

LoRa Module Data Transmission Base On ML51

Example Code Introduction for 8-bit NuMicro® Family

Information

Application	This sample code is based on the SPI and GPIO functions in the NuMicro ML51 chip to realize the data transmission of the LoRa module.
BSP Version	ML51_BSP_Keil_C51_V1.0.0
Hardware	NT-ML51PC 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 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 Function Description

1.1 Introduction

LoRa (LongRange) is popular used in the Low Power Wide Area Network (LPWAN). LoRa wireless communication technology was developed by the legal company Cycleo acquired by Semtech, and cooperated with IBM to develop specifications. Finally, Semtech, Cisco and IBM were the core companies to form the LoRa Alliance to promote related development, which is the most industrially supported LPWAN. technology.

LoRa' transmit mode also like Wi-Fi, and anyone can set up a base station to build a network environment. It has a high transmission bandwidth, and can save data in addition to power-saving communication capable of one-way transmission, and is suitable for use in some large smart factories. In addition, in order to meet different purposes, LoRa has three types: Class A, Class B, and Class C. Class A is used for basic timing transmission, which emphasizes power saving. Class B adds trigger transmission capability in addition to basic transmission functions, and Class C provides continuous transmission.

This sample code is based on the SPI and GPIO functions of the NuMicro® ML51 series MCU to implement data transfer within the LoRa module.

1.2 Demo Result

a. TX transmits 0x00, 0x01, 0x02, 0x03, 0x04, following shows data array of TxBuffer

The screenshot shows a code editor with three files: main.c, STARTUP.A51, and Operation.c. The Operation.c file contains the following code:

```

45 void TxStart ( void )
46 {
47     unsigned char i;
48     for ( i = 0; i < BUFFER_SIZE; i++)
49     {
50         TxBuffer[i] = i;
51     }
52     SX1276SetTxPacket ( TxBuffer, BufferSize );
53     TX_Init ();
54 }
55
56 void TxDoneAction ( void )
57 {
58     memset ( TxBuffer, 0, sizeof(TxBuffer) );
59     RX_Init ();
60 }
    
```

Below the code editor is a Watch window titled "Watch 1" showing the contents of the TxBuffer array:

Name	Value	Type
TxBuffer	X:0x000...	array[5]...
[0]	0x00	uchar
[1]	0x01	uchar
[2]	0x02	uchar
[3]	0x03	uchar
[4]	0x04	uchar
<Enter expression>		

b. RX receive 0x00, 0x01, 0x02, 0x03, 0x04 , following shows data array of RxBuffer

The screenshot shows a Watch window titled "Watch 1" showing the contents of the RxBuffer array:

Name	Value	Type
RxBuffer	0x2000001D RxBuffer[]...	unsigned char[5]
[0]	0x00	unsigned char
[1]	0x01	unsigned char
[2]	0x02	unsigned char
[3]	0x03	unsigned char
[4]	0x04	unsigned char
<Enter expression>		

2 Code Description

Setting #define TX 1 allows ML51 to control the LoRa module to transmit data as TX.

Setting #define TX 0 allows ML51 to control the LoRa module to transmit data as RX.

In the example, BoardInit() and Radio_Init() are called to initialize the hardware such as GPIO and SPI, and then the LoRa data transmission test is started.

```
#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();
        }
    }
}
```

The LoRa receiver wakes up through the GPIO interrupt, and the module data is processed

through the SPI. The initialization steps are as follows:

```
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;
}
```

The LoRa module sends and receives buffers as follows:

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

    SPI_SS0_PIN = 0;
    SpiOut( addr | 0x80 );
    for ( i = 0; i < size_in; i++ )
    {
        SpiOut( 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;
```

```
SpilnOut( addr & 0x7F );  
  
for ( i = 0; i < size_in; i++ )  
{  
    buffer[i] = SpilnOut( 0 );  
}  
  
SPI_SS0_PIN = 1;  
}
```

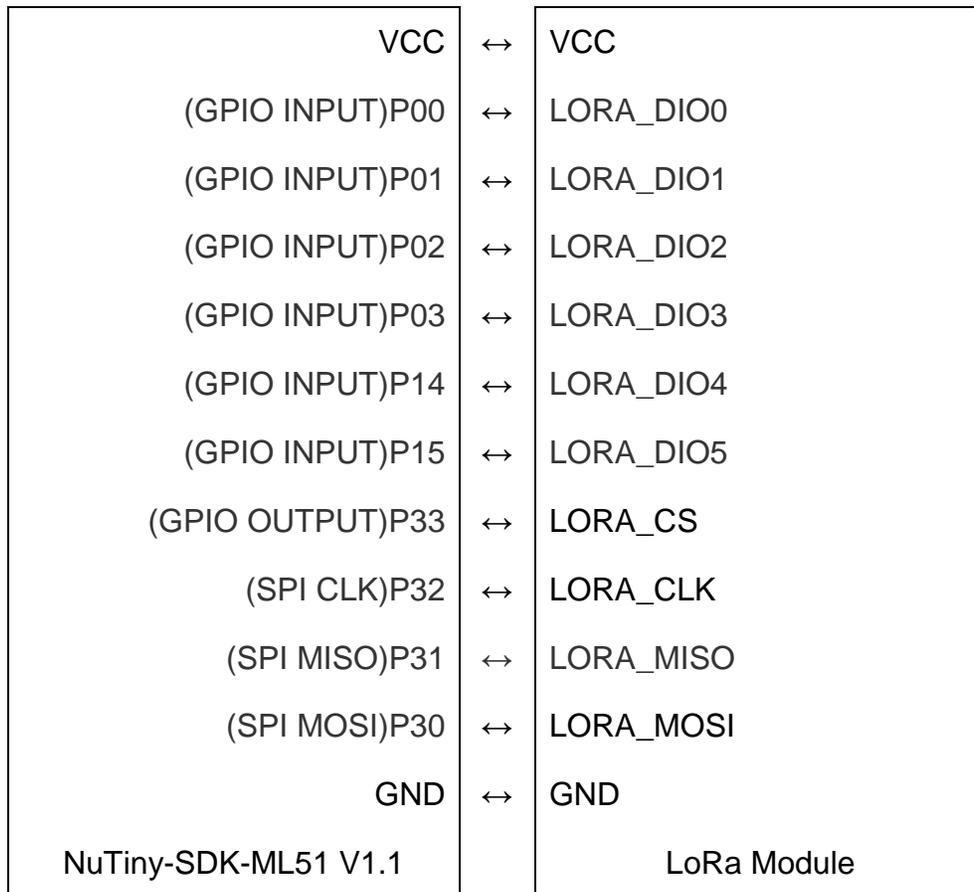
3 Hardware and Software Environment

- **Software Environment**

- BSP Version
 - ◆ ML51_BSP_Keil_C51_V1.0.0
- IDE Version
 - ◆ Keil uVersion 4.6 and PK51 Development Kit V9.52

- **Hardware Environment**

- Circuit components
 - ◆ NT-ML51PC V1.1 tiny board
 - ◆ SX1276 LoRa module board
- Diagram



4 Directory Information

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.

SampleCode

 **Example:** Source file of example code.

5 How to Execute a Sample Code

1. The project can be run in Keil uVision4 PK51 Development Kit V9.52 and above version.
2. Enter the ExampleCode\LoRa_Control\KEIL folder according to the Directory Information section and double-click LoRa_Control.uvproj to open the project.
3. Enter Keil compiler mode
 - a. Build
 - b. Download
 - c. Enter / Exit Debug Mode
4. When in under Debug mode
 - a. Free run

6 Revision History

Date	Revision	Description
Oct. 24, 2019	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.*