR95HF returns a specific correct code for ACK or NAK contactless response, coded on 4 bits, as described in *Table 13*.

Table 13. Specific success code for ACK or NAK contactless response

Parameter name	Byte	Bit	Value	Comments
Result Code	0	7:0	0x90	Success code
Length	1	7:0	0x04	Number of valid bits: 4
Data	2:X	7:0	0x01: NAK 0x05: NAK 0x0A: ACK	Contactless tag response

Table 14 describes the CR95HF success response value for ISO/IEC 14443-3 Type B.

Table 14. CR95HF success response value for ISO/IEC 14443-3 Type B

Parameter name	Byte	Bit	Value	Comments	Example
Result Code	0	7:0	0x80	Success code	0x800F50AA0B000500000000B371716A2100 where: 80: Success 0F: Number of byte 50AA0B000500000000B371716A21: ATQB 00: CRC Ok
Length	1	7:0	0xXX	Number of bytes of Data + control bytes	
Data	2:X	7:0	0xXX	Contactless tag response	
Control bytes	X+1	7:2	0b0000	RFU	
		1	1: CRC error 0: no CRC error	CRC error or contactless response	
		0	-	RFU	

For more information, refer to CR95HF datasheet.

If CR95HF_SendRecv function fails, the CR95HF returns an error code in the pResponse parameter (see *Table 15*). As an example, if no contactless tag is present in the RF field, the CR95HF response is 0x8700.

Table 15. CR95HF error response value for ISO/IEC 14443-3 Type A and B

Parameter name	Byte	Bit	Value	Example
Result code	0	7:0	0x08	0x8700: no contactless tag in the field
Length	1	7:0	0x00	

For more information about CR95HF error code, refer to CR95HF datasheet.

4.8 Idle function

This function sends an Idle command to the CR95HF to switch the CR95HF device into low consumption mode. *Table 16* describes the `CR95HF_Idle` function.

Table 16. CR95HF_Idle function description

Prototype	`result = int8_t CR95HF_Idle(uc8 Length,uc8 *Data,uint8_t *pResponse);`
Input parameter	**Length**: Number of data bytes **Data**: Pointer to data
Output parameter	**pResponse** : Pointer to the CR95HF response
Return parameter	**CR95HF_ERRORCODE_PARAMETER**: Function failed **CR95HF_SUCCESS_CODE:** Function is successful. **CR95HF_ERRORCODE_PARAMETERLENGTH**: Parameter length is incorrect.

4.9 RdReg function

This function sends an RdReg command to the CR95HF in order to read the CR95HF internal register. *Table 17* describes the `CR95HF_RdReg` function.

Table 17. CR95HF_RdReg function description

Prototype	`result = int8_t CR95HF_RdReg(uc8 Length,uc8 Address,uc8 RegCount,uc8 Flags,uint8_t *pResponse);`
Input parameter	**Length** : Number of bytes (Address+RegCount+Flags) **Address:** Register address **RegCount:** Number of bytes to read **Flags:** ST reserved (must be 0x00)
Output parameter	**pResponse** : Pointer to CR95HF response
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful.

4.10 BaudRate function

This function sends a BaudRate command to the CR95HF. It allows configuring the UART baudrate. *Table 18* describes the `CR95HF_BaudRate` function.

Table 18. CR95HF_BaudRate function description

Prototype	`result = int8_t CR95HF_BaudRate (uc8 BaudRate,uint8_t *pResponse);`
Input parameter	**BaudRate:** New baud rate = 13.56/(2*BaudRate+2) Mbps
Output parameter	**pResponse** : Pointer to the CR95HF response
Return parameter	**CR95HF_SUCCESS_CODE:** Function is successful.

For more information about this command, please refer to CR95HF datasheet.

4.11 SendEOF function

This function emits an RF pulse (EOF). *Table 19* describes the `CR95HF_SendEOF` function.

Table 19. CR95HF_SendEOF function description

Prototype	`result = int8_t CR95HF_SendEOF(uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful.

4.12 FieldOff function

This function switches off the RF field. *Table 20* describes the `CR95HF_FieldOff` function.

Table 20. CR95HF_FieldOff function description

Prototype	`result = int8_t CR95HF_FieldOff(void);`
Input parameter	None
Output parameter	None
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful.

4.13 HexCommandToStringCommand function

This function returns the command ASCII code. *Table 21* describes the `CR95HF_HexCommandToStringCommand` function.

Table 21. CR95HF_HexCommandToStringCommand function description

Prototype	`result = int8_t CR95HF_HexCommandToStringCommand (uint8_t CmdCode, uint8_t *StringCommand, uint8_t *CmdNameNbByte);`
Input parameter	**CmdCode**: Command code (one byte)
Output parameter	**StringCommand**: Command string (ASCII format) **CmdNameNbByte**: Number of bytes in the command code
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful.

4.14 IsReaderResultCodeOk function

This function returns `CR95HF_SUCCESS_CODE` if the CR95HF command has succeeded. *Table 22* describes the `CR95HF_IsReaderResultCodeOk` function.

Table 22. CR95HF_IsReaderResultCodeOk function description

Prototype	`result = int8_t CR95HF_IsReaderResultCodeOk (uint8_t CmdCode,uc8 *ReaderReply);`
Input parameter	**CmdCode**: Command code (one byte) **ReaderReply**: Pointer to CR95HF response
Output parameter	None
Return parameter	**CR95HF_NOREPLY_CODE**: No CR95HF response **CR95HF_SUCCESS_CODE**: Function is successful. **CR95HF_ERROR_CODE**: Function failed. CR95HF does not return a correct code.

4.15 IsReaderErrorCode function

This function returns `CR95HF_SUCCESS_CODE` value if the CR95HF command has returned an error code. *Table 23* describes the `CR95HF_IsReaderErrorCode` function.

Table 23. CR95HF_IsReaderErrorCode function description

Prototype	`result = int8_t CR95HF_IsReaderErrorCode (uint8_t CmdCode,uint8_t *ReaderReply);`
Input parameter	**CmdCode**: Command code (one byte) **ReaderReply**: Pointer to CR95HF response
Output parameter	None
Return parameter	**CR95HF_NOREPLY_CODE**: No CR95HF response **CR95HF_SUCCESS_CODE**: Function is successful. **CR95HF_ERROR_CODE**: Function failed. CR95HF does not return an error code.

4.16 IsCommandExists function

This function returns `CR95HF_SUCCESS_CODE` if the `CmdCode` value exists. *Table 24* describes the `CR95HF_IsCommandExists` function.

Table 24. CR95HF_IsCommandExists function description

Prototype	`result = int8_t CR95HF_IsCommandExists(uint8_t CmdCode);`
Input parameter	**CmdCode**: Command code (one byte)
Output parameter	None
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful. **CR95HF_ERRORCODE_COMMANDUNKNOWN**: The command code is unknown.

4.17 GetReaderErrorCode function

This function returns a description of the error contained in the `ReaderReply` parameter. *Table 25* describes the `CR95HF_GetReaderErrorCode` function.

Table 25. CR95HF_GetReaderErrorCode function description

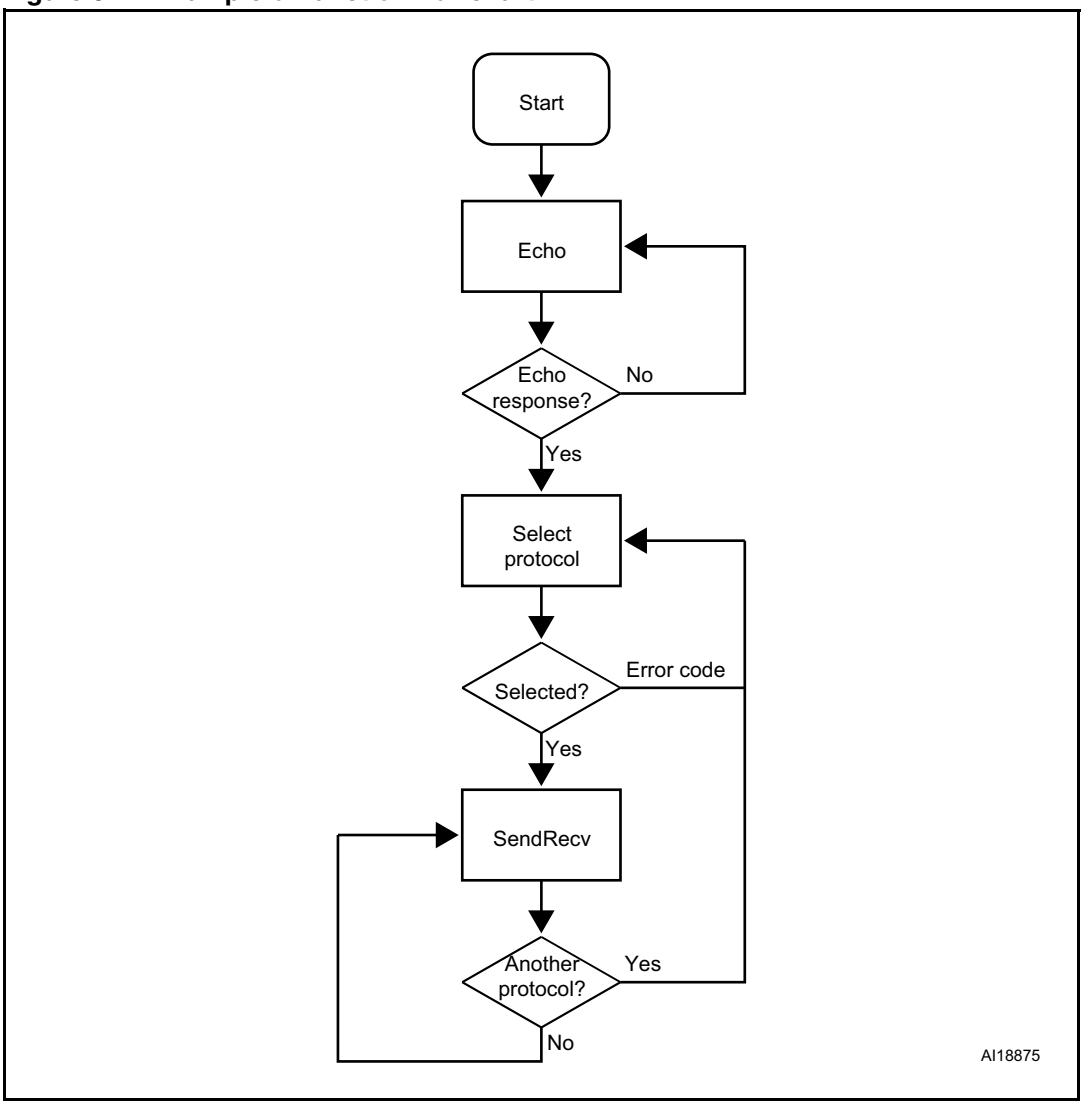
Prototype	`result = int8_t CR95HF_GetReaderErrorCode (uc8 CmdCode,uc8 *ReaderReply,char *ErrorDescription);`
Input parameter	**CmdCode**: Command code (one byte) **ReaderReply**: Pointer to CR95HF response
Output parameter	**ErrorDescription**: Pointer to ASCII string
Return parameter	**CR95HF_SUCCESS_CODE**: Function is successful. **ERRORCODE_GENERIC**: Function failed. CR95HF does not return an error code.

4.18 Application example: protocol selection and communication

To communicate with a contactless tag, the application must select the RF protocol by sending a ProtocolSelect command. Then, the application can use SendRecv commands to send data to a contactless tag.

The user can select another protocol or change RF parameters (e.g. choose another datarate) at any time by issuing again a ProtocolSelect command. *Figure 3* shows an example of function flowchart.

Figure 3. Example of function flowchart



5 ISO/IEC 14443-3 layers

For a brief description of ISO/IEC 14443-3 layers, refer to *Section 3.2: ISO/IEC 14443-3 overview*.

The ISO/IEC 14443-3 layers include two layers:

- the ISO/IEC 14443-3 Type A layer
- the ISO/IEC 14443-3 Type B layer

5.1 ISO/IEC 14443-3 Type A layer

The ISO/IEC 14443-3 Type A layer is composed of:

- the `lib_iso14443A.c` source file
- the `lib_iso14443A.h` include file

5.1.1 ISO/IEC 14443-3 Type A command format

The ISO/IEC 14443-3 Type A specification defines two command framings: a short frame and a standard frame.

- Short frame:

SOF	Data	CRC	EOF

with:

SOF: Start Of Frame

Data: 7 bits

EOF: End Of Frame

- Standard frame:

SOF	Command code	Parity	Data	Parity	Data	Parity	...	EOF

with:

SOF: Start Of Frame

Command code: 1 byte

Parity: 1 bit

Data: 1 byte

EOF: End Of Frame

5.1.2 EOF and SOF

The EOF and SOF are managed by the CR95HF.

5.1.3 Parity bit management

The parity bit of a standard frame is managed by the CR95HF.

5.1.4 ISO/IEC 14443-3 Type A CRC16 management

The ISO/IEC 14443-3 Type A specification defines a two-byte CRC. It is appended to some RF commands to check the data transmission between CR95HF and a contactless tag.

The 5th bit of the transmission flag of SendRecv command controls the CRC. When this bit is set to 1, the CR95HF appends the CRC to RF commands.

Table 26 lists the ISO/IEC 14443-3 Type A commands and notifies if a CRC is required. The functions of ISO/IEC 14443-3 Type A layer listed in *Table 26* manage the transmission flag and the CRC.

Table 26. ISO/IEC 14443-3 Type A commands

ISO/IEC 14443-3 Type A command	CRC	SendRecv example	Comments	ISO/IEC 14443_A function
REQA	No	04 02 26 07	Command on 7 bits without CRC	`ISO14443A_REQA`
WUPA	No	04 02 52 07	Command on 7 bits without CRC	`ISO14443A_WUPA`
HLTA	Yes	04 03 50 00 28	The last byte has 8 bits with CRC	`ISO14443A_HLTA`
Anticollision	No	04 02 93 20 08	The last byte has 8 bits without CRC	`ISO14443A_AntiCollisionLevelX`
Select	Yes	04 02 93 70 88...28	The last byte has 8 bits with CRC	`ISO14443A_SelectLevelX`

If the user application does not use the previous functions, it has to manage the CRC by using bit 5 of the transmission flag.

5.1.5 Functions of ISO/IEC 14443_A layer

The `lib_iso14443A.c` file includes all the commands defined in the ISO/IEC 14443-3 Type A specifications.

Table 27 lists the command to configure the CR95HF in order to communicate with an ISO/IEC 14443-3 Type A tag.

Table 27. Configuration of the CR95HF

Function	Description
`ISO14443A_ProtocolSelect`	Configures CR95HF to communicate with an ISO/IEC 14443 Type A contactless tag.

Table 28 lists the functions of this layer which emit an ISO/IEC 14443-3 Type A command.

Table 28. ISO/IEC 14443-3 Type A layer function description

Function	Description
`ISO14443A_REQA`	Sends a REQA command to a contactless tag.
`ISO14443A_WUPA`	Sends a WUPA command to a contactless tag.
`ISO14443A_HLTA`	Sends an HLTA command to a contactless tag.
`ISO14443A_AntiCollisionLevel1`	Sends an anti-collision command (Level 1) to a contactless tag.
`ISO14443A_AntiCollisionLevel2`	Sends an anti-collision command (Level 2) to a contactless tag.
`ISO14443A_AntiCollisionLevel3`	Sends an anti-collision command (Level 3) to a contactless tag.
`ISO14443A_SelectLevel1`	Sends a select command (Level 1) to a contactless tag.
`ISO14443A_SelectLevel2`	Sends a select command (Level 2) to a contactless tag.
`ISO14443A_SelectLevel3`	Sends a select command (Level 3) to a contactless tag.

A CR95HF response includes different information fields extracted by the following functions. *Table 29* describes the `ISO14443A_ProtocolSelect` function.

Table 29. ISO14443A_ProtocolSelect function description

Function	Description
`ISO14443A_ProtocolSelect`	Configures CR95HF to communicate with an ISO/IEC 14443 Type A contactless tag.

The Is functions return RESULTOK if the function is successful, otherwise the Is functions return ERRORCODE_GENERIC. *Table 30* lists the Is functions.

Table 30. ISO/IEC 14443-3 Type A Is function description

Function	Description
ISO14443A_IsCollisionDetected	Checks if the collision bit of the CR95HF control byte is set (collision detected).
ISO14443A_IsCRCError	Checks if the CRC bit of the CR95HF control byte is set (CRC error detected).
ISO14443A_IsParityError	Checks if the Parity bit of the CR95HF control byte is set (parity error detected).
ISO14443A_IsCorrectBCC	Checks if BCC byte matches the data.
ISO14443A_IS14443_4Compatible	Checks if the contacless tag is compliant with ISO/IEC 14443_4 specification.
ISO14443A_ISUIDComplete	Checks if the UID is completed.
ISO14443A_IsPresent	Checks if a 14443-3 Type A tag is present in the RF field.

The GET functions return an information field. *Table 31* describes the GET functions.

Table 31. ISO/IEC 14443-3 Type A GET function description

Function	Description
ISO14443A_GetSignificantBit	Returns the number of significant bit of the byte. This information is extracted from control bytes of CR95HF.
ISO14443A_GetFirstCollisionByte	Returns the index of the first byte where the collision is detected. This information is extracted from control bytes of CR95HF.
ISO14443A_GetFirstCollisionBit	Returns the index of the first bit where the collision is detected. This information is extracted from control bytes of CR95HF.

The advanced functions use the select sequence of one tag to manage a specific sequence. *Table 32* describes the advanced functions.

Table 32. ISO/IEC 14443-3 Type A Advanced function description

Function	Description
ISO14443A_SelectSequence	Runs a select sequence as defined in ISO/IEC 14443-3 specification.
ISO14443A_GetUIDsize	Returns UID size (information extracted from ATQA contactless tag response).
ISO14443A_GetUID	Returns UID of a contacless tag.

5.1.6 CR95HF configuration function

The CR95HF configuration function sends a ProtocolSelect command to CR95HF device in order to select the ISO/IEC 14443-3 Type A protocol. The transmission and reception data rates are integrated in the command. This command turns on the RF field.

Table 33 describes the `ISO14443A_ProtocolSelect` function.

Table 33. ISO14443A_ProtocolSelect function description

Prototype	`int8_t ISO14443A_ProtocolSelect (uc8TXDataRate, uc8 RxDataRate, uint8_t*pResponse);`
Input parameter	**TxDataRate**: Transmission data rate (2 bits) **RxDataRate**: Reception data rate (2 bits)
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The ISO/IEC 14443-3 Type A protocol is selected. I**SO14443A_ERRORCODE_PARAMETER:** One parameter is incorrect. **CR95HF_ERROR_CODE**: Function failed. The ISO/IEC 14443-3 Type A protocol is not selected.

5.1.7 ISO/IEC 14443-3 Type A command function

These functions send a basic RF command to a contactless tag and check the CR95HF response.

The `ISO14443A_REQA` function sends a REQA command to a contactless tag. *Table 34* describes the `ISO14443A_REQA` function.

Table 34. ISO14443A_REQA function description

Prototype	`ISO14443A_REQA (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The REQA command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_WUPA` function sends a WUPA command to a contactless tag. *Table 35* describes the `ISO14443A_WUPA` function.

Table 35. ISO14443A_WUPA function description

Prototype	`int8_t ISO14443A_WUPA (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The WUPA command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_HLTA` function sends an HLTA command to a contactless tag. *Table 36* describes the `ISO14443A_HLTA` function.

Table 36. ISO14443A_HLTA function description

Prototype	`int8_t ISO14443A_HLTA (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The HLTA command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code

The `ISO14443A_AnticollisionLevel1` function sends an anticollision (level 1) command to a contactless tag.
Table 37 describes the `ISO14443A_AnticollisionLevel1` function.

Table 37. ISO14443A_AnticollisionLevel1 function description

Prototype	`int8_t ISO14443A_AnticollisionLevel1 (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The anticollision (level 1) command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_AnticollisionLevel2` function sends an anticollision (level 2) command to a contactless tag.
Table 38 describes the `ISO14443A_AnticollisionLevel2` function.

Table 38. ISO14443A_AnticollisionLevel2 function description

Prototype	`int8_t ISO14443A_AnticollisionLevel2 (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The anticollision (level 2) command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_AnticollisionLevel3` function sends an anticollision (level 3) command to a contactless tag.
Table 39 describes the `ISO14443A_AnticollisionLevel3` function.

Table 39. ISO14443A_AnticollisionLevel3 function description

Prototype	`int8_t ISO14443A_AnticollisionLevel3 (uint8_t *pResponse);`
Input parameter	None
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The anticollision (level 3) command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_SelectLevel1` function sends a Select (level 1) command to a contactless tag. *Table 40* describes the `ISO14443A_SelectLevel1` function.

Table 40. ISO14443A_SelectLevel1 function description

Prototype	`int8_t ISO14443A_SelectLevel1 (uc8 NbByteUIDin, uc8*UIDin, uint8_t*pResponse);`
Input parameter	**NbByteUIDin**: Number of bytes of UID in parameter **UIDin**: Pointer of contacless tag UID
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The Select (level 1) command is successfully emitted to a contactless tag. **ISO14443A_ERRORCODE_PARAMETER**: `NbByteUIDin` is incorrect. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443A_SelectLevel2` function sends a Select (level 2) command to a contactless tag. *Table 41* describes the `ISO14443A_SelectLevel2` function.

Table 41. ISO14443A_SelectLevel2 functiond description

Prototype	`int8_t ISO14443A_SelectLevel2 (uc8 NbByteUIDin, uc8*UIDin, uint8_t*pResponse);`
Input parameter	**NbByteUIDin**: Number of bytes of UIDin parameter **UIDin**: Pointer of contacless tag UID
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The Select (level 2) command is successfully emitted to a contactless tag. **ISO14443A_ERRORCODE_PARAMETER:** `NbByteUIDin` is incorrect. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an error code.

The `ISO14443A_SelectLevel3` function sends a Select (level 3) command to a contactless tag. *Table 42* describes the `ISO14443A_SelectLevel3` function.

Table 42. ISO14443A_SelectLevel3 function description

Prototype	`int8_t ISO14443A_SelectLevel3 (uc8 NbByteUIDin, uc8*UIDin, uint8_t*pResponse);`
Input parameter	**NbByteUIDin**: Number of bytes of the UIDin parameter **UIDin**: Pointer of a contacless tag UID
Output parameter	**pResponse**: Pointer to the CR95HF response
Return parameter	**RESULTOK**: The Select (level 3)command is successfully emitted to contactless tag. **ISO14443A_ERRORCODE_PARAMETER:** `NbByteUIDin` is incorrect. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an error code.

The `ISO14443A_SelectSequence` function described in *Section 5.1.11: ISO/IEC 14443-3 Type A Advanced functions* carries out a whole sequence using the Select and Anticollision commands.

For more information, refer to *Section 5.1.5: Functions of ISO/IEC 14443_A layer*.

5.1.8 ISO/IEC 14443-3 Type A Split function

The `ISO14443A_SplitATQA` function splits the ATQA contactless tag response. *Table 43* describes the `ISO14443A_SplitATQA` function.

Table 43. ISO14443A_Split ATQA function description

Prototype	`int8_t ISO14443A_SplitATQA (uint8_t *ReaderResponse, uint8_t*ProprietaryCoding, uint8_t*UIDsizeframe, uint8_t*BitFrameAnticol, uint8_t*CR95HFControlByteIndex);`
Input parameter	**ReaderResponse:** Pointer of the CR95HF response
Output parameter	**ProprietaryCoding**: Proprietary coding (4 bits) **UIDsizeframe**: UID size bit frame (2 bits): 0 for single UID size, 1 for double UID size, 2 for triple UID size **BitFrameAnticol**: Bit frame anti-collision (5 bits) **CR95HFControlByteIndex**: Index on control byte of CR95HF response (3 bytes)
Return parameter	**RESULTOK**: Function is successful. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

5.1.9 ISO/IEC 14443-3 Type A Is function

The Is functions return `RESULTOK` if the function is successful, otherwise the Is functions return `ERRORCODE_GENERIC`.

The Is functions return `RESULTOK` if the bit collision is detected. The `ISO14443A_IsCollisionDetected` function returns `RESULTOK` if the bit `CollisionDetected` of the CR95HF control byte is set.
Table 44 describes the `ISO14443A_IsCollisionDetected` function.

Table 44. ISO14443A_IsCollisionDetected function description

Prototype	`int8_t ISO14443A_IsCollisionDetected(uc8*CR95HFControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer to the CR95HF control byte (3 bytes)(1).
Output parameter	None
Return parameter	**RESULTOK**: `CollisionDetected` bit is set. **ERRORCODE_GENERIC**: `CollisionDetected` bit is reset.

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

The `ISO14443A_IsCRCError` function returns `RESULTOK` if the bit `CRC Error` of CR95HF control bytes is set. Otherwise, the `ISO14443A_IsCRCError` function returns `ERRORCODE_GENERIC`. *Table 45* describes the `ISO14443_IsCRCError` function.

Table 45. ISO14443A_IsCRCError function description

Prototype	`int8_t ISO14443A_IsCRCError(uc8*CR95HFControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer of the CR95HF control bytes (3 bytes)(1).
Output parameter	None
Return parameter	**RESULTOK**: `CRC Error` bit is set. **ERRORCODE_GENERIC**: `CRC Error` bit is reset.

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

The `ISO14443A_IsParityError` function returns `RESULTOK` if the bit `Parity Error` of CR95HF control bytes is set. Otherwise, the `ISO14443A_IsParityError` function returns `ERRORCODE_GENERIC`. *Table 46* describes the `ISO14443A_IsParityError` function.

Table 46. ISO14443A_IsParityError function description

Prototype	`int8_t ISO14443A_IsParityError(uc8*CR95HFControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer on CR95HF control bytes (3 bytes)(1).
Output parameter	None
Return parameter	**RESULTOK**: `Parity Error` bit is set. **ERRORCODE_GENERIC**: `Parity Error` bit is reset.

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

The `ISO14443A_Is14443_4Compatible` function returns `RESULTOK` if the bit `Contactless Tag` is compliant with ISO/IEC 14443_4. Otherwise, the `ISO14443A_Is14443_4Compatible` function returns `ERRORCODE_GENERIC`. *Table 47* describes the `ISO14443A_Is14443_4Compatible` function.

Table 47. ISO14443A_Is14443_4Compatible function description

Prototype	`int8_t ISO14443A_Is14443_4Compatible(uc8 SAKbyte);`
Input parameter	**SAKbyte**: Contactless tag response to the Select command
Output parameter	None
Return parameter	**RESULTOK**: The contactless tag is compatible with ISO/IEC 14443_4. **ERRORCODE_GENERIC**: The contactless tag is not compatible with the ISO/IEC 14443_4.

The `ISO14443A_IsUIDComplete` function returns `RESULTOK` if the UID is complete. Otherwise, the `ISO14443A_IsUIDComplete` function returns `ERRORCODE_GENERIC`. *Table 48* describes the `ISO14443A_IsUIDComplete` function.

Table 48. ISO14443A_IsUIDComplete function description

Prototype	`int8_t ISO14443A_IsUIDComplete(uc8 SAKbyte);`
Input parameter	**SAKbyte**: Contactless tag response to select command
Output parameter	None
Return parameter	**RESULTOK**: UID is complete. The anti-collision loop is over. **ERRORCODE_GENERIC**: UID is not complete.

The `ISO14443A_IsPresent` function returns `RESULTOK` if a 14443 Type A contactless tag is present into RF field. Otherwise, the `ISO14443A_IsPresent` function returns `ERRORCODE_GENERIC`. *Table 49* describes the `ISO14443A_IsPresent` function.

Table 49. ISO14443A_IsPresent function description

Prototype	`int8_tISO14443A_IsPresent (void);`
Input parameter	None
Output parameter	None
Return parameter	**RESULTOK**: An ISO/IEC 14443 Type A contactless tag is present. **ERRORCODE_GENERIC**: No ISO/IEC 14443 Type A contactless tag available

The `ISO14443_IsCorrectBCC` function returns `RESULTOK` if the XOR of Data is equal to `BCC`. Otherwise, the `ISO14443_IsCorrectBCC` function returns `ERRORCODE_GENERIC`. *Table 50* describes the `ISO14443A_IsCorrectBCC` function.

Table 50. ISO14443A_IsCorrectBCC function description

Prototype	`int8_t ISO14443A_IsCorrectBCC (uint8_t NbByte, uint8_t*Data, uint8_t BCC);`
Input parameter	**NbByte**: Number of bytes of Data **Data**: Data to check (without BCC byte) **BCC**: BCC byte of contactless tag response
Output parameter	None
Return parameter	**RESULTOK**: BCC byte is equal to the XOR of Data. **ERRORCODE_GENERIC**: BCC byte is different to XOR of Data.

5.1.10 ISO/IEC 14443-3 Type A GET functions

The `ISO14443A_GetSignificantBit` function returns the number of significant bits of the first byte. This information is extracted from the control bytes of the CR95HF response. *Table 51* describes the `ISO14443A_GetSignificantBit` function.

Table 51. ISO14443A_GetSignificantBit function description

Prototype	`int8_t ISO14443A_GetSignificantBit (uc8*CR95HF ControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer of the CR95HF control bytes (3 bytes)(1).
Output parameter	None
Return parameter	Number of significant bit of the first byte

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

The `ISO14443A_GetFirtsCollisionByte` function returns the index of the first byte where the collision is detected. This information is extracted from the control bytes of CR95HF. *Table 52* describes the `ISO14443A_GetFirstCollisionByte` function.

Table 52. ISO14443A_GetFirstCollisionByte function description

Prototype	`int8_t ISO14443A_GetFirstCollisionByte (uc8*CR95HF ControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer of the CR95HF control bytes (3 bytes)(1).
Output parameter	None
Return parameter	Index of the first byte where the collision is detected.

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

The `ISO14443A_GetFirstCollisionBit` function returns the index of the first bit where the collision is detected. This information is extracted from control byte of CR95HF. *Table 53* describes the `ISO14443A_GetFirstCollisionBit` function.

Table 53. ISO14443A_GetFirstCollisionBit function description

Prototype	`int8_t ISO14443A_GetFirstCollisionBit (uc8*CR95HF ControlByteIndex);`
Input parameter	**CR95HFControlByteIndex**: Pointer of the CR95HF control bytes (3 bytes)(1).
Output parameter	None
Return parameter	Index of the first bit where the collision is detected.

1. Three control bytes are appended by CR95HF to contactless tag response. Refer to *Table 12: CR95HF success response value for ISO/IEC 14443-3 Type A*.

5.1.11 ISO/IEC 14443-3 Type A Advanced functions

The `ISO14443A_SelectSequence` function runs a select sequence. For more information, refer to the ISO/IEC 14443-3 documentation. *Table 54* describes the `ISO14443A_SelectSequence` function.

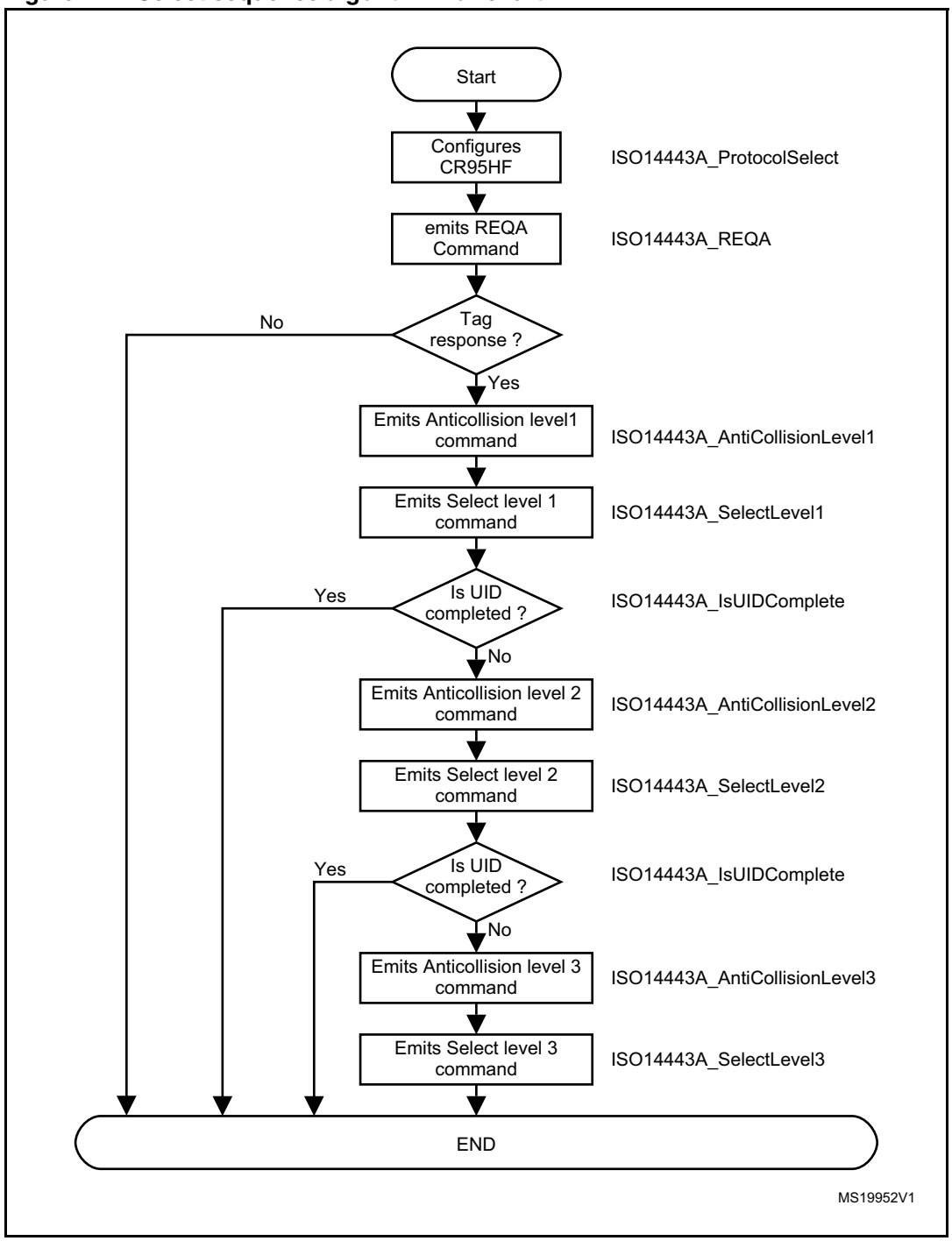
Table 54. ISO14443A_SelectSequence function description

Prototype	`int8_t ISO14443A_SelectSequence(uint_t*SAKByte);`
Input parameter	None
Output parameter	**SAKByte**: The last SAK byte received from a contactless tag during select sequence.
Return parameter	**RESULTOK**: Function is successful. The Select sequence is completed. **ERRORCODE_GENERIC**: Function failed. An error occured during a select sequence.

For more information about the `ISO14443A_SelectSequence` function, refer to *Table 32: ISO/IEC 14443-3 Type A Advanced function description*.

The following flowchart illustrates the select sequence algorithm.

Figure 4. Select sequence algorithm flowchart



The `ISO14443A_GetUIDsize` function returns the UID size of a contactless tag. *Table 55* describes the `ISO14443A_GetUIDsize` function.

Table 55. ISO14443A_GetUIDsize function description

Prototype	`int8_t ISO14443A_GetUIDsize(uint8_t*UIDsize);`
Input parameter	None
Output parameter	**UIDsize**: 2-bit value: 0 for single, 1 for double, 2 for triple and 3 for RFU.
Return parameter	**RESULTOK**: Function is successful. **ERRORCODE_GENERIC**: Function failed.

The `ISO14443A_GetUID` function returns the UID of a contactless tag. *Table 56* describes the `ISO14443A_GetUID` function.

Table 56. ISO14443A_GetUID function description

Prototype	`int8_t ISO14443A_GetUID(uint8_t*SAKByte,uint8_t*NbUIDByte, uint8_t*UIDout);`
Input parameter	None
Output parameter	**SAKByte**: The last SAK byte received from a contactless tag during a `Select` sequence. **NbUIDByte**: Number of bytes of UID **UIDout**: The last SAK byte received from a contactless tag during a select sequence.
Return parameter	**RESULTOK**: Function is successful. The select sequence is completed. **ERRORCODE_GENERIC**: Function failed. An error occured during the select sequence. **ISO14443A_ERRORCODE_BCC**: A BCC byte is incorrect. The received data from a contactless tag is corrupted.

5.2 ISO/IEC 14443-3 Type B layer

The ISO/IEC 14443-3 Type B layer is composed of:

- the `lib_iso14443B.c` source file
- the `lib_iso14443B.h` include file

5.2.1 ISO/IEC 14443-3 Type B command format

The ISO/IEC 14443-3 Type B specification defines the following command frame:

SOF	Data	CRC	EOF

with:

SOF: Start of Frame

Data: 1 byte or more

CRC: 2 bytes

EOF: End of Frame

5.2.2 EOF and SOF

The EOF and SOF are managed by CR95HF.

5.2.3 Data management

The command code and data must be managed by the user application.

5.2.4 ISO/IEC 14443-3 Type B CRC16 management

The ISO/IEC 14443-3 Type B specification defines a two-byte CRC. It is appended to RF commands to check the data transmission between CR95HF and a contactless tag.

The bit number 0 of a parameter byte of ProtocolSelect command controls the CRC. When this bit is set to 1, the CR95HF appends the CRC to all RF commands.

The ProtocolSelect command sets this bit, thus the CR95HF manages the CRC.

5.2.5 Functions of ISO/IEC 14443-3 Type B layer

Table 57 describes the function to configure the CR95HF to communicate with an ISO/IEC 14443 Type B contactless tag.

Table 57. ISO/IEC14443-3 Type B configuration function description

Function	Description
`ISO14443B_ProtocolSelect`	Configures CR95HF to communicate with an ISO/IEC 14443 Type B contactless tag.

Table 58 describes the `ISO14443B_SetParamByte` function.

Table 58. ISO14443B_SetParamByte function description

`ISO14443B_SetParamByte`	Returns the `PARAM` byte.

Table 59 lists the functions of this layer which emit an ISO/IEC 14443-3 Type B command.

Table 59. ISO/IEC 14443-3 Type B command functions description

`ISO14443B_REQB`	Sends an `REQB` command to a contactless tag.
`ISO14443B_WUPB`	Sends a `WUPB` command to a contactless tag.
`ISO14443B_SlotMarker`	Sends a `SlotMarker` command to a contactless tag.
`ISO14443B_Attrib`	Sends an `Attrib` command to a contactless tag.
`ISO14443B_HLTB`	Sends an `HLTB` command to a contactless tag.

Table 60 lists the `Split` functions. A CR95HF response can include different information fields extracted by the following functions.

Table 60. ISO/IEC 14443-3 Type B Split functions description

`ISO14443B_SplitATQB`	Splits an `ATQB` contactless tag response.
`ISO14443B_SplitApplicationDataField`	Splits an `ApplicationData` field.
`ISO14443B_SplitProtocolInfoField`	Splits a `ProtocolInfo` field.

Table 61 lists the `Is` function. This function returns `RESULTOK` if the function is successful. Otherwise, the Is function returns `ERRORCODE_GENERIC`.

Table 61. ISO/IEC 14443-3 Type B Is function description

`ISO14443B_IsPresent`	Checks if a 14443-3 Type B tag is present in the RF field.

5.2.6 CR95HF configuration function

The CR95HF configuration function sends a ProtocolSelect command to the CR95HF device to select the ISO/IEC 14443-3 Type B protocol. The transmission and reception data rates are integrated in the command. This command turns on the RF field.
Table 62 describes the `ISO14443B_ProtocolSelect` function.

Table 62. ISO14443B_ProtocolSelect function description

Prototype	`int8_t ISO14443B_ProtocolSelect (uc8TxDataRate, uc8 RxDataRate, uint8_t*pResponse);`
Input parameter	**TxDataRate**: Transmission data rate (2 bits) **RxDataRate**: Reception data rate (2 bits)
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful.The ISO14443-4 Type B protocol is selected. **ISO14443B_ERRORCODE_PARAMETER**: One parameter is incorrect. **CR95HF_ERROR_CODE**: Function failed. The ISO 14443_4 Type B protocol is not selected.

Note: The Bit 0 (Append CRC) is automatically set inside this function enabling the CR95HF to manage the CRC of the RF command.

For more information, refer to *Section 5.2.4: ISO/IEC 14443-3 Type B CRC16 management*.

5.2.7 ISO/IEC 14443-3 Type B Set function

The `ISO14443B_SetParamByte` function returns `PARAM` byte, which is the result of the concatenation of the input parameters. A `PARAM` byte is an input parameter of REQB and WUPB commands.

Table 63 describes the `ISO14443B_SetParamByte` function.

Table 63. ISO14443B_SetParamByte function description

Prototype	`int8_t ISO14443B_SetParamByte(uc8ExtendedBit, uc8 REQBWUPBbit, uc8NumberOfSlots);`
Input parameter	**ExtendedBit**: Indicates the PCD capability to support an extended `ATQB` response. **REQBWUPBbit**: This bit defines either REQB (0) or WUPB (1). **NumberOfSlots:** Number of slots
Output parameter	None
Return parameter	**PARAM**: PARAM byte

5.2.8 ISO/IEC 14443-3 Type B command function

These functions send an ISO/IEC 14443-3 Type B RF command to a contactless tag and check the CR95HF response.

The `ISO14443B_REQB` function sends an `REQB` command to a contactless tag. *Table 64* describes the `ISO14443B_REQB` function if an `REQB` command is sent.

Table 64. ISO14443B_REQB function description

Prototype	`int8_t ISO14443B_REQB(uc8 AFIin,uc8 Param, uint8_t *pResponse);`
Input parameter	AFIin: An AFI byte (Application Family Identifier) represents the type of targeted application. **Param**: PARAM byte
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The REQB command is successfully transmitted to a contactless tag. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an errror code.

The `ISO14443B_WUPB` function sends a `WUPB` command to a contactless tag. *Table 65* describes the `ISO14443B_WUPB` function if a `WUPB` command is sent.

Table 65. ISO14443B_WUPB function description

Prototype	`int8_t ISO14443B_WUPB(uc8 AFIin,uc8 Param, uint8_t *pResponse);`
Input parameter	**AFIin**: AFIbyte (Application Family identifier) represents the type of targeted application. **Param**: PARAM byte
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The WUPB command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an errror code.

The `ISO14443B_SlotMarker` function sends a SlotMarker command to a contactless tag. *Table 66* describes the `ISO14443B_SlotMarker` function.

Table 66. ISO14443B_SlotMarker function description

Prototype	`int8_t ISO14443B_SlotMarker(uc8 SlotNumber, uint8_t *pResponse);`
Input parameter	**SlotNumber**: Slot number value (4 bits) between 1 and 15
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an errror code.

The ISO14443B_Attrib function sends an Attrib command to a contactless tag. *Table 67* describes the ISO14443B_Attrib function.

Table 67. ISO14443B_Attrib function description

Prototype	int8_t ISO14443B_Attrib(uc8 *Identifier, uc8 *Parameters, uc8 NbByteHigherlayerData, uc8*HigherLayerData, uint8_t *pResponse);
Input parameter	**Identifier**: Pointer to the identifier parameter (4 bytes). It is the value of the PUPI sent by contactless tag in the ATQB. **Parameters** input**:** Pointer on Parameters parameter (4 bytes) **NbByteHigherlayerData:** Number of bytes of HigherLayesDataparameter **HigherLayersData:** Pointer on Parameters parameter (NBByteHigherlayerDatabytes (optional))
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to contactless tag. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an errror code.

The ISO14443B_HLTB function sends an HLTB command to a contactless tag. *Table 68* describes the ISO14443B_HLTB function.

Table 68. ISO14443B_HLTB function description

Prototype	int8_t ISO14443B_HLTB(uc8 *Identifier, uc8 *Identifier, uint8_t *pResponse);
Input parameter	**Identifier**: Pointer on identifier parameter (4 bytes). It is the value of the PUPI sent by a contactless tag in the ATQB contactless tag.
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The HLTB command was successfully transmitted to a contactless tag. **CR95HF_ERROR_CODE:** Function failed. The CR95HF returns an errror code.

5.2.9 ISO/IEC 14443-3 Type B Split function

The `ISO14443B_SplitATQB` function splits the `ATQB` contactless tag response. *Table 69* describes the ISO14443B_SplitATQB contactless response.

Table 69. ISO14443B_SplitATQB function description

Prototype	`int8_t ISO14443B_SplitATQB (uint8_t*ReaderResponse, uint8_t*PUOIindex, uint8_t*ApplicationDataindex, uint8_t*ProtocolInfoindex);`
Input parameter	**ReaderResponse**: Pointer on CR95HF response
Output parameter	**PUPIindex:** Index of PUPI field (4bytes) **ApplicationDataindex**: Index of ApplicatioData field (4bytes) **ProtocolInfoindex:** Index of Protocol field (3 or 4 bytes depending on Extended ATQB supported bit.) Refer to ***Table 64: ISO14443B_REQB function description***.
Return parameter	**RESULTOK:** Function is successful. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443B_SplitApplicationDataField` function splits the ApplicationData field. The application data field is a part of ATQB response. *Table 70* describes the `ISO14443B_SplitApplicationDataField` function.

Table 70. ISO14443B_SplitApplicationDataField function description

Prototype	`int8_t ISO14443B_SplitApplicationDataField( uc8_t*ApplicationDataField, uint8_t*AFlout, uint8_t*CRC_BAIDoutIndex, uint8_t*ProtocolInfoindex);`
Input parameter	**ApplicationDataField**: Application Data field (part of ATQB response, 4 bytes)
Output parameter	**AFIout:** AFI byte **CRC_BAIDoutIndex**: Index of CRC field (2 bytes) **NumberOfApplication:** Number of application byte
Return parameter	**RESULTOK:** Function is successful. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `ISO14443B_SplitProtocolInfoField` function splits the ProtocolInfo field. The ProtocolInfo field is a part of `ATQB` response. *Table 71* describes the `ISO14443B_SplitProtocolInfoField` function.

Table 71. ISO14443B_SplitProtocolInfoField function description

Prototype	`int8_t ISO14443B_SplitProtocolInfoField(uc8*ProtocolInfoField, uint8_t*BitRateCapability,uint8_t*MaxFrameSize,uint8*ProtocolType, uint8_t*FWIbits, uint8_t*ADCbits, uint8_t*FObits, uint8_t*SFGIbits, uint8_t*RFUbits);`
Input parameter	**ProtocolInfoField**: Protocol info field (part of `ATQB` response, 4bytes)
Output parameter	**BitRateCapability:** Bite rate supported by the contactless tag (8 bit) **MaxFrameSize**: Max frame size (4bit) **ProtocolType:** (4 bit) b1=1 compliant with 14443_4 b1=0 not compliant with 14443_4. b2&b3 minimum TR2 coding b4: RFU **FWIbits:** Frame waiting time integer (4 bit) **ADCbits:** Application data coding (2bit). 0b00: application is proprietary 0b01: application is coded (refer to ISO/IEC 14443-3, RFU specification) **FObits:** Frame option (2bit) 0b1X: NAD suported x1b 0bX1: CID supported **SFGIbits:** SFGI bits (4 bit). Optional, depending on Extended ATQB supported bit. Refer to ***Table 64: ISO14443B_REQB function description***. **RFUbits:** RFU bits (4bit). Optional, depending on Extended ATQB supported bit. Refer to ***Table 64: ISO14443B_REQB function description***.
Return parameter	**RESULTOK:** Function is successful. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

5.2.10 ISO/IEC 14443-3 Type B Is function

The `ISO14443B_IsPresent` function returns `RESULTOK` if an ISO/IEC 14443-3 Type B is present into the RF field. Otherwise, it returns `ERRORCODE_GENERIC`. *Table 72* describes the `ISO14443B_IsPresent` function.

Table 72. ISO14443B_IsPresent function description

Prototype	`int8_t ISO14443B_IsPresent (void);`
Input parameter	None
Output parameter	None
Return parameter	**RESULTOK**: A 14443-3 Type B contactless tag is present. **ERRORCODE_GENERIC**: No 14443-3 Type B contactless tag available

For more information about Is function, refer to *Table 61: ISO/IEC 14443-3 Type B Is function description*.

6 SRIX4K library

For a brief description of SRIX4K library, refer to *Section 3.3: SRIX4K product overview*.

The SRIX4K library is composed of:

- the `lib_SRX4K.c` source file
- the `lib_SRX4K.h` include file

This library is valid for all SRIX series.

6.1 SRIX4K command format

The SRIX4K is compliant with the frame format specified in *Section 5.2.1: ISO/IEC 14443-3 Type B command format*.

6.1.1 SRIX4K CR16 management

The ISO/IEC 14443-3 Type B specification defines a two-byte CRC. It is appended to RF commands to check the data transmission between CR95HF and a contactless tag.

The CRC is managed by the first bit of parameter byte of ProtocolSelect (CR95HF layer) command. If it is set, the CR95HF appends the CRC to RF commands.

The `SRIX4K_ProtocolSelect` function sets this function, so the CR95HF manages the CRC.

6.1.2 Functions of SRIX4K layer

The `SRIX4K_ProtocolSelect` function permits to configure the CR95HF to communicate with an SRI contactless tag.

*Table 73*describes the `SRIX4K_ProtocolSelect` function.

Table 73. SRIX4K_ProtocolSelect function description

`SRIX4K_ProtocolSelect`	Configures CR95HF to communicate with an SRI contactless tag.

Table 74 lists the functions of this layer which emit a command described in SRIX4K datasheet.

Table 74. Commands included in the lib_SRIX4k.c file

Function	Description
`SRIX4K_Initiate`	Sends an Initiate command to a contactless tag.
`SRIX4K_Pcall16`	Sends an Pcall16 command to a contactless tag.
`SRIX4K_SlotMarker`	Sends a SlotMarker command to a contactless tag.
`SRIX4K_SelectChipID`	Sends a SelectChipID command to a contactless tag.
`SRIX4K_Completion`	Sends a Completion command to a contactless tag.
`SRIX4K_ResetToInventory`	Sends a ResetToInventory command to a contactless tag.
`SRIX4K_ReadSingleBlock`	Sends a ReadSingleBlock command to a contactless tag.
`SRIX4K_WriteSingleBlock`	Sends a WriteSingleBlock command to a contactless tag.
`SRIX4K_GetUID`	Sends a GetUID command to a contactless tag.

For more information, refer to the SRIX4K datasheet.

A CR95HF response can include different information fields extracted by the `Split` functions. *Table 75* lists the Split functions.

Table 75. SRIX4K Split functions

Function	Description
`SRIX4K_SplitInitiateResponse`	Splits Initiate response of contactless tag.
`SRIX4K_SplitPCall16Response`	Splits PCall16 response of contactless tag.
`SRIX4K_SplitSlotMarkerResponse`	Splits SlotMarker response of contactless tag.
`SRIX4K_SplitSelectChipIDResponse`	Splits SelectChipID response of contactless tag.
`SRIX4K_SplitCompletionResponse`	Splits Completion response of contactless tag.
`SRIX4K_SplitReadBlockResponse`	Splits ReadSingleBlock response of contactless tag.
`SRIX4K_SplitWriteBlockResponse`	Splits WriteSingleBlock response of contactless tag.
`SRIX4K_SplitGetUIDResponse`	Splits GetUID response of contactless tag.

The Is functions return `RESULTOK` if the function is successful, otherwise they return `ERRORCODE_GENERIC`. *Table 76* lists the `SRXI4K_Is` functions.

Table 76. SRIX4K_Is functions

Function	Description
`SRIX4K_IsPresent`	Checks if an SR tag is in the RF field.
`SRIX4K_IsSRIX4KPresent`	Checks if an SRIX4K contactless tag is in the RF field.

The Advanced functions use the previously function to manage a specific sequence. *Table 77* lists the `SRIX4K_Advanced` function.

Table 77. SRIX4K_Advanced functions

Function	Description
`SRIX4K_GetUIDadvanced`	Returns UID field of a contactless tag.
`SRIX4K_GetChipIDAndSelectIt`	Returns a ChipID of a contactless tag and sends a SelectChipID command.
`SRIX4K_Anticollision`	Runs an anti-collision sequence. For more information, refer to the SRIX4K datasheet.

6.1.3 CR95HF configuration function

The `SRIX4K_ProtocolSelect` function sends a ProtocolSelect command to the CR95HF device to select the ISO/IEC 14443-3 Type B protocol. The transmission and reception data rates are integrated in the command. *Table 78* describes the `SRIX4K_ProtocolSelect` function.

Table 78. SRIX4K_ProtocolSelect function description

Prototype	`int8_t SRIX4K_ProtocolSelect(uc8TxDataRate ,uint8_tRxDataRate,uint8_t*pResponse);`
Input parameter	**TxDataRate**: Transmission data rate (2bits) **RxDataRate**: Reception data rate (2bits)
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: The ISO/IEC 14443_4 Type B protocol is selected. **SRIX4K_ERRORCODE_PARAMETER**: One parameter is incorrect. **CR95HF_ERROR_CODE**: Function failed. The ISO/IEC 14443_4 Type B protocol is not selected.

Note: The Bit 0 (Append CRC) is automatically set inside this function enabling the CR95HF to manage the CRC of the RF command.

For more information, refer to *Section 5.2.4: ISO/IEC 14443-3 Type B CRC16 management*.

6.1.4 SRIX4K command function

These functions send a RF command defined in SRIX4 datasheet to a contactless tag and check the CR95HF response.

The `SRIX4K_Initiate` function sends an Initiate command to a contactless tag. *Table 79* describes the `SRIX4K_Initiate` function.

Table 79. SRIX4K_Initiate function description

Prototype	`int8_t SRIX4K_Initiate(uint8_t*pResponse);`
Input parameter	None
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `SRIX4K_Pcall16` function sends a Pcall16 command to a contactless tag. *Table 80* describes the `SRIX4K_Pcall16` function.

Table 80. SRIX4K_Pcall16 function description

Prototype	`int8_t SRIX4K_Pcall16(uint8_t*pResponse);`
Input parameter	None
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `SRIX4K_SlotMarker` function sends a SlotMarker command to a contactless tag. *Table 81* describes the `SRIX4K_SlotMarker` function.

Table 81. SRIX4K_SlotMarker function description

Prototype	`int8_t SRIX4K_SlotMarker(uc8 SlotNumber, uint8_t*pResponse);`
Input parameter	**SlotNumber:** Slot number value (4bits) between 1 and 15
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `SRIX4K_SelectChipID` function sends a SelectChipID command to a contactless tag. *Table 82* describes the `SRIX4K_SelectChipID` function.

Table 82. SRIX4K_SelectChipID command description

Prototype	`int8_t SRIX4K_SelectChipID(uc8 ChipID, uint8_t*pResponse);`
Input parameter	**ChipID:** Chip ID of a contactless tag
Output parameter	**pResponse:** Pointer to CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The SRIX4K_Completion function sends a Completion command to a contactless tag. *Table 83* describes the SRIX4K_Completion function.

Table 83. SRIX4K_Completion function description

Prototype	int8_t SRIX4K_Completion(uint8_t*pResponse);
Input parameter	None
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The SRIX4K_ResetToInventory function sends a ResetToInventory command to a contactless tag. *Table 84* describes the SRIX4K_ResetToInventory function.

Table 84. SRIX4K_ResetToInventory command description

Prototype	int8_t SRIX4K_ResetToInventory(uint8_t*pResponse);
Input parameter	None
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command has been successfully emitted to contactless a tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The SRIX4K_ReadSingleBlock function sends a ReadSingleBlock command to a contactless tag. *Table 85* describes the SRIX4K_ReadSingleBlock function.

Table 85. SRIX4K_ReadSingleBlock function description

Prototype	int8_t SRIX4K_ReadSingleBlock(uc8 BlockAddress, uint8_t*pResponse);
Input parameter	**BlockAddress:** Block address to read into contactless tag memory.
Output parameter	**pResponse:** Pointer to CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `SRIX4K_WriteSingleBlock` function sends a WriteSingleBlock command to a contactless tag. *Table 86* describes the `SRIX4K_WriteSingleBlock` function.

Table 86. SRIX4K_WriteSingleBlock function description

Prototype	`int8_t SRIX4K_WriteSingleBlock(uc8 BlockAddress,uc8*DataToWrite, uint8_t*pResponse);`
Input parameter	**BlockAddress (1 byte):** Block address to read in contactless tag memory. **DataToWrite (4 bytes):** Pointer to Data to write in contactless tag memory.
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful. The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

The `SRIX4K_GetUID` function sends a GetUID command to a contactless tag. *Table 87* describes the `SRIX4K_GetUID` function.

Table 87. SRIX4K_GetUID function description

Prototype	`int8_t SRIX4K_GetUID( uint8_t*pResponse);`
Input parameter	None
Output parameter	**pResponse:** Pointer to the CR95HF response
Return parameter	**RESULTOK**: Function is successful.The command is successfully emitted to a contactless tag. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code.

6.1.5 SRIX4K Split functions

The `SRIX4K_SplitInitiateResponse` function splits the Initiate contactless tag response. *Table 88* describes the `SRIX4K_SplitInitiateResponse` function.

Table 88. SRIX4K_SplitInitiateResponse function description

Prototype	`int8_t SRIX4K_SplitInitiateResponse(uc8 *ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer on CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitPCall16Response` function splits the PCall16 contactless tag response. *Table 89* describes the `SRIX4K_SplitPCall16Response` function.

Table 89. SRIX4K_SplitPCall16Response function description

Prototype	`int8_t SRIX4K_SplitPCall16Response(uc8* ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. **CR95HF_ERROR_CODE**: Function failed.The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitSlotMarkerResponse` function splits the SlotMarker contactless tag response. *Table 90* describes the `SRIX4K_SplitSlotMarkerResponse` function.

Table 90. SRIX4K_SplitSlotMarkerResponse function description

Prototype	`int8_t SRIX4K_SplitSlotMarkerResponse(uc8*ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response validated. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitSelectChipIDResponse` function splits the SelectChipID contactless tag response.
Table 91 describes the `SRIX4K_SplitSelectChipIDResponse` function.

Table 91. SRIX4K_SplitSelectChipIDResponse function description

Prototype	`int8_t SRIX4K_SplitSelectChipIDResponse (uc8*ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response is validated. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitCompletionResponse` function splits the Completion contactless tag response. *Table 92* describes the `SRIX4K_SplitCompletionResponse` function.

Table 92. SRIX4K_SplitCompletionResponse function description

Prototype	`int8_t SRIX4K_SplitCompletionResponse(uc8*ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response is validated. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitReadBockResponse` function splits the ReadSingleBlock contactless tag response. *Table 94* describes the `SRIX4K_SplitReadBockResponse` function.

Table 93. SRIX4K_SplitReadSingleBlockResponse function description

Prototype	`int8_t SRIX4K_SplitReadBlockResponse(uc8*ReaderResponse, uc8 Length, uint8_t*DataReadIndex);`
Input parameter	**ReaderResponse:** Pointer on CR95HF response **Length:** Number of byte of Reader response
Output parameter	**DataReadIndex:** Index on ReaderResponse of data read into a contactless tag.
Return parameter	**RESULTOK**: Function is successful. Contactless tag response is validated. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitWriteBlockResponse` function splits the WriteSingleBlock contactless tag response. *Table 94* describes the `SRIX4K_SplitWriteBlockResponse` function.

Table 94. SRIX4K_SplitWritedBlockResponse function description

Prototype	`int8_t SRIX4K_SplitWriteBlockResponse(uc8*ReaderResponse, uc8 Length, uint8_t*ChipID);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response validated. **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

The `SRIX4K_SplitGetUIDResponse` function splits the GetUID contactless tag response. *Table 95* describes the `SRIX4K_SplitGetUIDResponse` function.

Table 95. SRIX4K_SplitGetUIDResponse function description

Prototype	`int8_t SRIX4K_SplitGetUIDResponse(uc8*ReaderResponse, uc8Length,uint8_t*UIDindex);`
Input parameter	**ReaderResponse:** Pointer to the CR95HF response **Length:** Number of byte of Reader response
Output parameter	**ChipID:** Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response validated **CR95HF_ERROR_CODE**: Function failed. The CR95HF returns an error code. **SRIX4K_ERRORCODE_CRCRESIDUE:** CRC16 residue is incorrect.

6.1.6 SRIX4K Is function

The `SRIX4K_IsPresent` function returns `RESULTOK` if an SRI contactless tag is present into the RF field. Otherwise, the `SRIX4K_IsPresent` function returns `ERRORCODE_GENERIC`. *Table 96* describes the `SRIX4K_IsPresent` function.

Table 96. SRIX4K_IsPresent function description

Prototype	`SRIX4K_IsPresent(void);`
Input parameter	None
Output parameter	None
Return parameter	**RESULTOK**: Function is successful. An SRI contactless tag is present. **ERRORCODE_GENERIC**: Function failed. No SRI contactless tag available.

For more information about `SRIX4K_Is` function, refer to *Table 76: SRIX4K_Is functions*.

The `SRIX4K_IsSRIX4KPresent` function returns `RESULTOK` if an SRIX4K contactless tag is present into RF field. Otherwise, the `SRIX4K_IsSRIX4KPresent` function returns `ERRORCODE_GENERIC`. *Table 97* describes the `SRIX4K_IsSRIX4KPresent` function.

Table 97. SRIX4K_IsSRIX4KPresent function description

Prototype	`int8_t SRIX4K_IsSRIX4KPresent(void);`
Input parameter	None
Output parameter	None
Return parameter	**RESULTOK**: Function is successful. An SRIX4K contactless tag is present. **ERRORCODE_GENERIC**: Function failed. No SRIX4K contactless tag available.

6.1.7 SRIX4K Advanced function

The `SRIX4K_GetUIDadvanced` function returns UID field of a contactless tag. *Table 98* describes the `SRIX4K_GetUIDadvanced` function.

Table 98. SRIX4K_GetUIDadvanced function description

Prototype	`int8_t SRIX4K_GetUIDadvanced(uint8_t*UID field);`
Input parameter	None
Output parameter	**UIDfield**: Pointer to the UID field of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response validated. **ERRORCODE_GENERIC**: Function failed. CR95HF returns an error code.

The `SRIX4k_GetChipIDAndSelectIt` function returns a ChipId of a contactless tag and sends a SelectChipId command.
Table 99 describes the `SRIX4k_GetChipIDAndSelectIt` function.

Table 99. SRIX4K_GetChipIDAndSelectIt function description

Prototype	`int8_t SRIX4K_GetChipIDAndSelectIt(uint8_t*ChipID);`
Input parameter	None
Output parameter	**ChipID**: Chip ID of a contactless tag
Return parameter	**RESULTOK**: Function is successful. Contactless tag response validated. **ERRORCODE_GENERIC**: Function failed. CR95HF returns an error code.

The `SRIX4K_Anticollision` function runs an anti-collision sequence. *Table 100* describes the `SRIX4K_Anticollision` function.

Table 100. SRIX4K_Anticollision function description

Prototype	`int8_t SRIX4K_Anticollision(uint8_t*NbTagInventoried, uint8_t*ChipIdArray);`
Input parameter	None
Output parameter	**NbTagInventoried:** Number of tags inventoried in `RF` field ***ChipIdArray:** Pointer to the ChipId of the tag inventoried (the length of a ChipId is one byte).
Return parameter	**RESULTOK**: Function is successful. **ERRORCODE_GENERIC**: Function failed. A parameter value is incorrect or CR95HF returns an error code.

7 Project example

This section provides an example of project. In this example, the application handles ISO/IEC 14443-3 Type A and Type B contactless tags. The results are displayed on the LCD screen.

7.1 Hardware

The project runs on an STM3210B-EVAL evaluation board.

7.2 Keil µvision®

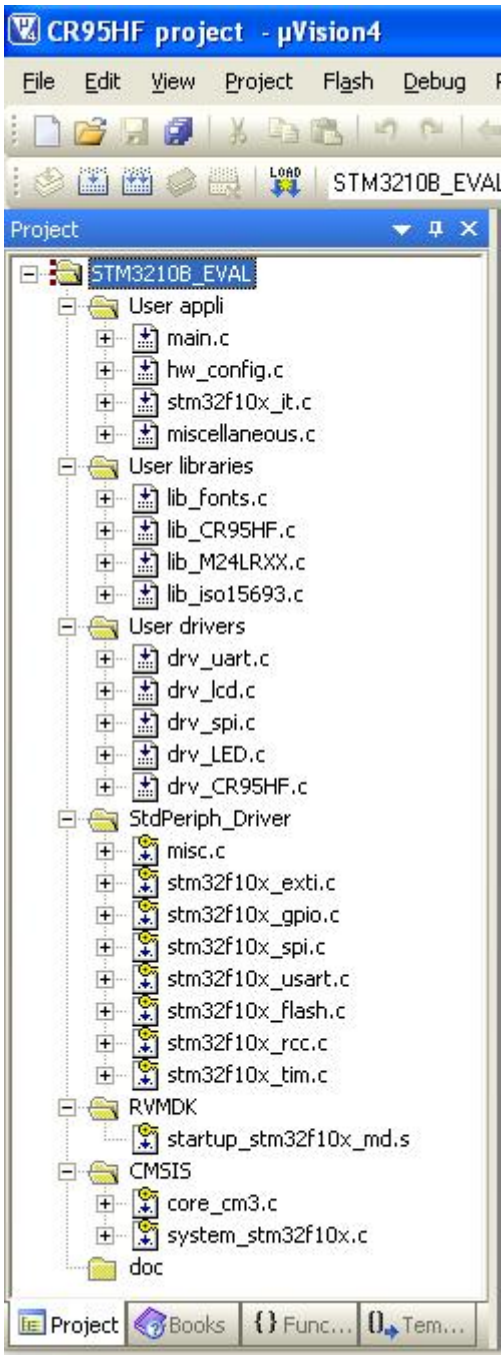
Keil µvision® is a full-featured development environment. It includes a C compiler for STM32 microcontroller family. Keil µvision® software is available from the internet Keil site at http://www.keil.com.

An evaluation version is available free of charge for projects which do not exceed 32 Kbytes.

7.3 Project structure on Keil µvision®

Figure 5 shows an example of project.

Figure 5. Example of project



7.3.1 Standard peripheral drivers

The Keil µvision® project was built from the MCD standard peripheral library (StdPeriph_Driver). It can be downloaded from the STMicroelectronics internet site at http://www.st.com.

It is used to drive all the peripherals used by your application.

7.3.2 Drivers

Some complementary drivers were developped for this application. They are grouped together in the User Drivers directory:

- `drv_UART`: driver on UART serial interface for the communication to CR95HF. The MCU can communicate with CR95HF using either UART or SPI serial interface.
- `drv_SPI`: driver on SPI serial interface for the communication with CR95HF
- `drv_lcd`: driver of LCD of STM3210B-EVAL board
- `drv_LED`: driver of LED of STM3210B-EVAL board
- `drv_CR95HF`: driver of CR95HF. This driver groups functions linked to MCU hardware. (the function to configure the GPIO of the MCU).

7.3.3 Libraries

The libraries used in this project are:

- CR95HF library
- ISO/IEC 14443_A library
- ISO/IEC 14443_B library
- SRIX4K library

The libraries are independent of the hardware and can be reused on a different board.

Furthermore, the `lib-fonts.c` library defines the test fonts for LCD drivers.

7.4 Application functions

The application is intended to handle three kinds of contactless tag:

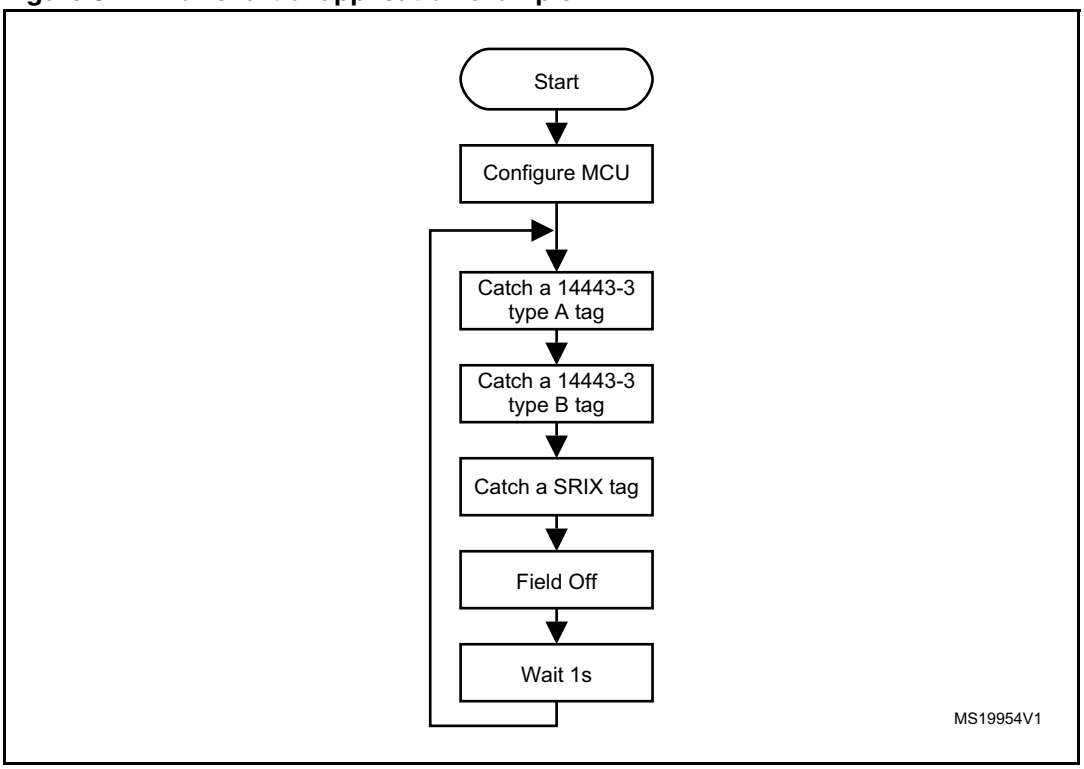
- ISO/IEC 14443-3 Type A contactless tag (function `User_Get14443ATag()`)
- ISO/IEC 14443-3 Type B contactless tag (function `User_Get14443ATag()`)
- SRIx contactless tag (function `USer_Get14443SRTag()`)

The end of the infinite loop turns off the RF field (function `CR95HF_FieldOff()`).

7.4.1 Application example flowchart

Figure 6 shows a flowchart of an application example.

Figure 6. Flowchart of application example



7.4.2 Detect an ISO/IEC 14443-3 Type A tag function

The `ISO14443A_IsPresent` function is used to detect an ISO/IEC 14443-3 Type A contactless tag.

The following table describes how the `ISO14443A_IsPresent` function detects an ISO/IEC 14443-3 Type A contactless tag.

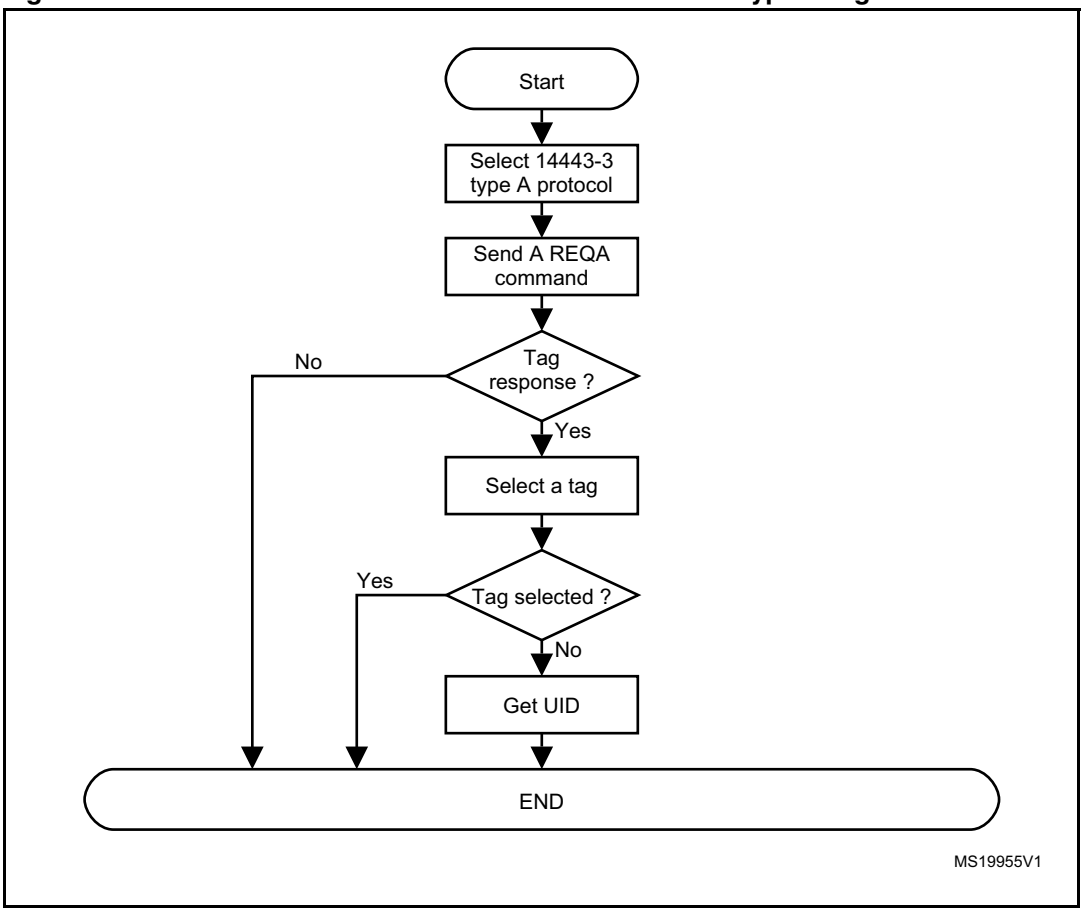
Table 101. Detection of an ISO/IEC 14443-3 Type A contactless tag

Step	Action
1	The `ISO14443A_IsPresent` function configures the CR95HF to communicate with an ISO/IEC 14443-3 Type A contactless tag.
2	If a tag has been detected, it selects the tag by using the Advanced function `ISO14443A_SelectSequence`.
3	It gets the UID. The user function is an Advanced function `ISO14443A_GetUID`.

Figure 7 described how to detect an ISO/IEC 14443-3 Type A tag.

The name of the used function is `User_Get14443Atag (main.c)`.

Figure 7. Flowchart of User detects an ISO/IEC 14443-3 Type A tag



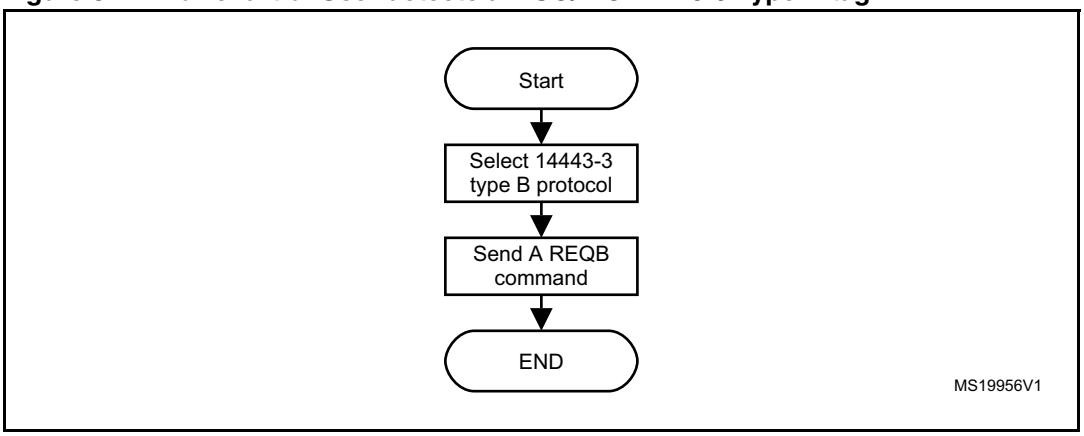
7.4.3 Detect an ISO/IEC 14443-3 Type B tag function

The `ISO14443B_IsPresent` function is used in order to detect an ISO/IEC 14443-3 Type B contactless tag. This function configures CR95HF to communicate with an ISO/IEC 14443-3 Type B contactless tag and sends an REQB command.

If a tag response is detected, the `ISO14443B_IsPresent` function returns `RESULTOK`.

Figure 8 describes how to detect an ISO/IEC 14443-3 Type B tag.

Figure 8. Flowchart of User detects an ISO/IEC 14443-3 Type B tag



7.4.4 Detect an SRIX tag function

The `SRIX4K_IsPresent` function is used in order to detect an SRIX contactless tag. This function configures CR95HF in order to communicate with an ISO/IEC 14443-3 Type B contactless tag and sends an Initiate command.

The following table describes how the `SRIX4K_IsPresent` function detects an SRIX tag.

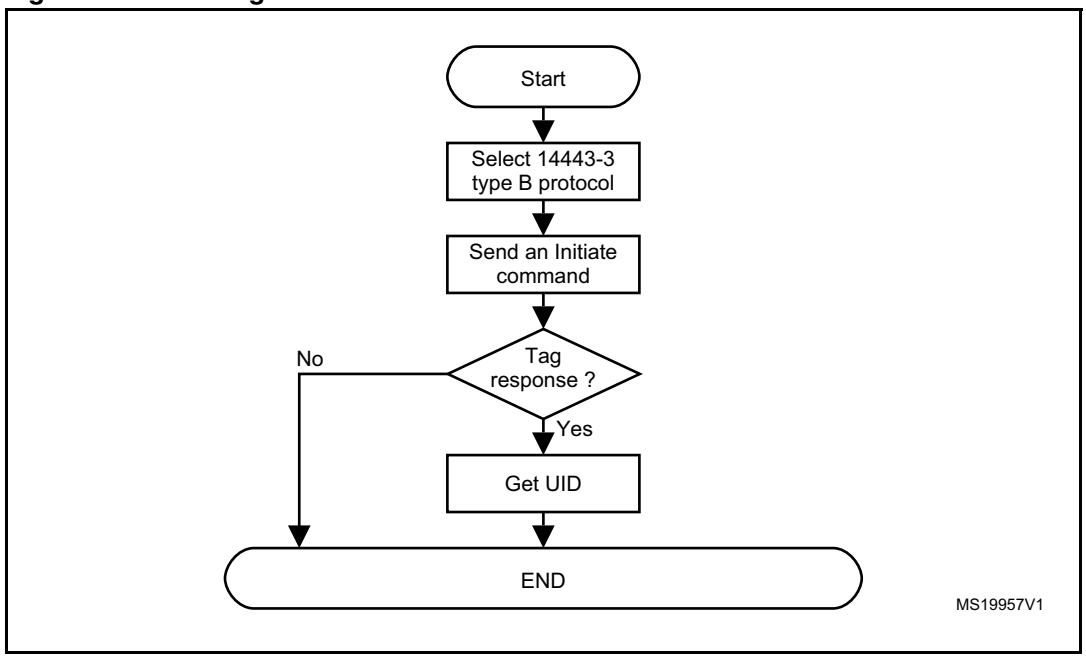
Table 102. Detection of an SRIX tag function

Step	Action
1	The `SRIX4K_IsPresent` function configures the CR95HF to communicate with an ISO/IEC 14443-3 Type B contactless tag and sends an Initiate command.
2	If a tag has been detected, it returns `RESULTOK`.
3	It gets the UID by using the Advanced function `SRIX4K_GetUIDadvanced`.

Figure 9 describes how to detect an ISO/IEC 14443-3 Type B tag.

The name of the used function is `User_Get14443SRTag (main.c)`.

Figure 9. SRIX tag detection flowchart



8 Revision history

Table 103. Document revision history

Date	Revision	Changes
20-Sep-2011	1	Initial release.