AN4593
Application note

CR95HF C++ Library for Linux User

Introduction

This application note explains how to use CR95HF C++ Library for Linux user (STSW-95HF004). CR95HF library is developed on Linux platform to expose a number of APIs that can be used by Linux users to communicate with CR95HF RF transceiver board.

This board is delivered with the M24LR-DISCOVERY kit.

The CR95HF RF transceiver board is powered through the USB port and no external power supply is required. It is made up of a CR95HF contactless transceiver, a 48 x 34 mm 13.56 MHz inductive etched antenna and its associated tuning components. The CR95HF communicates with the STM32F103CB 32-bit core MCU via the SPI bus.

A dynamic link library (.so) file is available that can be used by Linux host computer to manage several functions and communicate with the STM32 MCU and the CR95HF IC.

A Linux command line test application is also available to validate and test the functionality of developed library.

Figure 1. CR95HF RF transceiver board

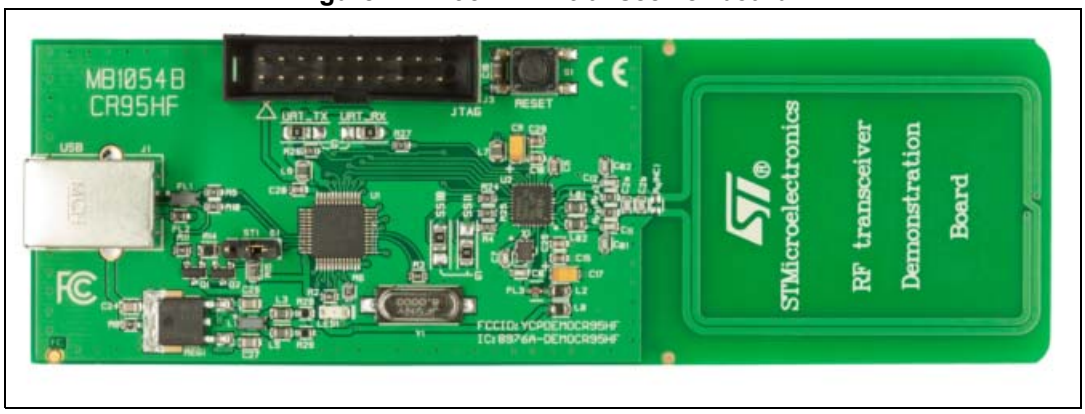


Table 1. Applicable tools and software

Type	Root Part numbers
Evaluation Tools	M24LR-DISCOVERY
Software	STSW-95HF004

Contents

List of tables

List of figures

1 Getting started

This application note and the test application have been written in order to help Linux developers to easily use CR95HF reader. This application note explains all the functions exposed by the Linux dynamic link library (.so file) and their description. The extension ".so" in Linux stands for shared object.

1.1 Connecting the board to your computer

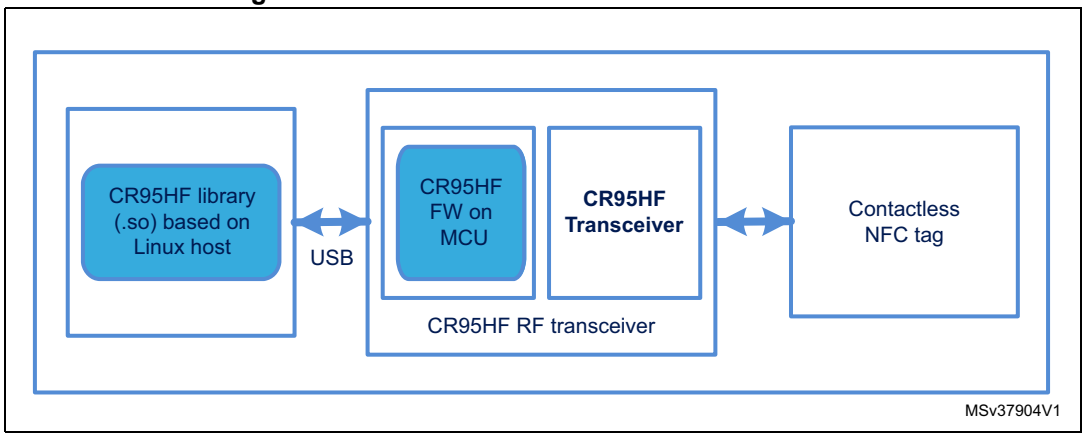
The CR95HF RF transceiver board is connected to the Linux host computer through its USB port.

The developed library for CR95HF uses "libusb" at the lower level to communicate with USB connected devices on Linux platform.

libusb is a C library that gives applications easy access to USB devices on many different operating systems. libusb is an open source project (http://www.libusb.org/).

Figure 2 shows the connection of CR9HF RF transceiver demo board with Linux machine is through the USB. Libusb based Linux library for CR95HF reader is available on host side that communicate with the CR95HF firmware running on MCU. CR95HF transceiver available on demo board communicates with NFC tag through RF communication.

Figure 2. Connection of board with Linux Machine



1.2 Using the Linux Library

Below steps are required to be followed to create the library and to run the test application on any Linux machine (here, Ubuntu machine is taken as Linux host machine).

Pre-Requisite:

"**build-essential**", "**g++**" and "**libusb**" packages are required to be installed on Linux machine to proceed for library creation.

If not, please follow below steps:

- For "build-essential" pkg installation, run the command:

```
apt-get install build-essential
```

- For "g++" pkg installation, run the command:

```
apt-get install g++
```

- For "libusb" pkg installation, run the command:

```
apt-get install libusb-1.0-0-dev
```

1.2.1 Library creation:

Unzip the package locally on Linux machine and run the below commands from the location where package is unzipped.

- Step 1: Compile the source code and generate object files.

```
g++ -g -c -Wall -Werror -fPIC -I. HIDManager.cpp libcr95hf.cpp
```

- Step 2: Create the library (.so) from object files.

```
g++ -g -shared -o libCR95HF.so libcr95hf.o HIDManager.o
```

- Step 3: Export the path of library to System Library Path so that library can be used by other applications.

```
export LD_LIBRARY_PATH= < Path of the generated libCR95HF.so
file>:$LD_LIBRARY_PATH
```

1.2.2 Test application compilation and execution:

In the package there is a file named TestApp.cpp, it is the test application that can be used to test and validate the developed Linux library.

- Step 1: Compilation of test application:

```
g++ -g -L<Path_to_libusb_library>  -L<path_to_CR95HF_library>  -Werror
TestApp.cpp -o TestApp  -lCR95HF -lusb-1.0
```

Example:

```
g++ -g -L/lib/x86_64-linux-gnu  -L.  -Werror TestApp.cpp -o TestApp  -
lCR95HF -lusb-1.0
```

Here, path of libusb library is /lib/x86_64-linux-gnu and path of CR95HF library is same from where the command is executing.

- Step 2: Execution of test application

```
./TestApp
```

Note: Please note, this application is needed to be run from "root" to make the application able to open the device for USB communication.

On the execution, this application displays a number of options for the user to select depending on which action he want to perform. A screen shot of TestApp execution is available in the Appendix B.

Whenever a new terminal is opened to build/execute the application, it is required to run the setp3 to include the path of ".so" file in environment variable "LD_LIBRARY_PATH".

2 Function Description

This section explains about the different functions exposed by the library that can be used by any other application to communicate with CR95HF reader.

List of Library functions are:

1. CR95HFlib_USBConnect
2. CR95HFlib_MCUVer
3. CR95HFDll_Echo
4. CR95HFlib_Idn
5. CR95HFlib_Select
6. CR95HFlib_SendReceive
7. CR95HFlib_Read_Block
8. CR95HFlib_Write_Block
9. CR95HFlib_FieldOff
10. CR95HFlib_ResetSPI
11. CR95HFlib_SendIRQPulse
12. CR95HFlib_getInterfacePinState

2.1 Functions to check USB connection

2.1.1 CR95HFlib_USBConnect

This function detects if the CR95HF board is properly connected with Linux machine and lower level driver Libusb available on host side is able to properly communicate with the board.

Declaration:	int CR95HFlib_USBConnect()
Prototype:	int iResult; iResult=CR95HFlib_USBConnect();
Input parameter:	None
Output parameter:	None
Returned value:	iResult= 0: No Error 1: CR95HF RF transceiver board not connected

Example:

Test Application code to validate CR95HFlib_USBConnect()

```
void Device_Connect()
{
char entry3;
   int iResult;
   iResult=CR95HFlib_USBConnect();
   printf("\n Establishing CR95HR Reader connection through USB \n");
   printf("\n --> Library function call : CR95HFlibUSB_Connect() \n");
   printf("\n <-- Return from Library function : 0X%x \n",iResult);

   if (iResult == 0)
      printf("\n SUCCESS : Board connected successfully and ready to use
\n");
   else
      printf("\n ERROR : Connection failed \n");
      printf("\n Selected Task is completed, To proceed for another task put
   your    choice");
      printf("\n");
   scanf("%c", &entry3);
}
```

2.2 Functions to communicate with the STM32 MCU

2.2.1 CR5HFlib_Echo

This function sends a USB request to the STM32 MCU that executes an Echo request on the CR95HF. The STM32 MCU sends back the answer of the CR95HF, if success, or returns an error code ‘1’ if there is no answer.

Declaration:	int CR95HFlib_Echo (char* strAnswer);
Prototype:	char strAnswer[50]=""; int iresult; iresult= CR95HFlib_Echo (strAnswer);
Input parameter:	None
Output parameter:	strAnswer: The CR95HF IC answer to the Echo request is “5500” if there is no error. Answer example: “5500”
Returned value:	iResult= 0: No Error 5: CR95HF RF transceiver board not connected

Example

Test Application code to validate CR95HFlib_Echo

```
void Echo ()
{
   char strAnswer[50]="";
   char entry3;
   printf("\n Echo command sent to MCU \n");
   int iresult= CR95HFlib_Echo (strAnswer);
   printf("\n --> Library function call : CR95HFlib_Echo (strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if(iresult == 0)
      printf("\n SUCCESS : Echo command answer = %s \n", strAnswer);
   else
      printf("\n ERROR : Echo command failed, no answer \n");

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);

}
```

2.2.2 CR95HFlib_MCUrev

This function sends a USB request to the STM32 MCU on the CR95HF RF transceiver board that sends back the revision number of its firmware.

Declaration:	int CR95HFlib_MCUVer(char* StringReply);
Prototype:	char strAnswer[50]=""; int iresult; iresult= CR95HFlib_MCUVer(strAnswer);
Input parameter:	None
Output parameter:	strAnswer: Firmware revision of the STM32 MCU on CR95HF RF transceiver board. Answer example: ““0003010300” Where: 00: Status byte 03: Size of answer (in bytes) 010300: Revision 1.3.0
Returned value:	iResult= 0: No Error 5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate CR95HFlib_MCUrev

```
void Get_MCU_rev ()
{
   char strAnswer[50]="";
   int iresult;
   char entry3;

   iresult = CR95HFlib_MCUVer(strAnswer);
   printf("\n Get MCU Version request is sent \n");
   printf("\n --> Library function call : CR95HFlibUSB_MCUVer(strAnswer)
\n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
      printf("\n SUCCESS: MCU Version Reply=%s \n", strAnswer);
   }

   else
   {
      printf("\n ERROR : Get MCU Version Failed \n");
```

```
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);

}
```

2.2.3 CR95HFlib_getInterfacePinState

This function verifies the communication path between the STM32 MCU and the CR95HF IC (either SPI or UART).

The STM32 MCU checks which communication configuration is selected on the CR95HF RF transceiver board.

Declaration: int CR95HFlib_getInterfacePinState(char*);

Prototype: char strAnswer[50]="";
int iresult;
CR95HFlib_getInterfacePinState(strAnswer);

Input parameter: None

Output parameter: strAnswer: strAnswer: Interface Pin state
Answer example: “80010X”
Where:
80: Status byte
01: Size of answer (in bytes)
0X: Communication mode
With X:
0: Communication in UART mode
1: Communication in SPI mode

Returned value: iResult= 0: No Error
5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate CR95HFlib_getInterfacePinState

```
void Get_InterfacePinState ()
{
   char strAnswer[50]="";
   int iresult;
   char entry3;

   iresult = CR95HFlib_getInterfacePinState(strAnswer);
   printf("\n Request for getInterfacePinState is sent \n");
   printf("\n --> Library function call :
   CR95HFlib_getInterfacePinState(strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
```

```
      printf("\n SUCCESS: getInterfacePinState completed successfully,Answer
   received= %s \n", strAnswer);
   if(strAnswer[3]=='0')
   {
      printf("\n  Communication is in UART mode \n");
   }
   else
   {
      printf("\n  Communication is in SPI mode \n");
   }
   }
   else
   {
      printf("\n ERROR : getInterfacePinState failed, No answer
received\n");
   }
   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3 Functions to communicate with the CR95HF IC

2.3.1 CR95HFlib_Idn

This function sends a USB request to the STM32 MCU that requests the IDN of the CR95HF IC. The STM32 MCU sends back the answer of the CR95HF containing the IDN value (ASCII codes), if success, or returns an error code ‘1’ if there is no answer.

Declaration:	int CR95HFlib_Idn(char*);
Prototype:	int iresult; char strAnswer[50]=""; iresult= CR95HFDll_Idn (strAnswer);
Input parameter:	None
Output parameter:	strAnswer: IDN of the CR95HF IC (if no error) Answer example: “000F4E4643204653324A415354320075D2” Where: 00: Status byte 0F: Size of answer (in bytes) 4E4643204653324A41535432: ASCII transcription of the CR95HF IDN 00: protocol status 75D2: CRC value
Returned value:	iresult: 0: No error 5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate CR95HFlib_Idn

```
void Idn ()
{
   int iresult;
   char strAnswer[50]="";
   char entry3;
   printf("\n IDN command is sent \n");

   iresult= CR95HFlib_Idn (strAnswer);
   printf("\n --> Library function call : CR95HFlib_Idn (strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
      printf("\n SUCCESS : Idn command response = %s \n", strAnswer);
```

```
   else
      printf("\n ERROR : no Idn returned \n",strAnswer);

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3.2 CR95HFlib_Select

This function sends a USB request to the STM32 MCU that prepares the CR95HF for communication by executing a Select request containing the selected RF parameters to the CR95HF IC.

The STM32 MCU sends back the answer of the CR95HF, if success, or returns an error code ‘1’ if there is no answer. In addition to selecting the correct RF communication parameters, this function activates the RF field.

Note: This is necessary at the start of communications if the RF field was previously switched off.

Declaration: int CR95HFlib_Select(char*, char*);

Prototype: int iresult;
char strRequest[50]="";
char strAnswer[50]="";
iresult = CR95HFlib_Select(strRequest ,strAnswer);

Input parameter: strRequest: Selected RF communication protocol and certain protocol-related parameters. (This configuration is used for SendReceive requests.)
For ex: strRequest for ISO15693 HighDataRate 10% One subcarrier is = "010D":
Where:
“01” is the ISO 15693 configuration
“0D” are the parameters

Output parameter: Answer: The CR95HF RF transceiver sends back an answer if the CR95HF is configured correctly and the RF field is on.
Answer example: “0000”
Where:
“00” is the status byte
“00” is the size of the answer

Returned value: iresult:
0: No error
5: CR95HF RF transceiver board not connected
2: Empty argument error
3: Command parameter error

Example:

Test Application code to validate CR95HFlib_Select

```
void Select_ISO15693 ()
{
   int iresult;
   char strRequest[50]="";
   char strAnswer[50]="";
```

```
   char entry3;

   printf("\n ISO 15693 Protocol selection for future NFC
communication:\n");
   strcpy(strRequest,"010D");
   iresult = CR95HFlib_Select(strRequest ,strAnswer);
   printf("\n --> Library function call : CR95HFlib_Select(strRequest
,strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
      printf("\n SUCCESS : ISO15693 protocol selected : Answer Received=%s
\n", strAnswer);
   else
      printf("\n ERROR : ISO15693 protocol selection failed :No Answer
Received \n", strAnswer);
      printf("\n Selected Task is completed, To proceed for another task put
your choice");
      printf("\n");
      scanf("%c", &entry3);
}
```

2.3.3 CR95HFlib_SendReceive

This function sends a USB request to the STM32 MCU that executes a SendRecv command with data to the CR95HF IC. The STM32 MCU sends back the answer of the CR95HF, if success, or returns an error code ‘1’ if there is no answer.

Note: The request uses the SendRecv command to send data using previously selected protocol and to receive the tag response. For more information, refer to the CR95HF transceiver datasheet.

Declaration: int CR95HFlib_SendReceive(char*,char*);

Prototype:

int iresult;

char strRequest[50]="";

char strAnswer[50]="";

iresult=CR95HFlib_SendReceive(strRequest,strTagAnswer);

Input parameter: strRequest: The RF Request to be sent by the CR95HF IC to the Tag (with previously selected ISO format).

ISO 15693 Inventory example: “260100”

Where:

“260100” is the ISO 15693 Inventory command.

Output parameter: strTagAnswer: The Tag answer if the CR95HF has received an answer from the Tag in the field; otherwise, an error code.

ISO 15693 Inventory example:

800D00FF6820492F6A5C02E00CD800

Where

“80” is the status byte

“0D” is the length of the entire data field

“00FF6820492F6A5C02E0” is the data received from the tag

“0CD8” is the original received CRC value

“00” is the protocol error status

Returned value: iresult:

0: No error

5: CR95HF RF transceiver board not connected

4: Communication error

Example:

Test Application code to validate library function CR95HFlib_SendReceive

```
void Send_ISO15693_Inventory ()
{
   int iresult;
```

```
   char strRequest[50]="";
   char strTagAnswer[50]="";
   char entry3;

   strcpy(strRequest,"260100");

   iresult=CR95HFlib_SendReceive(strRequest,strTagAnswer);

   printf("\nISO15695 inventory using CR95HF SendReceive command:\n");
   printf("\n --> Library function call : CR95HFlib_Select(260100
,strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if ((strTagAnswer[0] == '8') & (strTagAnswer[1] == '0')) //CR95HF Tag
answer OK
   {
      printf("\n SUCCESS : Tag answer=%s \n",strTagAnswer);
   }
   else
   {
      printf("\n ERROR : No tag answer received \n",strTagAnswer);
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3.4 CR95HFlib_Read_Block

This function will read the particular block of Tag memory. This function takes the block no. as a input parameter and sends back the 4 byte data available at that particular block.

Declaration:	int CR95HFlib_Read_Block(int,unsigned char*);
Prototype:	nt iresult; int RegAdd; unsigned char strTagAnswer[50]=""; iresult = CR95HFlib_Read_Block(RegAdd, strTagAnswer);
Input parameter:	RegAdd: Address of register of which data is required
Output parameter:	strTagAnswer: array of unsigned char to contain the 4 bytes data available at RegAdd
Returned value:	iresult: 0: No error 5: CR95HF RF transceiver board not connected 4: Communication error

Example:

Test Application code to validate library function CR95HFlib_Read_Block

```
void Read_Block()
{
   char entry3;
   unsigned char strTagAnswer[50]="";
   int RegAdd=0;
   printf("\n This option will read 4 byte data of the block entered by user
   \n");
   printf("\n");
   printf("\n please enter the Block address in hex \n");
   printf("\n");
   scanf("%x", (int*)&RegAdd);
   int r = CR95HFlib_Read_Block(RegAdd, strTagAnswer);
   printf("\n --> Library function call : CR95HFlib_Read_Block(RegAdd,
strTagAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",r);
   if(r==0)
   {
      printf("\n SUCCESS : Received Tag answer=%s \n",strTagAnswer);

      for(int i=3;i<=6;i++)
      {
         printf("%x",(int)strTagAnswer[i]);
         printf(" ");
```

```
      }
   }
   else
   {
      printf("\n ERROR : No tag answer received \n",strTagAnswer);
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

This test application asks the user to enter the register address of which data is needed to be read. After the execution of function it returns the 4 bytes data available at that particular location separated by space.

2.3.5 CR95HFlib_Write_Block

This function writes the data into particular block of Tag memory. This function takes the block number and the data needed to be written on memory as the input parameter.

Declaration: int CR95HFlib_Write_Block(int,unsigned char*,unsigned char*);

Prototype:
int iresult;
int WriteAdd;
unsigned char strTagAnswer[50]="";
unsigned char strBytestowrite[50]="";
iresult = CR95HFlib_Write_Block(WriteAdd, strTagAnswer,strBytestowrite);

Input parameter: WriteAdd: Address at which data is needed to be written
strBytestowrite: array of unsigned char to store the 4 bytes of data which is required to be written on memory of Tag

Output parameter: strTagAnswer : The Tag answer if the CR95HF has received an answer from the Tag in the field; otherwise, an error code

Returned value: iresult:
0: No error
5: CR95HF RF transceiver board not connected
4: Communication error

Example:

Test Application code to validate library function CR95HFlib_Write_Block

```
void Write_Block()
{
   char entry3;
   unsigned char strBytestowrite[50]="";
   unsigned char strTagAnswer[50]="";
   int WriteAdd;
   printf("\n This option will write 4 bytes of data entered by user into
the selected block\n");
   printf("\n");
   printf("\n please enter the Block address in hex");
   printf("\n");
   scanf("%x", (int*)&WriteAdd);

   printf("\n enter first byte \n");
   scanf("%x",(int*)&strBytestowrite[0]);
   printf("\n enter second byte \n");
   scanf("%x",(int*)&strBytestowrite[1]);
   printf("\n enter third byte \n");
```

```
   scanf("%x",(int*)&strBytestowrite[2]);
   printf("\n enter fourth byte \n");
   scanf("%x",(int*)&strBytestowrite[3]);
   printf("\n");
   int r = CR95HFlib_Write_Block(WriteAdd, strTagAnswer,strBytestowrite);

   printf("\n --> Library function call : CR95HFlib_Write_Block(WriteAdd,
strTagAnswer,strBytestowrite) \n");
   printf("\n <-- Return from Library function : 0X%x \n",r);

   if(r==0)
   {
      printf("\n SUCCESS : Data written into the block
successfully\n",strTagAnswer);

   }
   else
   {
      printf("\n ERROR : data write on block is failed \n");
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);

}
```

2.3.6 CR95HFlib_FieldOff

This function sends a USB request to the STM32 MCU to switch off the CR95HF RF Field. The STM32 MCU sends back the answer of the CR95HF, if success, or returns an error code ‘5’ if there is no answer.

Declaration:	Int CR95HFlib_FieldOff(char*);
Prototype:	int iresult; char strAnswer[50]=""; iresult= CR95HFlib_FieldOff (strAnswer);
Input parameter:	None
Output parameter:	strAnswer: The CR95HF RF transceiver sends back an answer and the RF Field is switched off. Answer example: “0000” Where: “00” is the status byte “00” is the size of the answer
Returned value:	result: 0: No error 5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate library function CR95HFlib_FieldOff

```
void FieldOff ()
{
   int iresult;
   char strAnswer[50]="";
   char entry3;
   printf("\n FieldOff command is sent \n");
   iresult= CR95HFlib_FieldOff (strAnswer);
   printf("\n --> Library function call : CR95HFDll_FieldOff(strAnswer)
\n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
      printf("\n SUCCESS : RF Field Off ok = %s \n", strAnswer);
   else
      printf("\n ERROR : RF Field Off command error \n",strAnswer);
      printf("\n Selected Task is completed, To proceed for another task put
      your choice");
      printf("\n");
      scanf("%c", &entry3);
}
```

2.3.7 CR95HFlib_ResetSPI

This function resets the CR95HF IC in case of a problem. This function only resets the CR95HF IC and not the STM32 MCU.

Declaration:	int CR95HFlib_ResetSPI(char*);
Prototype:	int iresult; char strAnswer[50]=""; iresult = CR95HFlib_ResetSPI(strAnswer);
Input parameter:	None
Output parameter:	strAnswer: The CR95HF RF transceiver sends back an answer if the SPI has been correctly reset. Answer example: “8000” Where: “80” is the status byte “00” is the size of the answer
Returned value:	iresult: 0: No error 5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate library function CR95HFlib_ResetSPI

```
void Reset_SPI ()
{

   char strAnswer[50]="";
   int iresult;
   char entry3;

   iresult = CR95HFlib_ResetSPI(strAnswer);
   printf("\n Reset_SPI request is sent \n");
   printf("\n --> Library function call : CR95HFlib_ResetSPI(strAnswer)
\n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
      printf("\n SUCCESS: Reset SPI successfully,Answer received= %s \n",
strAnswer);
   }
else
   {
```

```
      printf("\n ERROR : Reset SPI failed, No answer received\n");
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3.8 CR95HFlib_SendIRQPulse

This function must be used when the CR95HF RF transceiver is configured in SPI mode (communication between the STM32 MCU and the CR95HF IC). The interrupt pulse is sent to the CR95HF IRQ pin.

Declaration: int CR95HFlib_SendIRQPulse(char*);

Prototype: int iresult;

char strAnswer[50]="";

iresult = CR95HFlib_SendIRQPulse(strAnswer);

Input parameter: None

Output parameter: strAnswer: The CR95HF RF transceiver board sends back an answer if the IRQ

Pulse was correctly sent.

Answer example: “8000”

Where:

“80” is the status byte (see Appendix A for error codes)

“00” is the size of the answer

Returned value: iresult:

0: No error

5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate library function CR95HFlib_SendIRQPulse

```
void Send_IRQPulse ()
{
   char strAnswer[50]="";
   int iresult;
   char entry3;

   iresult = CR95HFlib_SendIRQPulse(strAnswer);
   printf("\n Send IRQPulse request is sent \n");
   printf("\n --> Library function call : CR95HFlib_SendIRQPulse(strAnswer)
   \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
   printf("\n SUCCESS: Send IRQPulse completed successfully,Answer
received= %s \n", strAnswer);
   }
```

```
   else
   {
      printf("\n ERROR : Send IRQPulse failed, No answer received\n");
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3.9 CR95HFlib_SendNSSPulse

This function sends an interrupt to wake up the CR95HF IC. It can be used when the CR95HF Demo board is configured in UART mode. The interrupt pulse is sent to the CR95HF NSS pin.

Declaration:	int CR95HFlib_SendNSSPulse (char*);
Prototype:	it iresult char strAnswer[50]=""; iresult = CR95HFlib_ SendNSSPulse (strAnswer);
Input parameter:	None
Output parameter:	strAnswer: The CR95HF RF transceiver board sends back an answer if NSS Pulse was correctly sent. Answer example: “8000” Where: “80” is the status byte (see Appendix A for error codes) “00” is the size of the answer
Returned value:	result: 0: No error 5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate library function CR95HFlib_SendNSSPulse

```
void Send_NSS_Pulse ()
{

   char strAnswer[50]="";
   int iresult;
   char entry3;

   iresult = CR95HFlib_SendNSSPulse(strAnswer);
   printf("\n Request for Send NSS_Pulse \n");
   printf("\n --> Library function call : CR95HFlib_SendNSSPulse(strAnswer)
\n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
      printf("\n SUCCESS: Send NSS pulse successfully,Answer received= %s
      \n", strAnswer);
```

```
   }

   else
   {
      printf("\n ERROR : Send NSS Pulse failed, No answer received\n");
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n");
   scanf("%c", &entry3);
}
```

2.3.10 CR95HFlib_STCmd

This function is used to send any request to CR95HF IC. The STM32 MCU receives the frame contained in the request and send it directly to the CR95HF IC. The STM32 MCU sends back the answer from CR95HF IC to the PC through USB port. The frame has to be formatted according to the CR95HF datasheet.

This function can be defined as a “Transparent Mode” command.

The CR95HF formatted frame has to be sent through this CR95HFlib_STCmd function. “01” is prepend in the frame as the header byte.

Declaration: int CR95HFlib_STCmd(char*,char*);

Prototype: it iresult
char strAnswer[50]="";
char strAnswer[50]="";
iresult = CR95HFlib_STCmd(strRequest,strAnswer);

Input parameter: strRequest : is the frame which will be directly send to CR95HF IC. This
frame is prepended by an additional byte “01”
Example:
For select protocol ISO 15693, strRequest will be “010202010D”
Where:
“01” is the transparent command header byte
“0202010D” is the protocol select frame
“02” is Protocol Select Command
“02” is the request length
“010D” are protocol select parameters

Output parameter: strAnswer: The CR95HF RF transceiver board sends back an answer
Answer example: “0000”
Where:
“00” is the status byte (see Appendix A for error codes)
“00” is the size of the answer

Returned value: iresult:
0: No error
5: CR95HF RF transceiver board not connected

Example:

Test Application code to validate library function CR95HFlib_STCmd(char*,char*)

```
void STCmd_ISO15693()
{
```

```
   char strRequest[50]="";
   char strAnswer[50]="";
   int iresult;
   char entry3;
   strcpy(strRequest,"010202010D");

   iresult = CR95HFlib_STCmd(strRequest,strAnswer);
   printf("\n ISO15693 Protocol select using CR95HFlib_STCmd function \n");
   printf("\n --> Library function call :
CR95HFlib_STCmd(strRequest,strAnswer) \n");
   printf("\n <-- Return from Library function : 0X%x \n",iresult);

   if (iresult == 0)
   {
      printf("\n SUCCESS: ISO15693 protocol is selected through STCmd,
      Answer received= %s \n", strAnswer);
   }

   else
   {
      printf("\n ERROR : ISO15693 protocol select through STCmd failed,
      Answer received= %s \n", strAnswer);
   }

   printf("\n Selected Task is completed, To proceed for another task put
your choice");
   printf("\n"); scanf("%c", &entry3);
}
```

Appendix A Error codes

Table 2. Error codes

Error code	Description
0000	Answer OK
8000	Answer OK
8200	Invalid command length
8300	Invalid protocol
8600	Communication error
8700	Frame wait time out OR no tag
8800	Invalid Start Of Frame
8900	Receive buffer overflow (too many bytes received)
8A00	Framing error (start bit = 0, stop bit = 1)
8B00	EGT time out (for ISOIEC 14443-B)
8C00	Invalid length. Used in Felica, when field length < 3
8D00	CRC error (Used in Felica protocol)
8E00	Reception lost without EOF received
8F00	No field
FD00	Time out - no answer from Tag detected by the CR95HF IC
FE00	Unknown error

Appendix B TestApp execution screenshot

Below is the screen shot of TestApp execution to explain how application executes, what are different options available and how the output will appear on the screen for user.

Figure 3. TestApp USER MENU screen shot

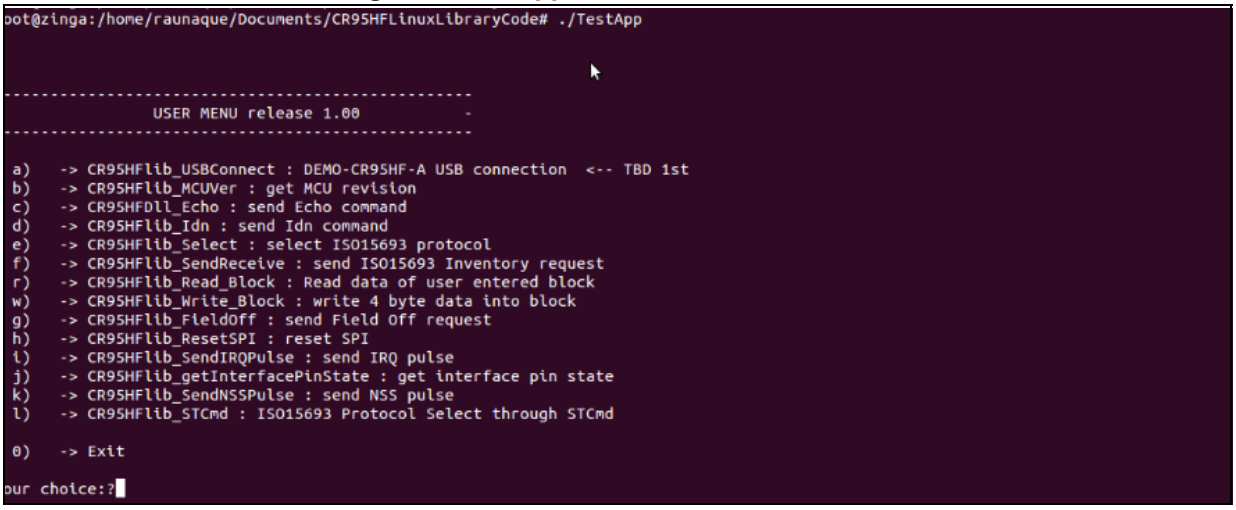


Figure 4. Option “a” TestApp execution

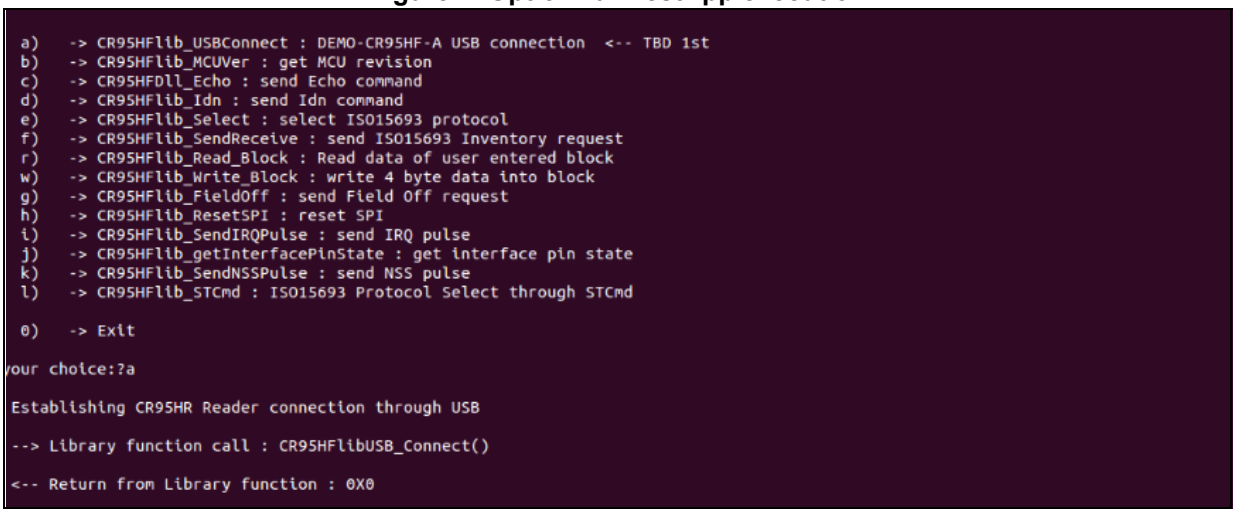


Figure 5. Option “b” TestApp execution

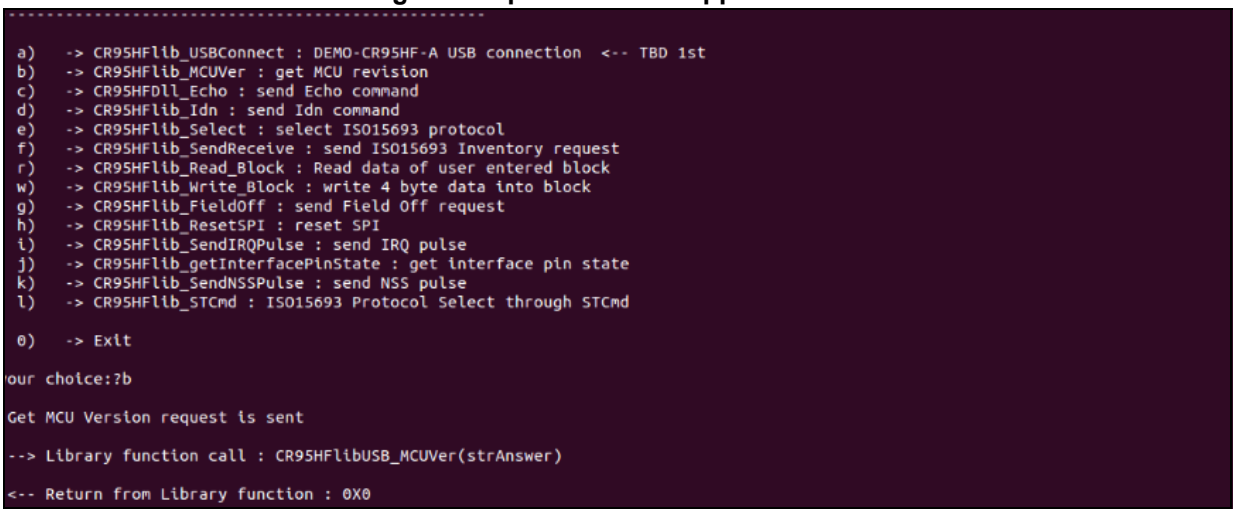


Figure 6. Option “c” TestApp execution

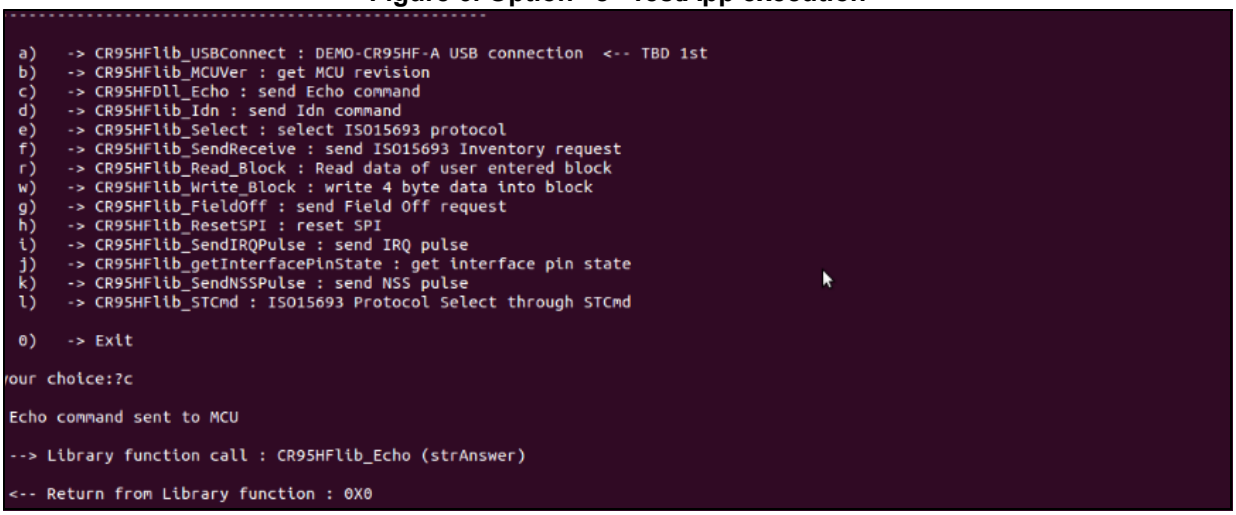


Figure 7. Option “d” TestApp execution

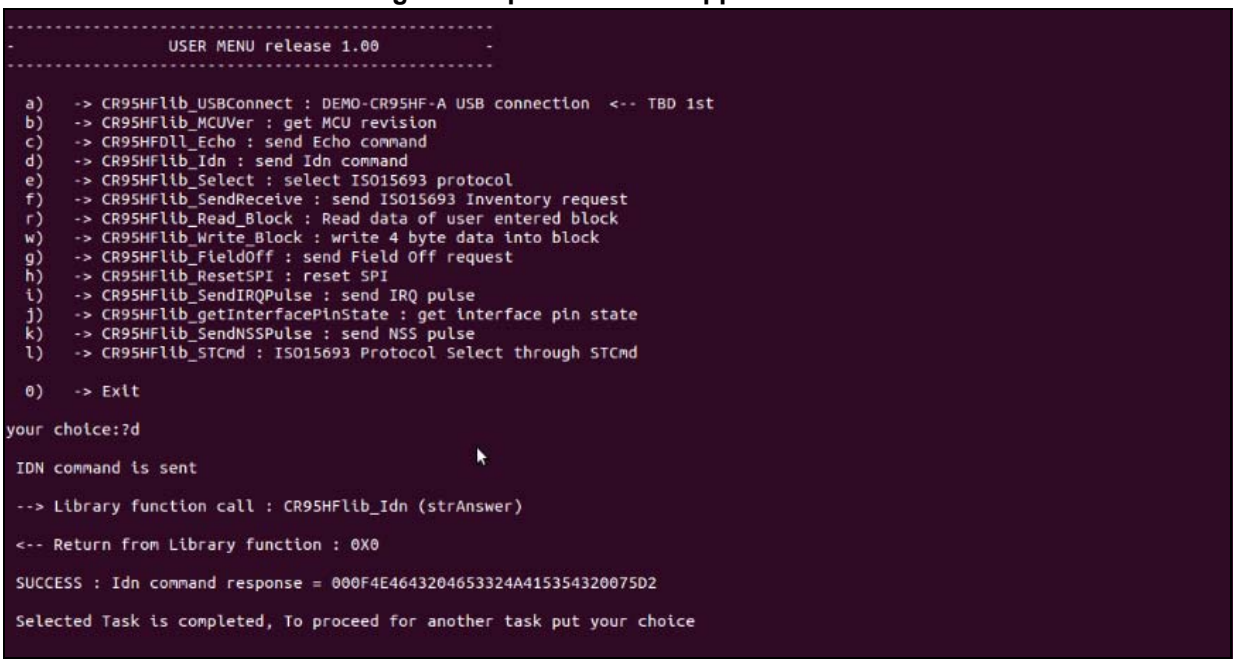


Figure 8. Option “e” TestApp execution

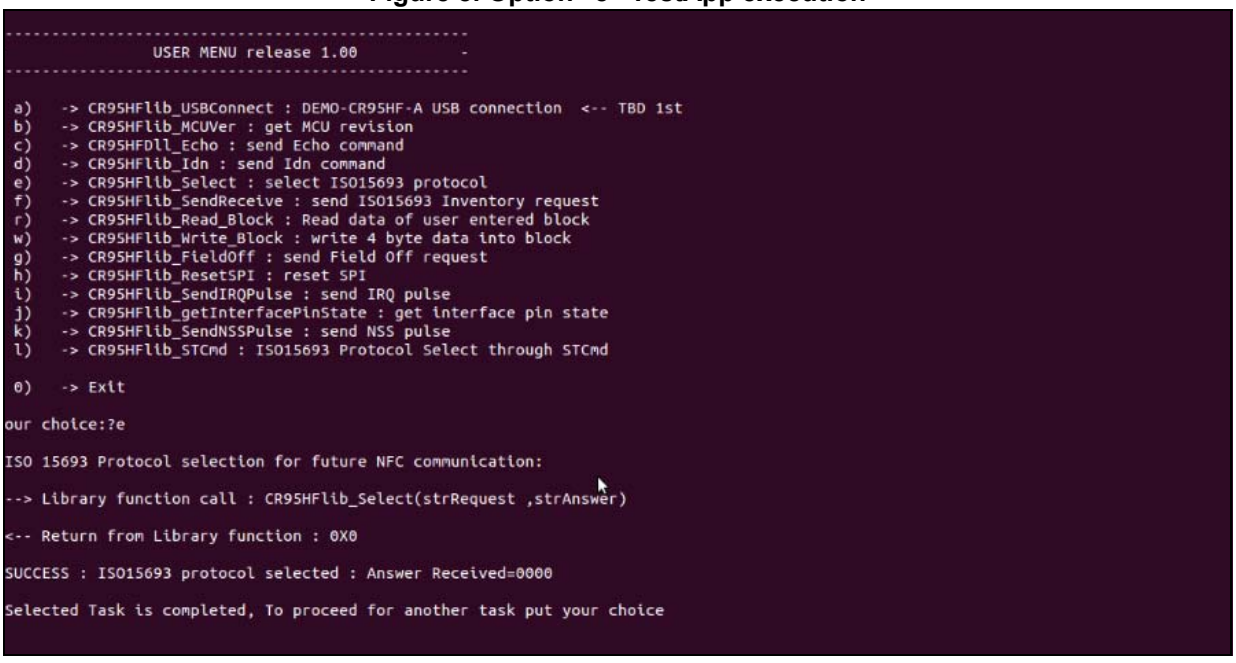


Figure 9. Option “f” TestApp execution

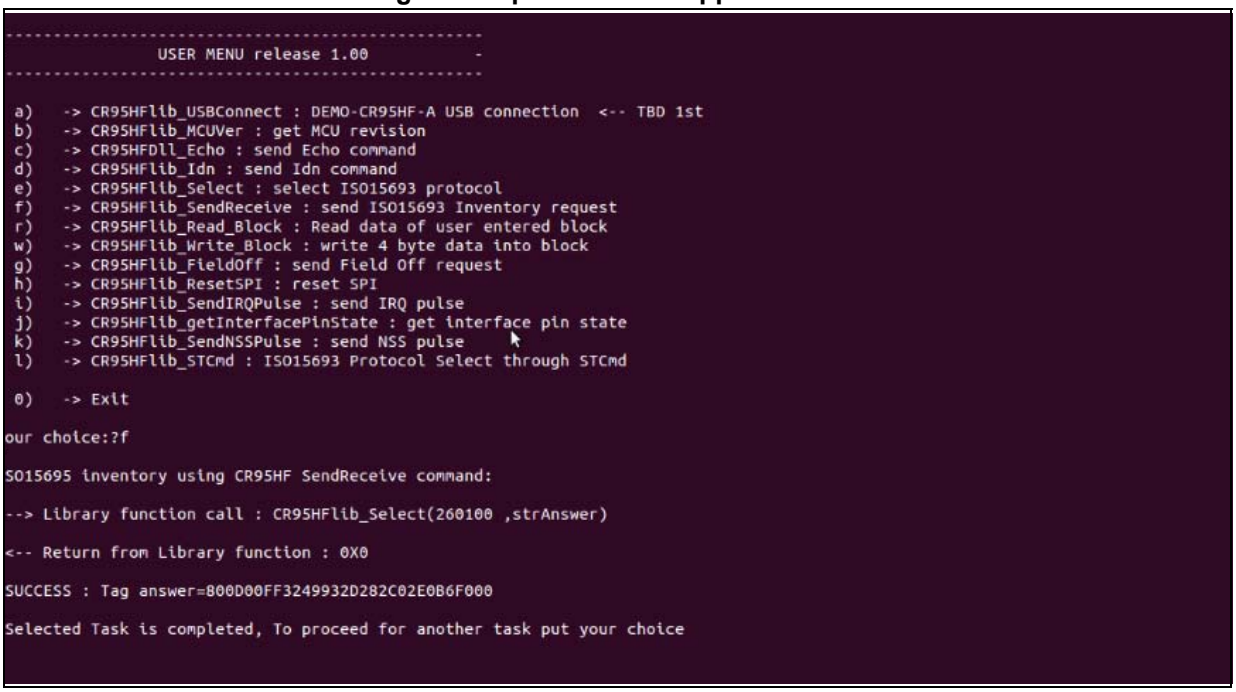


Figure 10. Option “r” TestApp execution

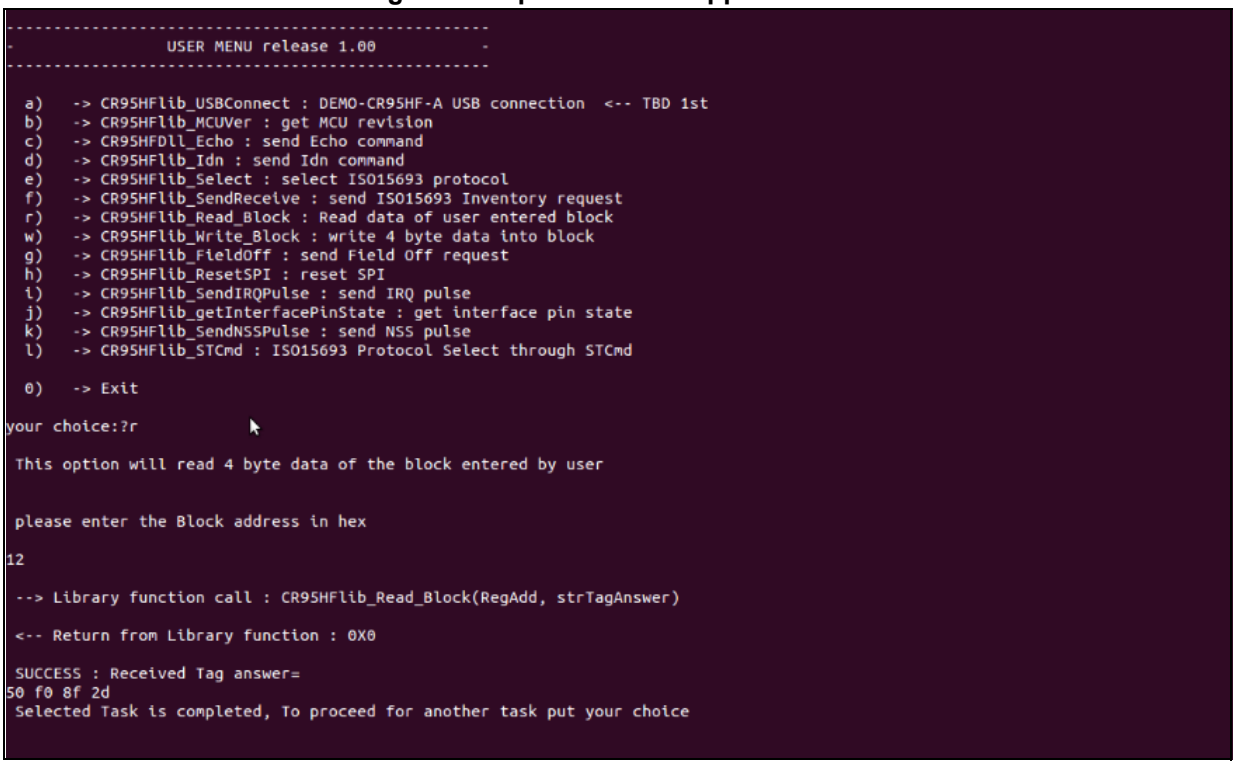


Figure 11. Option “w” TestApp execution

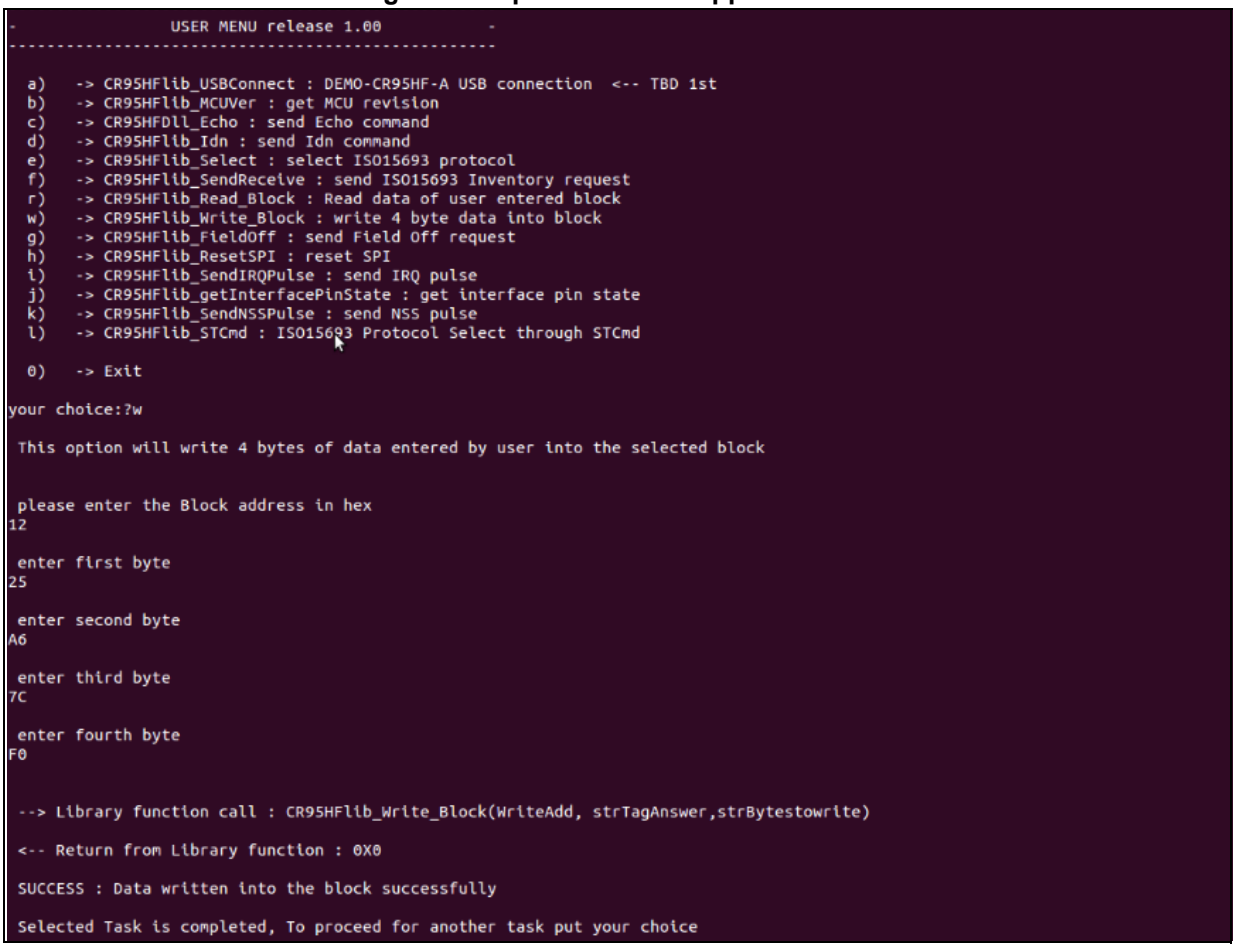


Figure 12. Option “r” TestApp execution after write

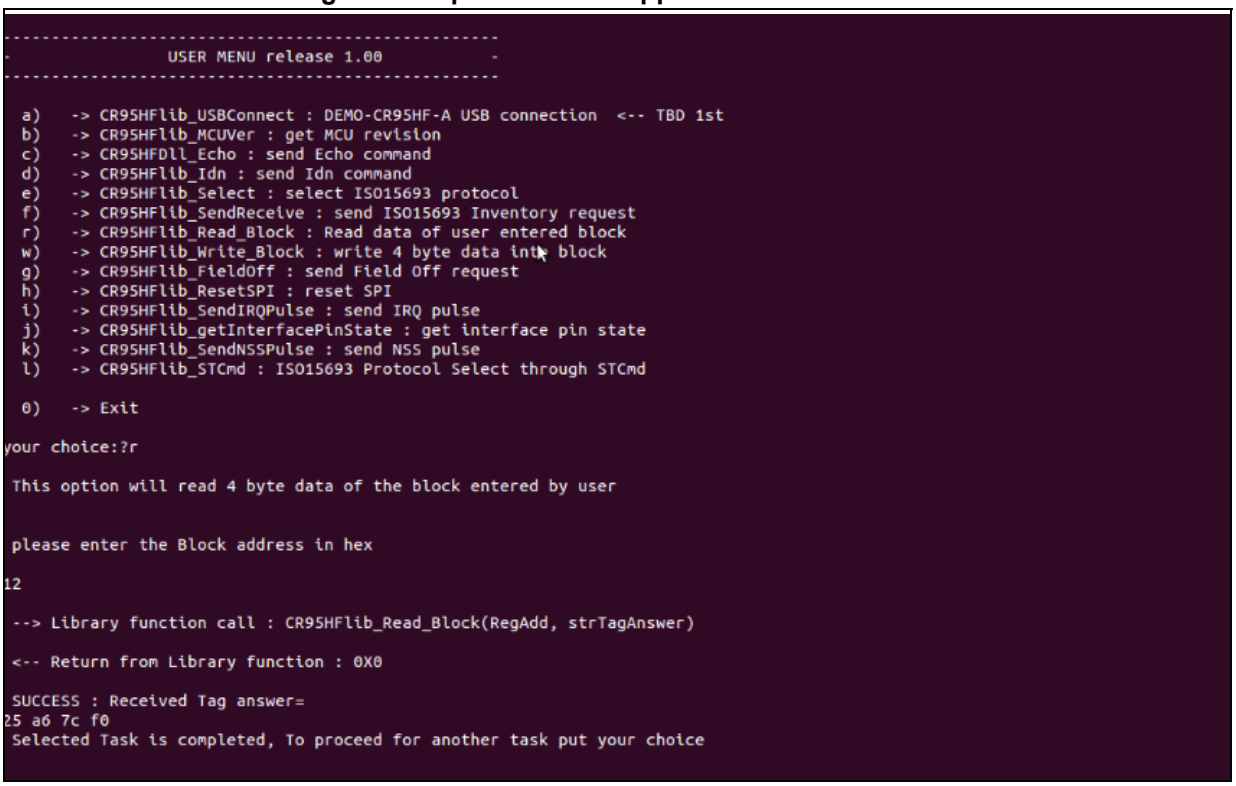


Figure 13. Option “g” TestApp execution

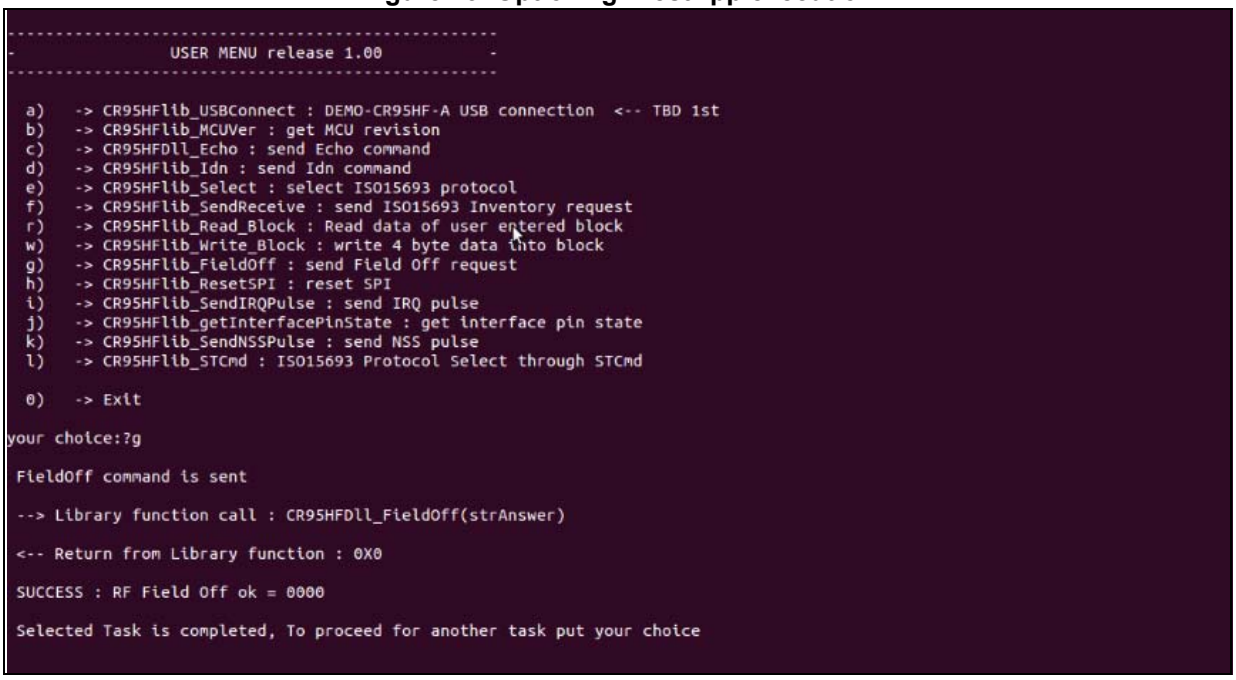


Figure 14. Option “h” TestApp execution

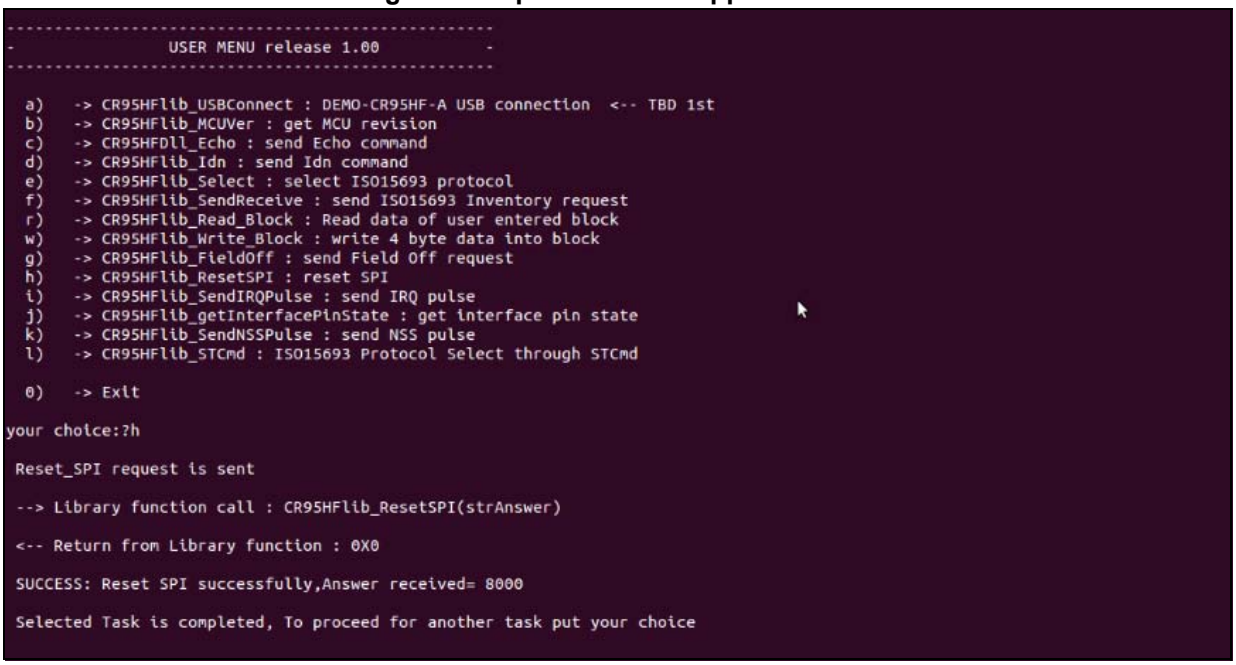


Figure 15. Option “i” TestApp execution

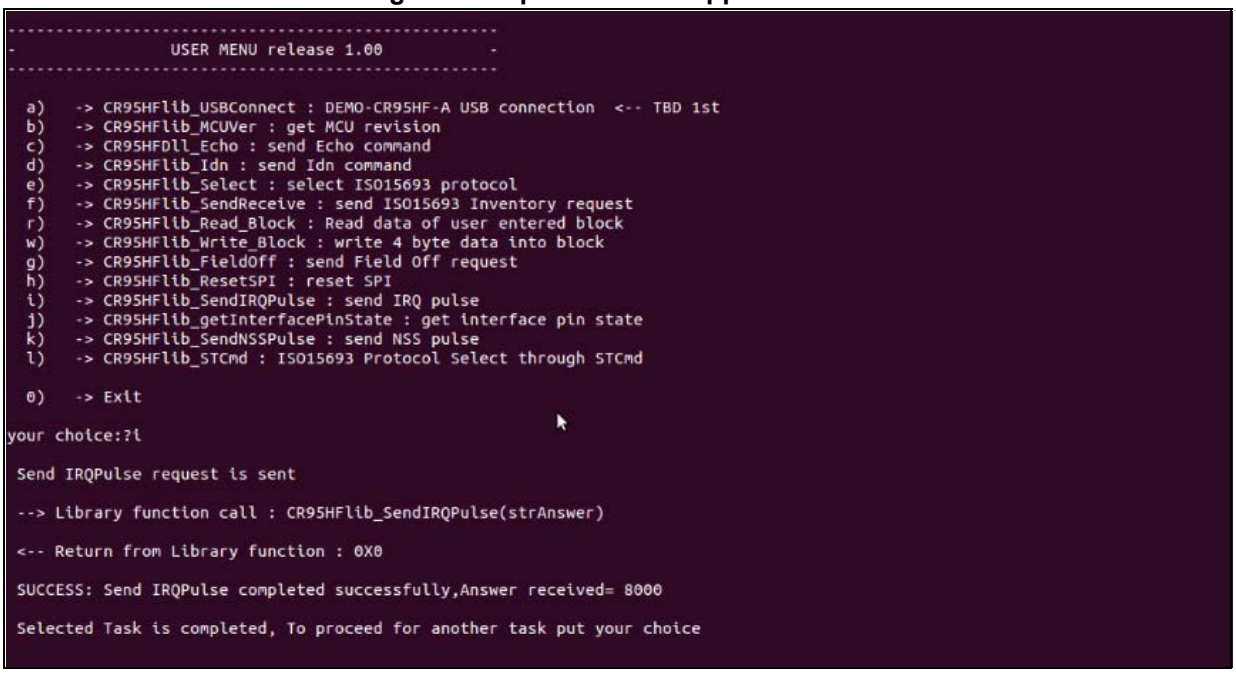


Figure 16. Option “j” TestApp execution

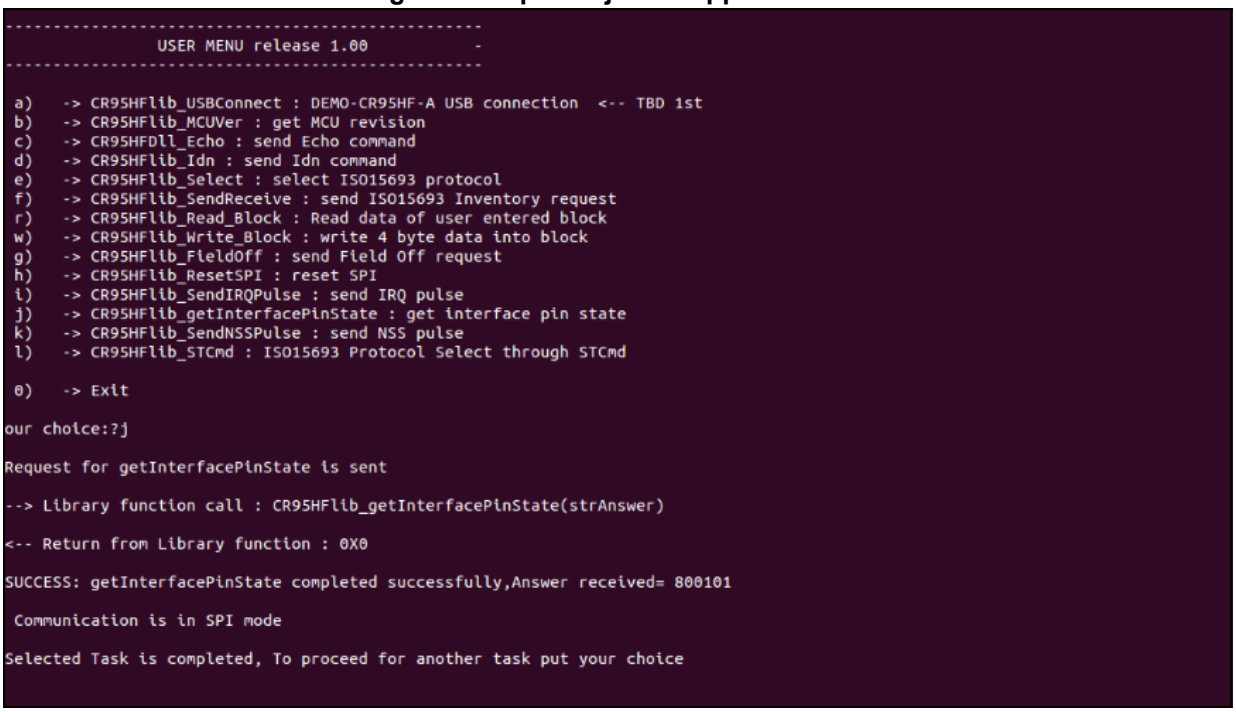


Figure 17. Option “k” TestApp execution

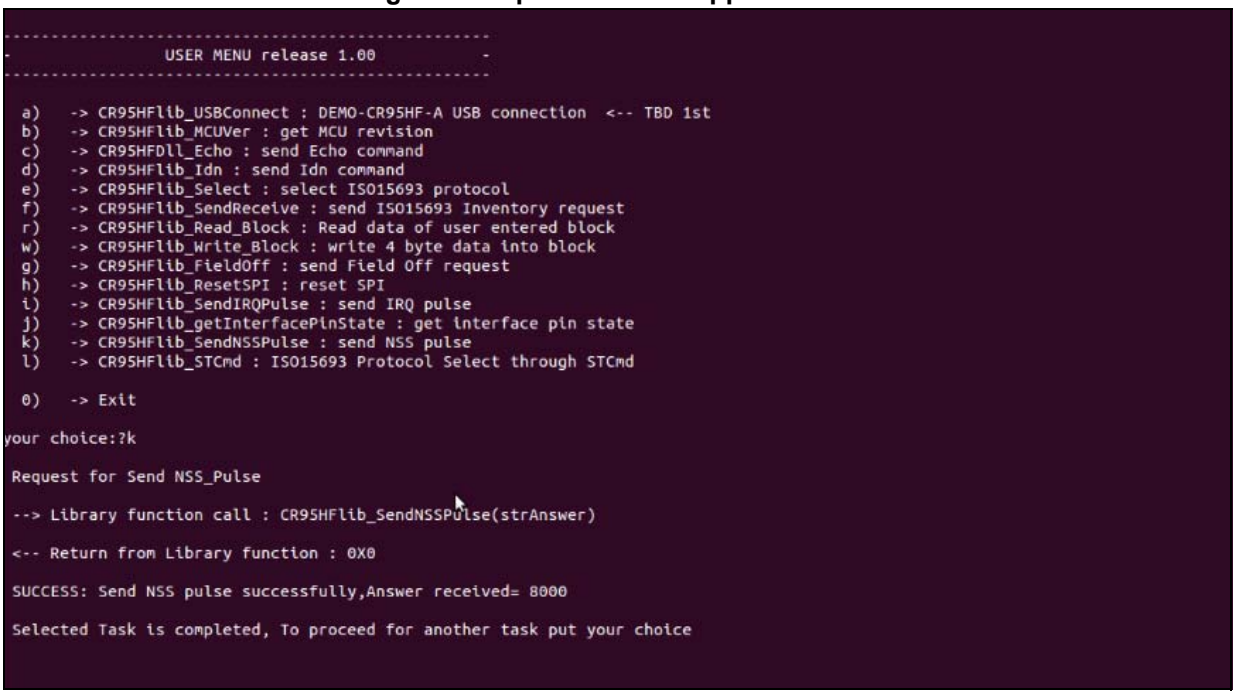
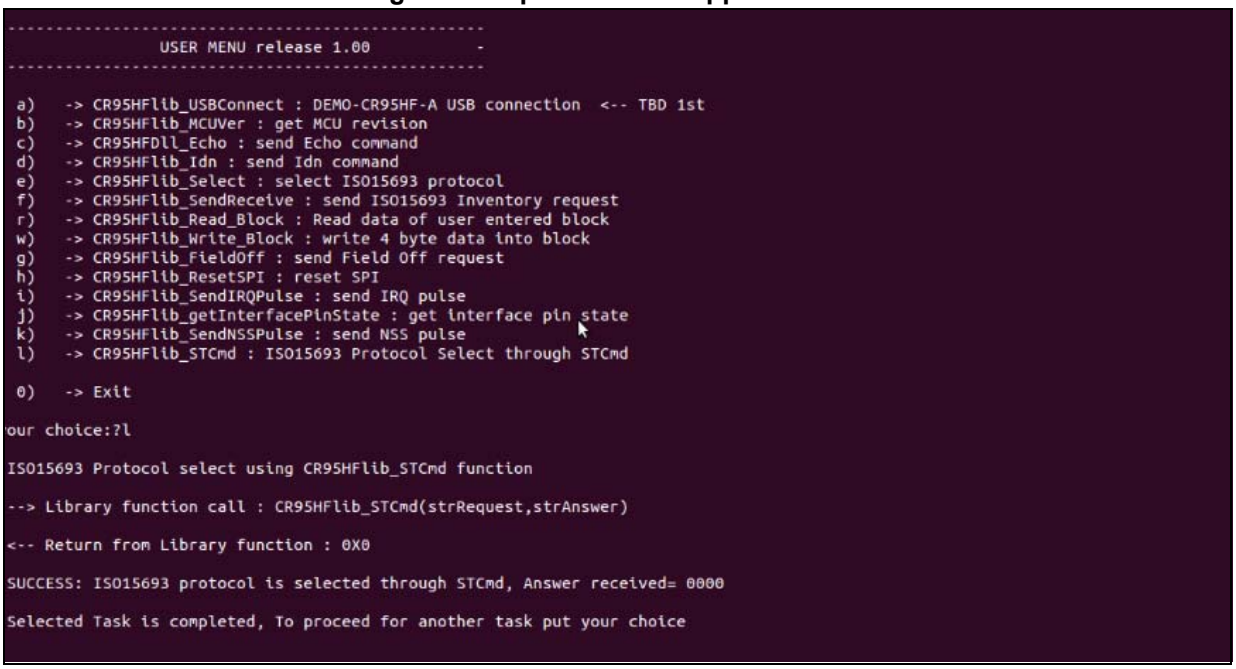


Figure 18. Option “l” TestApp execution



3 Revision history

Table 3. Document revision history

Date	Revision	Changes
14-Apr-2015	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2015 STMicroelectronics – All rights reserved