Index

1 Overview- 简介

本 API code 提供了一个用于 SX1212 简单的收发控制代码供用户参考。
This API provides transmitting and receiving demo cdoe using SX1212.

API code 包括两个文件：SX1212.c 和 SX1212.h。附加的 main.c 是用来调用上述 API 作测试的主程序文件。
API includes 2 files: SX1212.c and SX1212.h. The additional main.c is a test file to show how to call API.

代码全部用标准 C 语言编写，用户基本不需改动代码，只需更改一些宏定义和 RF 配置数据，就可用于任何的 MCU 平台。
Code is written with stand C. Basically user doesn’t need to change too much code. By define some marco/funtion and RF configurations, API can be used on any MCU platform.

本 API code 只是用简单的方法帮助用户快速调试 RF 通信，并不是最有效和最完善的机制。例如某些等候没有超时机制、接收数据包采用查询而非中断方式。用户可以在 RF 连通后逐步加入。
The API is using a simple way to help user establish a RF communication quickly. It may not be the best way. For example, no time out when waiting for a signal, uses polling to receive RF packet. User can optimize it later.

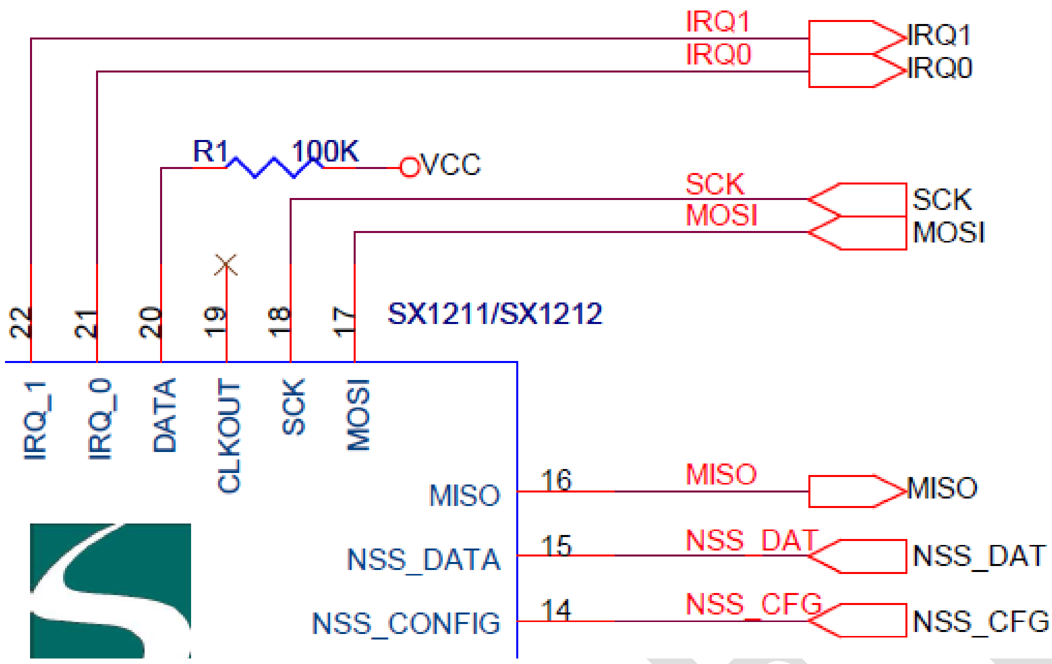
SX1212 支持 3 种工作模式：
SX1212 supports 3 operation modes

- continuous mode
- buffer mode
- packet mode

这里只提供了 packet 和 buffer 两种模式的示范代码。基本上，buffer mode 是可以替代 continuous mode 的。
API only has packet and buffer mode reference code. Basically, continuous mode can be replaced by buffer mode.

本 API code 在 Microcip PIC16F677 单片机下验证工作正常。
This API code is proved working on Microchip PIC16F677 MCU.

2 Hardware connection - 硬件连接



SX1212 通过 7 根信号线和 MCU 连接。
SX1212 has 7 signals to connect with MCU.

Name	Direction	Description
NSS_CFG	Out	SPI 配置寄存器选择 configure SPI select
NSS_DAT	Out	SPI 数据缓冲区选择 data SPI select
MISO	In	SPI Master In Slave Out
MOSI	Out	SPI Master Out Slave In
SCK	Out	SPI clock
IRQ0	In	中断信号 0 Interrupt 0
IRQ1	In	中断信号 1 Interrupt 1

Note 1：方向是对 MCU 而言
Direction stands on MCU side.

Note 2：IRQ 尽量连到 MCU 中具有中断唤醒功能的引脚上，尤其是 IRQ0，如果此信号不连到具有中断唤醒功能的引脚上的话，MCU 只能以查询方式接收信号
IRQ are recommended connection to MUC pins with IRQ wake up function. For example, if IRQ0 used as PayloadReady and connected to a non-irq pin, then MCU only can receive RF packet by polling.

Note 3：上面各引脚的初始化由用户在自己的初始化程序（例如 InitMCU）中完成（包括方向，初始值等）。本 API code 并不包括这些代码，用户需自己编写。

Above pins initiate code including direction, power on default value must be done by user(i.e. in InitMCU). This API code doesn’t have this kind of code.

3 API source code

用户只需用 3 个文件(main.c、SX1212.c、SX1212.h)建立一个基于自己 MCU 平台的工程，并做简单的修改，即可验证系统的基本功能。（带“*”的是必须修改的内容）
User only need to use 3 files(main.c, SX1212.c, SX1212.h) to build a project iwht some simply change based on his own MCU, then the basic function of SX1212 is ready to go.

通过在源文件中查找“// !!!”可以找到需要用户改动的内容，下面详细描述这些需要改动的地方。其它没有提及的地方均不需要修改。
Search “// !!!” in source file to locate the contents which user may need to change. Below is the detail about it. Others which are not mentioned in this document do not need to change.

建议用户开始时只更改那些必须修改的内容（有*的是必须修改的内容，只有 3 处），RF 通信成功后再修改其它参数。
Recommend user only change the contents which must be changed(contents with * must be changed. only 3 groups). The other changes can be done step by step after RF communication is established.

3.1 SX1212.h

3.1.1 Sync word - 同步字

4 bytes sync word

```
#define SYNC_WORD1            0x63
#define SYNC_WORD2            0x64
#define SYNC_WORD3            0x65
#define SYNC_WORD4            0x66
```

3.1.2 RF Tx power - 发射功率

```
#define RF_TX_POWER           RF_TX_POWER_PLUS10
```

3.1.3 Operation mode –工作方式

可选 packet mode 或 buffer mode。如果 RF packet 的长度不超过 64 bytes，建议选用 packet mode。
There are 2 options: packet mode or buffer mode. If RF packet length less than 64 bytes, recommend to use packet mode

```
#define DATA_MODE    PACKET_MODE
    #define PACKET_MODE        1
    #define BUFFER_MODE        2
```

3.1.4 RF packet len type - 数据包长度格式

```
#define LEN_TYPE                FIXED_LEN
    #define FIXED_LEN           3       // fixed length
    #define VARIABLE_LEN        4       // variable length
```

可以选择固定长度（FIXED_LEN）或可变长度（VARIABLE_LEN）
There are 2 options: fixed length(FIXED_LEN) and variable length(VARIABLE_LEN)

3.1.5 CRC - 是否使用 CRC

```
//#define CRC                   1
```

注：buffer mode 是不支持 CRC 的
Note: buffer mode doesn’t support CRC

3.1.6 For test - 测试用

```
#define TEST_ID         0xBB
#define FILLS           40
```

只是供测试示范用的，最后可以删除。FILLS 的大小在测试中决定了测试 packet 的长度。
This is for test. It can be deleted finally. FILLS is to change to length of test RFpacket.

3.1.7 RF packet definition - 数据包定义

```
#if LEN_TYPE == FIXED_LEN
    typedef struct {
        unsigned char fill[FILLS];
        unsigned char test_id;
        unsigned char counter;
        unsigned char key;
    }_RF_PACKET;
    #if DATA_MODE == PACKET_MODE
        #if FILLS > 61
            #error "packet too big"
        #endif
    #endif
#endif
```

```
#if LEN_TYPE == VARIABLE_LEN
    typedef struct {
        unsigned char len;
        unsigned char fill[FILLS];
        unsigned char test_id;
        unsigned char counter;
        unsigned char key;
    }_RF_PACKET;
    #if DATA_MODE == PACKET_MODE
        #if FILLS > 60
            #error "packet too big"
        #endif
    #endif
#endif
```

程序已经提供了一个数据包（RF packet）定义的示例，用户通常需要根据自己的实际需要重新定义。例如，下面定义了一个最简单的 64 bytes 数据包：
API has defined a demo RF packet structure. User usually needs to re-define it. For example, below define a simplest 64 bytes RF packet:

```
unsigned char my_test_pld[64];
```

3.1.8 IRQ0 and IRQ1 pin *

用户必须定义 SX1212 的 IRQ0 和 IRQ1 引脚和 MCU 的连接关系。
User MUST define the pin assignment between SX1212 IRQ0/IRQ1 and MCU.

```
#define RF_IRQ0()       RA2
#define RF_IRQ1()       RA4
```

例如，如果 SX1212 的 IRQ0 引脚接到 8051 的 P11，则：
For example, if IRQ0 is connected to 8051 P11, then:

```
#define RF_IRQ0()    P11
```

3.2 *SX1212.c*

3.2.1 SPI pins - SPI 引脚 *

用户必须定义 SX1212 的 SPI 总线引脚和 MCU 的连接关系。
User MUST define the pin assignment between SX1212 SPI bus and MCU.

```
#define SPI_MISO()          RA5
#define SPI_mosi(i)         RC5 = i
#define SPI_sck(i)          RC4 = i
#define SPI_nss_cfg(i)      RC7 = i
#define SPI_nss_dat(i)      RC6 = i
```

例如，如果 SX1212 的 MISO 引脚接到 8051 的 P12，则：
For example, if SX1212 MISO is connected to 8051 P12, then:

```
#define SPI_MISO()          P12
```

如果 SX1212 的 MOSI 引脚接到 8051 的 P13，则：
For example, if SX1212 MOSI is connected to 8051 P13, then:

```
#define SPI_mosi(i)      P13 = i
```

3.2.2 SPI NOP - SPI 空操作

_nop_spi()：spi 时钟线延时空操作，通常不需要。
SPI bus NOP, usually doesn't need.

```
#define _nop_spi()                // do nothing, use nop if needed
```

如果 MCU 速度很快，则需要加入若干 NOP 以使得时序符合要求。例如：
If MCU runs very fast, some NOP are needed.

```
#define _nop_spi()                 do {                 \
                                       asm("NOP");  \
                                       asm("NOP");  \
                                       asm("NOP");  \
                                       asm("NOP");  \
                                   }while(0)
```

3.2.3 SPIInit

初始化 MCU 对应的 SPI 引脚，通常不需要更改。
Initiate MCU pins of SPI. Usually it doesn't need to change.

```
#define SPIInit()                 do {                     \
                                   SPI_nss_cfg(1);          \
                                   SPI_nss_dat(1);          \
                                   SPI_mosi(1);             \
                                   SPI_sck(0);              \
                                }while(0)
```

例如，对于 8051 的单片机，也可用多加一行代码。
For 8051 MCU, it may define as bleow:

```
#define SPIInit()                 do {                     \
                                   SPI_nss_cfg(1);          \
                                   SPI_nss_dat(1);          \
                                   SPI_mosi(1);             \
                                   SPI_sck(0);              \
                                   SPI_MISO() = 1;          \
                                }while(0)
```

这样可以确保 MISO 引脚处于输入状态。
This may sure MISO is input mode.

3.2.4 Delay – 延时程序 *

```
#define DelayTS_OS()            Delay(55536)            // 5ms
#define DelayTS_FS()            Delay(63936)            // 800us
#define DelayTS_RE()            Delay(64536)            // 500us
#define DelayTS_TR()            Delay(64536)            // 500us
```

用户必须提供这些基于所用 MCU 的延时程序，分别延时 5ms、800us 和 500us。
User MUST provide a delay routine based on current MCU platform. They are to delay 5ms, 800us and 500us.

例如，假定用户有一个函数 void My8051Delay_us (unsigned int us)，参数就是延时的 us 值，则：
For example, if user has defined a function void My8051Delay_us (unsigned int us), the parameter 'us' is the delay time in unit of us(microsecond), then

```
#define DelayTS_OS()          My8051Delay_us(5000)           // 5ms
#define DelayTS_FS()          My8051Delay_us(800)            // 800us
#define DelayTS_RE()          My8051Delay_us(500)            // 500us
#define DelayTS_TR()          My8051Delay_us(500)            // 500us
```

3.2.5 RF configuration - RF 参数配置

```
#define DEF_MCPARAM1     (RF_MC1_SLEEP|RF_MC1_BAND_400_440|RF_MC1_SUB_BAND_1st)
static const _SX1212_REG RegistersCfg[] = { // !!! user can reconfigure regist
//   {addr,            val},
     {REG_MCPARAM1, DEF_MCPARAM1},

     #if DATA_MODE == BUFFER_MODE
     {REG_MCPARAM2, RF_MC2_BUFFER_MODE|RF_MC2_FSK},
     #endif

     #if DATA_MODE == PACKET_MODE
     {REG_MCPARAM2, RF_MC2_PACKET_MODE|RF_MC2_FSK},
     #endif

     {REG_FDEV, RF_FDEV_50},
```

band，Fdev，Bit Rate，Center Frequency etc…

3.2.6 BuildPacket

```
void BuildPacket (unsigned char test_val) {
        unsigned char i, *p;
        unsigned char j = 0x00;
        static unsigned char counter = 0;

    #if LEN_TYPE == FIXED_LEN
        p = (unsigned char *)(&RF_Pkt_fill);
```

程序提供了一个示例的 packet builder，用户最后要定义自己的 packet builder。示例的 packet builder 只是往包中填写一些有规律的数据，最后 3 个 bytes 则是 TEST_ID，counter 和 key（see also *3.1.7RF packet definition - 数据包定义*）
API provides a demo packet builder. User need to define his own packet builder. Demo packet builder is to fill some incremental data pattern in packet. Last 3 bytes are test_id, coutner and key for debug.

- TEST_ID：默认是 0xBB，Rx 可以检查收到的这个 byte 是否是该值
  TEST_ID: default is 0xBB, Rx can check if this byte received match
- counter：每次发送增加 1，在 RF sniffer 中（例如 SX1212 SKA）更容易辨认
  counter: increased by 1 for each transmit. Easy to debug in RF sniffer(such as SX1212 SKA)

- key：为了便于调试时识别，用户可以传送一个任意值作为测试用（test_val）
  for debug: user can send an arbitrary value to Rx(test_val)

3.3 *The simplest project - 建立简单的测试工程*

用 main.c、SX1212.c、SX1212.h 共 3 个文件即可建立一个最简单的测试工程以验证 SX1212 的 RF 通信。
User main.c, SX1212.c and SX1212.h total 3 files can build a simplest project to verify SX1212 RF communication.

3.3.1 main.c

用户需根据自己的 MCU 平台，完成 InitMCU 函数。
User need to implement InitMCU based on his own MCUplatform.

```
static void InitMCU (void) {
    // ....
    /*
    Initiate MCU here
    including clock, IO, .....
    */
}
```

如 3.2.4 所述，用户还必须提供一个基于所用 MCU 的延时程序。
As described at 3.2.4, use must provide a delay function.

程序在初始化 MCU 和 SX1212 完毕后，会对 SX121 SPI 进行一个读写测试：

After initiate MCU and SX1212, code will do a SX1212 SPI write/read test:

```
#if 1          // to test SPI
SpiWriteCfg(REG_SYNCBYTE4, 0x55);
if(SpiReadCfg(REG_SYNCBYTE4) != 0x55) {
    while(1);   // something wrong with SPI
}

SpiWriteCfg(REG_SYNCBYTE4, 0xAA);
if(SpiReadCfg(REG_SYNCBYTE4) != 0xAA) {
    while(1);   // something wrong with SPI
}

SpiWriteCfg(REG_SYNCBYTE4, SYNC_WORD4);
if(SpiReadCfg(REG_SYNCBYTE4) != SYNC_WORD4) {
    while(1);   // something wrong with SPI
}
#endif
```

如果没有进入 while(1)的死循环，则证明 SPI 读写没有问题。如果进入了死循环，则可能的问题是：
If code doesn’t fall into dead loop while(1), it means SPI write/ read is OK.

- 芯片没有焊接好
- SPI 引脚定义错误
- SPI 引脚方向初始化错误
- …

如果 SPI 无误，系统发出一个测试包：
If everything is OK, a test packet will be sent out:

```
BuildPacket(0x12);
SendRfFrame((unsigned char *)(&RF_Pkt), sizeof(RF_Pkt));
```

最后进入 Rx 模式，通过查询方式接收来自 Tx 的数据：
Finally, try to receive RF packet by polling:

```
while(1) {
    ReceiveRfFrame((unsigned char *)(&RF_Pkt), sizeof(RF_Pkt), &rc);
    if(rc == OK) {
        //got packet
        // ...
    }
    //...
}
```

用户可以根据需要在合适的地方，调用 BuildPacket、SendRfFrame、ReceiveRfFrame 即可发送和接收 RF 数据包。
User can call BuildPacket, SendRfFrame, Receive RfFrame to send and receive RF packet.

4 API functions for user - 用户可以调用的函数

API code 中提供了下列的函数（或宏）可供用户调用。
API provides below functions(marcos):

4.1 SpiReadCfg

read register

```
#define SpiReadCfg(addr)
```

addr：unsigned char。address of register to read
返回值：unsigned char。current value of the register

Note: User mustn’t call `_SpiConfig` directely to read register.

4.2 SpiWriteCfg

write register

```
#define SpiWriteCfg(addr, val)
```

addr：unsigned char。address of register to write to
val：unsigned char。value to write to
返回值：unsigned char 。old value of the register

Note: User mustn’t call `_SpiConfig` directely to write register.

4.3 SendRFFrame

```
void SendRFFrame (unsigned char *buffer, unsigned char len);
```

call this routine once will send a RFpacket out

```
input
buffer: pointer to send buffer
len: packet length

call example:
SendRFFrame((unsigned char *)(&RF_Pkt), sizeof(RF_Pkt));
```

buffer: buffer to send

len: length to send

Example, if the RF packet is define as *3.1.7RF packet definition - 数据包定义*

```
unsigned char my_test_pld[64];
```

then:

```
SendRFFrame(my_test_pld, sizeof(my_test_pld));
```

4.4 ReceiveRfFrame

程序中，可以不断地调用该函数来接收来自 Tx 的 RF packet，如果返回代码 rc = OK 的话，表示收到了 Tx 的 packet，并已存入 buffer 中。

User can call this routine continuously to receive RF packet. If return code rc = OK, that means receive a RF packet from Tx, and packet has been stored into buffer.

```
void ReceiveRfFrame (unsigned char *buffer, unsigned char len, unsigned char *rc);
input
buffer: pointer to receive buffer
len: packet length

output
rc: return code
    - RF_RX_RUNNING  Rx running
    - ERROR          error
    - OK             packet received

call example:
ReceiveRfFrame((unsigned char *)(&RF_Pkt), sizeof(RF_Pkt), &rc);
```

buffer: buffer to receive packet

len: buffer length

rc: return code

Example, if the RFpacket is defined as *3.1.7RF packet definition - 数据包定义*

```
unsigned char my_test_pld[64];
```

then:

```
ReceiveRfFrame(my_test_pld, sizeof(my_test_pld), &rc);
```

Note: for variable length(VARIABLE_LEN), buffer[0] will return the actual received packet length.

4.5 SetRfMode

change SX1212 的 RF mode

```
unsigned char SetRfMode (unsigned char mode);
```

```
#define RF_SLEEP
#define RF_STANDBY
#define RF_SYNTHESIZER
#define RF_RECEIVER
#define RF_TRANSMITTER
```

4.6 clearFIFO

To clear SX1212 FIFO

```
#define clearFIFO()      SpiWriteCfg(REG_IRQPARAM2, 0x1F)
```