使用ML51实现LoRa模块数据对传

NuMicro® 8 位系列微控制器范例代码介绍

文件信息

代码简述	本范例代码基于NuMicro ML51芯片中的SPI及GPIO功能，实现LoRa模块数据对传
BSP 版本	ML51_BSP_Keil_C51_V1.0.0
开发平台	NT-ML51PC V1.1

1 功能介绍

1.1 简介

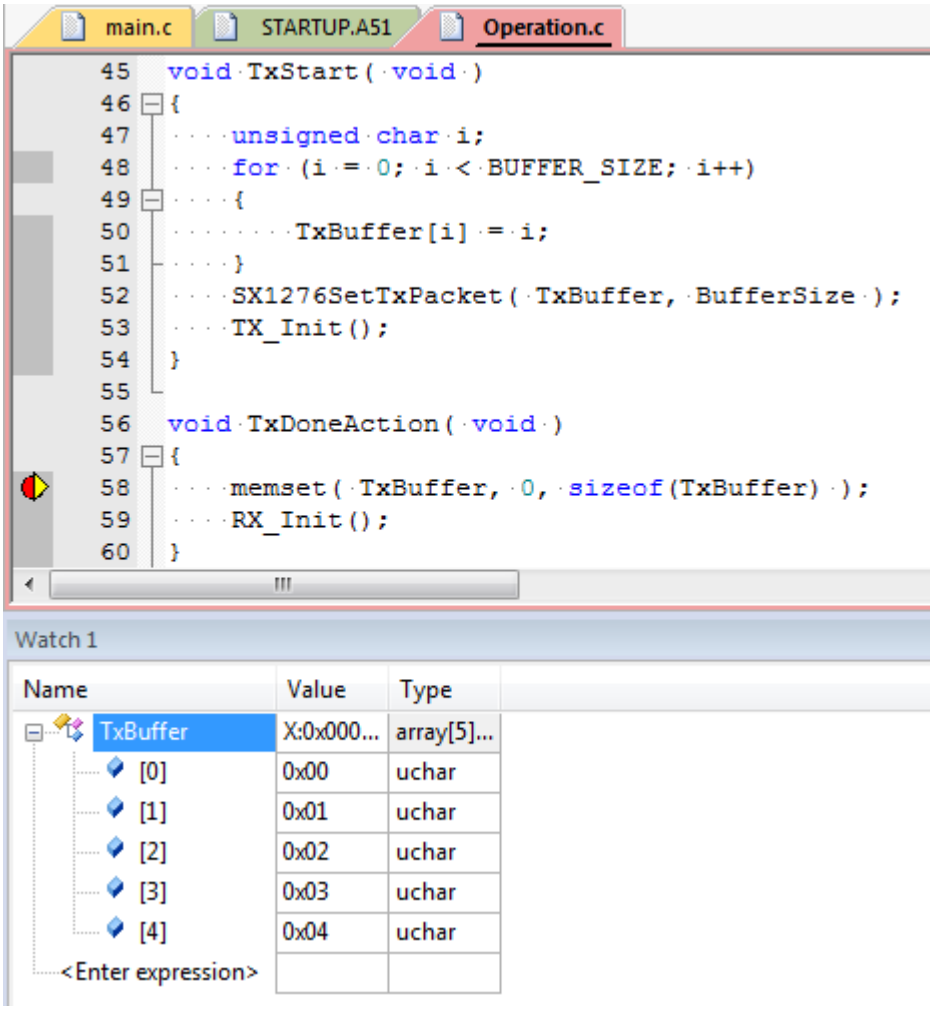
LoRa是LongRange的简称，应用于物联网的低功耗广域网传输技术(Low Power Wide Area Network, LPWAN)。LoRa无线通信技术由美商Semtech并购的法商Cycleo所开发，并与IBM合作制定规范，最后由Semtech、Cisco、IBM三大公司作为核心，组成LoRa联盟推动相关发展，为现今最受产业支持的LPWAN技术。

LoRa的模式如Wi-Fi般，任何人都可以设置基地台来建置网络环境。其具有较高的传输带宽，除了能进行单向传输的省电通讯外，也能够进行数据交换，适合应用于一些较大型的智能工厂中。除此之外，为了因应不同使用目的，LoRa有Class A, Class B, Class C 3个种类。Class A做基本的定时传输用，强调省电、Class B 除基本传输功能外，还增加触发性传输能力、Class C 则提供持续传输功能。

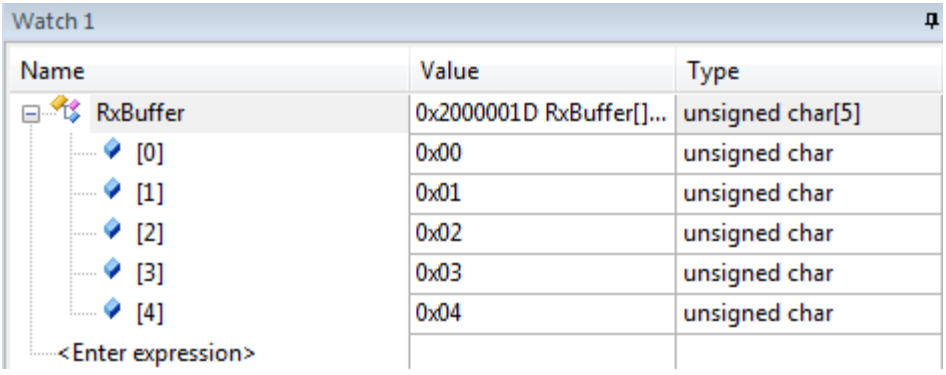
本范例代码基于NuMicro ML51芯片中的SPI及GPIO功能，实现LoRa模块数据对传。

1.2 执行结果

a. TX传送0x00, 0x01, 0x02, 0x03, 0x04，如TxBuffer数组中的数据



b. RX接收到0x00, 0x01, 0x02, 0x03, 0x04，如RxBuffer数组中的数据



2 代码介绍

设定#define TX 1，可以让ML51控制LoRa模块作为TX传送数据。
设定#define TX 0，可以让ML51控制LoRa模块作为RX传送数据。
范例中先呼叫BoardInit()与Radio_Init()进行GPIO与SPI等硬件的初始化，然后开始进行LoRa数据传输测试。

```
#include <stdio.h>
#include "platform.h"
#include "ML51.h"
#include "Operation.h"

/*---------------------------------------------------------------------------------------------------------*/
/*Define                                                                                                   */
/*---------------------------------------------------------------------------------------------------------*/
#define TX 1

/*---------------------------------------------------------------------------------------------------------*/
/*Global variables                                                                                         */
/*---------------------------------------------------------------------------------------------------------*/
extern unsigned char F_RxStart;
extern unsigned char F_TxStart;

/*---------------------------------------------------------------------------------------------------------*/
/* Functions.                                                                                              */
/*---------------------------------------------------------------------------------------------------------*/
int main(void)
{
   BoardInit();
   Radio_Init();

#if TX
    TxStart();
#else
    RX_Init();
#endif

    while (1)
    {
        if (F_RxStart)
        {
            F_RxStart = FALSE;
            RX_Done();
            RxDoneAction();
            RX_Init();
        }
        if (F_TxStart)
        {
            F_TxStart = FALSE;
            TX_Done();
            TxDoneAction();
            TxStart();
        }
    }
}
```

LoRa 接收通过GPIO中断唤醒，模块数据通过SPI进行，初始化步骤如下

```
void SX1276InitIo( void )
{
    /* SPI1 IO define */
    MFP_P32_SPI1_CLK;
```

```
    P32_PUSHPULL_MODE;
    MFP_P31_SPI1_MISO;
    P31_INPUT_MODE;
    P31_PULLUP_ENABLE;
    MFP_P30_SPI1_MOSI;
    P30_PUSHPULL_MODE;
    MFP_P33_GPIO;         /*setting SS use gpio mode */
    P33_PUSHPULL_MODE;

    /*Init DIO0 */
    MFP_P00_GPIO;
    P00_INPUT_MODE;
    P00_PULLDOWN_ENABLE;
    GPIO_EnableInt(PIT0, RISING, EDGE, Port0, 0); /*set DIO0 use pin interrupt */
```

```
void SpiInit( void )
{
    unsigned char u8MasterSlave = SPI_MASTER;
    unsigned char u8SPICLKDIV = 9;
    unsigned char u8SPIMode = SPI_MODE_0;
    unsigned char u8MSBLSB = MSB_FIRST;

    SX1276InitIo();
    set_SPI1SR_DISMODF;       /*Mode fault error detection disable*/
    clr_SPI1CR0_SSOE;         /*SS pin use as GPIO*/
    SFRS = 0;
    SPI1CR0 = 0;
    SPI1CR0 |= (u8MasterSlave << 4) | (u8SPICLKDIV & 0x03) | (u8SPIMode << 2) | (u8MSBLSB
<< 5);
    SPI1CR1 = 0 ;
    SPI1CR1 |= (u8SPICLKDIV & 0x0C) << 2;
    set_SPI1CR0_SPIEN;
```

LoRa模块发送及接收buffer的动作如下

```
void SX1276WriteBuffer( unsigned char addr, unsigned char *buffer, unsigned char size_in )
{
    unsigned char i;

    SPI_SS0_PIN = 0;
    SpiInOut( addr | 0x80 );
    for ( i = 0; i < size_in; i++ )
    {
        SpiInOut( buffer[i] );
    }
    SPI_SS0_PIN = 1;
}

void SX1276ReadBuffer( unsigned char addr, unsigned char *buffer, unsigned char size_in )
{
    unsigned char i;

    SPI_SS0_PIN = 0;
    SpiInOut( addr & 0x7F );

    for ( i = 0; i < size_in; i++ )
    {
        buffer[i] = SpiInOut( 0 );
    }

    SPI_SS0_PIN = 1;
}
```

3 软件与硬件环境

- 软件环境
  - BSP 版本
    - ML51_BSP_Keil_C51_V1.0.0
  - IDE 版本
    - Keil uVersion 4.6 及 PK51 Development Kit V9.52
- 硬件环境
  - 硬件需求
    - NT-ML51PC V1.1 tiny board
    - SX1276 LoRa module board
  - 接线图

NuTiny-SDK-ML51 V1.1		LoRa Module
VCC	↔	VCC
(GPIO INPUT)P00	↔	LORA_DIO0
(GPIO INPUT)P01	↔	LORA_DIO1
(GPIO INPUT)P02	↔	LORA_DIO2
(GPIO INPUT)P03	↔	LORA_DIO3
(GPIO INPUT)P14	↔	LORA_DIO4
(GPIO INPUT)P15	↔	LORA_DIO5
(GPIO OUTPUT)P33	↔	LORA_CS
(SPI CLK)P32	↔	LORA_CLK
(SPI MISO)P31	↔	LORA_MISO
(SPI MOSI)P30	↔	LORA_MOSI
GND	↔	GND

4 目录信息

- EC_ML51_LoRa_Control
  - Library: Sample code header and source files.
    - Device: ML51 device header files.
    - Startup: Keil51 startup files
    - StdDriver: All peripheral driver header and source files.
  - Sample_Code
    - Example: Source file of example code.
      - LoRa_Control LoRa module SX1276 TX/RX driver sample code

5 如何执行范例程序

1. 本项目支持 Keil uVersion 4.6 及以上版本
2. 根据目录信息章节进入 ExampleCode\LoRa_Control\KEIL 文件夹，双击 LoRa_Control.uvproj 开启专案
3. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
4. 进入除错模式接口
   a. 执行代码

6 修订纪录

日期	版本	描述
Oct. 24, 2019	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.