AN3375
Application note

CR95HF library for ISO15693 and Dual Interface EEPROM products

Introduction

This document describes the CR95HF library which allows a microcontroller to drive the CR95HF 13.56 MHz multiprotocol contactless transceiver using an SPI or UART interface in order to perform ISO15693 wireless communications with Dual Interface EEPROM devices.

The library was developed to speed up the development of applications using the CR95HF in conjunction with a contactless tag based on an M24LRXX dual interface EEPROM.

The CR95HF library is split into three layers:

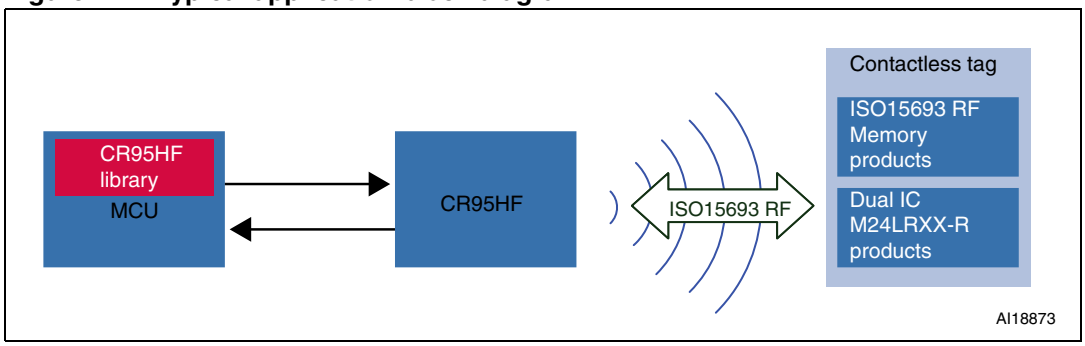
- Low level CR95HF layer
- Standard ISO15693 protocol layer
- Dual Interface EEPROM specific layer

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

Reference documents

- CR95HF datasheet
- ISO/IEC FCD 15693-3
- M24LRXX datasheets

Figure 1. Typical application block diagram



Contents

List of tables

List of figures

1 CR95HF description

1.1 CR95HF overview

The CR95HF is an RF transceiver IC for contactless application. It supports ISO15693, 14443A,14443B and ISO/IEC 18092 protocols and manages frame coding, RF modulation and contactless tag response decoding both from RFID or NFC tags.

The CR95HF operates as a slave device, controlled by a host MCU (see *Figure 1*). The transceiver is delivered with a library which allows to interface the user application with the SPI and UART hardware drivers.

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

1.2 Library overview

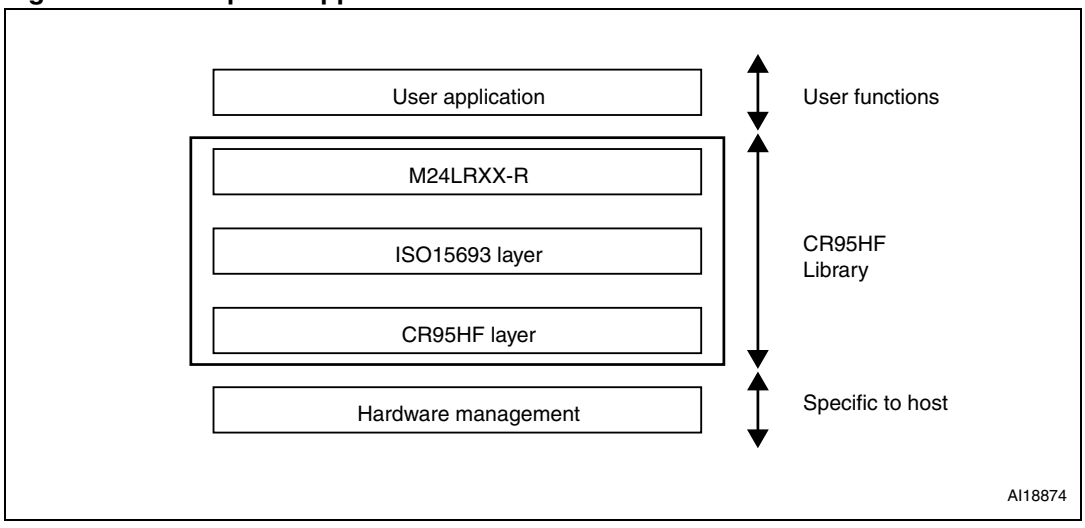
The library described in this application note is composed of three layers;

- Low level layer supporting the commands described in the CR95HF datasheet. This level is fully supported by the CR95HF library
- Intermediate layer based on ISO/IEC FCD 15693-3 protocol
- Upper layer supporting all the commands described in the M24LRXX-R datasheet.

The library can be downloaded from http://www.st.com.

Figure 2 shows the interaction between a typical user application and the CR95HF library layers.

Figure 2. Example of application architecture



2 CR95HF low level layer

This library is composed of:

- The lib_CR95HF.c source file
- The lib_CR95HF.h include file

2.1 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;
```

2.2 Low layer overview

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 2: CR95HF library additional function*.

Application developers can use the functions described in *Table 1: CR95HF library functions based on CR95HF commands* to create their own higher level functions.

Refer to the CR95HF datasheet for more details on the commands.

2.3 CR95HF layer functions

Table 1. CR95HF library functions based on CR95HF commands

Function name	Brief description
CR95HF_IDN	Send Idn command
CR95HF_Echo	Send EchoCode command
CR95HF_ProtocolSelect	Send Protocol Select command
CR95HF_SendRecv	Send SendRecv command
CR95HF_Idle	Send Idle command
CR95HF_RdReg	Send RdReg command
CR95HF_BaudRate	Send BaudRate command

Table 2. CR95HF library additional function

Function name	Brief description
CR95HF_SendEOF	Send EOF pulse
CR95HF_FieldOff	Switch off RF field
CR95HF_HexCommandToStringCommand	Translate hexadecimal command code to command name
CR95HF_GetReaderErrorCode	Translate hexadecimal error code to error code description
CR95HF_IsReaderResultCodeOk	Checks if returned code is succeeded code
CR95HF_IsReaderErrorCode	Checks if returned code is error code
CR95HF_IsCommandExists	Checks if command is available

2.3.1 IDN function

This function sends an IDN command. The CR95HF returns its version number.

Table 3. CR95HF_IDN function description

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

2.3.2 Echo function

This function sends an EchoCode command to the CR95HF which returns an Echo code response (0x55). The Echo function checks that communications can be started between the MCU and the CR95HF.

Table 4. 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

2.3.3 ProtocolSelect function

This function sends a Protocol Select command to the CR95HF. It selects the RF communication protocol, configures RF parameters, and switches the RF field on.

To set up communications with a contactless tag, the Protocol Select command shall be sent to the CR95HF before the SendRecv command.

Table 5. CR95HF_ProtocolSelect function description

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 **Data**: pointer to data
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	CR95HF_ERRORCODE_PARAMETER: function failed CR95HF_SUCCESS_CODE: function successful CR95HF_ERRORCODE_PARAMETERLENGTH: parameter length is erroneous

The Data parameter signification depends on the selected protocol (Protocol parameter). *Table 6* shows the parameter values for the ISO 15693 protocol.

Table 6. CR95HF_ProtocolSelect ISO15693 protocol parameter

Parameter name	Byte	Bit	Value	Example
Command Code	0	7:0	0x02: Select command code	0x02020126
Length	1	7:0	0x02: number of bytes	
Protocol	2	7:0	0x01: ISO15693 protocol	
Parameter	3	7:6	RFU	
		5:4	00b: 26 kbps	
			01b: 52 kbps	
			10b: 6 kbps	
			11b: RFU	
		3	0b: respect 312µs delay	
			1b: wait for SOF	
		2	0b: 100 % modulation	
			1b: 10 % modulation	
		1	0b: single subcarrier	
			1b: dual subcarrier	
		0	0b: Do not append CRC	
			1b: append CRC	

For more details concerning this command, refer to ISO15693 specifications.

2.3.4 SendRecv function

This function sends a SendRecv command. The parameter passed to the command is the frame which is coded according to the protocol previously selected by issuing a Protocol Select 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 7. CR95HF_SendRecv function description

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

The input parameter depends on the selected protocol. *Table 8* gives ISO15693 protocol parameters.

Table 8. CR95HF_SendRecv ISO15693 protocol parameters

Parameter name	Byte	Bit	Value	Example
Command Code	0	7:0	0x04	0x0403260100
Length	1	7:0	0xXX	
Data	2:X	7:0	0xXX	

If the CR95HF_SendRecv function is successful, it returns the CR95HF_SUCCESS_CODE code (see *Table 9*)

Table 9. CR95HF_SUCCESS_CODE values

Parameter name	Byte	Bit	Value	Example
Result Code	0	7:0	0x80	80 0D 0000D625563C172 202E0 4515 00
Length	1	7:0	0xXX	
Data	2:X	7:0	0xXX	
CRC	X + 2: X + 3	7:0	0xXX	
Control byte	X + 4	7:2	RFU	
		1	CRC error if set	
		0	Collision detected if set	

If CR95HF_SendRecv function fails, the CR95HF returns an error code in the pResponse parameter (see *Table 10*). As an example, if no contactless tag is present in the RF field, the CR95HF response is 0x8700.

For more detail concerning the CR95HF error code, refer to CR95HF datasheet.

Table 10. CR95HF_ERROR_CODE_PARAMETERLENGTH values

Parameter name	Byte	Bit	Value	Example
Result code	0	7:0	0x8X	0x8700: no contactless tag in the field
Length	1	7:0	0x00	

2.3.5 Idle function

This function sends an Idle command to the CR95HF to switch the CR95HF device into low consumption mode.

Table 11. CR95HF_Idle function description

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 successful CR95HF_ERRORCODE_PARAMETERLENGTH: parameter length is erroneous

2.3.6 RdReg function

This function sends a RdReg command to the CR95HF to read the CR95HF internal register.

Table 12. CR95HF_RdReg function description

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:** nb of byte to read **Flags:** ST reserved (must be 0x00)
Output parameter	**pResponse** : pointer to CR95HF response
Return parameter	CR95HF_ERRORCODE_PARAMETER: function failed CR95HF_SUCCESS_CODE: function successful CR95HF_ERRORCODE_PARAMETERLENGTH: parameter length is erroneous

2.3.7 BaudRate function

This function sends a BaudRate command to the CR95HF. It allows to configure the UART baudrate.

Table 13. CR95HF_BaudRate function description

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

For more details on this command, please refer to CR95HF datasheet.

2.3.8 SendEOF function

This function emits an RF pulse (EOF) required to perform an ISO15693 inventory or a write command.

Table 14. CR95HF_SendEOF function description

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

2.3.9 FieldOff function

This function switches the RF field off.

Table 15. CR95HF_FieldOff function description

Function description	
Prototype	`result = int8_t CR95HF_FieldOff(void);`
Input parameter	none
Output parameter	none
Return parameter	CR95HF_SUCCESS_CODE

2.3.10 HexCommandToStringCommand function

This function returns the ASCII code of the command.

Table 16. CR95HF_HexCommandToStringCommand function description

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

2.3.11 IsReaderResultCodeOk function

This function returns CR95HF_SUCCESS_CODE if the CR95HF command has succeeded.

Table 17. CR95HF_IsReaderResultCodeOk function description

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: CR95HF returns a succeeded code CR95HF_ERROR_CODE: CR95HF did not return a succeeded code

2.3.12 IsReaderErrorCode function

This function returns CR95HF_SUCCESS_CODE value if the CR95HF command has returned as error code.

Table 18. CR95HF_IsReaderErrorCode function description

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: CR95HF returns an error code CR95HF_ERROR_CODE: CR95HF did not return an error code

2.3.13 IsCommandExists function

This function returns CR95HF_SUCCESS_CODE if the CmdCode value exists.

Table 19. CR95HF_IsCommandExists function description

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: the command code exists CR95HF_ERRORCODE_COMMANDUNKNOWN: the command code is unknown

2.3.14 GetReaderErrorCode function

This function returns a description of the error contained in ReaderReply parameter.

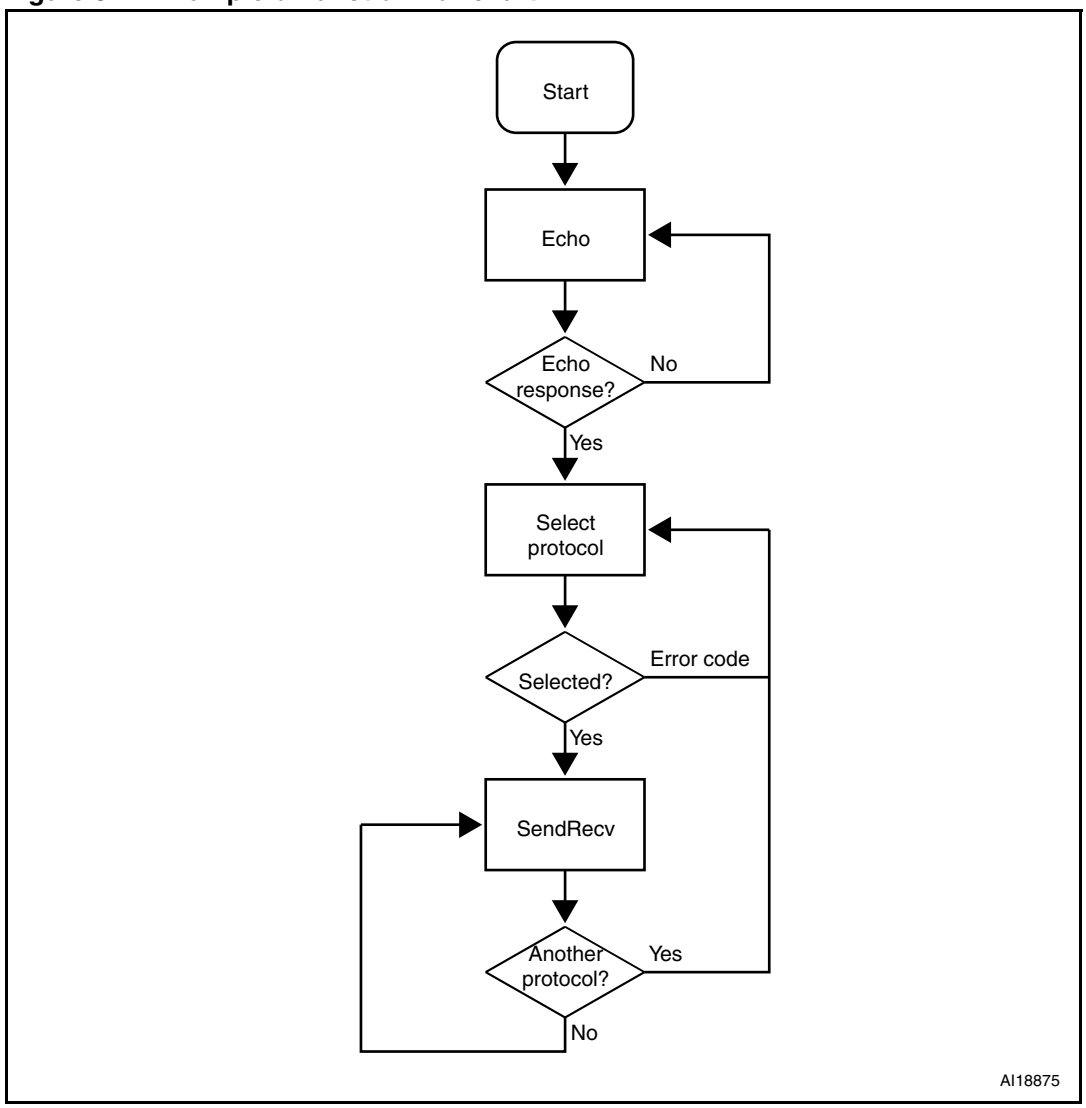
Table 20. CR95HF_GetReaderErrorCode function description

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	ERRORCODE_GENERIC: CR95HF did not return an error code CR95HF_SUCCESS_CODE: function successful executed

2.3.15 Application example: protocol selection and communication

To communicate with a contactless tag, the application must first 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*).

Figure 3. Example of function flowchart



3 ISO15693 library (intermediate layer)

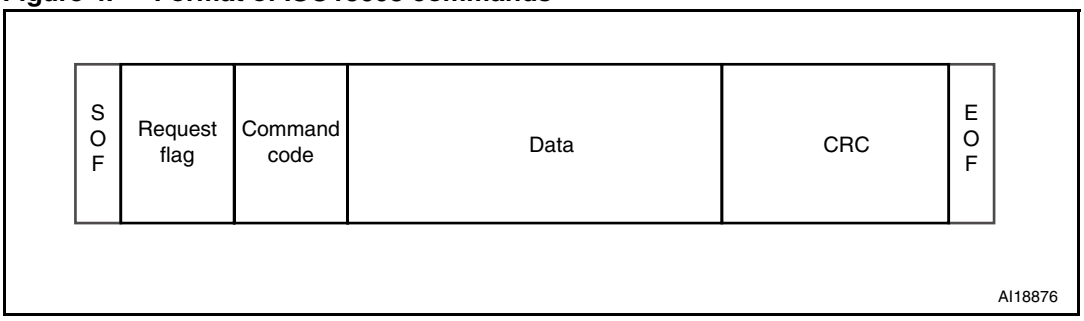
The library is composed of:

- The ISO15693.c source file
- The ISO15693.h include file

3.1 ISO15693 command format

Figure 4 shows ISO15693 command format.

Figure 4. Format of ISO15693 commands



Where

- SOF: start of frame
- Request flag: 1 byte
- Command code: 1 byte
- Data: 1 or more bytes
- CRC: 2 bytes
- EOF: end of frame

3.1.1 EOF and SOF

The EOF and SOF are managed by the CR95HF.

3.1.2 CRC management

The ISO15693 standard specifies a two-byte CRC which it appended to the ISO15693 command and allows to check the integrity of data transmission between CR95HF and contactless tag.

The two-byte CRC can be either computed by the user or by the CR95HF device, depending on the value of the AppendCRC input parameter:

- If AppendCRC is set to ‘0’, the CRC is calculated by the user application.
- If AppendCRC is set to ‘1’, the CRC is calculated by the CR95HF. This configuration is recommended to avoid calculation errors.

The CRC16 input parameter is appended to ISO15693 commands if the user has chosen to manage the CRC.

3.1.3 Request flag management

The request flags byte is managed by the user application. Each bit or flag specifies the actions to be performed by the contactless tag and whether the corresponding fields are present or not.

Bit 3 (Inventory_flag) of the request flag defines the contents of the 4 MSBs (bits 5 to 8). When bit 3 is reset (0), bits 5 to 8 contain the contactless tag selection criteria. When bit 3 is set (1), bits 5 to 8 define the contactless tag Inventory parameters.

For more information, please refer to ISO15693 specification or to the M24LRXX-R datasheets.

Request flags and CR95HF_ProtocolSelect functions

The CR95HF_ProtocolSelect function (see *Section 2.3.3*) selects the RF protocol and defines the CR95HF datarate and contactless tag response format (single or double subcarrier). Once the protocol and the RF parameters are configured, the CR95HF can decode only contactless tag responses of same format and datarate.

The datarate and contactless tag response format are also defined in the request flag byte of each ISO15693 command. The user application must ensure that the datarate and subcarrier flags match the CR95HF_ProtocolSelect function parameters. This check is performed by the functions of the ISO15693 layer. If the parameters do not match, the functions are not executed and the corresponding commands are not sent to the CR95HF.

3.1.4 Command code and data management

The command code is given by the function while data are managed by the user application.

3.2 ISO15693 functions

The CR95HF library includes all the commands defined in ISO/IEC FCD 15693-3 specifications. For more information, refer to the ISO15693 specifications.

The table below lists all the functions of the ISO15693 layer.

Table 21. ISO15693 functions

Function name	Brief description
CRC management functions	
ISO15693_CRC16	Compute CRC according to ISO15693 specification
ISO15693_IsCorrectCRC16Residue	Checks CRC residue according to ISO15693 specification
ISO15693 command functions	
ISO15693_CreateRequestFlag	Create a request flags
ISO15693_Inventory	Sends an inventory command
ISO15693_InventoryOneSlot	Sends an inventory command 1 slot command
ISO15693_Inventory16Slots	Send an inventory command 16 slots command

Table 21. ISO15693 functions (continued)

Function name	Brief description
ISO15693_ReadSingleBlock	Send a read single block command
ISO15693_WriteSingleBlock	Send a write single block command
ISO15693_LockSingleBlock	Send a lock single block command
ISO15693_ReadMultipleBlocks	Send a read multiple block command
ISO15693_Select	Send a select command
ISO15693_ResetToReady	Send a reset to ready command
ISO15693_WriteAFI	Send a write AFI command
ISO15693_LockAFI	Send a lock AFI command
ISO15693_WriteDSFID	Send a write DSFID command
ISO15693_LockDSFID	Send a lock DSFID command
ISO15693_GetMultipleBlockSecutityStatus	Send a Get multiple block security status command
ISO15693_StayQuiet	Send a stay quiet command
ISO15693_GetSystemInfo	Send a get system info command
ISO15693_SendEOF	Send an EOF pulse
Advanced functions	
ISO15693_SelectProtocol	Sends a select protocol
ISO15693_GetUID	Returns UID of a tag
ISO15693_GetDSFID	Returns DSFID field of a tag
ISO15693_GetAFI	Returns AFI field of a tag
ISO15693_GetMemSizeInfo	Returns memory size information of a tag
ISO15693_GetIcRef	Returns IC reference of a tag
ISO15693_IsPresent	Checks is a ISO15693 contactless tag is in RF field
ISO15693_IsCollisionDetected	Returns RESULTOK is collision was detected
ISO15693_IsATagInTheField	Returns RESULTOK is a contactless tag is in RF field
ISO15693_IsTagErrorCode	Returns RESULTOK if the error flag is set to ‘1’
Get flag functions from request flag byte	
ISO15693_GetSubCarrierFlag	Returns subcarrier flag from request flags
ISO15693_GetDataRateFlag	Returns data rate flag from request flags
ISO15693_GetInventoryFlag	Returns inventory flag from request flags
ISO15693_IsInventoryFlag	Returns RESULTOK if Inventory flag is set
ISO15693_GetProtocolExtensionFlag	Returns Protocol extension flag from request flags
ISO15693_GetSelectOrAFIFlag	Returns Select or AFI flag from request flags

Table 21. ISO15693 functions (continued)

Function name	Brief description
ISO15693_GetAddressOrNbSlotsFlag	Returns address or Number of slot flag from request flags
ISO15693_IsAddressOrNbSlotsFlag	Returns RESULTOK if AddressOrNbSlots flag is set
ISO15693_GetOptionFlag	Returns option flag from request flags
ISO15693_GetRFUFlag	Returns RFU flag from request flags
Get flag functions from response flag byte	
ISO15693_GetErrorFlag	Returns error flag from response flag
Get flag functions from information flag byte	
ISO15693_GetDSFIDFlag	Returns DSFID flag from information flag
ISO15693_GetAFIFlag	Returns AFI flag from information flag
ISO15693_GetMemorySizeFlag	Returns memory size flag from information flag
ISO15693_GetICReferenceFlag	Returns IC reference flag from information flag
Split CR95HF response functions	
ISO15693_SplitInventoryResponse	Splits inventory response of contactless tag
ISO15693_SplitGetSystemInfoResponse	Splits get system info response of contactless tag
ISO15693_SplitMemorySizeInfo	Splits memory size field from system info response
ISO15693_SplitReadSingleBlockResponse	Splits read single block response of contactless tag
ISO15693_SplitWriteSingleBlockResponse	Splits write single block response of contactless tag
ISO15693_SplitWriteMultipleBlockResponse	Splits write multiple blocks response of contactless tag
ISO15693_SplitWriteAFIResponse	Splits write AFI response of contactless tag
ISO15693_SplitWriteDSFIDResponse	Splits write DSFID response of contactless tag
ISO15693_SplitLockBlockResponse	Splits lock block response of contactless tag
ISO15693_SplitLockAFIResponse	Splits lock AFI response of contactless tag
ISO15693_SplitLockDSFIDResponse	Splits lock DSFID response of contactless tag
ISO15693_SplitReadMultipleBlockResponse	Splits read multiple blocks response of contactless tag
ISO15693_SplitSelectResponse	Splits select response of contactless tag
ISO15693_SplitResetToReadyResponse	Splits reset to ready response of contactless tag
ISO15693_SplitGetMultipleBlockSecutityRespon se	Splits get multiple block security status response of contactless tag

3.3 CRC management functions

These functions compute either CRC or CRC residue according to the ISO15693 specifications.

3.3.1 ISO15693_CRC16 function

This function returns a 2-byte CRC value as defined in ISO15693 specifications.

Table 22. ISO15693_CRC16 function description

Function description	
Prototype	`result = int16_t ISO15693_CRC16 (uc8 *DataIn, uc8 Length);`
Input parameter	**DataIn**: pointer on input data **Length**: Number of byte of data
Output parameter	none
Return parameter	computed 2 byte CRC

3.3.2 ISO15693_IsCorrectCRC16Residue function

This function returns RESULTOK if the 2-byte CRC is correct.

The CR95HF library provides all the commands defined in the ISO/IEC FCD 15693-3 standard. Only the main command is described in the present section.

Table 23. ISO15693_IsCorrectCRC16Residue function description

Function description	
Prototype	`result = int8_t ISO15693_IsCorrectCRC16Residue (uc8 *DataIn,uc8 Length);`
Input parameter	**DataIn**: input data **Length**: Number of byte of data
Output parameter	none
Return parameter	RESULTOK: CRC Residue is conform to ISO15693 ERRORCODE_GENERIC: CRC Residue is not conform to ISO15693

3.4 ISO15693 command functions

This section provides a detailed description of the library functions which send ISO15693 commands. These functions use the CR95HF_SendRecv function of the CR95HF layer (low level layer) to send commands to a contactless tag and return the tag response.

The output parameter pResponse of the ISO15693 command functions is a pointer to the CR95HF response (see *Section 2.3.4: SendRecv function* for a detailed description of the response format).

3.4.1 ISO15693_CreateRequestFlag function

This command creates the request flags.

Table 24. ISO15693_CreateRequestFlag function description

Function description	
Prototype	`result = int8_t ISO15693_CreateRequestFlag (uc8 SubCarrierFlag,uc8 DataRateFlag,uc8 InventoryFlag,uc8 ProtExtFlag,uc8 SelectOrAFIFlag,uc8 AddrOrNbSlotFlag,uc8 OptionFlag,uc8 RFUFlag);`
Input parameter(1)	**SubCarrierFlag**: single or double subcarrier 0: single subcarrier 1: double subcarrier **DataRateFlag**: contactless tag reply data rate 0: 6 Kbps 1: 26 Kbps **InventoryFlag**: 1: for inventory command 0: otherwise **ProtExtFlag**: protocol extension flag **SelectOrAFIFlag**: Select or AFI flag (depending of inventory flag) **AddrOrNbSlotFlag**: Add or number of slot flag (depending of inventory flag) **OptionFlag**: Option flag **RFUFlag**: RFU flag
Output parameter	None
Return parameter	Request flags

1. The input parameters are described in the ISO15693 specification and are defined in ISO15693.h.

3.4.2 ISO15693_Inventory function

This command sends an inventory command to the contactless tag.

Table 25. ISO15693_Inventory function description

Function description	
Prototype	`result = int8_t ISO15693_Inventory( uc8 Flags , uc8 AFI, uc8 MaskLength, uc8 *MaskValue, uc8 AppendCRC, uc8 *CRC16, uint8_t *pResponse );`
Input parameter	**Flags**: request flags. The inventory flag of request flags must be set to '1'. **AFI**: AFI byte **MaskLength**: number of bits in the mask **MaskValue**: mask **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code ISO15693_ERRORCODE_PARAMETERLENGTH: MaskLength value is erroneous

3.4.3 ISO15693_InventoryOneSlot function

This command issues an inventory command to the CR95HF with the number of slot flag in the request flags set to '1'.

Table 26. ISO15693_InventoryOneSlot function description

Function description	
Prototype	`result = int8_t ISO15693_InventoryOneSlot (uc8 Flags, uc8 AFI, uc8 MaskLength, uc8 *MaskValue, uc8 AppendCRC, uc8 *CRC16, uint8_t *pResponse );`
Input parameter	**Flags**: Request flags with inventory flag set, and number of slot flag set. **AFI**: AFI byte **MaskLength**: number of bits in the mask **MaskValue**: mask **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code ISO15693_ERRORCODE_PARAMETERLENGTH: MaskLength value is erroneous

3.4.4 ISO15693_Inventory16Slots function

This command sends an inventory command to the contactless tag with the number of slot flag in the request flags reset (0).

Table 27. ISO15693_Inventory16Slots function description

Function description	
Prototype	`result = int16_t ISO15693_Inventory16Slots( uc8 Flags , uc8 AFI, uc8 MaskLength, uc8 *MaskValue, uc8 AppendCRC, uc8 *CRC16, uint8_t *NbTag, uint8_t *pResponse );`
Input parameter	**Flags**: request flags with inventory flag set, and number of slot flag reset. **AFI**: AFI byte **MaskLength**: number of bits in the mask **MaskValue**: mask **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response **NbTag**: number of contactless tags identified by the inventory function
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code ISO15693_ERRORCODE_PARAMETERLENGTH: MaskLength value is erroneous

3.4.5 ISO15693_ReadSingleBlock function

This function sends a ReadSingleBlock command to the contactless tag.

Table 28. ISO15693_ReadSingleBlock function description

Function description	
Prototype	`result = int8_t` *`ISO15693_ReadSingleBlock`* `(uc8 Flags, uc8 *UID, uc8 BlockNumber,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UID**: pointer to contactless tag UID **BlockNumber**: block index **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.6 ISO15693_WriteSingleBlock function

This function sends a WriteSingleBlock command to the contactless tag.

Table 29. ISO15693_WriteSingleBlock function description

Function description	
Prototype	`result = int8_t ISO15693_WriteSingleBlock(uc8 Flags, uc8 *UIDin, uc8 BlockNumber,uc8 BlockLength,uc8 *DataToWrite,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **BlockLength**: number of bytes in a block **DataToWrite**: pointer to data to be written into contactless tag memory **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.7 ISO15693_LockSingleBlock function

This function sends a LockSingleBlock command to the contactless tag.

Table 30. ISO15693_LockSingleBlock function description

Function description	
Prototype	`result = int8_t ISO15693_LockSingleBlock (uc8 Flags, uc8 *UIDin, uc8 BlockNumber,uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.8 ISO15693_ReadMultipleBlocks function

This function sends a ReadMultipleBlocks command to the contactless tag.

Table 31. *ISO15693*_ReadMultipleBlocks function description

Function description	
Prototype	`result = int8_t ISO15693_ReadMultipleBlocks (uc8 Flags, uc8 *UIDin, uc8 BlockNumber, uc8 NbBlocks,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **NbBlocks**: number of blocks to read **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.9 ISO15693_Select function

This function sends a Select command to the contactless tag.

Table 32. *ISO15693*_Select function description

Function description	
Prototype	`result = int8_t ISO15693_Select (uc8 Flags, uc8 *UIDin, uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.10 ISO15693_ResetToReady function

This function sends a ResetToReady command to the contactless tag.

Table 33. *ISO15693*_ResetToReady function description

Function description	
Prototype	`result = int8_t ISO15693_ResetToReady (uc8 Flags, uc8 *UIDin, uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.11 ISO15693_WriteAFI function

This function sends a WriteAFI command to the contactless tag.

Table 34. *ISO15693*_WriteAFI function description

Function description	
Prototype	`result = int8_t ISO15693_WriteAFI(uc8 Flags, uc8 *UIDin,uc8 AFIToWrite,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AFIToWrite**: AFI byte to be written to the contactless tag memory **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.12 ISO15693_LockAFI function

This function sends a LockAFI command to the contactless tag.

Table 35. *ISO15693*_LockAFI function description

Function description	
Prototype	`result = int8_t ISO15693_LockAFI (uc8 Flags, uc8 *UIDin, uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.13 ISO15693_WriteDSFID function

This function sends a WriteDSFID command to the contactless tag.

Table 36. *ISO15693*_WriteDSFID function description

Function description	
Prototype	`result = int8_t ISO15693_WriteDSFID (uc8 Flags, uc8 *UIDin,uc8 DSFIDToWrite,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **DSFIDToWrite**: DSFID to write into contactless tag memory **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.14 ISO15693_LockDSFID function

This function sends a LockDSFID command to the contactless tag.

Table 37. *ISO15693*_LockDSFID function description

Function description	
Prototype	`result = int8_t ISO15693_LockDSFID (uc8 Flags, uc8 *UIDin, uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.15 ISO15693_GetMultipleBlockSecutityStatus function

This function sends a GetMultipleBlockSecutityStatus command to the contactless tag.

Table 38. *ISO15693*_GetMultipleBlockSecutityStatus function description

Function description	
Prototype	`result = int8_t ISO15693_GetMultipleBlockSecutityStatus (uc8 Flags, uc8 *UIDin, uc8 BlockNumber, uc8 NbBlocks,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **NbBlocks**: number of block **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse:** pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.16 ISO15693_StayQuiet function

This function sends a StayQuiet command to the contactless tag.

Table 39. *ISO15693*_StayQuiet function description

Function description	
Prototype	`result = int8_t ISO15693_StayQuiet(uc8 Flags,uc8 *UIDin,uc8 AppendCRC,uc8 *CRC16);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.17 ISO15693_GetSystemInfo function

This function sends a GetSystemInfo command to the contactless tag.

Table 40. *ISO15693*_GetSystemInfo function description

Function description	
Prototype	`result = int8_t ISO15693_GetSystemInfo (uc8 Flags, uc8 *UIDin, uc8 AppendCRC,uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.4.18 ISO15693_SendEOF function

This function transmits an RF pulse (EOF). This is useful for Inventory or write command (ISO15693 command).

Table 41. ISO15693_SendEOF function description

Function description	
Prototype	`result = int8_t ISO15693_SendEOF (uint8_t *pResponse );`
Input parameter	none
Output parameter	**pResponse** : pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

3.5 Advanced functions

The functions described in this section use the command functions to perform high level tasks, e.g. returning the contactless tag UID, the AFI or the DFID field.

3.5.1 ISO15693_SelectProtocol function

This function sends a SelectProtocol command to the CR95HF. For more information, please refer to CR95HF_ProtocolSelect function.

Table 42. ISO15693_SelectProtocol function description

Function description	
Prototype	`result = int8_t ISO15693_SelectProtocol (uc8 DataRate, uc8 TimeOrSOF, uc8 Modulation, uc8 SubCarrier, uc8 AppendCRC);`
Input parameter	**DataRate**: Datarate 0: 26 Kbps 1: 52 Kbps 2: 6 Kbps **TimeOrSOF**: 0: Respect 312 µs delay 1: Wait for SOF **Modulation**: modulation depth (0 for 100 % and 1 for 10 %) **SubCarrier**: single or dual subcarrier 0: single subcarrier 1: double subcarrier **AppendCRC**: append CRC 0: CRC manages by user 1: CRC will be append by CR95HF

Table 42. ISO15693_SelectProtocol function description (continued)

Function description	
Output parameter	none
Return parameter	RESULTOK: CR95HF returns a succeeded response ERRORCODE_GENERIC: Parameter error or CR95HF returns an error code

3.5.2 ISO15693_GetUID function

This function returns the UID of the ISO15693 contactless tag extracted from the ISO15693_Inventory command response.

Table 43. *ISO15693*_GetUID function description

Function description	
Prototype	`result = int8_t ISO15693_GetUID (uint8_t *UIDout);`
Input parameter	none
Output parameter	**UIDout**: pointer to contactless tag UID
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code ISO15693_ERRORCODE_PARAMETERLENGTH: MaskLength value is erroneous

3.5.3 ISO15693_GetDSFID function

This function returns the DSFID byte of the ISO15693 contactless tag, extracts from theISO15693_GetSystemInfo command response.

Table 44. *ISO15693*_GetDSFID function description

Function description	
Prototype	`result = int8_t ISO15693_GetDSFID (uint8_t *DSFIDout);`
Input parameter	none
Output parameter	**DSFIDout**: DSFID byte of the ISO15693 contactless tag
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

3.5.4 ISO15693_GetAFI function

This function returns the AFI byte of the ISO15693 contactless tag, extracted from the ISO15693_GetSystemInfo command response.

Table 45. *ISO15693*_GetAFI function description

Function description	
Prototype	`result = int8_t ISO15693_GetAFI (uint8_t *AFIout);`
Input parameter	none
Output parameter	**AFIout**: AFI byte of the ISO15693 contactless tag
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

3.5.5 ISO15693_GetMemSizeInfo function

This function returns the memory size information of the ISO15693 contactless tag, extracted from the ISO15693_GetSystemInfo command response.

Table 46. *ISO15693*_GetMemSizeInfo function description

Function description	
Prototype	`result = int8_t ISO15693_GetMemSizeInfo (uint8_t *NbBlock,uint8_t *MemSize);`
Input parameter	none
Output parameter	**NbBlock**: number of block of contactless tag memory **MemSize**: size of a block of contactless tag memory
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

3.5.6 ISO15693_GetIcRef function

This function returns the IC reference byte of ISO15693 card, extracted from the ISO15693_GetSystemInfo command response.

Table 47. *ISO15693*_GetIcRef function description

Function description	
Prototype	`result = int8_t ISO15693_GetIcRef (uint8_t *IcRefOut);`
Input parameter	none
Output parameter	**IcRefOut**: IC reference byte of the ISO15693 contactless tag
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

3.5.7 ISO15693_IsPresent function

This function returns RESULTOK if an ISO15693 contactless tag is present in the RF field.

Table 48. ISO15693_IsPresent function description

Function description	
Prototype	`result = int8_t ISO15693_IsPresent (void);`
Input parameter	none
Output parameter	none
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

3.5.8 ISO15693_IsCollisionDetected function

This function returns RESULTOK if a collision between two contactless tags has been detected.

Table 49. ISO15693_IsCollisionDetected function description

Function description	
Prototype	`result = int8_t ISO15693_IsCollisionDetected(uc8 *pTagReply);`
Input parameter	**pTagReply**: pointer of contactless tag response
Output parameter	none
Return parameter	RESULTOK: collision detected ERRORCODE_GENERIC: collision not detected

3.5.9 ISO15693_IsATagInTheField function

This function returns RESULTOK if at least one contactless tag is present in RF field.

Table 50. ISO15693_IsATagInTheField function description

Function description	
Prototype	`result = int8_t ISO15693_IsATagInTheField(uc8 *pTagReply);`
Input parameter	**pTagReply**: pointer to contactless tag response
Output parameter	none
Return parameter	RESULTOK: a contactless tag is present in the RF field ERRORCODE_GENERIC: no contactless tag detected

3.5.10 ISO15693_IsTagErrorCode function

This function returns RESULTOK if error flag is set, ERRORCODE_GENERIC otherwise

Table 51. ISO15693_IsTagErrorCode function description

Function description	
Prototype	`result = uint8_t ISO15693_IsTagErrorCode (uc8 Flag);`
Input parameter	**Flag**: response flags
Output parameter	**none**
Return parameter	RESULTOK: no error ERRORCODE_GENERIC: error detected

3.6 Extract flag from request flags functions

These functions return a bit extracted from the request flags (a byte). For more information, refer to ISO15693 specification and to M24LRXX-R datasheets.

Table 52. Description of request flags 1 to 4

Bit No	Flag
Bit 1	Subcarrier_flag
Bit 2	Data_rate_flag
Bit 3	Inventory_flag
Bit 4	Protocol_extension_flag

Table 53. Description of request flags 5 to 8 when Bit 3 = 0

Bit No	Flag
Bit 5	Select flag
Bit 6	Address flag
Bit 7	Option flag
Bit 8	RFU

Table 54. Description of request flags 5 to 8 when Bit 3 = 1

Bit No	Flag
Bit 5	AFI flag
Bit 6	Nb_slots flag
Bit 7	Option flag
Bit 8	RFU

3.6.1 ISO15693_GetSubCarrierFlag function

This function returns the Subcarrier flag.

Table 55. *ISO15693*_GetSubCarrierFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetSubCarrierFlag(uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	Subcarrier flag

3.6.2 ISO15693_GetDataRateFlag function

This function returns the DataRate flag.

Table 56. *ISO15693*_GetDataRateFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetDataRateFlag(uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	DataRate flag

3.6.3 ISO15693_GetInventoryFlag function

This function returns the inventory flag.

Table 57. *ISO15693*_GetInventoryFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetInventoryFlag(uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	Inventory flag

3.6.4 ISO15693_IsInventoryFlag function

This function returns RESULTOK if the Inventory flag is set.

Table 58. *ISO15693*_IsInventoryFlag function description

Function description	
Prototype	`result = int8_t ISO15693_IsInventoryFlag(uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	RESULTOK: Inventory flag set ERRORCODE_GENERIC: Inventory flag reset

3.6.5 ISO15693_GetProtocolExtensionFlag function

This function returns the Protocol extension flag.

Table 59. *ISO15693*_GetProtocolExtensionFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetProtocolExtensionFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	Protocol extension flag

3.6.6 ISO15693_GetSelectOrAFIFlag function

This function returns the Select or AFI flag.

Table 60. *ISO15693*_GetSelectOrAFIFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetSelectOrAFIFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	Select or AFI flag

3.6.7 ISO15693_GetAddressOrNbSlotsFlag function

This function returns the Address or the NbSlots flag.

Table 61. *ISO15693*_GetAddressOrNbSlotsFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetAddressOrNbSlotsFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	Address or NbSlots flag

3.6.8 ISO15693_IsAddressOrNbSlotsFlag function

This function returns RESULTOK if AddressOrNbSlots flag is set.

Table 62. *ISO15693*_IsAddressOrNbSlotsFlag function description

Function description	
Prototype	`result = int8_t ISO15693_IsAddressOrNbSlotsFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	RESULTOK: AddressOrNbSlots Flag is set ERRORCODE_GENERIC: AddressOrNbSlots Flag is reset

3.6.9 ISO15693_GetOptionFlag function

This function returns the option flag.

Table 63. *ISO15693*_GetOptionFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetOptionFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	option flag

3.6.10 ISO15693_GetRFUFlag function

This function returns the RFU flag.

Table 64. *ISO15693*_GetRFUFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetRFUFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: request flags
Output parameter	none
Return parameter	RFU flag

3.7 Get flag from response flag byte function

3.7.1 ISO15693_GetErrorFlag function

This function returns error flag from response flags.

Table 65. *ISO15693*_GetErrorFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetErrorFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: response flags
Output parameter	none
Return parameter	Error_flag

3.8 Get flag from information flag byte functions

Table 66. Description of information flag byte

Bit nb	Flag name	Description
1	DSFID	0: DSFID field is not present 1: DSFID field is present
2	AFI	0: AFI field is not present 1: AFI field is present
3	memory size	0: memory size field is not present 1: memory size field is present
4	IC reference	0: IC reference field is not present 1: IC reference field is present
5 to 8	RFU	

3.8.1 ISO15693_GetDSFIDFlag function

This function returns the DSFID flag extracted from the information flags.

Table 67. *ISO15693*_GetDSFIDFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetDSFIDFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: information flags
Output parameter	none
Return parameter	DSFID flag

3.8.2 ISO15693_GetAFIFlag function

This function returns AFI flag extracted from the information flags.

Table 68. *ISO15693*_GetAFIFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetAFIFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: information flags
Output parameter	none
Return parameter	AFI flag

3.8.3 ISO15693_GetMemorySizeFlag function

This function returns Memory Size flag extracted from the information flags.

Table 69. *ISO15693*_GetMemorySizeFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetMemorySizeFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: information flags
Output parameter	none
Return parameter	Memory Size flag

3.8.4 ISO15693_GetICReferenceFlag function

This function returns the IC reference flag.

Table 70. IISO15693_GetICReferenceFlag function description

Function description	
Prototype	`result = int8_t ISO15693_GetICReferenceFlag (uc8 FlagsByte);`
Input parameter	**FlagsByte**: information flags
Output parameter	none
Return parameter	IC reference flag

3.9 Split contactless tag response functions

An contactless tag response can contain various information. The functions described herein split the contactless tag response and return the offset byte into the pResponse parameter.

3.9.1 ISO15693_SplitInventoryResponse function

This function returns the response flags, DSFID byte and UID extracted from the inventory response.

Table 71. ISO15693_SplitInventoryResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitInventoryResponse (uc8 *pResponse,uc8 Length, uint8_t *Flags, uint8_t *DSFIDextract, uint8_t *UIDout);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **DSFIDextract**: DSFID byte **UIDout**: UID index
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.2 ISO15693_SplitGetSystemInfoResponse function

This function returns the different information extracted from a response to a GetSystemInfo command.

Table 72. ISO15693_SplitGetSystemInfoResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitGetSystemInfoResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *InfoFlags,uint8_t *UIDout,uint8_t *DSFIDout,uint8_t *AFIout,uint8_t *MemSizeOut,uint8_t *IcRefOut,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **InfoFlags**: informations flags **UIDout**: UID index **DSFIDout**: DSFID byte (optional) **AFIout**: AFI byte (optional) **MemSizeOut**: Memory size field (optional) **IcRefOut**: IC reference out (optional) **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.3 ISO15693_SplitMemorySizeInfo function

This function extracts the block size and number of blocks of MemSizeInfo field (response to GetSystemInfoResponse command, see *Table 73*).

For more information, refer to *Section 3.9.2: ISO15693_SplitGetSystemInfoResponse function*.

Table 73. Description of memory size field (MemSizeOut parameter)

Bit nb	Flag name	Description
16 to 14	RFU	
13 to 9	block size	Number of bytes of one contactless tag memory block
8 to 1	number of blocks	Number of blocks in the contactless tag memory

Table 74. ISO15693_SplitMemorySizeInfo function description

Function description	
Prototype	`result = int8_t ISO15693_SplitMemorySizeInfo (uc16 MemSizeInfo,uint8_t *BlockSize ,uint8_t *NbBlocks);`
Input parameter	**MemSizeInfo:** Memory size field
Output parameter	**BlockSize**: size of contactless tag memory block **NbBlocks**: number of blocks in the contactless tag memory
Return parameter	RESULTOK

3.9.4 ISO15693_SplitReadSingleBlockResponse function

This function returns the different information extracted from the response to a ReadSingleBlock command.

Table 75. ISO15693_SplitReadSingleBlockResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitReadSingleBlockResponse (uc8 *pResponse,uc8 Length,uc8 NbByteOfData,uc8 OptionFlag,uint8_t *Flags ,uint8_t *SecurityStatus,uint8_t *Data,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of bytes of pResponse **NbByteOfData**: block length expressed in bytes **OptionFlag**: Option flag of request flags
Output parameter	**Flags**: response flags **SecurityStatus**: block security status (optional) **Data**: pointer to a block **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is validated CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.5 ISO15693_SplitReadMultipleBlockResponse function

This function returns the different information from a response of ReadMultipleBlock command.

Table 76. ISO15693_SplitReadMultipleBlockResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitReadMultipleBlockResponse(uc8 *pResponse,uc8 Length,uc8 NbBlock,uc8 NbByteOfData,uc8 OptionFlag,uint8_t *Flags ,uint8_t *SecurityStatus,uint8_t *Data,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of bytes of pResponse **NbBlock**: Number of blocks to read **NbByteOfData**: block length (in bytes) **OptionFlag**: Option flag of request flags
Output parameter	**Flags**: response flags **SecurityStatus**: security status of block (optional) **Data**: pointer to data blocks **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.6 ISO15693_SplitWriteSingleBlockResponse function

This function returns the different information extracted from the response to a WriteSingleBlock command.

Table 77. ISO15693_SplitWriteSingleBlockResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitWriteSingleBlockResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.7 ISO15693_SplitWriteMultipleBlockResponse function

This function returns the different information extracted from the response to a Write Multiple Block command.

Table 78. ISO15693_SplitWriteMultipleBlockResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitWriteMultipleBlockResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.8 ISO15693_SplitWriteAFIResponse function

This function returns the different information extracted from the response to a Write AFI command.

Table 79. ISO15693_SplitWriteAFIResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitWriteAFIResponse(uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.9 ISO15693_SplitWriteDSFIDResponse function

This function returns the different information extracted from the response to a Write DSFID command.

Table 80. ISO15693_SplitWriteDSFIDResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitWriteDSFIDResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.10 ISO15693_SplitLockBlockResponse function

This function returns the different information extracted from the response to a Lock block command.

Table 81. ISO15693_SplitLockBlockResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitLockBlockResponse(uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.11 ISO15693_SplitLockAFIResponse function

This function returns the different information extracted from the response to a Lock AFI command.

Table 82. ISO15693_SplitLockAFIResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitLockAFIResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.12 ISO15693_SplitLockDSFIDResponse function

This function returns the different information extracted from the response to a Lock DSFID command.

Table 83. ISO15693_SplitLockDSFIDResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitLockDSFIDResponse (uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid

3.9.13 ISO15693_SplitSelectResponse function

This function returns the different information extracted from the response to a Select command.

Table 84. ISO15693_SplitSelectResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitSelectResponse(uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	Flags: response flags ErrorCode: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.14 ISO15693_SplitResetToReadyResponse function

This function returns the different information extracted from the response to a Reset to ready command.

Table 85. ISO15693_SplitResetToReadyResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitResetToReadyResponse(uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse
Output parameter	**Flags**: response flags **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

3.9.15 ISO15693_SplitGetMultipleBlockSecutityResponse function

This function returns the different information extracted from the response to a Get Multiple Block Security command.

Table 86. ISO15693_SplitGetMultipleBlockSecutityResponse function description

Function description	
Prototype	`result = int8_t ISO15693_SplitGetMultipleBlockSecutityResponse( uc8 *pResponse,uc8 Length,uc8 NbBlock,uint8_t *Flags ,uint8_t *DataOut,uint8_t *ErrorCode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of byte of pResponse **NbBlock**: number of blocks
Output parameter	**Flags**: response flags **DataOut**: pointer to data **ErrorCode**: CR95HF Error code
Return parameter	ISO15693_ERRORCODE_CRCRESIDUE: the contactless tag residue is erroneous RESULTOK: the contactless tag response is valid CR95HF_ERROR_CODE: the CR95HF returned an error code

4 M24LRXX-R Layer

The M24LRXX-R layer is composed of:

- the lib_M24LRXX.c source file
- the lib_M24LRXX.h include file

4.1 Overview

The library includes all the commands defined in the M24LRXX-R datasheets. Since M24LRXX-R devices are compliant with ISO15693 specifications, several functions of the M24LRXX-R layer are identical to ISO15693 functions.

4.1.1 CRC management

The two-byte CRC can be either computed by the user or by the CR95HF device, depending on the value of the AppendCRC input parameter.

- If AppendCRC input parameter is set to ‘0’, the two-byte CRC is calculated by the user application.
- If AppendCRC input parameter is set to ‘1’, the two-byte CRC is calculated by the CR95HF. This configuration is recommended to avoid calculation errors.

The CRC16 input parameter is appended to the ISO15693 command when the user has chosen to manage the CRC.

4.1.2 Request flag management

The request flag byte is managed by the user application. Each bit or flag specifies the actions to be performed by the contactless tag and whether the corresponding fields are present or not. Bit 3 (Inventory_flag) of the request flag defines the contents of the 4 MSBs (bits 5 to 8). When bit 3 is reset (0), bits 5 to 8 define the contactless tag selection criteria. When bit 3 is set (1), bits 5 to 8 define the contactless tag Inventory parameters.

For more information, please refer to ISO15693 specification and to the M24LRXX-R datasheets.

4.1.3 Request flags and CR95HF_ProtocolSelect functions

The CR95HF_ProtocolSelect function (defined into CR95HF layer) selects the RF protocol and defines for CR95HF device, datarate and contactless tag response format (single or double subcarrier). Once the protocol and the RF parameters are configured, the CR95HF is able to decode only contactless tag responses with same format and datarate.

The datarate and contactless tag response format are also defined in the request flag byte of each command. The user application must ensure that the datarate and subcarrier flags match the CR95HF_ProtocolSelect function parameters. This check is performed by the ISO15693 layer functions. If the parameters do not match, the functions are not executed and the corresponding commands are not sent to the CR95HF.

4.1.4 Extension flag

The protocol extension flag is set to ’1’ for the following functions as described in M24LRXX-R datasheet:

- M24LRXX_ReadSingleBlock
- M24LRXX_WriteSingleBlock
- M24LRXX_ReadMultipleBlocks
- M24LRXX_GetMultipleBlockSecutityStatus

4.2 M24LRXX-R layer commands

Table 87. M24LRXX-R global functions

Function name	Brief description	New or equivalent function
M24LRXX_SelectProtocol		ISO15693_SelectProtocol
M24LRXX_CreateRequestFlag		ISO15693_CreateRequestFlag
M24LRXX_Inventory		ISO15693_Inventory
M24LRXX_InventoryOneSlot		ISO15693_InventoryOneSlot
M24LRXX_Inventory16Slots		ISO15693_Inventory16Slots
M24LRXX_SendEOF		ISO15693_SendEOF
M24LRXX_GetSystemInfo		ISO15693_GetSystemInfo
M24LRXX_ReadSingleBlock	Send a read single block command	New function
M24LRXX_WriteSingleBlock	Send a write single block command	New function
M24LRXX_ReadMultipleBlocks	Send a read multiple block command	New function
M24LRXX_Select		ISO15693_Select
M24LRXX_ResetToReady		ISO15693_ResetToReady
M24LRXX_WriteAFI		ISO15693_WriteAFI
M24LRXX_LockAFI		ISO15693_LockAFI
M24LRXX_WriteDSFID		ISO15693_WriteDSFID
M24LRXX_LockDSFID		ISO15693_LockDSFID
M24LRXX_GetMultipleBlockSecutityStatus	Send a Get multiple block security status command	New function
M24LRXX_StayQuiet		ISO15693_StayQuiet

Table 88. M24LRXX-R specific functions

Function name	Brief description	New or equivalent function
M24LRXX_WriteSectorPassword	Send a Write Sector Password command	New function
M24LRXX_LockSectorPassword	Send a Lock Sector Password command	New function
M24LRXX_PresentSectorPassword	Send a PresentSectorPassword command	New function
M24LRXX_FastReadSingleBlock	Send a Fast Read Single Block command	New function
M24LRXX_FastReadMultipleBlock	Send a Fast Read Multiple Block command	New function
M24LRXX_FastInventoryInitiated	Send a Fast Inventory initiated command	New function
M24LRXX_FastInitiate	Send a Fast Inventory command	New function
M24LRXX_InventoryInitiated	Send a Fast Inventory Initiated command	New function
M24LRXX_Initiate	Send an Initiate command	New function
M24LRXX_GetUID		ISO15693_GetUID
M24LRXX_GetDSFID		ISO15693_GetDSFID
M24LRXX_GetAFI		ISO15693_GetAFI
M24LRXX_GetMemSizeInfo	Returns memory size flag from information flag	New function
M24LRXX_GetIcRef	Returns IC reference flag from information flag	New function
M24LRXX_IsAM24LR64	Checks if contactless tag is a M24LR64-R	New function

Table 89. M24LRXX-R flags functions

Function name	Brief description	New or equivalent function
M24LRXX_GetDSFIDFlag		ISO15693_GetDSFIDFlag
M24LRXX_GetAFIFlag		ISO15693_GetAFIFlag
M24LRXX_GetMemorySizeFlag		ISO15693_GetMemorySizeFlag
M24LRXX_GetICReferenceFlag		ISO15693_GetICReferenceFlag
M24LRXX_SplitInventoryResponse		ISO15693_SplitInventoryResponse
M24LRXX_SplitGetSystemInfoRespons e	Splits get system info response of contactless tag	New function
M24LRXX_SplitReadSingleBlockRespo nse		ISO15693_SplitReadSingleBlockR esponse
M24LRXX_SplitWriteSingleBlockRespo nse		ISO15693_SplitWriteSingleBlockR esponse
M24LRXX_SplitWriteAFIResponse		ISO15693_SplitWriteAFIResponse
M24LRXX_SplitWriteDSFIDResponse		ISO15693_SplitWriteDSFIDRespon se
M24LRXX_SplitLockAFIResponse		ISO15693_SplitLockAFIResponse
M24LRXX_SplitLockDSFIDResponse		ISO15693_SplitLockDSFIDRespon se
M24LRXX_SplitReadMultipleBlockResp onse		ISO15693_SplitReadMultipleBlock Response
M24LRXX_SplitSelectResponse		ISO15693_SplitSelectResponse
M24LRXX_SplitResetToReadyRespons e		ISO15693_SplitResetToReadyRes ponse
M24LRXX_SplitGetMultipleBlockSecutit yResponse		ISO15693_SplitGetMultipleBlockSe cutityResponse
M24LRXX_SplitWriteSectorPasswordRe sponse		ISO15693_SplitWriteSingleBlockR esponse
M24LRXX_SplitLockSectorPasswordRe sponse		ISO15693_SplitLockSingleBlockRe sponse
M24LRXX_SplitPresentSectorPassword Response		ISO15693_SplitWriteSingleBlockR esponse
M24LRXX_SplitFastReadSingleBlockRe sponse		ISO15693_SplitReadSingleBlockR esponse
M24LRXX_SplitInventoryInitiatedRespo nse		ISO15693_SplitInventoryResponse
M24LRXX_SplitFastInitiateResponse		ISO15693_SplitInventoryResponse
M24LRXX_SplitInitiateResponse		ISO15693_SplitInventoryResponse
M24LRXX_SplitFastReadMultipleBlockR esponse		ISO15693_SplitReadMultipleBlock Response
M24LRXX_SplitMemorySizeInfo		New function

4.3 M24LRXX-R global functions

4.3.1 M24LRXX_ReadSingleBlock function

This function sends a read single block command to a contactless tag. The protocol extension of the request flags is forced to '1'.

Table 90. M24LRXX_ReadSingleBlock function description

Function description	
Prototype	`result = int8_t M24LRXX_ReadSingleBlock (uc8 Flags, uc8 *UID, uc8 *BlockNumber, uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UID**: pointer to contactless tag UID **BlockNumber**: pointer to block index **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	RESULTOK: CR95HF returns a succeeded code ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code

4.3.2 M24LRXX_WriteSingleBlock function

This function sends a Write single block command to a contactless tag. The protocol extension of the request flags is forced to '1'.

Table 91. M24LRXX_WriteSingleBlock function description

Function description	
Protocol	`result = int8_t M24LRXX_WriteSingleBlock (uc8 Flags, uc8 *UIDin, uc8 *BlockNumber, uc8 BlockLength,uc8 *DataToWrite,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: pointer to block index **BlockLength**: number of byte of a block **DataToWrite**: pointer to data to write into contactless tag memory **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.3.3 M24LRXX_ReadMultipleBlocks function

This function sends a Read Multiple Blocks command to a contactless tag. The protocol extension of the request flags is forced to '1'.

Table 92. M24LRXX_ReadMultipleBlocks function description

Function description	
Protocol	`result = int8_t M24LRXX_ReadMultipleBlocks (uc8 Flags, uc8 *UIDin, uc8 *BlockNumber, uc8 NbBlocks,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UID**: pointer to contactless tag UID **BlockNumber**: pointer to block index **NbBlocks**: number of block to read **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.3.4 M24LRXX_GetMultipleBlockSecutityStatus function

This function sends a Get Multiple Block Security Blocks command to a contactless tag. The protocol extension of the request flags is forced to '1'.

Table 93. M24LRXX_GetMultipleBlockSecutityStatus function description

Function description	
Protocol	`result = int8_t M24LRXX_GetMultipleBlockSecutityStatus (uc8 Flags, uc8 *UIDin, uc8 *BlockNumber, uc8 *NbBlocks,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse);`
Input parameter	**Flags**: request flags **UIDin**: pointer on contactless tag UID **BlockNumber**: pointer to block index **NbBlocks**: number of block to read **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4 M24LRXX-R specific functions

4.4.1 M24LRXX_WriteSectorPassword function

This function sends a Write Sector Password command to a contactless tag. It is specific to M24LRXX-R devices. For more information, please refer to the M24LR64-R datasheet.

Table 94. M24LRXX_WriteSectorPassword function description

Function description	
Protocol	`result = int8_t M24LRXX_WriteSectorPassword (uc8 Flags, uc8 *UIDin, uc8 PasswordNumber,uc8 *Password,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer on contactless tag UID **PasswordNumber**: password index **Password**: password to write into contactless tag **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.2 M24LRXX_LockSectorPassword function

This function sends a Lock Sector Password command to a contactless tag. It is specific to M24LRXX-R devices. For more information, please refer to the M24LR64-R datasheet.

Table 95. M24LRXX_LockSectorPassword function description

Function description	
Protocol	`result = int8_t M24LRXX_LockSectorPassword (uc8 Flags, uc8 *UIDin, uc8 *SectorNumber,uc8 SectorSecurityStatus,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer on contactless tag UID **SectorNumber**: sector index **SectorSecurityStatus**: security status of the block **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.3 M24LRXX_PresentSectorPassword function

This function sends a Present Sector Password command to a contactless tag. It is specific to M24LRXX-R devices.

Table 96. M24LRXX_PresentSectorPassword function description

Function description	
Protocol	`result = int8_t M24LRXX_PresentSectorPassword(uc8 Flags, uc8 *UIDin, uc8 PasswordNumber,uc8 *Password,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **PasswordNumber**: password index **Password**: pointer to password to test **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.4 M24LRXX_FastReadSingleBlock function

This function sends a Fast Read Single Block command to a contactless tag. It is specific to M24LRXX-R products.

The protocol extension flag is forced to '1'.

Table 97. M24LRXX_FastReadSingleBlock function description

Function description	
Protocol	`result = int8_t M24LRXX_FastReadSingleBlock (uc8 Flags, uc8 *UIDin, uc8 *BlockNumber,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.5 M24LRXX_FastReadMultipleBlock function

This function sends a Fast Read Multiple Block command to a contactless tag. It is specific to M24LRXX-R devices.

The protocol extension flag is forced to '1'.

Table 98. M24LRXX_FastReadMultipleBlock function description

Function description	
Protocol	`result = int8_t M24LRXX_FastReadMultipleBlock (uc8 Flags, uc8 *UIDin, uc8 *BlockNumber, uc8 NumberOfBlock,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **UIDin**: pointer to contactless tag UID **BlockNumber**: block index **NumberOfBlock** : number of blocks to read **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.6 M24LRXX_FastInventoryInitiated function

This function sends a Fast Inventory Initiated command to a contactless tag. It is specific to M24LRXX-R devices.

Table 99. M24LRXX_FastInventoryInitiated function description

Function description	
Protocol	`result = int8_t M24LRXX_FastInventoryInitiated (uc8 Flags, uc8 AFIin, uc8 MaskLength,uc8 *Mask,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **AFI**: AFI word **MaskLength**: number of bit of mask **MaskValue**: mask **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.7 M24LRXX_FastInitiate function

This function sends a Fast Initiate command to a contactless tag. It is specific to M24LRXX-R devices.

Table 100. M24LRXX_FastInitiate function description

Function description	
Protocol	`result = int8_t M24LRXX_FastInitiate (uc8 Flags,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.8 M24LRXX_InventoryInitiated function

This function sends a Inventory Initiated command to a contactless tag. It is specific to M24LRXX-R devices.

Table 101. M24LRXX_InventoryInitiated function description

Function description	
Protocol	`result = int8_t M24LRXX_InventoryInitiated (uc8 Flags, uc8 AFIin, uc8 MaskLength,uc8 *Mask,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **AFI**: AFI word **MaskLength**: number of bits of the mask **MaskValue**: mask **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.9 M24LRXX_Initiate function

This function sends a Initiate command to a contactless tag. It is specific to M24LRXX-R devices.

Table 102. M24LRXX_Initiate function description

Function description	
Protocol	`result = int8_t M24LRXX_Initiate (uc8 Flags,uc8 AppendCRC, uc8 *CRC16,uint8_t *pResponse );`
Input parameter	**Flags**: request flags **AppendCRC**: CRC16 management **CRC16**: pointer to user CRC16
Output parameter	**pResponse**: pointer to CR95HF response
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.10 M24LRXX_GetMemSizeInfo function

This function returns the memory size information of a contactless tag, extracted from the response to a M24LRXX_GetSystemInfo command.

Table 103. M24LRXX_GetMemSizeInfo function description

Function description	
Protocol	`result = int8_t M24LRXX_GetMemSizeInfo (uint8_t *NbBlock,uint8_t *MemSize);`
Input parameter	none
Output parameter	**NbBlock**: pointer to number of blocks of contactless tag memory **MemSize**: size of contactless tag memory block
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.11 M24LRXX_GetIcRef function

This function returns the IC reference byte of a contactless tag, extracted from the response to an M24LRXX_GetSystemInfo command.

Table 104. M24LRXX_GetIcRef function description

Function description	
Protocol	`result = int8_t M24LRXX_GetIcRef (uint8_t *IcRefout);`
Input parameter	none
Output parameter	**IcRefOut**: IC reference byte of an ISO15693 contactless tag
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

4.4.12 M24LRXX_IsAM24LR64 function

This function returns RESULTOK if the contactless tag is an M24LR64-R device.

Table 105. M24LRXX_IsAM24LR64 function description

Function description	
Protocol	`result = int8_t M24LRXX_IsAM24LR64 (uint8_t *Mask,uint8_t MaskLength);`
Input parameter	**Mask**: Mask value (optional) **MaskLength**: number of bits of the mask
Output parameter	none
Return parameter	ERRORCODE_GENERIC: The contactless tag is not an M24LR64-R device RESULTOK: The contactless tag is an M24LR64-R device

4.5 M24LRXX-R flags functions

4.5.1 M24LRXX_SplitGetSystemInfoResponse function

This function returns the different information extracted from the response to a GetSystemInfo command.

Table 106. ISO15693_SplitGetSystemInfoResponse function description

Function description	
Protocol	`result = int8_t M24LRXX_SplitGetSystemInfoResponse(uc8 *pResponse,uc8 Length,uint8_t *Flags ,uint8_t *InfoFlags,uint8_t *UIDout,uint8_t *DSFIDout,uint8_t *AFIout,uint8_t *MemSizeOut,uint8_t *IcRefOut,uint8_t *errorcode);`
Input parameter	**pResponse**: pointer to CR95HF response **Length**: number of bytes in pResponse
Output parameter	**Flags**: response flags **InfoFlags**: informations flags **UIDout**: UID index **DSFIDout**: DSFID byte (optional) **AFIout**: AFI byte (optional) **MemSizeOut**: Memory Size field (optional) **IcRefOut**: IC reference out (optional) **ErrorCode**: CR95HF Error code
Return parameter	ERRORCODE_GENERIC: a parameter value is erroneous or CR95HF returns an error code RESULTOK: CR95HF returns a succeeded code

5 Project example

In this example, the ISO15693 protocol is selected and the CR95HF device performs an inventory of a single tag.

5.1 Project description

5.1.1 Hardware

The project runs on an STM3210B-EVAL evaluation board.

5.1.2 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 http://www.keil.com. An evaluation version is available free of charge for projects which do not exceed 32 Kbytes.

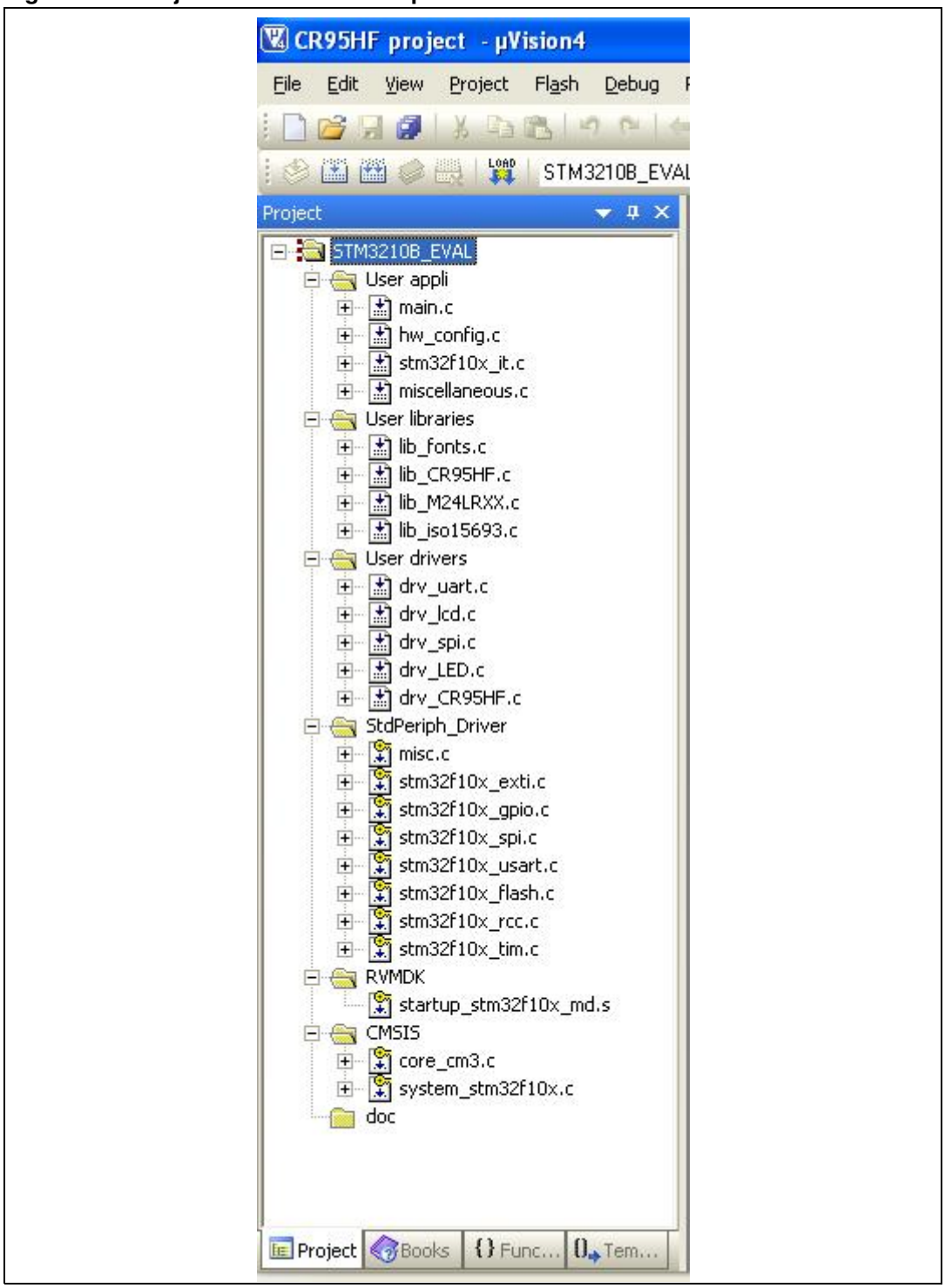
5.1.3 Project structure on Keil µvision

Figure 5 shows an example of project:

1. Prior to starting your project, download the STM32F10x_StdPeriph_Lib_V3.4.0 library from http://www.st.com. It allows to drive all the peripherals used by your application.
2. Install the library (lib_CR95HF.c, lib_M24LRxx.c, and lib_iso15693.c). It includes the three layers described in this application note.
3. An additional lib_fonts.c file can be delivered on request to drive the LCD display available on STM3210B-EVAL board.
4. Other dedicated peripheral drivers can be provided on request to control the UART, the LCD, LEDs, the SPI, and the CR95HF.

For the complete project, contact your local ST sales offices.

Figure 5. Project structure on Keil µvision



5.2 Application functions

The steps required to setup communications with an M24LRXX-R tag are the following:

1. Configure the CR95HF by issuing an M24LRXX_SelectProtocol command to select ISO15693 protocol. The parameters passed by the command are:
   a) Backscatter data rate set to 26 kbps (ISO15693_TRANSMISSION_26)
   b) Modulation depth of 100% (ISO15693_MODULATION_100)
   c) Single subcarrier for contactless tag response (ISO15693_SINGLE_SUBCARRIER)
   d) CRC16 management byte (M24LRXX_APPENDCRC)

```
M24LRXX_SelectProtocol(M24LRXX_TRANSMISSION_26,
   M24LRXX_WAIT_FOR_SOF,
   M24LRXX_MODULATION_100,
   M24LRXX_SINGLE_SUBCARRIER,
   M24LRXX_APPENDCRC);
```

2. Create a request flags according to the previous select command. The subcarrier and data rate must match.

Table 107. Select command and request flag corresponding parameters

Select command parameter	Request flags parameter
M24LRXX_SINGLE_SUBCARRIER	M24LRXX_REQFLAG_SINGLESUBCARRIER
M24LRXX_TRANSMISSION_26	M24LRXX_REQFLAG_HIGHDATARATE

```
InventoryFlagByte = M24LRXX_CreateRequestFlag (

M24LRXX_REQFLAG_SINGLESUBCARRIER,
   M24LRXX_REQFLAG_HIGHDATARATE,
   M24LRXX_REQFLAG_INVENTORYFLAGSET,
   M24LRXX_REQFLAG_NOPROTOCOLEXTENSION,
   M24LRXX_REQFLAG_NOTAFI,
   M24LRXX_REQFLAG_1SLOT,
   M24LRXX_REQFLAG_OPTIONFLAGNOTSET,
   M24LRXX_REQFLAG_RFUNOTSET);
```

3. Send an inventory command:

```
M24LRXX_InventoryOneSlot (
   InventoryFlagByte ,
   MAIN_DONTCARE,
   MAIN_DONTCARE,
   MAIN_DONTCARE,
   M24LRXX_APPENDCRC,
   MAIN_DONTCARE,
   ReaderRecBuf );
```

4. Extract the different elements from the CR95HF response. The inventory response contains a reply flag, the DSFID byte, and the contactless tag UID. The whole Split commands checks the CRC16 residue of contactless tag reply.

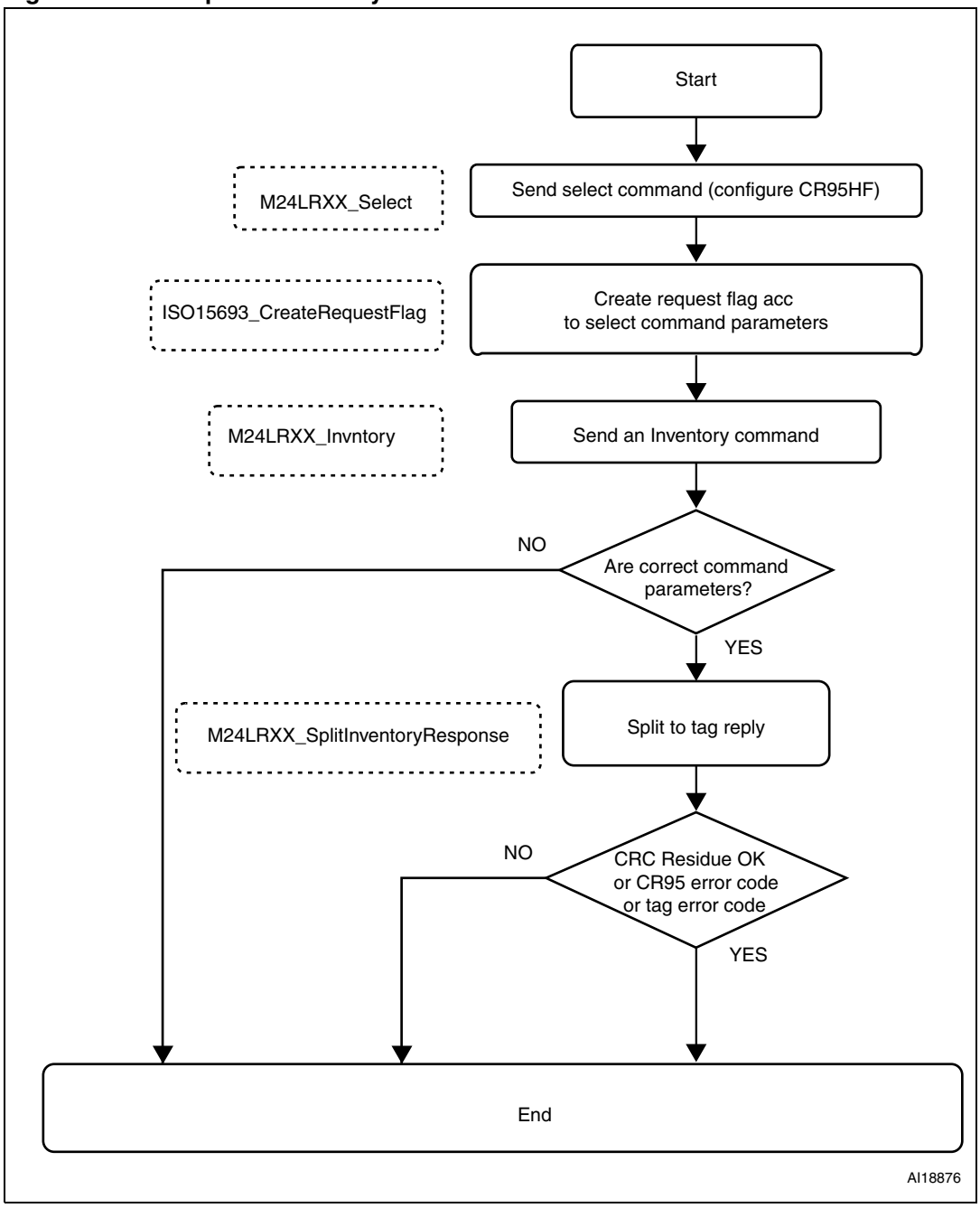
```
status = M24LRXX_SplitInventoryResponse

   ( ReaderRecBuf,
```

```
ReaderRecBuf[LENGTH_OFFSET],
&ResponseFlags ,
&DSFIDextract,
UIDout);
```

5.3 Typical inventory flowchart

Figure 6. Example of Inventory flowchart



5.4 Revision history

Table 108. Document revision history

Date	Revision	Changes
22-Apr-2011	1	Initial release.