Reading RFID Card by SPI Interface in M480

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses M480 SPI to communicate with the MFRC522 module to read the Unique Identification (UID) of RFID card.
BSP Version	M480 BSP CMSIS V3.03.000
Hardware	NuMaker-PFM-M487 V3.0, MFRC522 module, RFID card

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Overview

This example code uses the M480 series microcontroller as the host controller to communicate with the MFRC522 module to read the UID of RFID card. The MFRC522 is one of NXP MIFARE series chips. It supports all variants of the MIFARE identification protocols. The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56 MHz, and its internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 Type A cards and transponders without additional active circuitry.

From the definition in ISO/IEC 14443, the MFRC522 and RFID card work as the Proximity Coupling Device (PCD) and Proximity Card (PICC), respectively. The communication protocol between PCD and PICC must be compliant with ISO/IEC 14443.

1.1 Principle

The MFRC522 provides three kinds of host interfaces, including SPI, UART and I^2C interfaces. The M480 uses the SPI interface to communicate with MFRC522 in this sample code. While communicating with the M480, the MFRC522 acts as a slave, receiving data from the M480 for register settings, sending and receiving data relevant for RF interface communication. The SPI can handle data speeded up to 10 Mbit/s. The implemented interface is in accordance with the SPI standard as depicted in Figure 1-1.

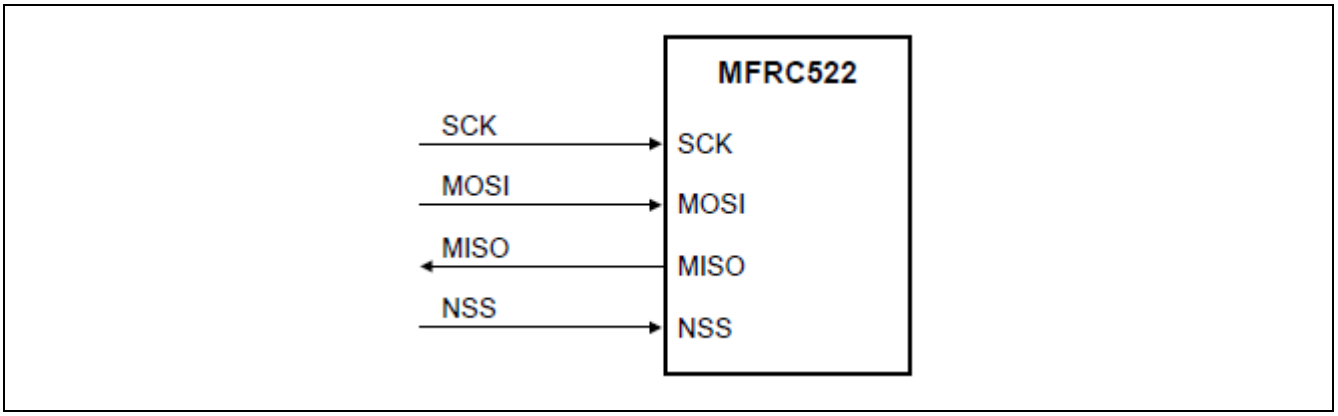

Figure 1-1 MFRC SPI Interface

The MFRC522 acts as the slave and the M480 is the master during SPI communication. The SPI clock signal SCK must be generated by the master. The data communication from the master to the slave uses the MOSI line. The MISO line is used to send data from the MFRC522 to the master. Each data byte on both MOSI and MISO lines is sent with the MSB first. The sent data on both MOSI and MISO lines must be stable on the rising edge of the clock and can be changed on the falling edge.

1.1.1 Reading Data by SPI

Reading data using the SPI requires the byte order shown in Table 1-1. It is possible to read out up to n-data bytes. The first byte sent byte (i.e., address 0) defines both the mode and the address.

x = Don't care

Line	Byte 0	Byte 1	Byte 0	…	Byte n	Byte n+1
MOSI	address 0	address 1	address 2	…	address n	address n+1
MISO	x	data 0	data 1	…	data n-1	data n

Table 1-1 MOSI and MISO Byte Order when SPI Reads Data

1.1.2 Writing Data by SPI

To write data to the MFRC522 using the SPI requires the byte order shown in Table 1-2. It is possible to write up to n data bytes by only sending one address byte. The first sent byte (i.e., address 0) defines both the mode and the address byte.

x = Don't care

Line	Byte 0	Byte 1	Byte 0	…	Byte n	Byte n+1
MOSI	address 0	data 0	data 1	…	data n-1	data n
MISO	x	x	x	…	x	x

Table 1-2 MOSI and MISO Byte Order when SPI Writes Data

1.1.3 SPI Address Byte

The address byte must meet the following format. The MSB of the first byte defines the mode used. To read data from the MFRC522, the MSB is set to logic 1. To write data to the MFRC522, the MSB must be set to logic 0. Bits 6 to 1 define the address of the MRFC522 register and the LSB is given to logic 0.

7 (MSB)	6	5	4	3	2	1	0 (LSB)
1 = read 0 = write	address						0

Table 1-3 Address Byte Format

1.1.4 MFRC522 Register Overview

Main registers in the MFRC522 are listed below.

Address (hex)	Register Name	Function
Page 0: Command and status register		
0x01	CommandReg	Starts and stops command execution
0x02	ComIEnReg	Interrupt request enable and disable control
0x03	DivlEnReg	Another interrupt request enable and disable control
0x04	ComIrqReg	Interrupt request flag bits
0x05	DivIrqReg	Another interrupt request flag bits
0x06	ErrorReg	Error bits showing the error status of the last command executed
0x07	Status1Reg	Communication status register 1
0X08	Status2Reg	Communication status register 2
0x09	FIFODataReg	Input and output of 64 byte FIFO buffer
0x0A	FIFOLevelReg	Number of bytes stored in the FIFO buffer
0x0C	ControlReg	Miscellaneous control registers
0x0D	BitFramingReg	Adjustments for bit-oriented frames
0x11	ModeReg	Defines general modes for transmitting and receiving
0x21	CRCResultReg1	Shows the MSB value of the CRC calculation
0x22	CRCResultReg2	Shows the LSB value of the CRC calculation
0x2A ~ 0x2D	TCtrlReg	Internal timer related control register
0x2E ~ 0x2F	TCounterValReg	Internal timer 16-bit counter value

Table 1-4 MFRC522 Main Registers

The CommandReg register plays the main role when the M480 communicates with the MFRC522; its bit allocation is listed in Table 1-5.

Bit	7	6	5	4	3	2	1	0
Symbol	Reserved		RcvOff	PowerDown	Command[3:0]			
Access			R/W	R/W	R/W			

Table 1-5 CommandReg Bit Allocation

The above Command[3:0] content is the command code and given from the external host controller. The MFRC522 is controlled by the command settings. Some key command codes in the Command[3:0] are listed in Table 1-6.

Command	Command code	Action
Idle	0000	No action, cancels current command execution
CalcCRC	0011	Activates the CRC coprocessor
Transmit	0100	Transmits data from the FIFO buffer
Receive	1000	Activates the receiver circuits
Transceive	1100	Transmits data from FIFO buffer to antenna and automatically activates the receiver after transmission
MFAuthent	1110	Performs the MIFARE standard authentication as a reader
SoftReset	1111	Resets the MFRC522

Table 1-6 Main Command Code in CommandReg

Other than the Transceive command, when the M480 writes any of the above commands to the Command[3:0] register, the MFRC522 shall execute the command immediately. To execute the Transceive command, the host shall write the command code to the CommandReg register first, and then the StartSend bit (b7) of BitFramingReg must also be set. Otherwise, this command cannot be executed exactly. The bit allocation of BitFramingReg is listed in Table 1-7.

Bit	7	6	5	4	3	2	1	0
Symbol	StartSend	RxAlign[2:0]			Reserved	TxLastBits[2:0]		
Access	W	R/W			-	R/W		

Table 1-7 BitFramingReg Bit Allocation

Those commands listed in Table 1-6 are just from the external host to the MFRC522 but not related to the ISO/IEC 14443 command protocols between the PCD and the PICC. Usually, to place the command from the PCD to the PICC, its content is written to the FIFO buffer before the host to activate the Transceive command. The procedure is described as follows.

1. The host prepares the PCD command to the FIFO buffer by writing the FIFODataReg register. It may consist of several bytes.
2. Write the Transceive (0xC) code to bits[3:0] of the CommandReg.
3. Enable the StartSend bit (b7) of BitFramingReg, and then the PCD command in the FIFO buffer will be sent to the PICC by the antenna and the receiver is also activated to receive the response from the PICC after transmission automatically.

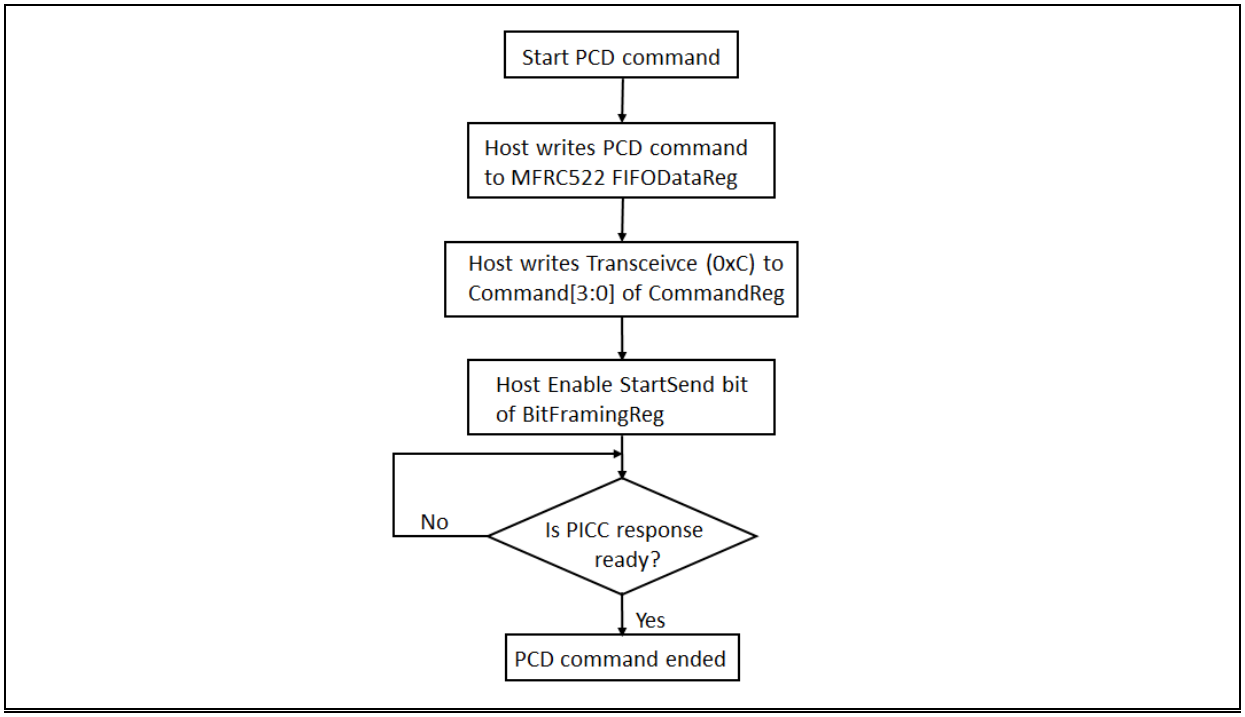

Figure 1-2 PCD Command to PICC

1.1.5 PICC States

After receiving some specific commands, the PICC shall be switched in the following states.

- POWER-OFF
- IDLE
- READY
- ACTIVE
- HALT

The detailed information of these states can be found in ISO/IEC 14443 standard. They are described briefly as follows.

- POWER-OFF state
  In the POWER-OFF state, the PICC is not energized due to lack of carrier energy.
- IDLE state
  After the field has been active, the PICC shall enter its IDLE state. In this state, the PICC is powered on, and is capable of demodulating and recognizing valid REQA and WAKE-UP Commands from the PCD.
- READY state
  This state is entered as soon as a valid REQA or WAKE-UP message has been received

and exited when the PICC is selected with its Unique Identification (UID).

- ACTIVE state

  This state is entered by selecting the PICC with its complete UID.

- HALT state

  In this state a PICC shall respond only to a WAKE-UP Command, which transits the PICC to its READY state.

1.1.6 ISO/IEC 14443 Type A Command

The ISO/IEC 14443 defines the command protocol between the PCD and the PICC. Two kinds of PICC are supported, including Type A and Type B. To detect PICCs that enter its energizing field, a PCD sends repeated Request commands and looks for an Answer To Request (ATQ) from PICCs. When a PICC is exposed to a non-modulated operating field (see ISO/IEC 14443-2), it shall be able to accept a request within five microseconds. The Request commands shall use the Request Command Type A (REQA) and Request Command Type B (REQB) to identify Type A PICC and Type B PICC, respectively. This process is designated “Polling”.

In this example program, the PCD uses the REQA to communicate with Type A PICC. All the commands used by the PCD to manage communication with several PICCs of Type A are

- REQA
- WAKE-UP
- ANTICOLLISION
- SELECT
- HALT

The detailed information of these commands can be found in ISO/IEC 14443 standard. They are explained below.

- REQA Command

  The REQA Command is sent by the PCD to probe the field for PICCs of Type A. If the PICC get the REQA command, it shall return Answer To Request of Type A (ATQA). The coding of ATQA is listed in Table 1-8.

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
Reserved								UID size bit frame		Reserved	Bit frame anticollision				

Table 1-8 Coding of ATQA

- WAKE-UP Command
  The WAKE-UP Command is sent by the PCD to put PICCs that have entered the HALT state back into the READY state. They shall then participate in further anticollision and selection procedures.

- ANTICOLLISION Command, SELECT Command
  These commands are used during an anticollision and select loop. The detailed information is described in the ISO/IEC 14443 standard.

- HALT Command
  The HALT Command lets the PICC enter the HALT state.

The state diagram for the PICC to receive the above commands is depicted in Figure 1-3.

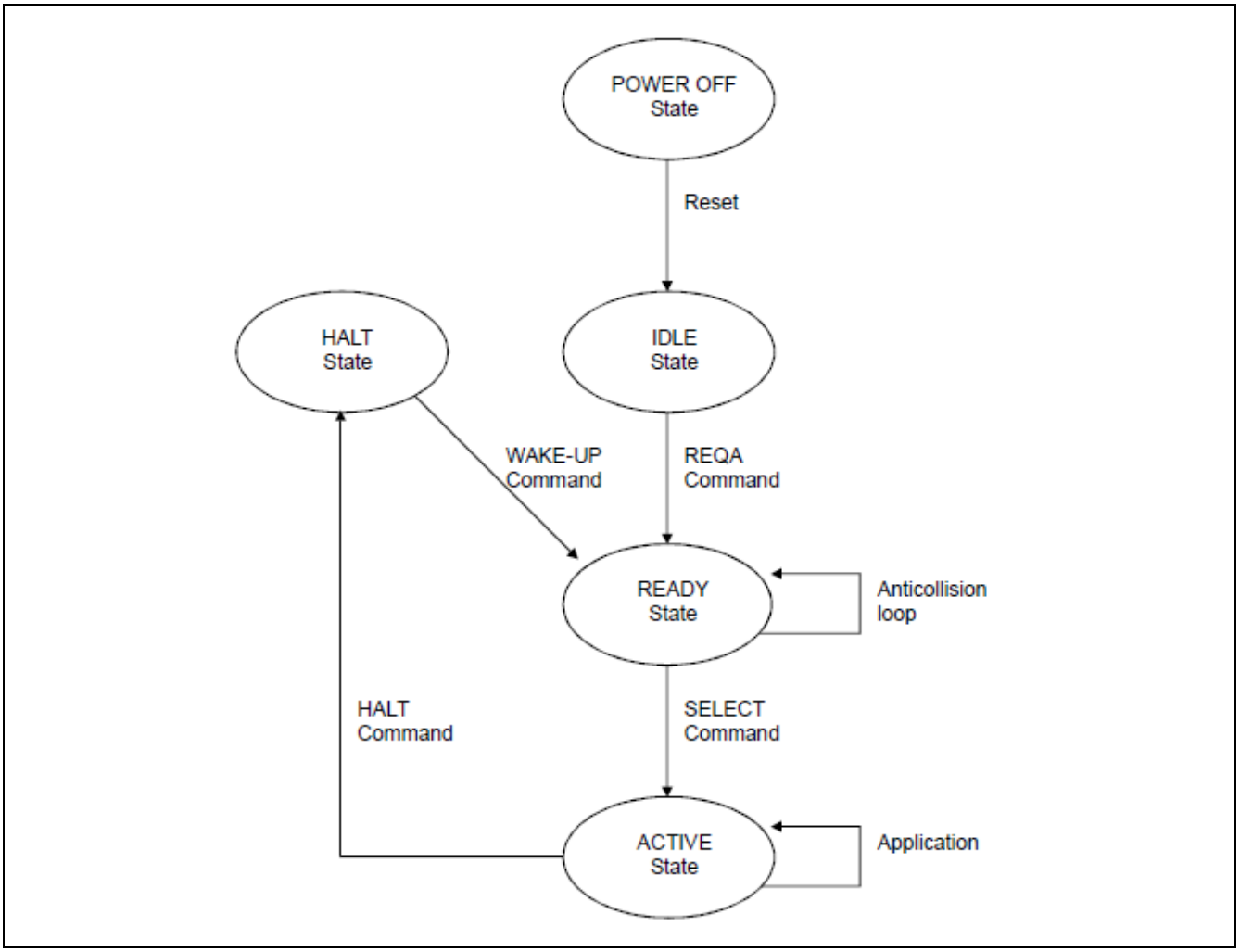

Figure 1-3 PICC of Type A State Diagram

In Figure 1-3, the anticollision loop is the algorithm used to prepare for dialogue between the PCD and one or more PICCs out of several in its energizing field. This sample code does not implement the algorithm for multiple PICCs and just for one PICC in the field.

The UID can be read from the PICC after ANTICOLLISION Command placed by the PCD. After getting the UID, the PCD sends the SELECT Command to confirm if the UID is complete. The UID is complete or not is indicated in the response of Select AcKnowledge (SAK).

If the UID is complete, the PICC shall transmit Select AcKnowledge (SAK) and transit from READY state to ACTIVE state.

If the UID is not complete, the PICC shall remain in READY State and the PCD shall initiate a new anticollision loop with increased cascade level.

B7	B6	B5	B4	B3	B2	B1	B0	Meaning
x	x	x	x	x	1	x	x	Cascade bit set: UID not complete
X	x	1	x	x	0	x	x	UID complete, PICC compliant with ISO/IEC 14443-4
x	x	0	x	x	0	x	x	UID complete, PICC not compliant with ISO/IEC 14443-4

Table 1-9 Coding of SAK

1.2 Operation Process

First, the related hardware connection must be ready as Figure 3-1. The MFRC522 module can be powered on by NuMaker-PFM-M487 Board. After around one second, the MFRC522 can be ready to receive the UID from the RFID card. When this sample code begins to execute on the board, the console can display the message as Figure 1-4.

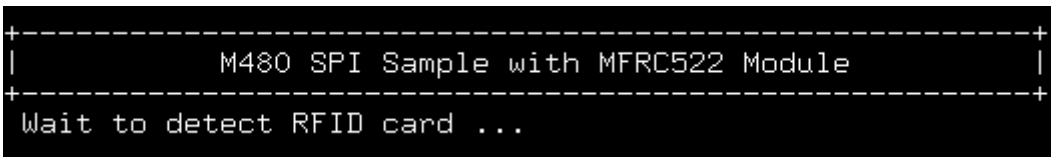

Figure 1-4 Console Message

The MFRC522 keeps polling the RFID card by placing the REQA command every 5 microseconds. If any RFID card is approximated and gets the REQA command, the ATQA response is returned from the RFID card. When the MFRC522 receives the ATQA, it begins the anticollision loop as depicted in Figure 1-3.

1.3 Demo Result

When an identifiable RFID card receives the REQA and ANTICOLLISION Commands, the UID can be read and displayed in the console. In the UID, the first four bytes are the UIDn (n: 0 ~

3), and the last one is the checkbyte. The checkbyte is calculated as exclusive-or over the four previous bytes.

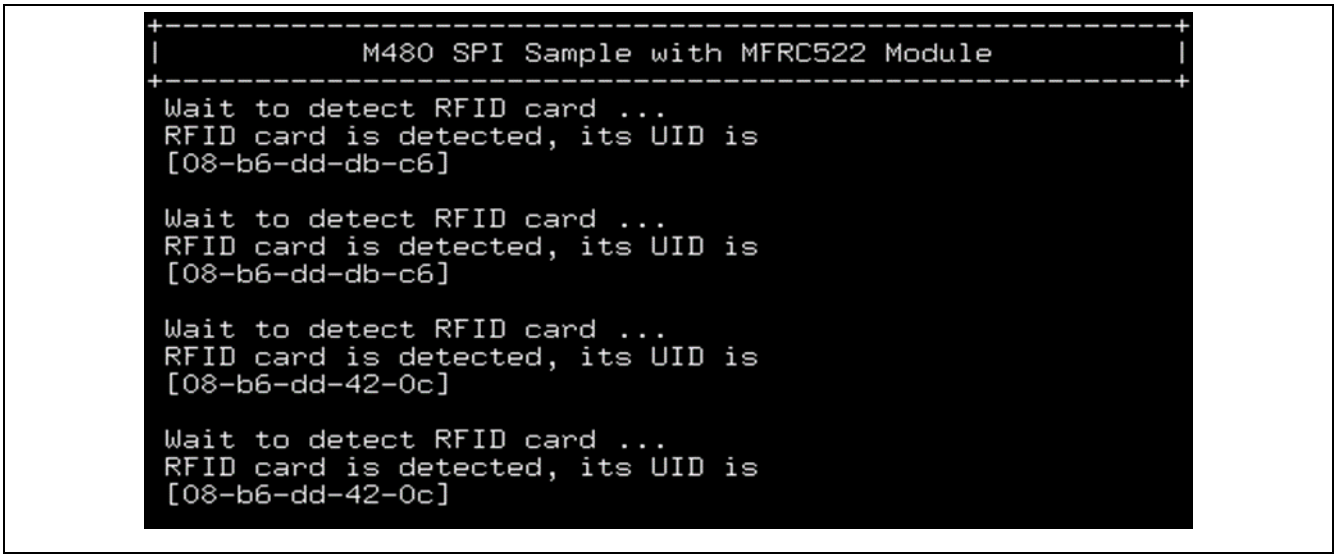

Figure 1-5 UID in Console

2 Code Description

The main program flow is that the MFRC522 keeps monitoring the approximated RFID card by placing the REQA command every 5 microseconds.

```
while(1)
{
    if (TM_MFRC522_Check(sn) == MI_OK)
    {
        printf(" RFID card is detected, its UID is \n");
        printf(" [%02x-%02x-%02x-%02x-%02x]\n\r\n", sn[0], sn[1], sn[2], sn[3], sn[4]);
        printf(" Wait to detect RFID card ...\n");
    }
}

TM_MFRC522_STS_T TM_MFRC522_Check(uint8_t* id)
{  ...

    /* Find cards, return card type */
    status = TM_MFRC522_Request(PICC_REQA, cardtype);
    ...
}
```

If any RFID card gets the REQA Command, the card shall return ATQA response.

```
TM_MFRC522_STS_T TM_MFRC522_Request(uint8_t reqMode, uint8_t* TagType)
{  ...

    status = TM_MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);
    /* ATQA (Answer to Request) from PICC will be 16-bit (0x10) length */
    if ((status != MI_OK) || (backBits != 0x10))
          status = MI_ERR;
    return status;
}
```

To get the UID, the MFRC522 shall place the ANTICOLLISION Command. After getting the UID from the RFID card, the MFRC522 sends the SELECT Command to confirm if the UID is complete.

```
TM_MFRC522_STS_T TM_MFRC522_Check(uint8_t* id)
{  ...

    /* Find cards, return card type */
    status = TM_MFRC522_Request(PICC_REQA, cardtype);

   /* Card detected */
   if (status == MI_OK)
   {
       /* Do anti-collision, return card serial number 4 bytes */
       status = TM_MFRC522_Anticoll(id);

       /* Check if anti-collision OK */
       if (status == MI_OK)
       {
           /* Select card */
           if(TM_MFRC522_SelectCard(id) == MI_OK)
           {
               if (id[0] & 0x04)
               /* Cascade bit set: UID not complete*/
               /* Note: PICC shall remain in READY State, and PCD shall initiate
                        a new anticollision loop with increased cascade level */
                   printf("UID not complete. \n");
           }
           ...
       }
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS V3.03.000
- IDE version
  - Keil uVersion 5.27

3.2 Hardware Requirements

- NuMaker-PFM-M487 V3.0
- MFRC522 Module, RFID card
- Pins Connection

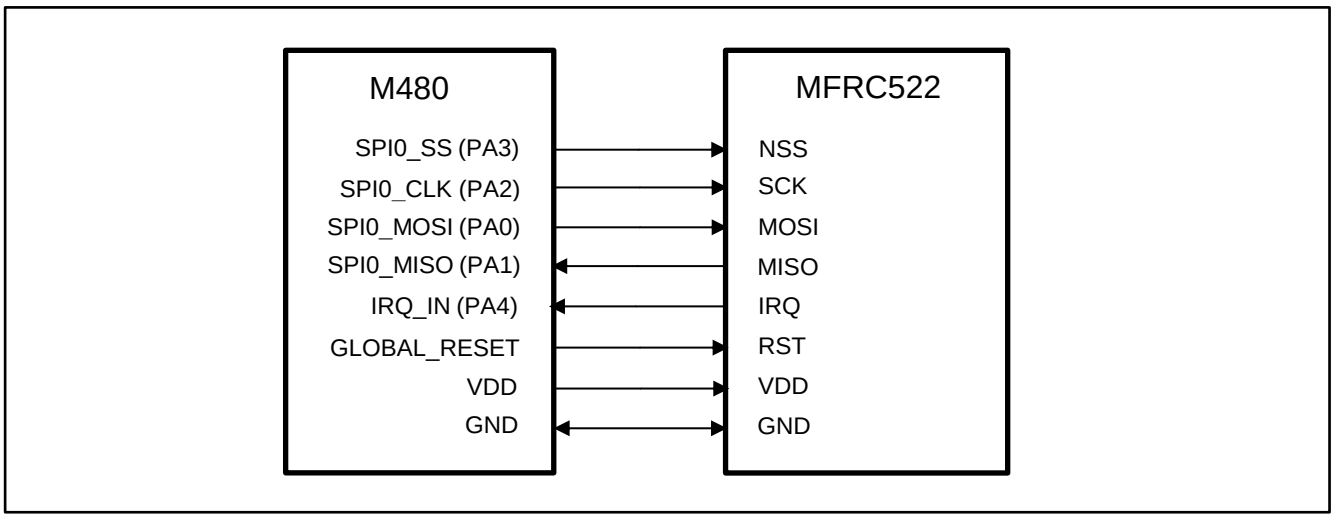

Figure 3-1 Hardware Environment

4 Directory Information

The directory structure is shown below.

EC_M480_SPI_RFID_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *SPI_RFID_MFRC522.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.11.03	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.