SP370 用户编程手册 V2.6

版本号	文件创建日期	最后修改日期	作者	签核
V1.0	2015-07-03	2016-03-21	Deathgod	
V2.0	2015-07-03	2016-07-12	Deathgod	
V2.5	2015-07-03	2016-10-24	Deathgod	
V2.6	2015-07-03	2016-11-08	Deathgod	
http://www.xiaojijiji.com/				
小鸡叽叽科技				
感谢您支持小鸡叽叽科技，请保护作者的劳动成果，未经允许不得将资料、代码传送他人或者网络				

SP370 编程环境

软件需求： 1. keil （C51） 本人使用的是 5.20 版本

2. SP370 编译工具 Infineon SP3x Keil Driver Setup.exe

安装 Keil 之后，需要安装 SP370 编译工具，才能编译出 SP370 的执行档!

1.如何创建第一个例程（01.LED）

1.1 创建需要的文件夹

INCLUDE： 用于存放各个需要包含为头文件

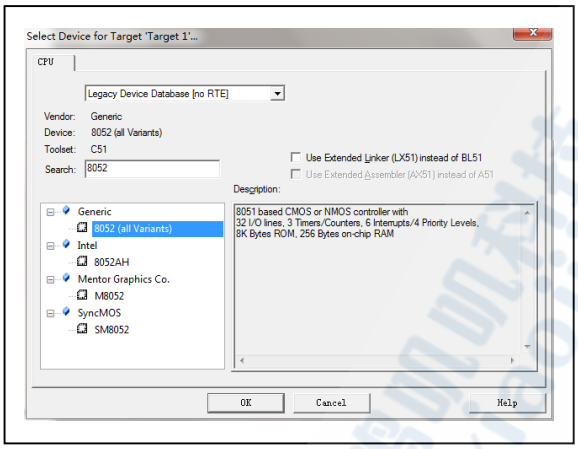
USER： 放置用户文件

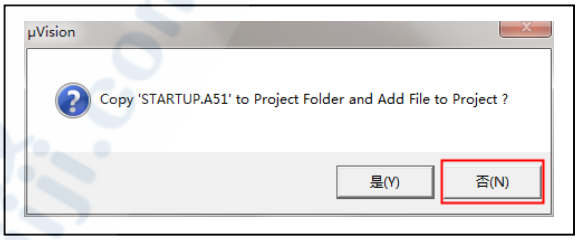
INCLUDE 需要包含 Reg_SP37.h (该文件来自 00_Empty_Project)

USER Main.c STARTUP.A51 (该文件来自 00_Empty_Project)

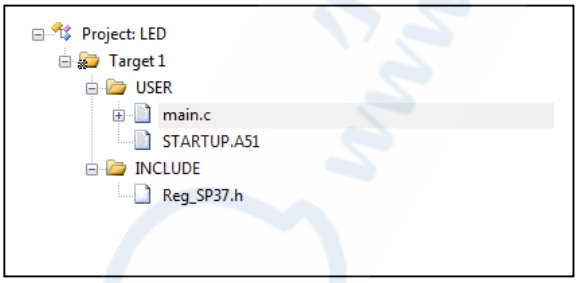
1.2 项目创建

SP370 为 8051 内核，所以可以选择一个 8051 的项目来创建。SP370 硬件与 8052 一致，这里选择 8052，其实 51 内核的编译器都是一致的，选不同的 IC 没有影响

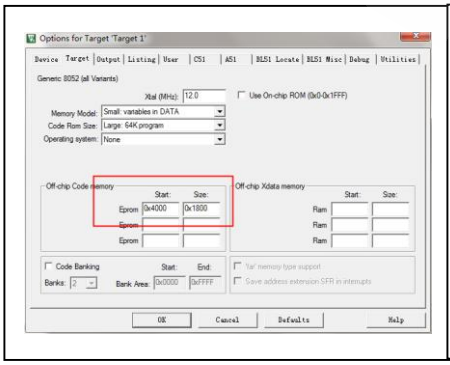


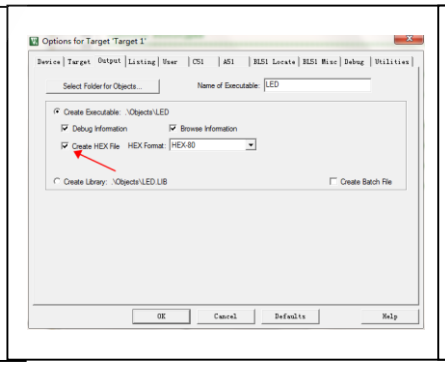


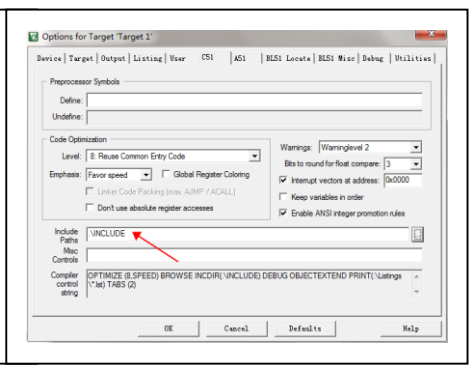
1.3 目录机构



1.4 配置（SP370 的 CODE 地址为 0x4000,大小为 0x1800）,并且添加头文件目录

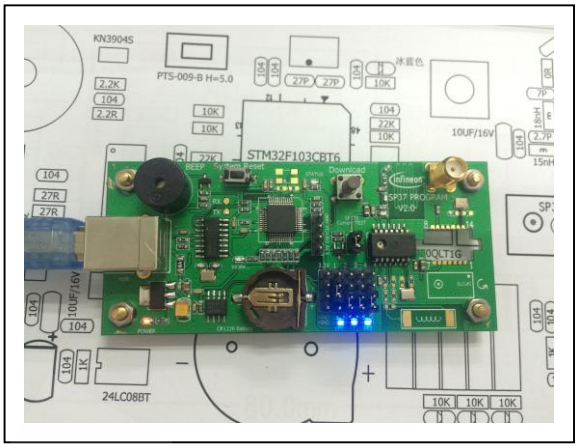






1.5 编译与结果

在 Object 文件下生成 LED.hex (通过烧录器烧录，可以看到 P0,P1,P2LED 灯全部点亮)



注: PP0,PP1,PP2 的跳线帽请接到”测试模式”，才能看到现象。
烧录软件的时候，需要将跳线帽还原至”烧录模式”
(不同模式的跳线帽连接方式请参考”使用说明书”)

实际烧录过程中并没有用到 PP2,下载的时候
可以不用拔插 PP2 的跳线帽

1.6 程序说明

整个程序是一个寄存器配置的过程（详见 SP37_Datasheet.pdf）
Page:113

```
/*****************************************************************
void main(void)
{
  P1DIR = 0x00;     //PP0,PP1,PP2配置为输出
  P1OUT &= ~((1<<0) | (1<<1) | (1<<2));     //管脚配置为低电平

  P1SENS &= ~0xFF;  //上拉电阻
  while(1);
}
```

2.使用 SP370 内部库函数(02.LED_FLASH)

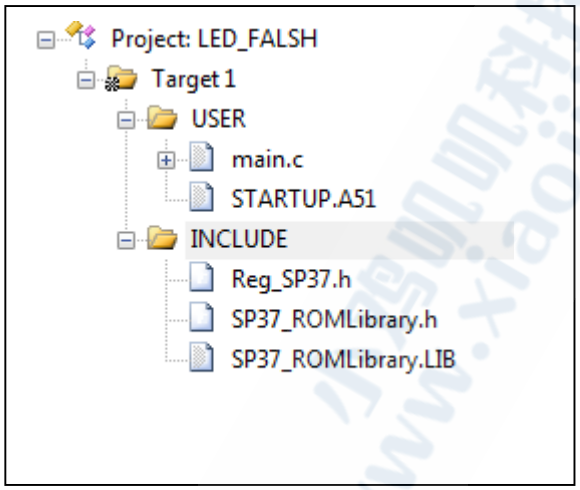
此例程使用库函数(Wait100usMultiples)
使用方法 SP37_Rom_Lib_Guide_V1.0.pdf Page:112

2.1 创建需要的文件夹

INCLUDE： 用于存放各个需要包含为头文件
USER： 放置用户文件

INCLUDE 需要包含 Reg_SP37.h （该文件来自 00_Empty_Project）
另需要添加 SP37_ROMLibrary.h 与 SP37_ROMLibrary.LIB 文件
（文件来自 00_Empty_Project）
USER Main 文件

2.2 目录机构

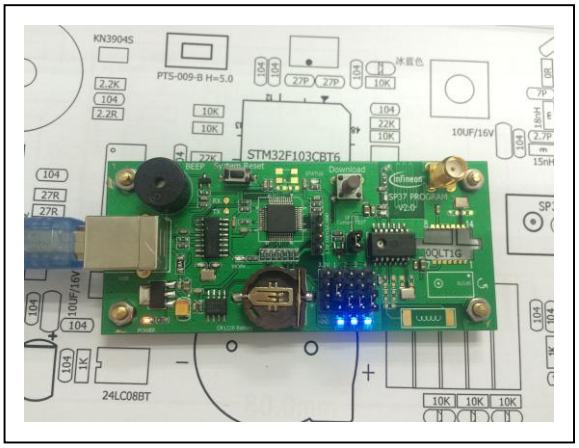


2.3 编译与结果

在 Object 文件下生成 LED_FLASH.hex (通过烧录器烧录，可以看到 P0,P1,P2LED 灯开始闪烁,1Hz)

注: PP0,PP1,PP2 的跳线帽请接到"测试模式"，才能看到现象。
烧录软件的时候，需要将跳线帽还原至"烧录模式"
(不同模式的跳线帽连接方式请参考"使用说明书")

实际烧录过程中并没有用到 PP2,下载的时候
可以不用拔插 PP2 的跳线帽



2.5 程序说明

通过内部库函数 Wait100usMultiples 来提供延时。

使用内部库函数，可以节省大量的程序空间。

main.c

```
#include "Reg_SP37.h"
#include "SP37_ROMLibrary.h"

//程序在PP0,PP1,PP2管教出产生方波
void MAIN(void)
{
  P1DIR = 0x00;     //PP0,PP1,PP2配置为输出
  P1OUT &= ~((1<<0) | (1<<1) | (1<<2));     //管脚配置为低电平
  P1SENS &= ~0xFF;  //上拉电阻

  while(1)
  {
    P1OUT &= ~((7<<0));
    Wait100usMultiples(5000);   //调用库函数 延时5000*100us = 500ms

    P1OUT |= (7<<0);
    Wait100usMultiples(5000);
  }
}
```

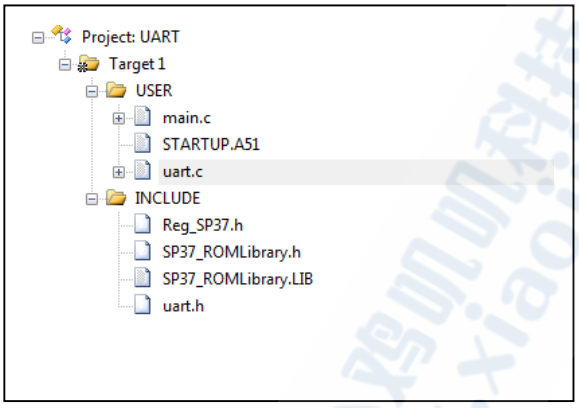
3.实现 Printf 函数(03.UART)

说明： 在调试过程中，Printf 函数是最好用一种方式，可打印任何用户想要看到的数据。
但是 SP370 这颗 IC，没有硬件 UART 功能，只能通过模拟的方式来产生。
本例子程序中 UART_TX 使用的是 PP2 管脚。并使用了 TIM 资源(详情:uart.c uart.h)

波特率：19200 N 8 1
UART_TX：PP2

注：Printf 会占用较大的程序空间，通过编译出来的文件可以很清楚知道这一点。
例子程序中大量用到 printf 函数，方便调试与教程。在实际产品中，由于程序空间的限制，一般都会把 printf 去掉。

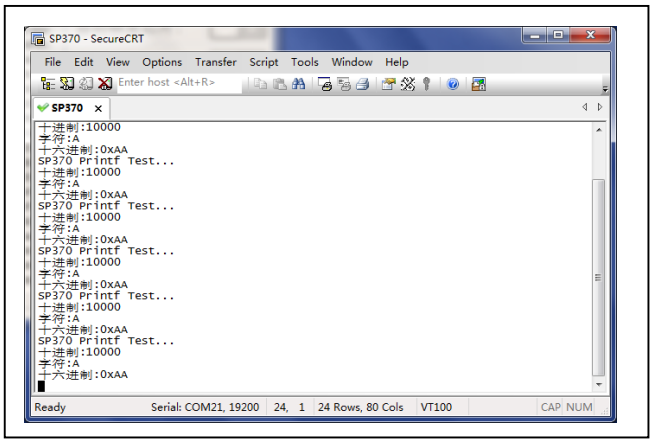
3.1 目录机构



3.2 编译与结果

在 Object 文件下生成 UART.hex (通过烧录器烧录，可以在串口调试助手中看到 来自 SP370 的 Printf 信息)

我使用的串口软件是 SecureCRT,可以选择自己习惯使用的串口调试助手

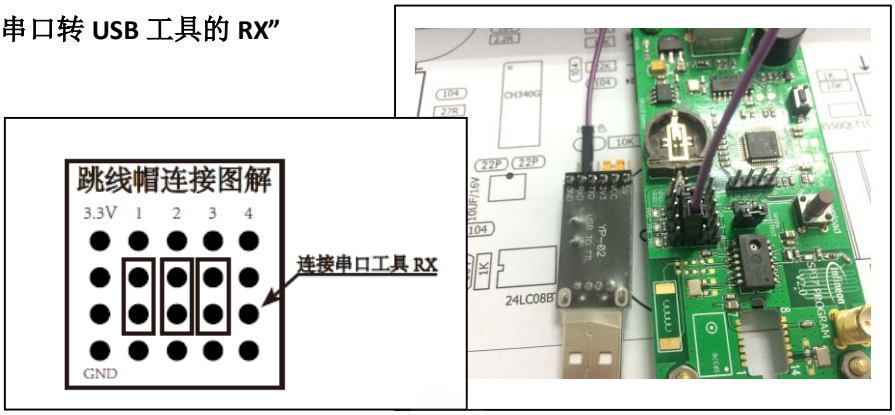


Printf 信息会从 SP37 的 PP2 打印出来，请按照接“3.3 接线方式”进行连线

注：烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）

3.3 接线方式

需要将开发板的 “PP2”与“串口转 USB 工具的 RX”连接（如右图）



3.4 程序说明

使用 printf 函数需要实现 putchar 函数

本例程中 putchar 通过定时器控制 PP2 管脚的电压，产生串口协议的波形，从而进行通信。

uart.c | main.c | Reg_SP37.h

```
#include "Reg_SP37.h"
#include "SP37_ROMLibrary.h"
#include <uart.h>
#include <stdio.h>
#include <ctype.h>

void MAIN(void)
{
  Uart_Init();

  while(1){

    WDRES = 1;
    printf("%s\r\n","SP370 Printf Test...");
    printf("十进制:%d\r\n",10000);
    printf("字符:%c\r\n",'A');
    //打印16进制需要将数据强制为int
    printf("十六进制:0x%X\r\n",(int)0xAA);

    //延时1S
    Wait100usMultiples(10000);
  }
}
```

芯片传感数据的获取

4.获取温度数据(04.Temperature)

5.获取电压数据(05. Voltage)

6.获取加速度数据(06. Acceleration)

7.获取气压数据(07. Pressure)

说明： 以上例子程序都用到库函数（详情 SP37_Rom_Lib_Guide_V1.0.pdf），数据通过 UART 输出。

UART：

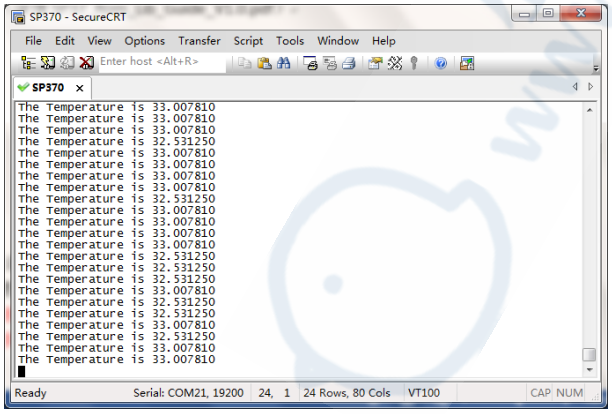
波特率：19200 N 8 1

UART_TX：PP2

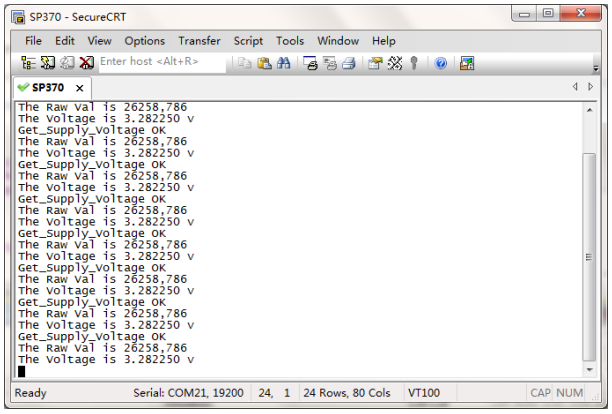
编译与结果

在 Object 文件下生成对应的 hex 文件(通过烧录器烧录，可以在串口调试助手中看到来自 SP370 的 Printf 信息)

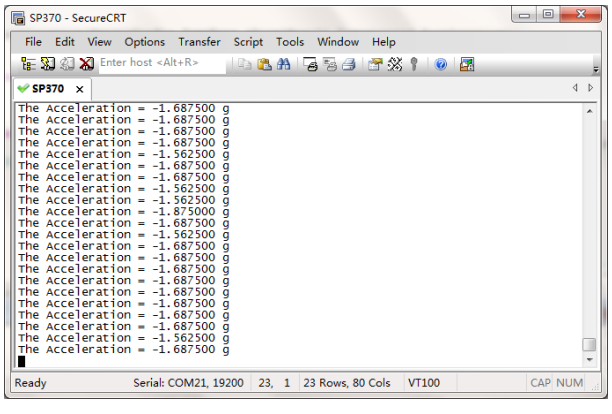
注: 烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）



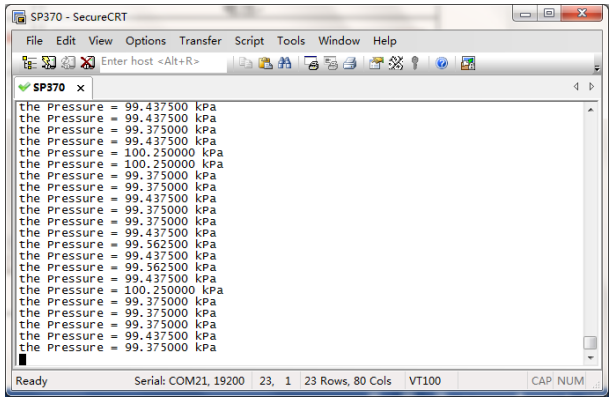
温度



电压



加速度



气压

8.ReadFalsh(08. Read_Falsh)

说明： Read_Falsh 可以读取 FLASH 空间中任意一个地址的数据，

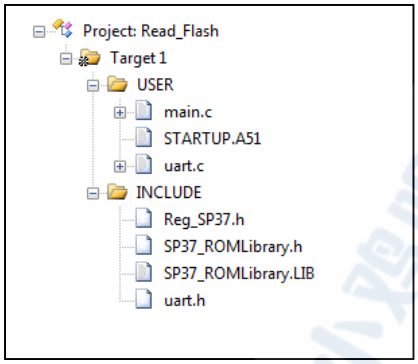
推荐用法： 获取芯片唯一序列号(唯一 ID)

SP370 芯片在出厂的时候会有一个唯一芯片 ID，地址: 5850H ~5853H，大小 4Byte。
由于此唯一 ID,在出厂的时候已经被固定，不可更改。所以可以很好地用来区别不同位置的轮胎，或者不同车子的轮胎。当然，也是加密算法的不二之选。

获取用户“特征 ID”

SP37 烧录器 增加一个“烧录特征 ID”的功能，烧录地址为 57E0H~57E1H ，大小 2Byte
同样读取 57E0 与 57E1 可以获取”特征 ID”
此”特征 ID”的作用完全取决于客户的定义，比如可以用来记录版本号，记录批次，记录芯片生产个数等。

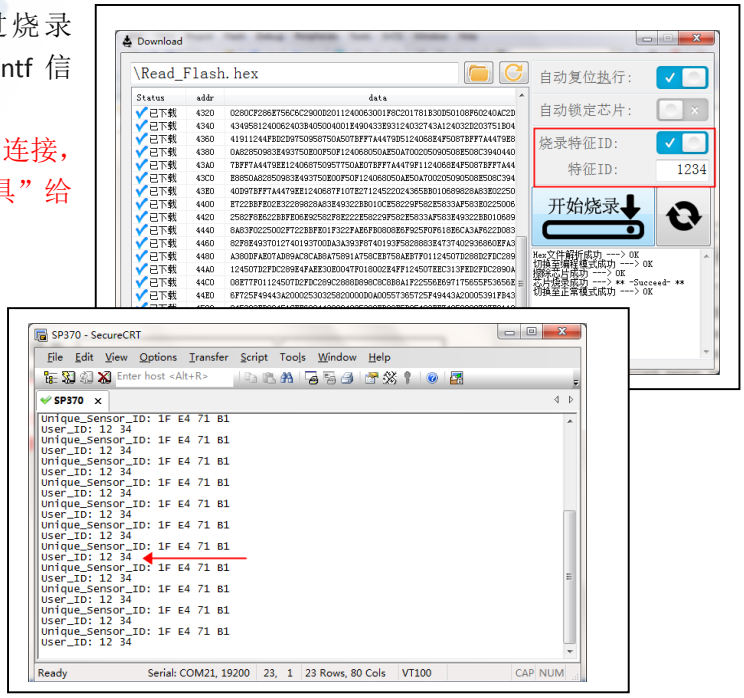
8.1 目录机构



8.2 编译与结果

在 Object 文件下生成 Read_Falsh.hex (通过烧录器烧录，可以在串口调试助手中看到 来自 SP370 的 Printf 信息)

注: 烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）



8.3 程序说明

用到绝对地址访问函数”CBYTE[addr]” 使用这一个函数，需要包含头文件#include <absacc.h>

```
//打印芯片唯一ID
addr = 0x5850;
printf("Unique_Sensor_ID: ");
for(i=0;i<4;i++){
  printf("%02X ",(int)CBYTE[addr]);
  addr++;
}
printf("\r\n");

//打印用户特征ID
addr = 0x57E0;
printf("User_ID: ");
for(i=0;i<2;i++){
  printf("%02X ",(int)CBYTE[addr]);
  addr++;
}
printf("\r\n");

Wait100usMultiples(10000);
}
```

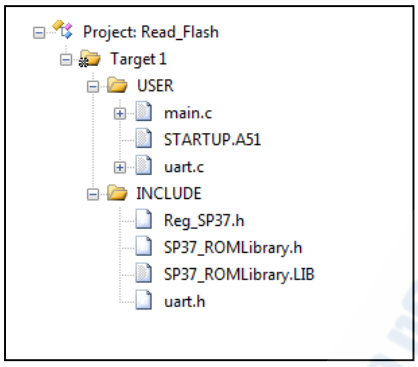
9.无线发射 RF 433MHz(09. RF)

无线 RF 数据协议如下：

频率: **433.92MHz**
频偏: **±35KHz**
调制方式: **FSK**
数据速率: **2.0Kbps**
编码方式: **曼切斯特**
同步码(SYNC): **0x15**

例子程序发送的数据为:0x12,34,56,78,AA,BB,CC,DD　　8Byte

9.1 目录机构



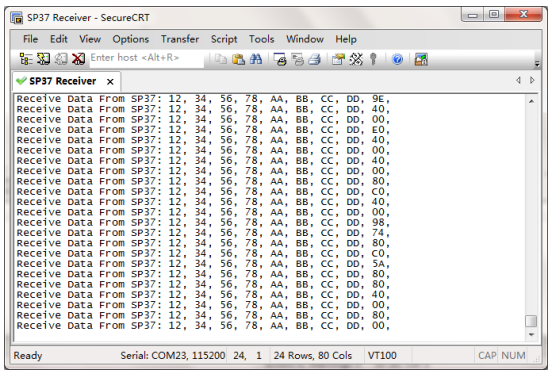
9.2 编译与结果

在 Object 文件下生成 RF.hex　　(通过烧录器烧录，可以在串口调试助手中看到 来自 SP370 的 Printf 信息)

接收端配套软件(以下软件都可配套):

TDA5235_RFData_Printf　　(查询方法)
TDA5235_RFData_Printf_INIT　　(中断)
TDA5235_RFData_OLED　　(中断+OLED 显示)

注：烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）



来自接收端的打印信息
接收端需要下载 ” TDA5235_RFData_Printf “项目下的 hex 文件

9.3 程序说明

主要是构造好发射的数据，如右图。

```
signed char StatusByte;
unsigned char idata descriptorPtr[] = {
  0x00,
  0x00,

  8*13+8*1+8*8,

  0x00,0x00,0x00,0x00,0x00,   //唤醒
  0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,

  0x15,   //SYNS

  0x12,   //DATA
  0x34,
  0x56,
  0x78,
  0xAA,
  0xBB,
  0xCC,
  0xDD,

  0xF1,
};
```

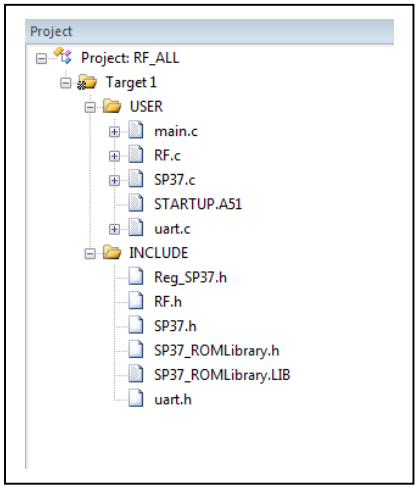
程序流程

1. 配置 RF
   a) RFTX 寄存器相关配置
2. 配置晶振参数
   a) XTAL1 与 XTAL2
3. 开启晶振
   a) StartXtalOsc
4. VCO_Tuning（）
5. 发送数据包
   a) Send_RF_Telegram

10.无线发射传感器数据(10. RF_ALL)

说明： 此例程通过 RF 发送： “用户特征 ID + 气压 + 温度 + 重力加速度 + 电压值+ CRC” 10Byte 的数据
实际项目中根据需求可自行调整（一般需要结合耗电情况考虑）

10.1 目录机构



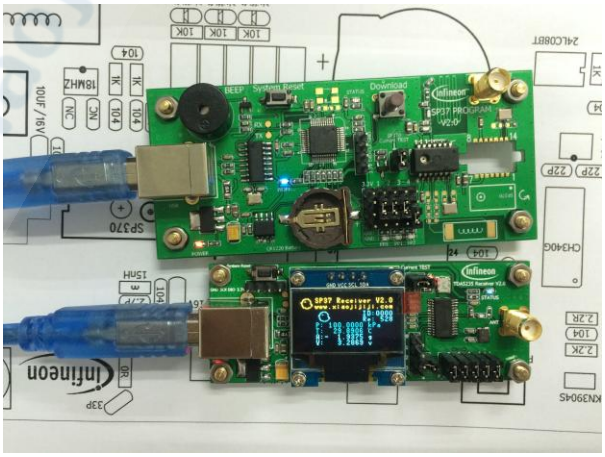
10.2 编译与结果

在 Object 文件下生成 RF_ALL.hex (通过烧录器烧录，可以在接收端看到来自 SP370 RF 的数据)

接收端配套软件(以下软件都可配套):

SP37_RFData_To_OLED_WithCRC

注: 烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）



接收端显示画面，接收端需要下载 ”SP37_RFData_To_OLED_WithCRC “项目下的 hex 文件

10.3 程序说明

主要是构造好发射的数据，如右图。

```
//原始数据
unsigned char idata RF_Array[]={
  0x00,
  0x00,

  (2+1+RF_DATA_BYTES)*8,     //DATA_LANGTH
                             //WAKE UP + SYNS + DATA

  0x00,
  0x00,   //WAKE UP

  0x15,   //SYNS

  0x00,   //DATA
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,

  0xF1,   //END
};
```

程序流程

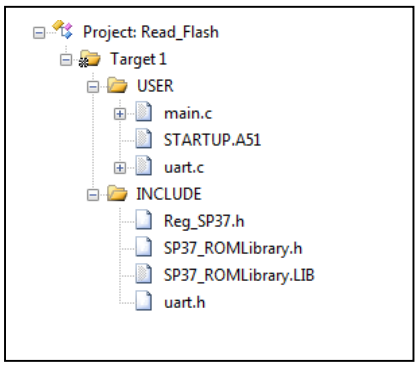
1. 配置 RF
   a) RFTX 寄存器相关配置
2. 配置晶振参数
   a) XTAL1 与 XTAL2
3. 开启晶振
   a) StartXtalOsc
4. VCO_Tuning（）
5. 发送数据包
   a) Send_RF_Telegram

11.休眠与唤醒(11.WU)

说明： 为了降低功耗休眠功能是必不可少的。合理使用休眠功能，可以很大提高的电池寿命

辅助说明：可以尝试固定时间内唤醒，检查 SP37 传感器加速度的大小。由于汽车轮子转动时，加速度必然会变化。

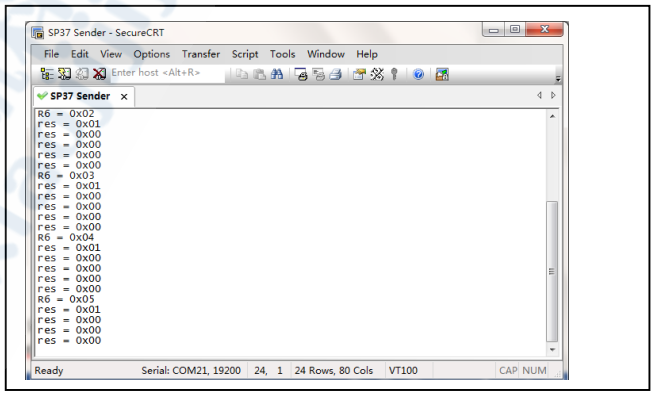
11.1 目录机构



11.2 编译与结果

在 Object 文件下生成 WU.hex (通过烧录器烧录，可以在串口调试助手中看到 来自 SP370 的 Printf 信息)

注: 烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）

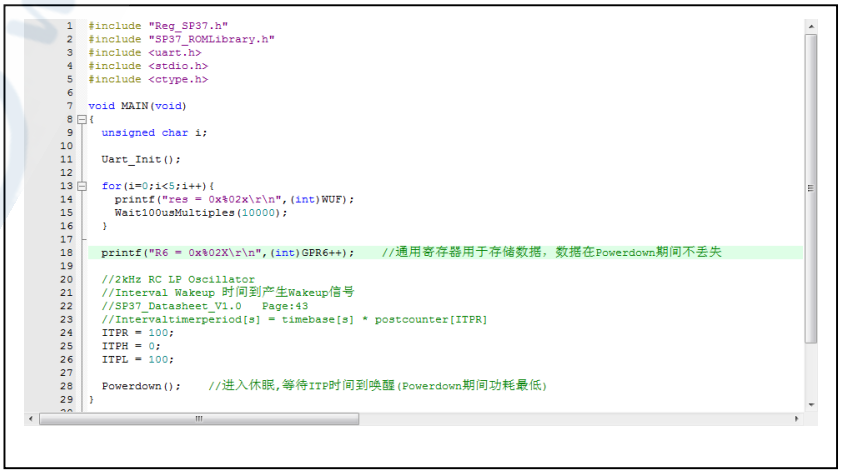


11.3 程序说明

本例子程序，用到了 Powerdown()函数，将整个系统关机。

在 Powerdown 与 Run 的整个过程中，通用寄存器的值不会被复位，其他寄存器的值会被复位。

内部定时器使用的是 2kHz 的 RC 震荡时钟源。所以本例子程序关机时间为 5 秒。



```
#include "Reg_SP37.h"
#include "SP37_ROMLibrary.h"
#include <uart.h>
#include <stdio.h>
#include <ctype.h>

void MAIN(void)
{
  unsigned char i;

  Uart_Init();

  for(i=0;i<5;i++){
    printf("res = 0x%02x\r\n",(int)WUF);
    Wait100usMultiples(10000);
  }

  printf("R6 = 0x%02X\r\n",(int)GPR6++);    //通用寄存器用于存储数据，数据在Powerdown期间不丢失

  //2kHz RC LP Oscillator
  //Interval Wakeup 时间到产生Wakeup信号
  //SP37_Datasheet_V1.0   Page:43
  //Intervaltimerperiod[s] = timebase[s] * postcounter[ITPR]
  ITPR = 100;
  ITPH = 0;
  ITPL = 100;

  Powerdown();    //进入休眠,等待ITP时间到唤醒(Powerdown期间功耗最低)
}
```

12.无线发射传感器数据(12. RF_ALL_V2)

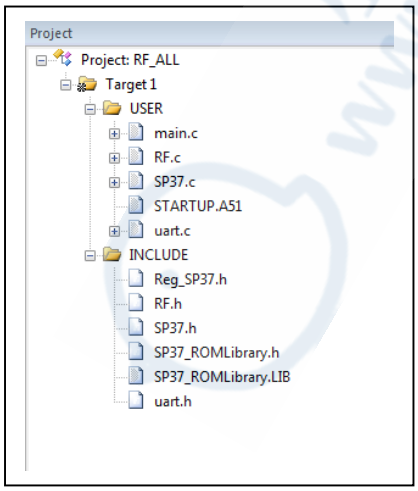
说明： 此例程通过 RF 发送：
“设备唯一序列号 + 轮胎位置识别码 + 气压 + 温度 + 重力加速度 + 电压值 + CRC” 14Byte 的数据
实际项目中根据需求可自行调整（一般需要结合耗电情况考虑）

与 10.RF_ALL 比较差别如下，主要如以下 3 点（可以对比两个工程的代码，进一步理解工作机制）

1. RF 传输波特率更改为 9600,原本为 2000 速率提高，时间就会相应缩短，功耗自然也会降低一点
2. RF 数据包总共发射 14Byte,原本 10Byte。
   2.1. 原本发射 2 个 Byte 的用户 ID，修改为 4 个 Byte 的芯片唯一序列号。以此能够更好区分每一颗发射器
   2.2. 原本发射 1 个 Byte 的压力信息，修改为了 2 个 Byte。之前发射 1 个 Byte，实际是通过”Scale_Pressure”这个函数进行转化。但是这个函数测量范围为 100~450Kpa，无法满足 900Kpa 的范围。如果产品的量程在 450Kpa 以下，推荐 1 个 Byte 就好。毕竟少发一个 Byte 可以省电的。
   当然 900Kpa 的客户也可以自己定一个转化关系，将他转化为 1 个 Byte 之后，进行无线传输
   温度也可以转化为 1 个 Byte。比如 “ -50℃ + 实际温度 ”，这样一个 Byte(0~255)可以涵盖-50℃~205℃的范围，基本可以满足胎压温度范围
   2.3. 添加发射轮胎位置识别码，可用于区分不同位置的轮胎。
3. 添加数据发射时，进行随机延时，再发射数据。(可避免 4 个轮胎之间的数据串扰)

由于波特率和数据总长度发生变化，接收端需要重新配置，才能正常收到信息。
具体配置教程详见 ” TDA5235 配置软件与资料\TDA5235 配置指导手册.pdf”，当然配套的接收端程序，我已经配置好了，可以直接使用

12.1 目录机构



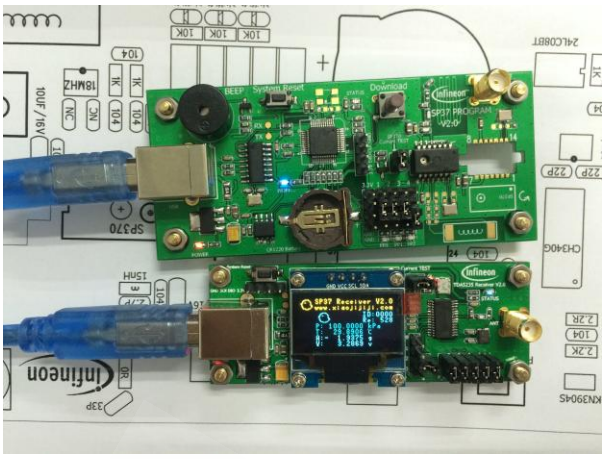
12.2 编译与结果

在 Object 文件下生成 RF_ALL_V2.hex （通过烧录器烧录，可以在接收端看到来自 SP370 RF 的数据)

接收端配套软件(以下软件都可配套):

SP37_RFData_To_OLED_WithCRC_V2

注: 烧录时 PP2 先断开与“USB 转串口工具”的连接，否则可能会导致下载失败（主要原因是“USB 转串口工具”给 PP2 供电，导致 SP370 无法进行完整的复位操作）



接收端显示画面，接收端需要下载 ”SP37_RFData_To_OLED_WithCRC_V2 “项目下的 hex 文件

12.3 程序说明

主要是构造好发射的数据，如右图。

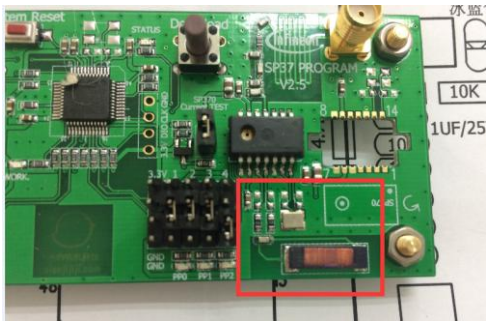
```
//原始数据
unsigned char idata RF_Array[]={
  0x00,                    //START of TABLE

  //以下配置发射数据包时 延时一个随机时间
  //随机时间来之SP37内部的随机数发生器
  0xDD,                    //Delay Pattern descriptor
  0,                       //DURATION = 0ms
  100,                     //RANDOM = 100,将会产生0-100ms的随机延时


  //以下配置数据包
  0x00,                    //TYPE for Manchester

  (2+1+RF_DATA_BYTES)*8,   //DATA_LANGTH = (WAKE UP + SYNS + DATA)*8bit

  0x00,
  0x00,                    //WAKE UP

  0x15,                    //SYNC

  0x00,                    //DATA(RF_DATA_BYTES 个 DATA) 这里默认14个数据
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
```

程序流程

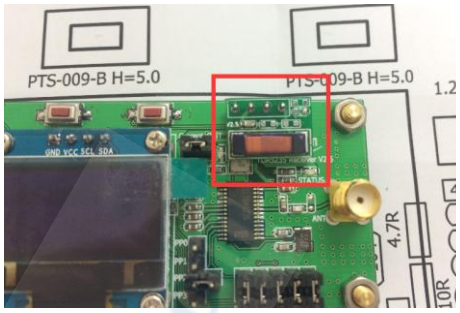
1. 配置 RF
   a) RFTX 寄存器相关配置
2. 配置晶振参数
   a) XTAL1 与 XTAL2
3. 开启晶振
   a) StartXtalOsc
4. VCO_Tuning（）
5. 发送数据包
   a) Send_RF_Telegram

13.低频唤醒(13.LF_WakeUp)

注：LF 功能，需要有硬件支持。请确认一下两块开发板上，是否都已经具备了”LF 相关的电子元件”
(若没有相关元件，可联系卖家，免费升级)
如下图：



烧录版



接收板

着重说明:

这里更正一下大部分人理解的 LF 唤醒功能：

可能大家都以为，SP370 在一般情况下，都是休眠的(省电)。遇到 LF 唤醒信号，从休眠中唤醒，然后发射数据包出来，提供接收器接收与处理。LF 是具有这样的功能，但是由于 LF 的通信(触发)距离只有 10cm 左右。在真正产品上，不可能通过 LF 来进行双向通信，或者通过 LF 来进行触发唤醒，请求数据。所以 SP370 还是主动唤醒（定时唤醒），检测情况，发射数据。

市面上 95%以上的内置型胎压产品，都不是通过 LF 来唤醒，从而请求数据包。相反 LF 功能主要在配对、激活、调换轮胎等操作时使用。也就是说，平时根本没有用到 LF 的功能。所以很多内置型的胎压产品也没有搭配 LF 相关的电路与电子元件，达到省空间省成本等目的。

具备 LF 功能，在硬件上需要搭配一个大电感，用于感应出电压。由于外置型胎压的空间限制，不能容下这颗电感。所以外置型一般不搭配 LF 的相关电路与电子元件。

LF 的主要功能体现:

1. 利于生产与配对。有了 LF 功能，发射端可以使用同一套程序，接收端也可以使用同一套程序。那么问题来了，如何区分前后左右轮胎以及不同车子的轮胎呢。一般有如下做法：

1.1 接收端通过按键或者其他方式进入”学习模式”:LF 唤醒胎压，胎压反馈信号(RF 信号出来)，信号包含 ID，此时接收端将 ID 保存，以便判断前后左右轮胎，或者过滤不同车子的轮胎。(前后左右逐一学习过去，即可配齐一套)

1.2 客户如果执行”轮胎调换”(一般前后轮胎磨损程度不一致，一段时间后可能需要将前后轮胎进行对换)的操作，可执行以上步骤重新学习一遍即可配套起来。同时如果客户某一只胎压损坏，只需购买一只胎压，重新学习一遍，即可配套)

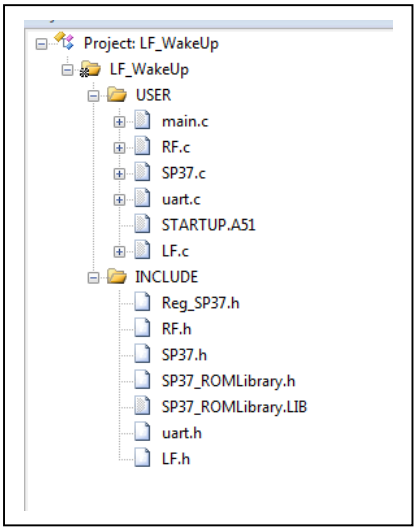
1.3 LF 还可以用于触发设备，检测设备是否正常（气压，温度、电池电压等状态）

以上是接收端主机进入学习模式,当然也可以让胎压(SP370),进入学习模式。

若没有 LF 功能:

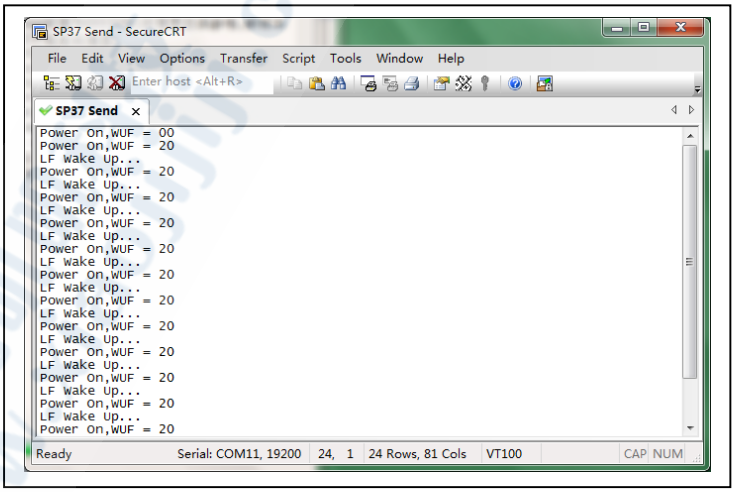
就需要提前烧录轮胎位置识别码，或者 ID 等信息。进行配套。(可能软件上不能共用一套)

13.1 目录机构



13.2 编译与结果

在 Object 文件下生成 LF_WakeUp.hex(通 过 烧录器烧录，可以在串口调试助手可看到 来自 SP370 的 Printf 信息，或者短接 PP2 的跳线帽，可以看到 LED 闪烁)



接收端需要下载如下项目的 Hex 文件

11.Send_125Khz_LF_Telegram

(接收端用来产生 LF 激励源)

测试时：两块开发板尽量靠近

13.3 程序说明

主要是配置过程。

此例程只要 LF 感应电压超过设定的阈值就会触发唤醒

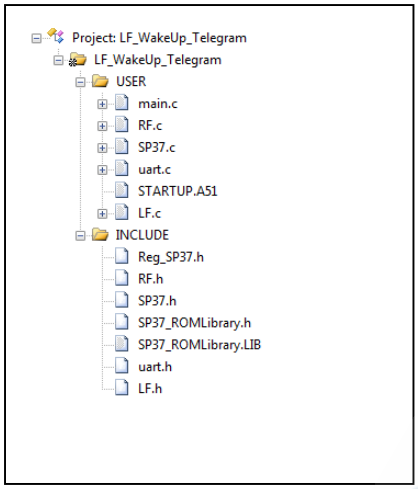
详情 Code 源码

```
}{
  LFCDM0 = 0x1C;            //Enable auto-calibration & freeze threshold

  LFRX0  = CBYTE[0x580F];   //这里选择唤醒灵敏度,即感应电压大小灵敏度
                            //务必从地址0x5810,0X580F,0X580E读入参数值(因为芯片出厂时,原厂保存了矫正值)
                            //0x5810: LF-sensitivity = 0.33 ... 3.35 mVpp(最高灵敏度)
                            //0x580F: LF-sensitivity = 2 ... 11 mVpp
                            //0x580E: LF-sensitivity = 10 ... 50 mVpp(最低灵敏度)
                            //此处我选择0x580F,
                            //灵敏度过高可能会导致，可能会造成误触发。比如还没使用LF激励，就被唤醒了
                            //灵敏度过低,需要较强的激励源，或者激励源需要 很靠近

  LFRX1  = 0x10;            //配置auto-calibration为侦测 载波的模式
  LFCDFLT = 0x00;           //过滤器配置
                            //关闭载波过滤功能，可以降低Powerdown时的 功耗
                            //此处的值也需要从地址0x580D,0x580C,0x580B读取
                            //0x580D: Detector Filter Time = 62...240  s
                            //0x580C: Detector Filter Time = 500 ... 800  s
                            //0x580B: Detector Filter Time = 800 ... 1150   s

  WUM &= ~(0x20);           //开启LF 唤醒中断(载波侦测)
  LFRXC = 0x04;             //LF Baseband disabled, LF-Receiver enabled
  ITPR  = 0x00;             //由于内部定时器唤醒，无法关闭。若使用默认值，0.5秒就会被唤醒一次
                            //即默认值将会干扰我们观测LF唤醒的状态
                            //此处修改为0x00,变成大约2min会被定时器唤醒一次
```

14.低频唤醒 Telegram 模式(14.LF_WakeUp_Telegram)

说明：此例程只有 LF 接收到 SYNC 信号并且 ID(0x1234)匹配之后,才会触发唤醒
也就是发射端 LF 出来的 ID，与 SP370 所配置的 ID 一致，才可以触发唤醒。
默认发射端 LF 出来的 ID 为 0x1234(当然这个 ID 客户完全可以自己定义)
详情源代码

14.1 目录机构



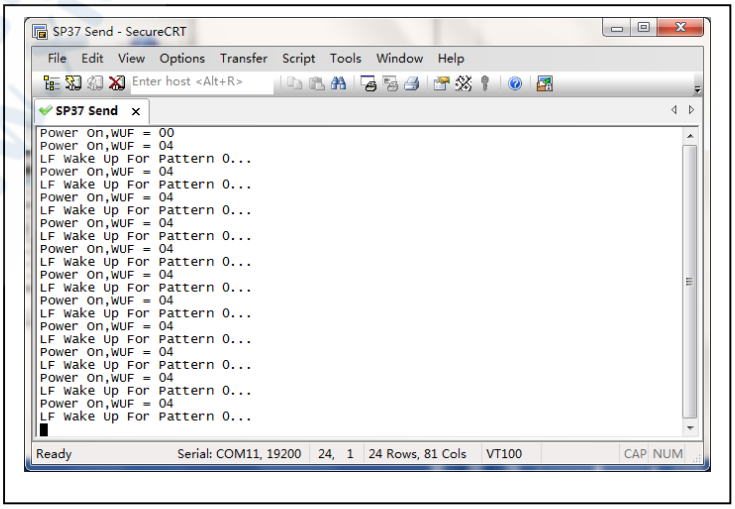
14.2 编译与结果

在 Object 文件下生成 LF_WakeUp_Telegram.hex (通过烧录器烧录，可以在串口调试助手可看到 来自 SP370 的 Printf 信息，或者短接 PP2 的跳线帽，可以看到 LED 闪烁)

接收端需要下载如下项目的 Hex 文件

11.Send_125Khz_LF_Telegram
(接收端用来产生 LF 激励源)

测试时：两块开发板尽量靠近



14.3 程序说明

主要是配置过程。

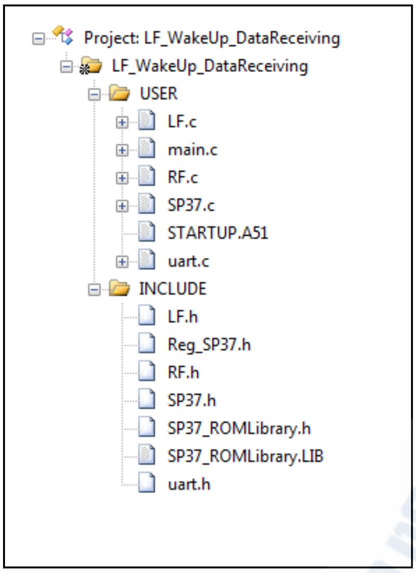
详情 Code 源码

```
void LF_Configure_Telegram(void)
{
  LFPCFG = 0x02;              //Use 16Bit pattern P0 for wakeup(16Bit ID匹配)
  LFP0H = 0x12;LFP0L = 0x34;//ID的高地位

  LFRX1  = 0x30;              //配置auto-calibration为侦测 报文的模式

  WUM &= ~(0x04);             //开启LF 唤醒中断 Pattern Match Wakeup
  LFRXC = 0x14;               //LF Baseband enabled, LF-Receiver enabled
  ITPR  = 0x00;               //由于内部定时器唤醒，无法关闭。若使用默认值，0.5秒就会被唤醒一次
                              //即默认值将会干扰我们观测LF唤醒的状态
                              //此处修改为0x00,变成大约2min会被定时器唤醒一次

  StartXtalOsc(40);           //Start RF quartz oscillator and wait 40x42.67µs
                              //开启外部晶振，实际目的是为了校准LF的波特率(为LFBaudrateCalibration函数服务)
  LFBaudrateCalibration(3906);//数据传输波特率3906Bit/s
  StopXtalOsc();              //关闭外部晶振
}
```

15.LF 唤醒，数据接收（15.LF_WakeUp_DataReceiving)

说明：此例程只有 LF 接收到 SYNC 信号并且 ID(0x1234)匹配之后,才会触发唤醒
唤醒之后进行数据的读取。（注:数据读取之前,不能耗费较长的时间。比如调用 delay，或者 printf 等比较耗费时间的函数）
详情源代码。

15.1 目录机构



15.2 编译与结果

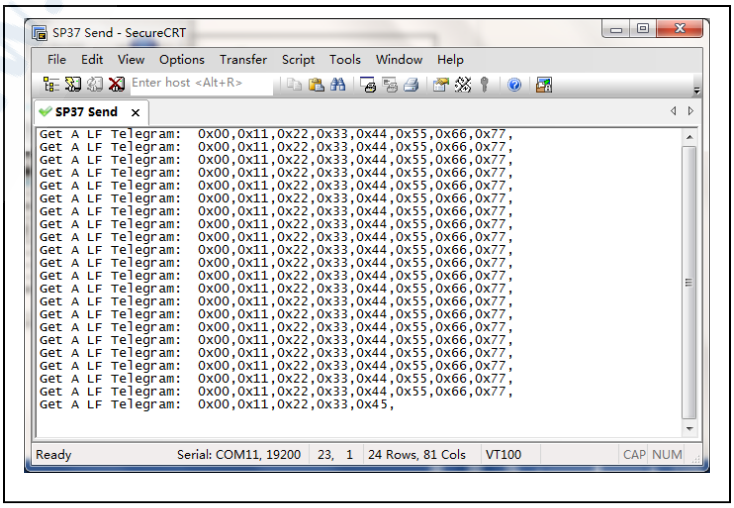
在 Object 文件下生成 LF_WakeUp_DataReceiving.hex.hex （通过烧录器烧录，可以在串口调试助手可看到 来自 SP370 的 Printf 信息，或者短接 PP2 的跳线帽，可以看到 LED 闪烁)

接收端需要下载如下项目的 Hex 文件

11.Send_125Khz_LF_Telegram

(接收端用来产生 LF 激励源)

测试时：两块开发板尽量靠近



15.3 程序说明

配置过程与 LF_WakeUp_Telegram 一致。

主要添加了 LF_DataReceive 函数，进行数据的接收

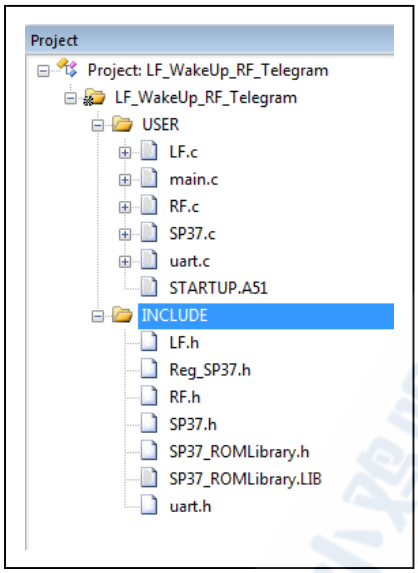
详情源代码

16.低频唤醒 Telegram 模式,并 RF 数据包(16.LF_WakeUp_RF_Telegram)

说明：此例程只有 LF 接收到 SYNC 信号并且 ID(0x1234)匹配之后,才会触发唤醒
也就是发射端 LF 出来的 ID，与 SP370 所配置的 ID 一致，才可以触发唤醒。
默认发射端 LF 出来的 ID 为 0x1234(当然这个 ID 客户完全可以自己定义)
唤醒之后采集传感器数据，并将数据 RF 出来。

综上本例程是"14.LF_WakeUp_Telegram"与" 12.RF_ALL_V2"的结合体
详情源代码

16.1 目录机构



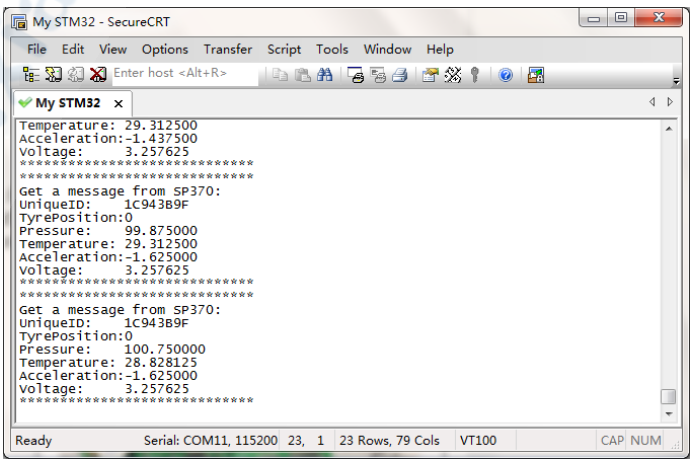
16.2 编译与结果

在 Object 文件下生成 LF_WakeUp_RF_Telegram.hex (通过烧录器烧录，当接收板靠近发射板的时候，发射板被触发唤醒，并 RF 数据。此时可以看到接收板屏幕更新来自 SP370 的传感器信息)

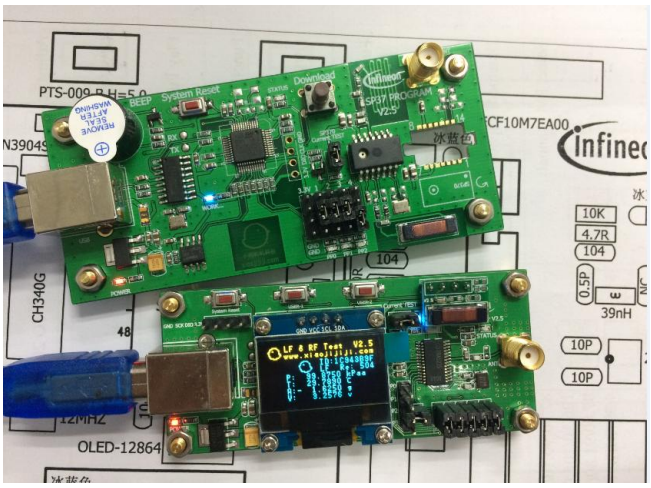
接收端需要下载如下项目的 Hex 文件

12.SendLFAndReceiveRF
(接收端用来产生 LF 激励源,
并在屏幕显示 RF 信息)

测试时：两块开发板尽量靠近



来自接收板的 Printf 信息



实物现象

16.3 程序说明

详情 Code 源码

结束语：

这里已经介绍了 **SP370** 几乎所有功能以及例子程序了，具体的当然请以源码为主。源码里面也相应做了很多注解，相信您可以很快的熟悉这一套。实际产品其实也是这些例子代码的组合体。当然不同的开发者，有着不同的想法，必然会有意想不到的结果。再次祝愿您的产品可以尽快量产，也祝愿我的这套开发套件可以给您带来更多的产值。

再次感谢您选择与信任小鸡叽叽科技!!!

By Deathgod
QQ : 813227539