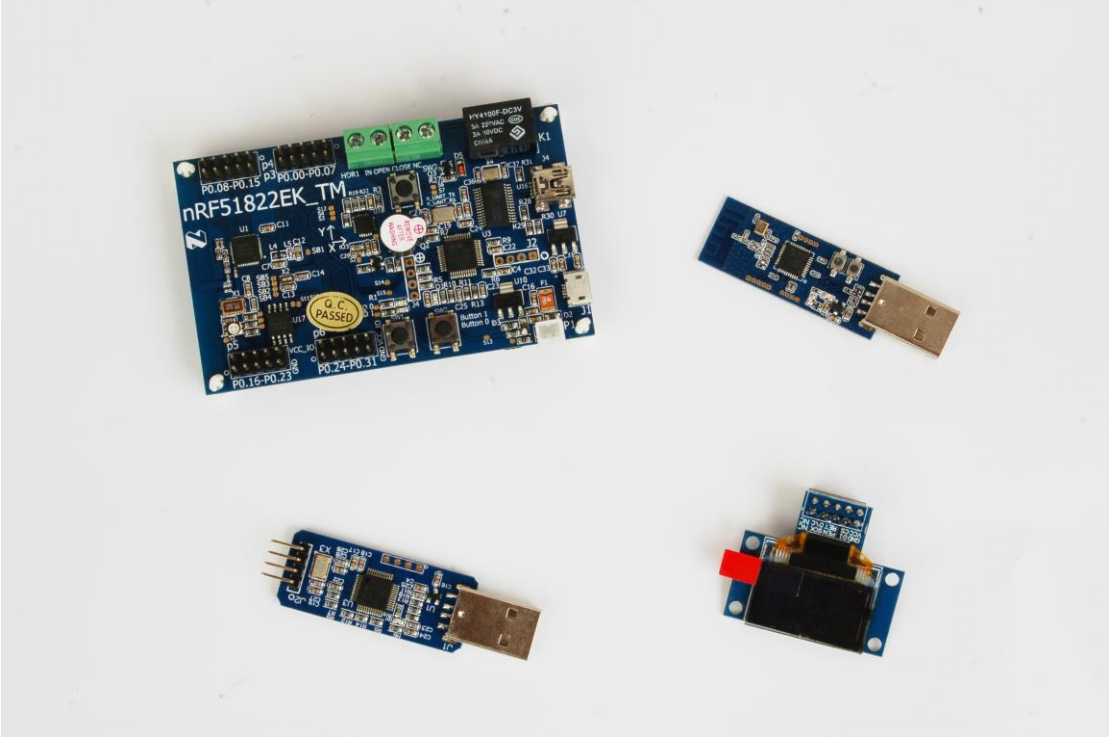
讯联电子nRF51822蓝牙4.0开发实战

IIC

V:1.0

作者：不离不弃　qq 574912883

申明：本教程版权归讯联电子所有。本教程仅供内部客户交流之用。如需引用，请注明出处。由于工程师水平有限，文档难免有所疏漏和错误，由此造成的损失，讯联电子不承担任何责任。



作者：不离不弃　qq 574912883

关于 IIC 总线的核心有以下几点：
：时钟线高电平期间必须保持数据线不变。
：时钟线低电平期间可以改变数据。
：时钟线和数据线上都要接上拉电阻，以使总线不工作时，两根线的电平都处于高电平状态。
：每个传输的字节后面需要由对方回送一个应答信号。

由上面可知，在时钟线为高电平的时候如果数据线改变，那么就是”不合法” 的。于是就刚好利用这种”不合法的”的跳变来作为数据 起始信号和停止信号。
于是规定：
：时钟线为高电平时，数据线由高到低跳变为起始信号
：时钟线为高电平时，数据线有低到高跳变为 停止信号。

关于 IIC 的原理和时序，网上很多文章，这里主要介绍 51822 硬件 Iic 的使用。

首先看先相关寄存器的说明：
STARTRX: 启动接收，即 iic 的读。
STARTTX：启动发送，即写。
ADDRESS: 设备地址寄存器，IIC 总线的通信，总是以地址+读写标识 开始，因为总线上可能挂了不止一个 IIC 设备，所以需要通过发送地址来说明要和哪个设备通信。

这里需要注意的是，51822 的 ADRESS 寄存器只有 7 位有效，不包含低 8 位的 读写指示，读写指示是 硬件通过 你是启动读(通过设置 STARTRX 寄存器)还是启动写(通过设置 STARTTX 寄存器)来自动 在 ADDRESS 的 7 位发送完后在发送的。
所以在使用时，你不需要自己根据是读还是写，而设置地址寄存器 ADDRESS =(ADDRESS<<8) |0x01 或 ADDRESS =(ADDRESS<<8)&0xfe。 而是 直接 ADDRESS = 7 位设备地址 就可以了。读写位 会有硬件自动发送。

STOP:停止 IIC
SUSPEND:挂起 IIC(暂停)，通常在 IIC 读中使用，是为了在收到一个字节后，暂停 IIC 的传输，以保证 接收数据寄存器不被后续的数据覆盖。
RESUME:恢复被暂停的 IIC，继续传输

关于事件寄存器，主要是如下两个事件需要关注：
RXDREADY: 指示数据接收完成。
TXDREADY:指示数据发送完成
BB：该事件在每一个字节发送或者接收之前产生，改事件通常使用在 读操作中，即接收操作。

SHORTS:该寄存器重要用来 将某个 event 和 task 短接。上面 说过，通过 设置 SUSPEND 可

以暂停 IIC 总线，这样可以避免后续的接收数据覆盖了接收寄存器中的数据，而 BB 事件在每次数据接收之前会产生。 于是在接收过程中，可以通过判断接收的数据量如果还大于 1 那么久应该 通过 SHORTS 寄存器将 BB 事件和 SUSPEND 任务短接，那么每次从接收寄存器 RXD 中提取数据时，IIC 总线就会自动被暂停，也就避免了后续数据覆盖了 RXD 内容，而如果接收的数据只剩最后一个了，那么久可以将 B B 事件和 STOP 任务短接，那么在接收最后一个数据后就会自动发送停止信号了。这块看后面的代码注释更好理解。

INTEN:
INTENSET:
INTENCLR:
以上三个寄存器都是用来设置 当产生各种事件是是否产生中断。本教程中并未使用中断。

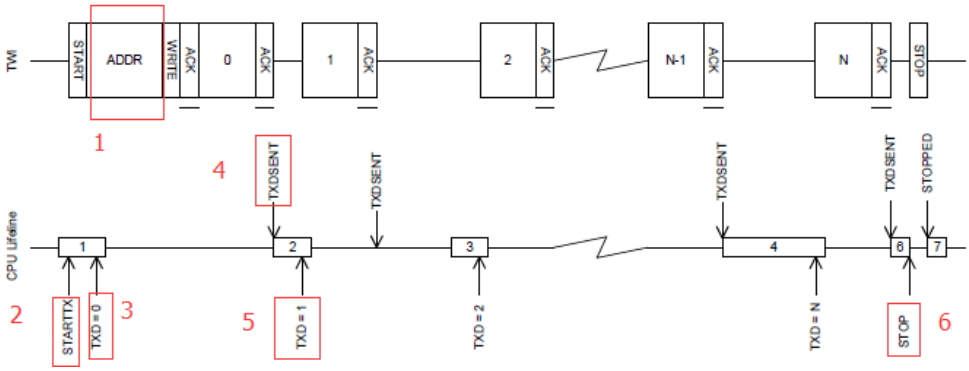
ERRORSRC:用来记录产生的错误原因

PSELSCL:用来选择哪个引脚作为 时钟线
PSELSDA:用来选择哪个引脚作为数据线

RXD:从该寄存器中提取接收到的数据
TXD：将要发送的数据填入该寄存器

FREQUENCY:设置 发送速率
ADDRESS:设置要通信的设备的地址

51822 的 IIC 写操作如下图所以所示：



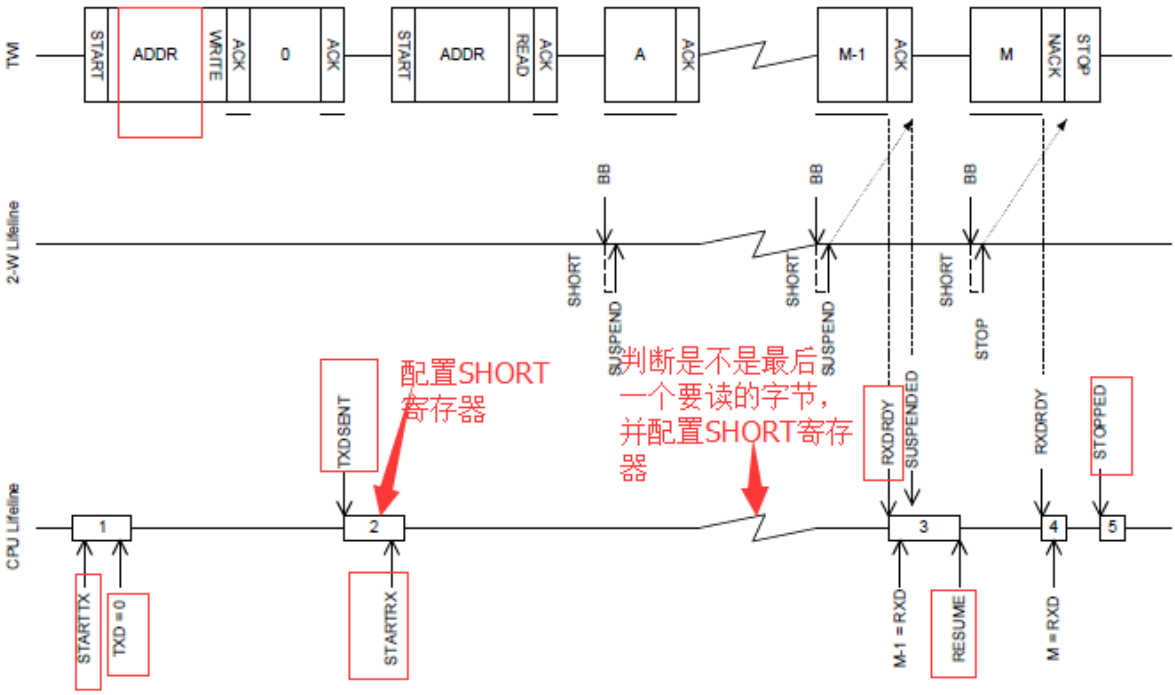
所以对于写需要如下几个步骤：
1：首先设置地址寄存器。
2：设置 STARTTX 启动写操作。
3：将要发送的数据放入 TXD 寄存器中。
4：等待 TXDSENT 信号

作者：不离不弃 qq 574912883

5：如果有数据，继续将后续数据放入 TXD 中，并会到步骤 4.否则到步骤 6

6：设置 STOP 寄存器，并等待 IIC 停止了。

对于读操作，一般 IIC 设备都需要先提供要读的寄存器或地址。所以读操作一般需要先有一个写操作，来设置要读的地址或寄存器，然后再跟随读操作。51822 提供的操作图如下所示：



所以对于需要先写地址再执行读的操作有如下几个步骤：

1：设置设备地址

2：设置 STARTTX 启动写操作

3：将要发送的数据(寄存器地址或数据地址)写入 TXD 寄存器中

4：等待 TXDSENT 事件，以确定数据发送完毕。

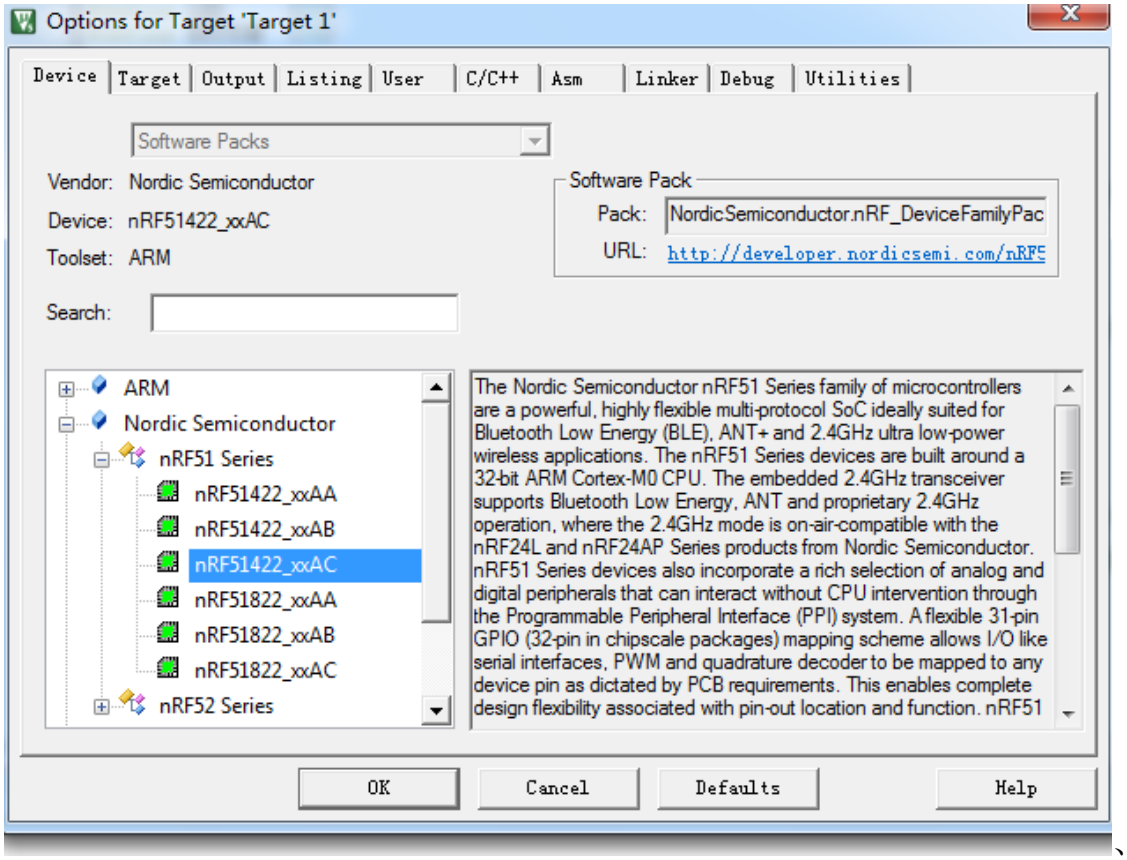
5：判断是否只有一个要读的数据，如果不是设置 SHORTS 将 BB event 和 SUSPEND task 短接(BB evnet 产生时自动触发 SUSPEND task)，否设置则 BB event 和 STOP task 短接。

6：设置 STARTRX 寄存器启动读操作。

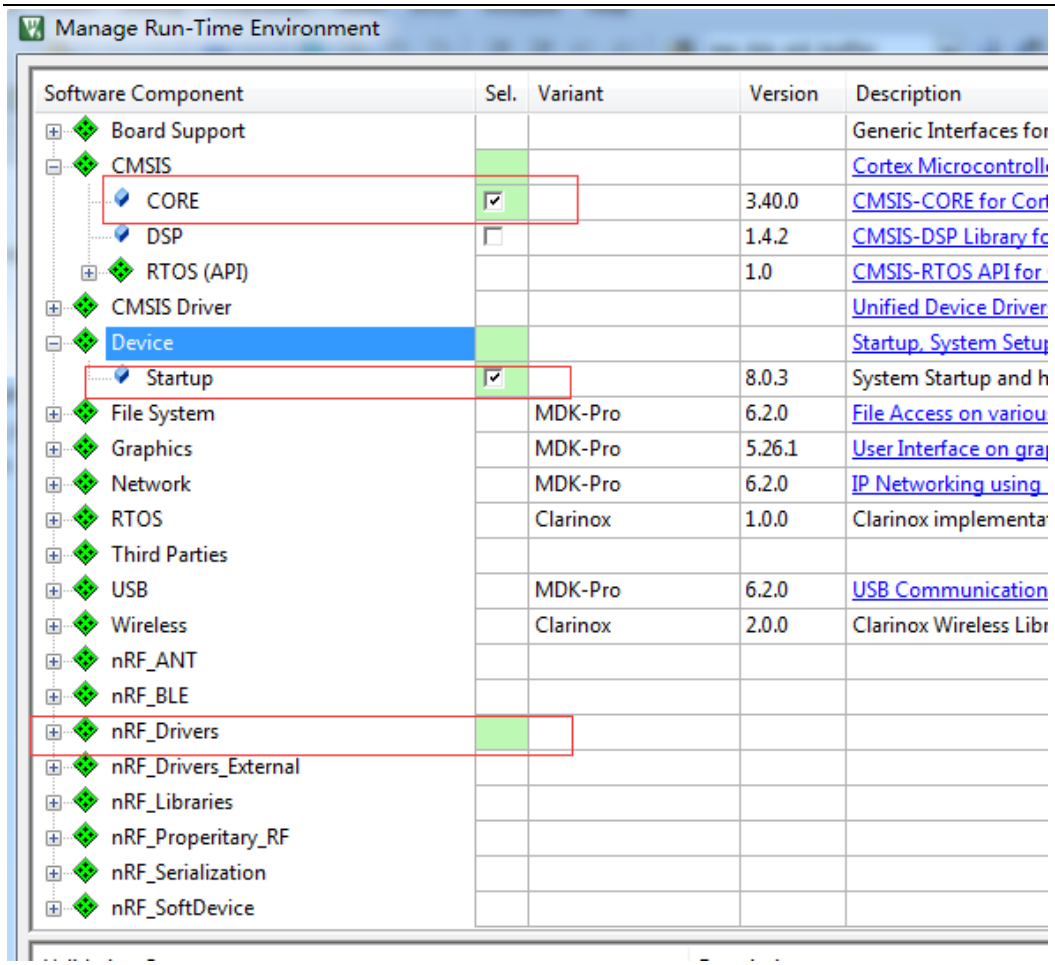
7：等待 RXDRDY 事件，提取数据。如果后续只有一个要读的数据了，则设置 BB event 和 STOP task 短接，并跳到 8。否则继续执行 7

8：等待 STOPED 信号。

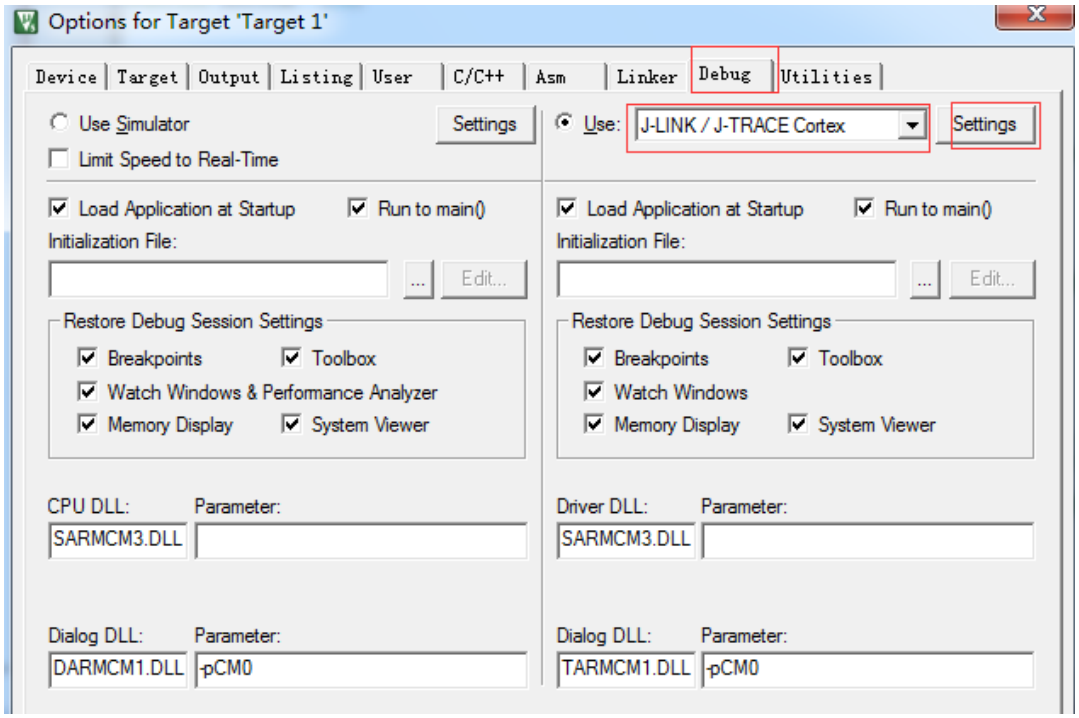
新建工程选择自己板子使用的芯片型号：

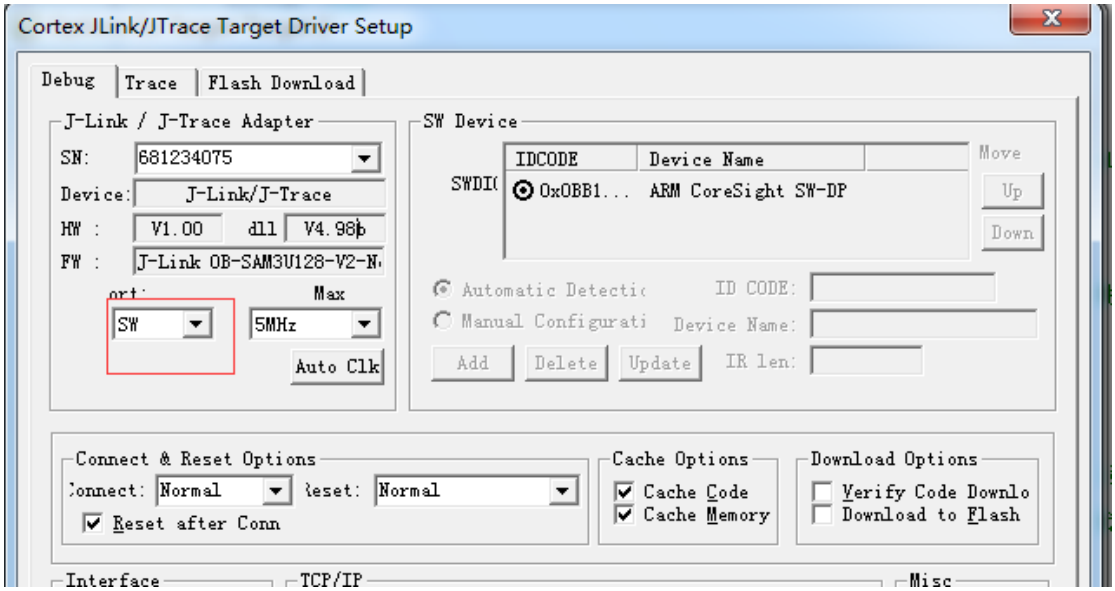


教程中为了更直接的理解模块的使用。不使用 sdk 中提供的库函数，而直接操作寄存器来实现。
所以运行时环境勾选下必要的 CMSIS 下的 CORE，Device 下的 Startup。因为用了 gpio 的函数 勾选一下 nRF_Drivers 下的 nrf_gpio 就可以了。

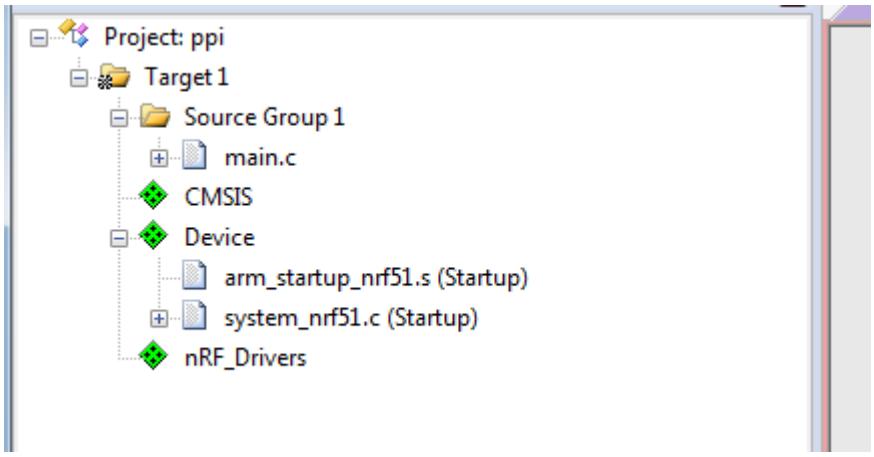

Manage Run-Time Environment
Software Component
Sel.
Variant
Version
Description
Board Support
Generic Interfaces for
CMSIS
Cortex Microcontroll
CORE
3.40.0
CMSIS-CORE for Cort
DSP
1.4.2
CMSIS-DSP Library fo
RTOS (API)
1.0
CMSIS-RTOS API for
CMSIS Driver
Unified Device Driver
Device
Startup, System Setup
Startup
8.0.3
System Startup and h
File System
MDK-Pro
6.2.0
File Access on variou
Graphics
MDK-Pro
5.26.1
User Interface on gra
Network
MDK-Pro
6.2.0
IP Networking using
RTOS
Clarinox
1.0.0
Clarinox implementa
Third Parties
USB
MDK-Pro
6.2.0
USB Communication
Wireless
Clarinox
2.0.0
Clarinox Wireless Libr
nRF_ANT
nRF_BLE
nRF_Drivers
nRF_Drivers_External
nRF_Libraries
nRF_Properitary_RF
nRF_Serialization
nRF_SoftDevice

然后配置 jlink 的设置(我的板子使用的是 jlink 的 sw 方式下载程序)。





创建 main.c 文件，然后添加到工程中



下面介绍 main.c 代码细节。讯联的板子上有一个 MPU6050 是通过 IIC 来操作的，所以这里就使用该设备来验证 IIC 驱动的正确性。根据板子的接线原理图，6050 的设备地址为 0x69，根据 6050 的手册知道该传感器 0x75 地址的寄存器中存放的数据始终未 0x68，所以下面就读这个寄存器中的值，来验证 IIC 是否正确通信。

PS:下面的驱动只是为了说明驱动原理，错误情况处理以及等待超时都没有做，如果自己的项目中需要使用 IIC，请使用 sdk 中提供的，或者将下面的驱动参考 SDK 中的驱动加上错误处理和超时处理的相关代码。

另外手册中说明为了使 TWI 在 system off 模式下，以及 TWI 关闭的情况下，使 TWI 使用的引脚保持正确性，应该在引脚作为 TWI 功能引脚使用前设置其 GPIO 为特定状态。如下图所示。官方给的驱动中有做这个设置，本教程中为方便并未设置。

To secure correct signal levels on the pins used by the TWI master when the system is in OFF mode, and when the TWI master is disabled, these pins must be configured in the GPIO peripheral as described in *Table 258: GPIO configuration* on page 145.

Only one peripheral can be assigned to drive a particular GPIO pin at a time, failing to do so may result in unpredictable behavior.

Table 258: GPIO configuration

TWI master signal	TWI master pin	Direction	Drive strength	Output value
SCL	As specified in PSELSCL	Input	S0D1	Not applicable
SDA	As specified in PSELSDA	Input	S0D1	Not applicable

```
#include "nrf51.h"
#include "nrf_gpio.h"
#include <stdbool.h>
#include <stdio.h>
```

作者：不离不弃 qq 574912883

```
#include <stdint.h>
#include "nrf_delay.h"

#define SCL_PIN      (1)
#define SDA_PIN      (5)

void iic_init(void){
    NRF_TWI0->PSELSCL = SCL_PIN;
    NRF_TWI0->PSELSDA = SDA_PIN;

    NRF_TWI0->FREQUENCY = 0x06680000;//400Khz
    NRF_TWI0->ENABLE = 5;

    //清零各种事件
    NRF_TWI0->EVENTS_STOPPED = 0;
    NRF_TWI0->EVENTS_RXDREADY = 0;
    NRF_TWI0->EVENTS_TXDSENT = 0;
    NRF_TWI0->EVENTS_BB = 0;
    NRF_TWI0->EVENTS_ERROR = 0;
}


void write_datas(uint8_t dev_addr, uint8_t arg_addr, uint32_t len,
uint8_t *p_data, bool is_stop){

    NRF_TWI0->ADDRESS = dev_addr;

    NRF_TWI0->TXD = arg_addr;
    NRF_TWI0->EVENTS_TXDSENT = 0;          //先清零一下事件

    NRF_TWI0->TASKS_STARTTX = 0X01;        //开始启动发送

    while(1){
        while( NRF_TWI0->EVENTS_TXDSENT == 0 ){ } //等待发送完成

        NRF_TWI0->EVENTS_TXDSENT = 0;      //清零
        if( (len--) == 0 ){
            break;

        }
        NRF_TWI0->TXD = *p_data++;
    }
//判断是否需要停止 IIC，对于单独的写操作，应该需要停止 IIC,
```

```
//对于读寄存器中值的操作，因为需要先写地址，后再发起读，所以前面的写操
//作之后就不需要 停止 IIC。

    if ( is_stop ){
        NRF_TWI0->EVENTS_STOPPED = 0;
        NRF_TWI0->TASKS_STOP = 1;
        while( NRF_TWI0->EVENTS_STOPPED == 0){};  //等待 iic 正确结束
    }
}


void read_data(uint8_t dev_addr, uint8_t arg_addr, uint32_t len, uint8_t
*p_data){

    write_datas(dev_addr, arg_addr, 0, NULL, false); //先写地址

//  NRF_TWI0->ADDRESS = dev_addr;   //设置地址

    if ( len == 1 ){
        NRF_TWI0->SHORTS = 2;
    }else{
        NRF_TWI0->SHORTS = 1;    //将 BB 事件和 SUSPEND 短接，则每次接收
到数据后，总线被挂起，目的是为了在提取数据的时候，防止对方又发送数据过
来导致接收的数据被覆盖
    }
    NRF_TWI0->EVENTS_RXDREADY = 0;          //先清零一下事件
    NRF_TWI0->EVENTS_STOPPED = 0;
    NRF_TWI0->TASKS_STARTRX = 0X01;         //开始启动接收

    while( (len--) > 0 ){
        while( NRF_TWI0->EVENTS_RXDREADY == 0 ){} //等待接收到数据
        NRF_TWI0->EVENTS_RXDREADY = 0; //清零事件
        *p_data++ = NRF_TWI0->RXD;

        if( len == 0){
            break;
        }
        if ( len == 1){
            NRF_TWI0->SHORTS = 2;
        }
        NRF_TWI0->TASKS_RESUME = 1;
    }
    while ( NRF_TWI0->EVENTS_STOPPED == 0){}
```

```
}


#define LED1 (18)
#define LED2 (19)
int main(void){
    uint8_t who_am_i;

    //讯联的 51822EK_TM 板子 是高电平点亮 LED
    nrf_gpio_cfg_output(LED1);
    nrf_gpio_pin_clear(LED1);
    nrf_gpio_cfg_output(LED2);
    nrf_gpio_pin_clear(LED2);

    nrf_delay_ms(50);
    iic_init();


    read_data(0x69, 0x75, &who_am_i, 1);
    if(who_am_i == 0x68){
           nrf_gpio_pin_set(LED1);
    }else{
           nrf_gpio_pin_set(LED2);
    }
    while(1);
    return 0;
}
```