NRF24l01-lib-3.0

欧阳俊源，于2020/02/20

更新日志：

1. 学习HAL库在寄存器映射方面的书写习惯，比1.0更新了数据结构，配置更方便。
2. 增加了中断服务函数，可以进行中断接收，比轮询接收来的高效，减小MCU时间占用。
3. 完成了Enhance Shockburst的 驱动代码。可以自动完成应答和应答带数据包。
4. 根据以上完成片上时分双工的通信。发射方发送信息后接收方发回带应答的数据包，并以中断通知。

目前功能：

1. 普通发送/接收
2. 中断接收
3. 带数据的应答信号，即发送方发送一包数据，接收方应答一包可编写的反馈数据(否则只能产生不带数据包应答)。发射方接收到应答信号后执行回调函数，接收方完成反馈数据发送后执行回调函数。

To do list：

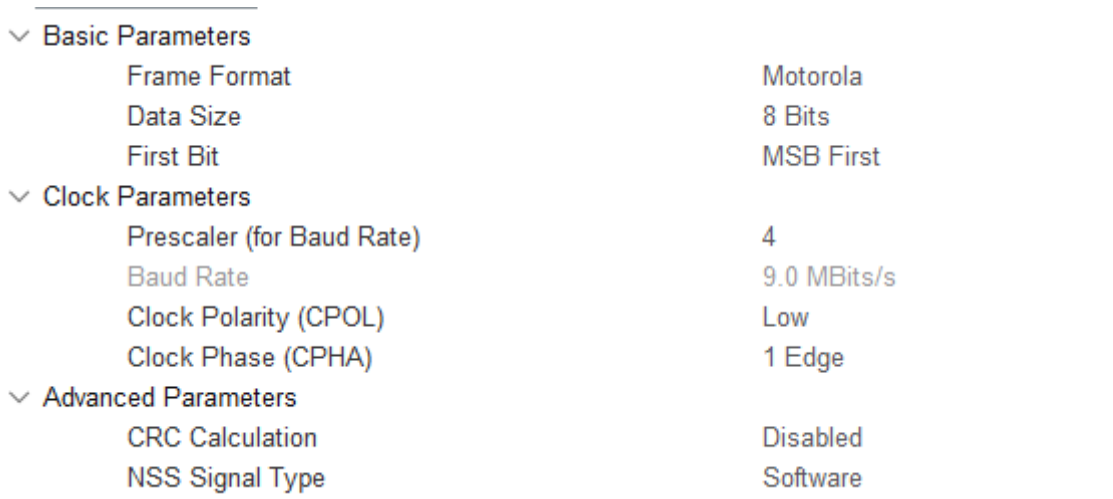
1. 增加接收中断模式
2. 增加配对模式
3. 增加调频模式
4. 增加接收方检波，和信号功率指示来进行低功率模式通信。0dBm时的收发模式电流为10ms，而下电模式只有900nA待机模式为300uA，对比赛等功耗无要求场合可不计较。

使用配置

接口匹配配：

SPI接口配置：

注意点：1. 8字节数据包。2. 高bit先行。3. 波特率控制在8MHz附近。4. 时钟极性低。5. 时钟相位为第一沿。 6. 软件控制



GPIO配置：供需配置3个GPIO

IRQ：可选择上拉输入。如果想用中断式函数，则选用上拉**下降沿**外部中断输入模式

CSN：SPI的从机片选信号。推挽上拉**输出**。

CE: IC的启动信号。使用推挽**输出**。

软件配置：

进入nrf24l01.c的最下方 **line 399** 处

只需修改此项：修改接收还是发送： NRF_InitStruct.Mode = NRF_MODE_TX 。

如需修改地址，则接收方的pipeadress的第一个地址和txmsgadress必须一致。

使用中断形式函数：

在stm32f1xx_it.c中对应管脚的中断服务函数处增加

```
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */
  NRF_IRQHandler(&hnrf24l01);
  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

使用带数据包应答：此处要指定存放地址和长度。

```
hnrf24l01.pAckBuffPtr=ack;
hnrf24l01.AckBuffsize=32;
while (1)
 {
/    hnrf24l01.Init.TxMsgAddr=pipeadress1[(n++)%6];
/    NRF24L01_Switch2_Tx(&hnrf24l01);
   NRF24L01_Get_Instance(&hnrf24l01);
/    buf[0]=count++;
   NRF24L01_Transmit(&hnrf24l01,buf,0xffff);
   HAL_Delay(1);
   /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
 }
```

其余函数：

```
void NRF_ReceiveACKCallback(NRF24L01_HandleTypeDef *hnrf);
void NRF_RxCpltCallback(NRF24L01_HandleTypeDef *hnrf,uint8_t pipex);
void NRF_RxPayloadAckCpltCallback(NRF24L01_HandleTypeDef *hnrf,uint8_t pipex);
```

这三个都是弱定义，用户可以自己实现。

- 第一个是发送方接收到应答数据包的回调函数。应答数据放在句柄下的pAckBuffPtr地址中。
- 第二个是接收回调函数，使用NRF24L01_Recieve_IT。
- 第三个是接收方的发送应答数据包完毕回调函数，必须使用NRF24L01_Recieve_IT。

主要的两个函数：为非堵塞式，在里面是循环轮询，所以需要设定超时时间，否则未接受或发送成功就死循环。

```
HAL_StatusTypeDef NRF24L01_Transmit(NRF24L01_HandleTypeDef *hnrf,uint8_t *tbuf,uint32_t timeout);
HAL_StatusTypeDef NRF24L01_Recieve(NRF24L01_HandleTypeDef *hnrf,uint8_t *tbuf,uint32_t timeout);
```

NRF_InitStruct.Mode = NRF_MODE_TX ;

```
static uint8_t pipeadress[6][5]={{0x12,0x34,0x56,0x78,0x9a},
/* ↓↓↓↓ */                        {0x1a,0xC2,0xC2,0xC2,0xC2},
                                  {0x1b,0x00,0x00,0x00,0x00},
/* this four should have same */ {0x1c,0x00,0x00,0x00,0x00},
/* four latter one */             {0x1d,0x00,0x00,0x00,0x00},
                                  {0x1e,0x00,0x00,0x00,0x00}};
/* ↑↑↑ */
static uint8_t txmsgadress[5] = {0x12,0x34,0x56,0x78,0x9a};
static uint8_t pipepayloadwidth[6] = {32,32,32,32,32,32};
```

- pipeadress：数据通道接收地址。

  每行为1个5位的地址，共6个。第一个地址可以是独立5Bytes，第二到第六个共用后4个Bytes，所以第三到第六个地址的尾4Bytes填0。然后这六个地址开头的第一个Bytes要相互不同。

- txmsgadress：发送地址。

  发送方该地址必须与自身数据通道接收**第一地址**相同

- pipepayloadwidth： 数据通道的数据包长度，最大1-32之间

```
NRF_SET_PIN(hnrf24l01,NRF_PIN_CE ,GPIOB,GPIO_PIN_11);
NRF_SET_PIN(hnrf24l01,NRF_PIN_CSN,GPIOB,GPIO_PIN_12);
NRF_SET_PIN(hnrf24l01,NRF_PIN_IRQ,GPIOB,GPIO_PIN_10);
```

- 绑定管脚： 软硬件结合的地方，修改后面的GPIOB,GPIO_PIN_xx 为对应引脚即可。

```
NRF_InitStruct.AutoReTransmitDelayTime    = NRF_AutoRetransmit_DelayTime_us(250*4);
NRF_InitStruct.AutoRetransmitCountMax     = NRF_AutoRetransmit_Max(15);
NRF_InitStruct.FrequncyChannel            = NRF_FREQUENCY_CHANNEL(50);
NRF_InitStruct.RfAirDataRate              = NRF_AIR_DATA_RATE_2MHz;
NRF_InitStruct.RfPower                    = NRF_RF_PWR_0dBm;
NRF_InitStruct.UseLNA                     = NRF_SETUP_LNA;
```

- AutoReTransmitDelayTime：自动重发时间，为等待接收方应答的超时时间。取值为250us-4000us，步进250us。如果使用带数据包应答，则最好将此值改为500us以上。

- AutoRetransmitCountMax：最大重发次数。
- FrequncyChannel：使用的信道。IC占用的通信频率2.4-2.525Hz，从2.4GHz开始步进1MHz。拥有125个1MHz带宽的信道。如果使用空中数据速率2MHz的，则只拥有一半的2MHz信道。
- RfAirDataRate：空中数据速率，倒数即GFSK解调无线信号1个bit的维持时间。
- RfPower：无线电发射频率。
- UseLNA：使用接收LNA。

```
NRF_InitStruct.Mode                         = NRF_MODE_TX ;
NRF_InitStruct.EnablePipe                   = NRF_DATE_PIPE_ALL ;
NRF_InitStruct.AutoAckPipe                  = NRF_DATE_ACK_PIPE_ALL;
NRF_InitStruct.RxTxAddrWidth2Regbit         = NRF_ADDRWIDTH_REGBIT_5BYTES;
NRF_InitStruct.RxTxAddrWidth                = NRF_ADDRWIDTH_5BYTES;
NRF_InitStruct.RxPipeAddr                   = pipeadress;
NRF_InitStruct.TxMsgAddr                    = (uint8_t*)txmsgadress;
NRF_InitStruct.RxPipePayLoadWidth           = (uint8_t*)pipepayloadwidth;
NRF_InitStruct.TxPayLoadWidth               = 32;
```

- Mode：初始化为接收模式还是发送模式
- EnablePipe：使用数据通道几。
- AutoAckPipe：启动数据通道x的自动应答

```
NRF_InitStruct.EnableAckPayLoad           = NRF_ENABLE_ACK_PAYLOAD;

NRF_InitStruct.EnableDyanmeicPayLoadWidth = NRF_ENABLE_DYNAMIC_PLYLOAD_WIDTH;
NRF_InitStruct.EnableDynamicPayLoadPipe   = NRF_DYNAMIC_PAYLOAD_WIDTH_P0;
```

- EnableAckPayLoad：使能带数据包应答
- EnableDyanmeicPayLoadWidth：使能动态长度数据包，如果使用带数据包应答则必须使能此项。
- EnableDynamicPayLoadPipe：对于接收方只需要使能P0，发送方使能希望的通道号。