使用M031来驱动SI4463

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	此范例代码使用 M031 的 SPI 接口驱动 SI4463 收发数据
BSP 版本	M031 BSP CMSIS V3.03.000
开发平台	NuMaker-M031SD V1.1

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 概述

此范例为使用M031通过SPI接口来驱动SI4463，来实现数据无线（433M）收发。

1.1 原理

Silicon Laboratories的Si446x器件是高性能、低电流功耗收发器，覆盖119到1050MHz的sub-GHz频段。具有-126dBm的高灵敏度，同时实现极低的工作和待机电流消耗。Si446x通过标准的4线SPI:SCLK，SDI，SDO和nSEL与主机MCU通讯，最大通讯速率可达10Mbps。

主机MCU通过SDI引脚写数据，并可以通过SDO引脚从设备读取数据。图 1-1为一个SPI写命令时序。nSEL引脚为低来启动一个SPI传输。SDI数据的第一个字节将是固件命令之一，后面是n字节的参数数据，参数数据将根据特定的命令而变化。SCLK的上升沿应与SDI数据位的中心对齐。

SPI通讯写时序：

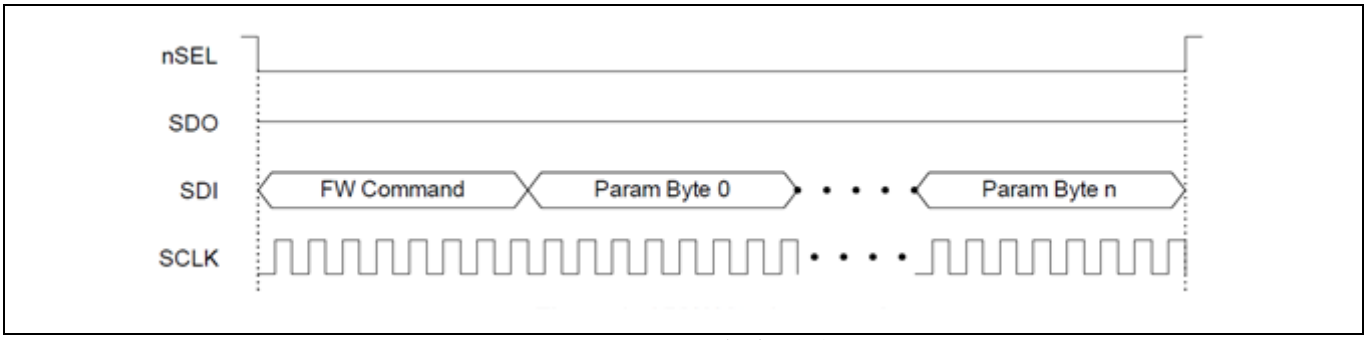

图 1-1 SPI 写命令时序

Si446x内含一个MCU，来控制RADIO的所有内部功能。对于SPI读命令，先检查流控命令CTS，以确保内部MCU已经执行了命令，并准备了要在SDO引脚上输出的数据。图1-2为SPI读命令的一般流程。一旦CTS值读取FFh，那么表示需要读取的数据已准备好可以被时钟输出到主机MCU。有效FFh值CTS读取的典型时间是20us。图 1-3演示了读取CTS为FFh后剩余的读取周期时序。内部MCU将在时钟下降沿从SDO输出数据，因此主机MCU应该在SCLK的上升沿上处理SDO数据。

SPI通讯读时序及流程：

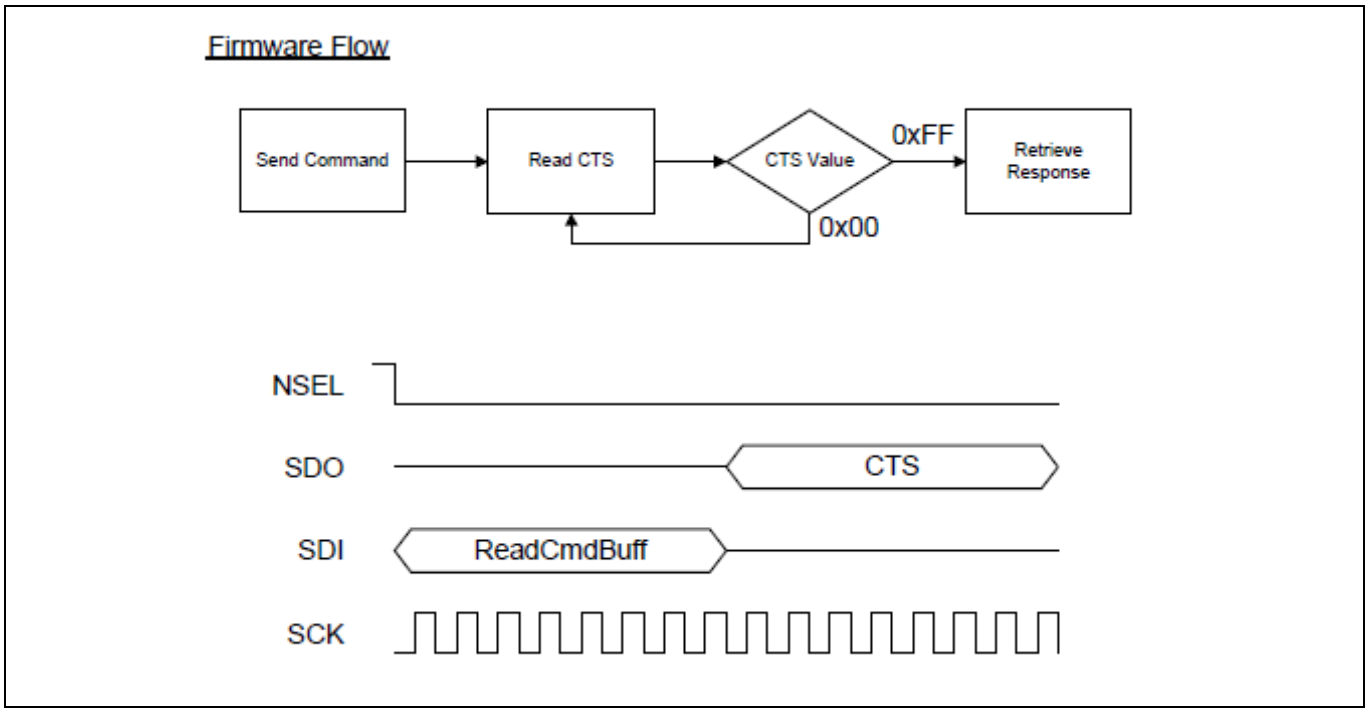

图 1-2 SPI 读命令-检查 CTS 值

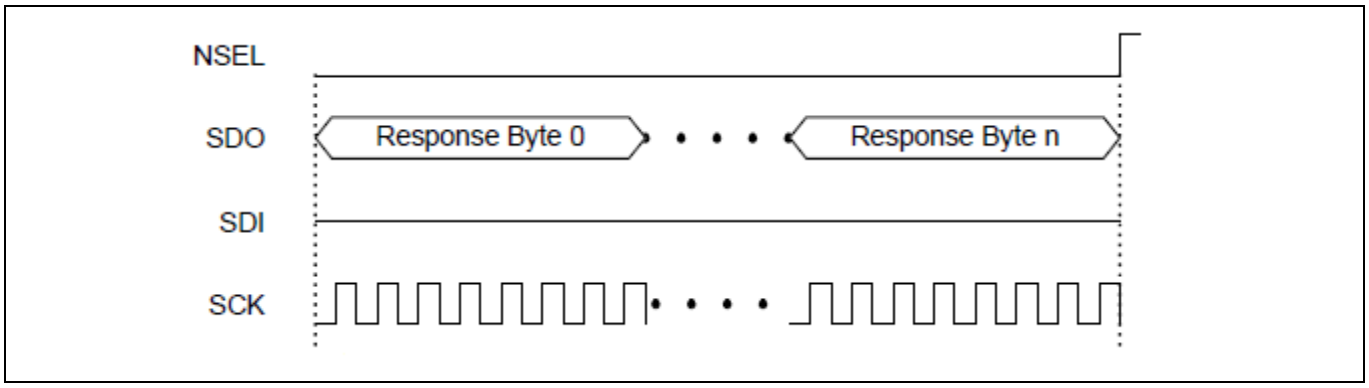

图 1-3 SPI 读命令-读数据

以上图 1-1，图1-2和图 1-3均来源于Si4463 数据手册。

关于Si4463命令，寄存器等详细资料可以参考以下文档：

https://www.silabs.com/documents/public/data-sheets/Si4463-61-60-C.pdf

1.2 执行结果

主机/从机通过串口连接电脑串口工具，打印信息分别如下：
作为主机，SI4463发送数据包并接收ACK：

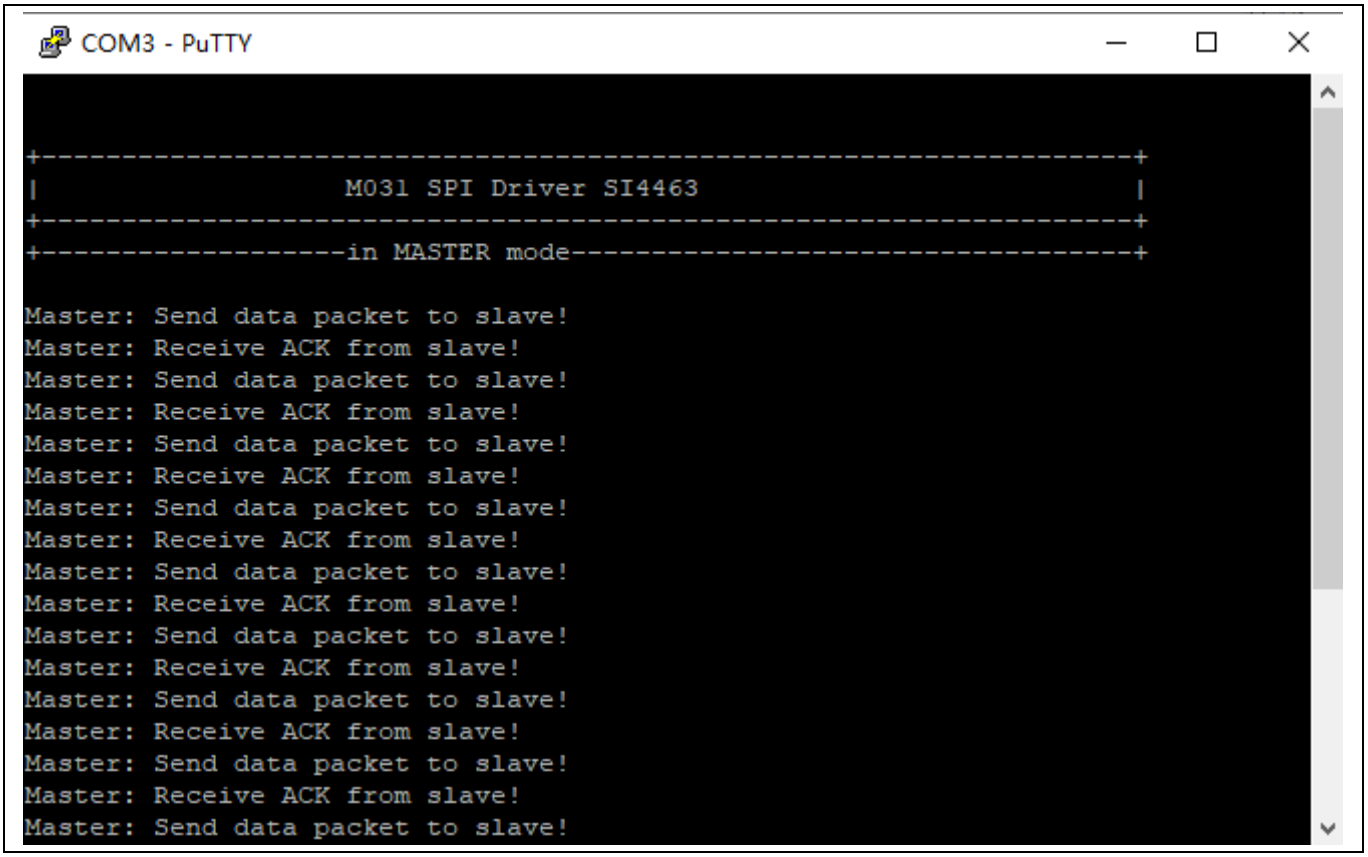

图 1-4 主机通讯结果

作为从机，SI4463接收数据包并发送ACK：

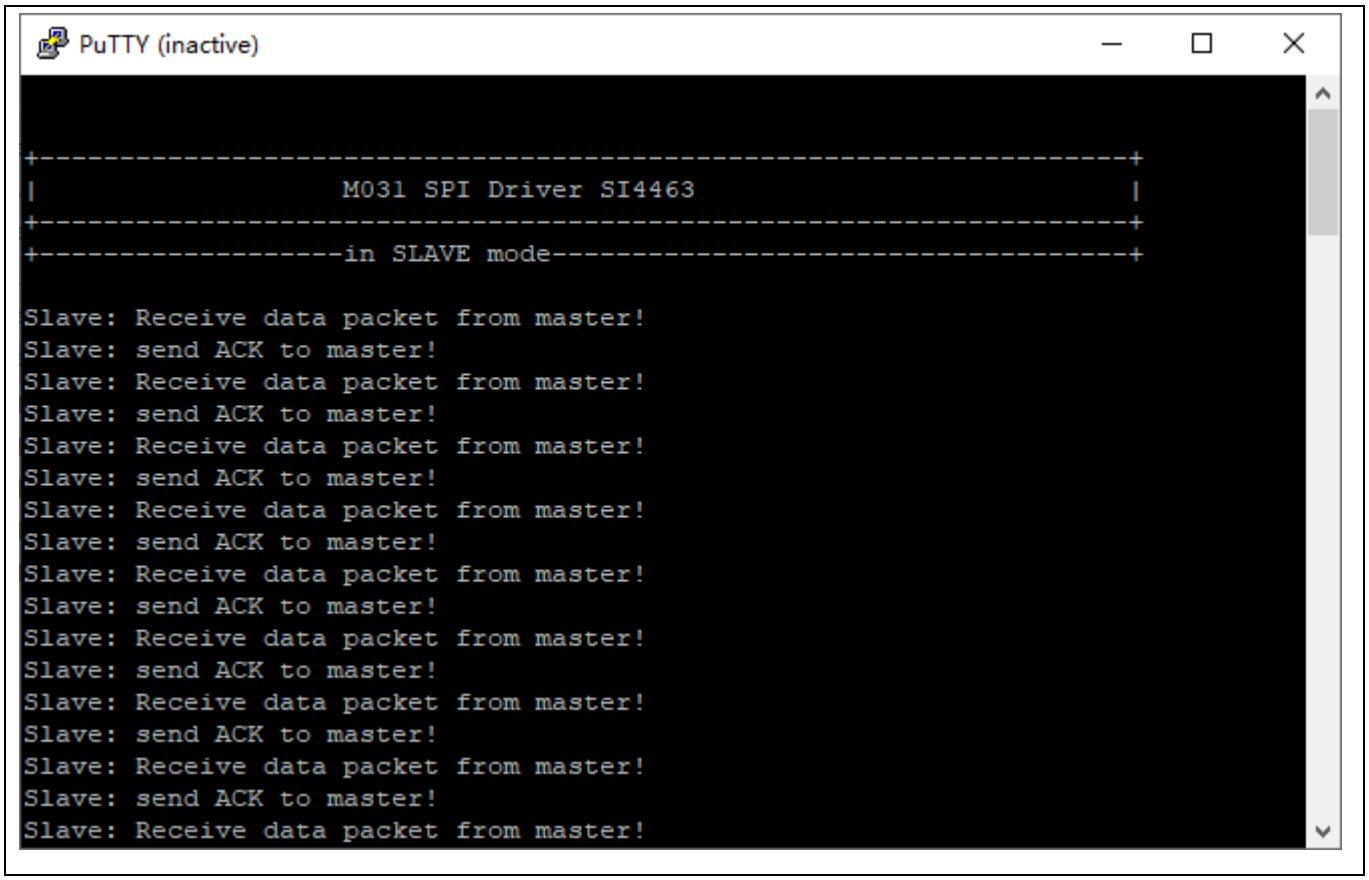

图 1-5 从机通讯结果

2 代码介绍

设置主机/从机：

```
#define EnableMaster   1       //in master mode
//#define EnableMaster   0        //in slave mode
```

初始化 SI4463

```
void vRadio_Init(void)
{
    uint16_t wDelay;

    /* Power Up the radio chip */
    vRadio_PowerUp();

    /*get  part information6 */
    si446x_part_info();

    /*part information check */
    if (Si446xCmd.PART_INFO.PART != 0x4463)
    {
        while (1)
        {
            /* ERROR!!!!  SPI communicate error  or part information error */
#ifdef RADIO_COMM_ERROR_CALLBACK
            RADIO_COMM_ERROR_CALLBACK();
#endif
        }
    }

    /* Load radio configuration */
    while (SI446X_SUCCESS != si446x_configuration_init(pRadioConfiguration-
>Radio_ConfigurationArray))
    {
        for (wDelay = 0x7FFF; wDelay--;) ;

        /* Power Up the radio chip */
        vRadio_PowerUp();
    }

    /* Read ITs, clear pending ones  */
    si446x_get_int_status(0, 0, 0);
}
```

SI4463 收发状态切换及处理：

```
uint8_t bRadio_Process()
{
    switch (RFstate)
    {
        case RFLR_STATE_RX_INIT:
            radio_TX(0);
            radio_RX(1);
            vRadio_StartRX(pRadioConfiguration->Radio_ChannelNumber, pRadioConfiguration-
>Radio_PacketLength);
```

```
            TickCounter = 0;
            RFstate = RFLR_STATE_RX_RUNNING;
            break;

        case RFLR_STATE_RX_RUNNING:
            //Delay_Us(300);
            //si446x_get_modem_status_fast_clear_read();//RSSI
            bMain_IT_Status = bRadio_Check_Tx_RX();

            if (bMain_IT_Status ==
SI446X_CMD_GET_INT_STATUS_REP_PH_PEND_PACKET_RX_PEND_BIT)
            {
                if (EnableMaster)
                {
                    RFstate = RFLR_STATE_RX_ACK_DONE;
                }
                else
                {
                    RFstate = RFLR_STATE_RX_DONE;
                }
            }

            if (TickCounter > 2000)
            {
                RFstate = RFLR_STATE_RX_TIMEOUT;
            }
            break;

        case RFLR_STATE_TX_INIT:
            radio_TX(1);
            radio_RX(0);

            gSampleCode_SendVariablePacket();
            TickCounter = 0;
            RFstate = RFLR_STATE_TX_RUNNING;
            break;

        case RFLR_STATE_TX_ACK_INIT:
            radio_TX(1);
            radio_RX(0);

            printf("Slave: send ACK to master! \n") ;
```

```
            vSampleCode_SendAcknowledge();
            TickCounter = 0;
            RFstate = RFLR_STATE_TX_RUNNING;
            break;

        case RFLR_STATE_TX_RUNNING:
            bMain_IT_Status = bRadio_Check_Tx_RX();

            if (bMain_IT_Status ==
SI446X_CMD_GET_INT_STATUS_REP_PH_PEND_PACKET_SENT_PEND_BIT)
            {
                if (EnableMaster)
                {
                    RFstate = RFLR_STATE_TX_DONE;
                }
                else
                {
                    RFstate = RFLR_STATE_TX_ACK_DONE;
                }
            }

            if (TickCounter > 2000)
            {
                RFstate = RFLR_STATE_TX_TIMEOUT;
            }
            break;

        case RFLR_STATE_RX_DONE:
            if (gSampleCode_StringCompare(customRadioPacket, tx_data, sizeof(tx_data)) ==
true)
            {
                RFstate = RFLR_STATE_TX_ACK_INIT;
                printf("Slave: Receive data packet from master! \n") ;
                HAL_Delay_nMS(200);
            }
            else
            {
                RFstate = RFLR_STATE_RX_INIT;
            }
            break;

        case RFLR_STATE_RX_ACK_DONE:
```

```
            if (gSampleCode_StringCompare(customRadioPacket, ACK_data, 3) == true)
            {
                printf("Master: Receive ACK from slave! \n") ;
            }

            HAL_Delay_nMS(100);
            RFstate = RFLR_STATE_TX_INIT;
            break;

        case RFLR_STATE_TX_DONE:
            printf("Master: Send data packet to slave! \n") ;
            RFstate = RFLR_STATE_RX_INIT;
            break;

        case RFLR_STATE_TX_ACK_DONE:
            HAL_Delay_nMS(20);
            RFstate = RFLR_STATE_RX_INIT;
            break;

        case RFLR_STATE_TX_TIMEOUT:
            if (EnableMaster)
            {
                RFstate = RFLR_STATE_TX_INIT;
            }
            else
            {
                RFstate = RFLR_STATE_RX_INIT;
            }
            break;

        case RFLR_STATE_RX_TIMEOUT:
            if (EnableMaster)
            {
                RFstate = RFLR_STATE_TX_INIT;
            }
            else
            {
                RFstate = RFLR_STATE_RX_INIT;
            }
            break;
```

```
        default:
            break;
    }

    return RFstate;
}
```

3 软件与硬件需求

- 软件需求
  - BSP 版本
    - M031_Series_BSP_CMSIS_V3.03.000
  - IDE 版本
    - Keil uVersion 5.26
- 硬件需求
  - 电路组件
    - NuMaker-M031SD V1.1
    - SI4463 module
  - 引脚连接

硬件电路使用M031连接SI4463模块，电路如下图 3-1：

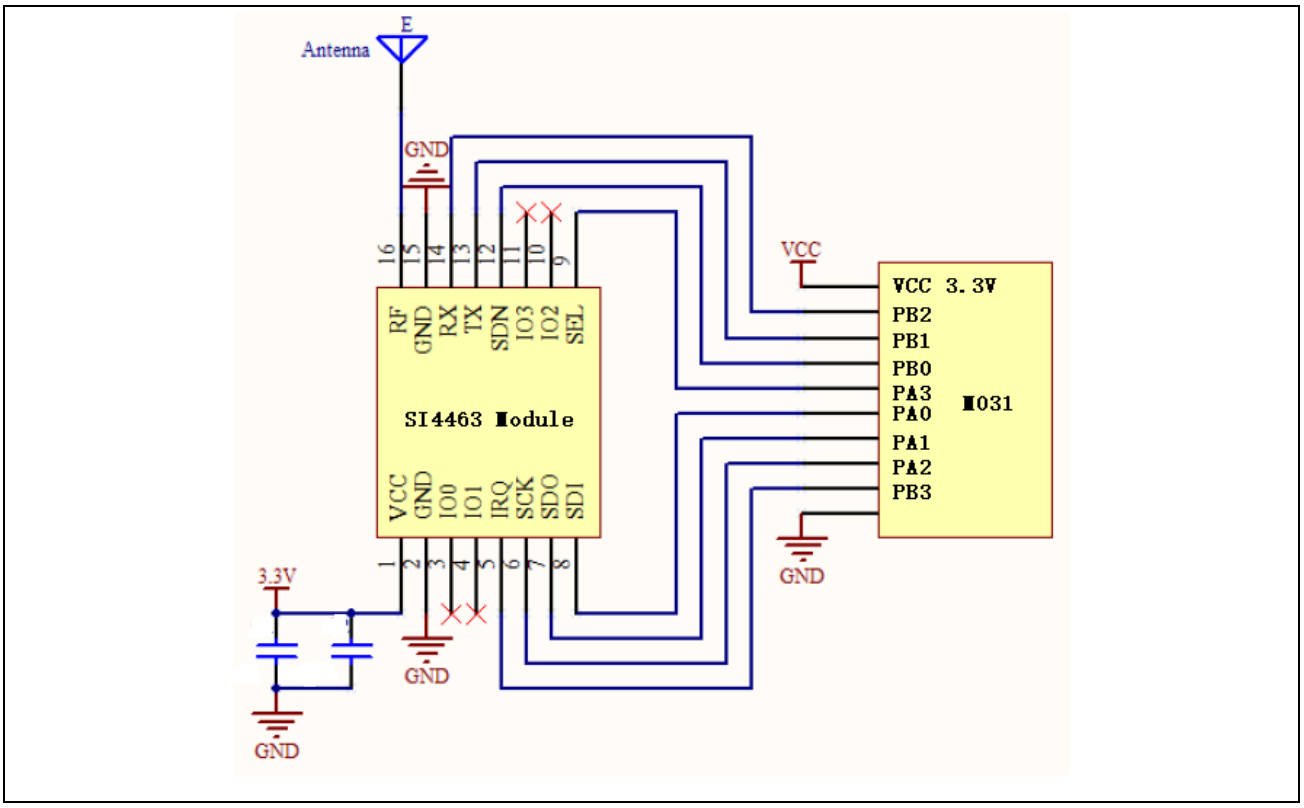

图 3-1 引脚连接

Si4463模块相关引脚功能描述如下：

引脚名	I/O	描述
IO0	I/O	通用数字 I/O
IO1	I/O	通用数字 I/O
IRQ	O	中断状态输出 当 Si446x 发生任何中断事件，IRQ 引脚将置低
SCK	I	SPI 时钟输入
SDO	O	SPI MISO 输出
SDI	I	SPI MOSI 输入
SEL	I	SPI 片选信号输入
IO2	I/O	通用数字 I/O
IO3	I/O	通用数字 I/O
SDN	I	关闭输入引脚 除 Shutdown 模式外的所有模式下，SDN 都应该是低 当 SDN=1 时，芯片将完全关闭，寄存器内容将会丢失
TX	I	发送输出控制端。TX 发送置高时 RX 必须为低
RX	I	接收输入控制端。RX 接收置高时 TX 必须为低
RF	I/O	RF 信号输入/输出端。 连接 50Ω 天线

表 3-1 引脚描述

4 目录信息

EC_M031_ SPI_SI4463_V1.00	
Library	Sample code header and source files
CMSIS	Cortex®Microcontroller Software Interface Standard (CMSIS) V4.5.0 definitions by ARM® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

图 4-1 目录信息

5 范例程序执行

1. 通过目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 M031_SPI_SI4463.uvproj
2. 进入编译模式界面：
   a. 编译
   b. 下载代码至内存
   c. 进入/离开仿真模式
3. 进入仿真模式界面：
   a. 执行代码

6 修订纪录

Date	Revision	Description
2021.06.25	1.00	1. 初始发布.

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.