实

验

报

告

之

串口控制LED灯 + OLED灯光师

实

验

作者：刘俊杰

班级：物网171

日期：20190622

指导老师：李军老师

**目录**

[1、实验目的 3](#_Toc477335449)

[2 、实验设备 3](#_Toc477335450)

[3 、实验内容 3](#_Toc477335451)

[4 、实验原理 3](#_Toc477335452)

[5、实验相应寄存器 4](#_Toc477335453)

[6、实验电路图 5](#_Toc477335454)

[7、实验程序实现 6](#_Toc477335455)

[8 、实验步骤和结果 8](#_Toc477335456)

[9 、实验感想 8](#_Toc477335457)

# 1、实验目的

掌握 CC2530 的 I/O 控制寄存器的配置，中断的配置和使用。

学会使用 CC2530 的 I/0 来控制外设。

# 2 、实验设备

cc2530 Zigbee 节点 1 个，USB 连接线 1 根，串口线一根。

# 3 、实验内容

1）通过串口发送命令控制LED亮灭，同时通过串口回传LED状态信息。

2）OLED屏幕显示开机页面，包含作者信息及欢迎标题。四个LED灯默认关闭状态。

3）通过按键（button2）控制LED亮灭，同时通过串口回传LED状态以及哪个按键被按下信息。通过按键（button2），按下一次时，实现跑马灯，同时屏幕显示相关内容，通过串口回传LED状态信息。按下第二次时，实现四个灯同时闪烁，同时屏幕显示相关内容，通过串口回传LED状态信息。按下第三次时，实现跑马灯。以此循环。

4）分秒计时器。

# 4 、实验原理

CC2530有21 个数字I/O引脚，可以配置为通用数字I/O或外设 I/O信号，配置为连接到 ADC、定时器或 USART 外设。这些 I/O 口的用途可以通过一系列寄存器配置，由用户软件加以实现**。**

I/O 端口具备如下重要特性：

 21 个数字 I/O 引脚

 可以配置为通用 I/O 或外部设备 I/O

 输入口具备上拉或下拉能力

 具有外部中断能力。

21 个 I/O 引脚都可以用作于外部中断源输入口。因此如果需要外部设备可以产生中断。外部中断功能也可以从睡眠模式唤醒设备。

用作通用 I/O 时，引脚可以组成 3 个 8 位端口，端口 0、端口 1和端口2，表示为 P0、P1和 P2。其中，P0 和 P1 是完全的 8 位端口，而 P2 仅有 5 位可用。所有的端口均可以通过 SFR 寄存器 P0、P1和 P2位寻址和字节寻址。每个端口引脚都可以单独设置为通用 I/O 或外部设备 I/O。

除了两个高驱动输出口 P1.0 和 P1.1 各具备 20mA 的输出驱动能力之外，所有的输出均具备 4mA 的驱动能力。

寄存器 PxSEL，其中 x 为端口的标号 0~2，用来设置端口的每个引脚为通用 I/O 或者是外部设备 I/O 信号。作为缺省的情况，每当复位之后，所有数字输入/输出引脚都设置为通用输入引脚。所以当x为0和1时，则P0SEL和P1SEL作为通用I/O 或者是外部设备 I/O 信号。

在任何时候，要改变一个端口引脚的方向，就使用寄存器 PxDIR 来设置每个端口引脚为输入或输出。因此只要设置 PxDIR 中的指定位为 1和0，其对应的引脚口就被设置为输出了。

当读取端口寄存器 P0、P1 和 P2 的值，不管引脚配置如何，输入引脚上的逻辑值都被返回。这在执行读-修改-写指令期间不适用。读-修改-写指令是：ANL，ORL，XRL，JBC，CPL，INC，DEC，DJNZ，MOV，CLR 和 SETB。在一个端口寄存器上操作，以下是正确的：当目标是端口寄存器 P0、P1 或P2 中一个独立的位，寄存器的值，而不是引脚上的值，被读取、修改并写回端口寄存器。

用作输入时，通用 I/O 端口引脚可以设置为上拉、下拉或三态操作模式。作为缺省的情况，复位之后，所有的端口均设置为带上拉的输入。要取消输入的上拉或下拉功能，就要将 PxINP 中的对应位设置为 1。I/O 端口引脚 P1.0和 P1.1 没有上拉/下拉功能。注意配置为外设 I/O 信号的引脚没有上拉/下拉功能，即使外设功能是一个输入。

在电源模式 PM1、PM2 和 PM3 下 I/O 引脚保留当进入 PM1/PM2/PM3时设置的 I/O 模式和输出值（如果可用的话）。

CC2530有两个串行通信接口USART0和USART1，它们能够分别运行于异步UART模式或者同步SPI模式。两个USART接口具有相同的功能，通过PERCFG寄存器可以设置两个USART接口对应外部I/O引脚的映射关系：

        位置1：RX0 --- P0\_2 TX0 --- P0\_3

             RX1 --- P0\_5 TX1 --- P0\_4

        位置2：RX0 --- P1\_4  TX0 --- P1\_5

RX1 --- P1\_7  TX1 --- P1\_6

        对每个USART串口通信编程，本质是设置相关的5个寄存器：

        <1> UxCSR：   USARTx的控制和状态寄存器。

        <2> UxUCR：  USARTx的UART控制寄存器。

        <3> UxGCR：  USARTx的通用控制寄存器。

        <4> UxDBUF：USARTx的接收/发送数据缓冲寄存器。

        <4> UxBAUD：USARTx的波特率控制寄存器。

中断函数的两个重要概念：

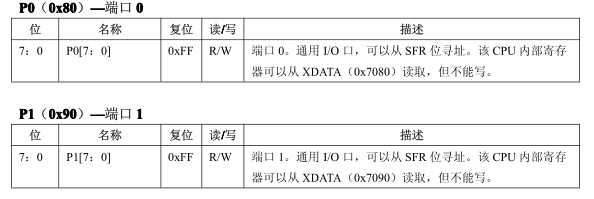
<1> **中断服务函数**： 内核响应中断后执行的相应处理程序。

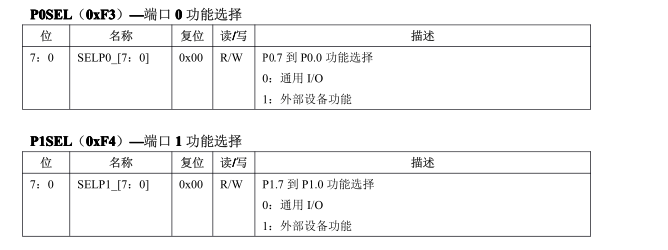
   <2>**中断向量**：中断服务程序的入口地址。每个中断源都对应一个固定的入口地址。当内核响应中断请求时，就会暂停当前的程序执行，然后跳转到该入口地址执行代码。

CC2530中断系统



# 5、实验相应寄存器

****

****

# 6、实验电路图

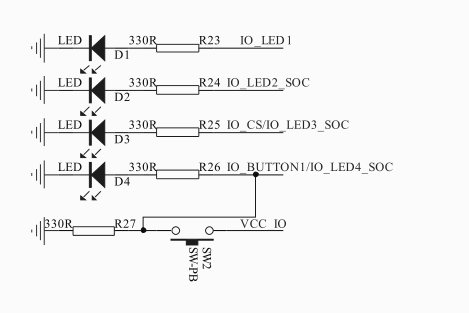
****

图1：LED电路图

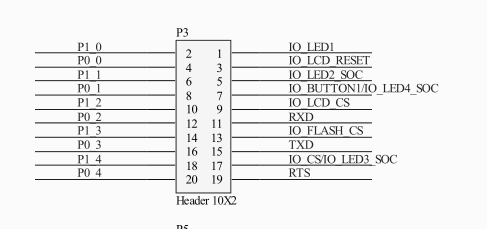
****

图2：LED对应的GPIO表

**LED 和 GPIO 对应表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LED1** | **LED2** | **LED3** | **LED4** |
| **P1\_0** | **P1\_1** | **P1\_4** | **P0\_1** |

# 实验程序实现

1. main.c文件

#include "ioCC2530.h"

#include "LCD.h"

#include "UART.h"

#include "timer.h"

#include "LED.h"

#include "Date.h"

int key\_num=0;

void key\_1();

void key\_2();

void main(void)

{

//外部中断

Init32M();

InitUART();

//定时器1中断

Init\_Timer1();

//按钮中断

P0IE=1;

P0IEN |= 0X02;

//按钮初始化

button\_Init();

//初始化LED

P1DIR|=0X13;

All\_OFF();

//zijian(); //自检函数，想写的时候再说吧

//初始化液晶屏

LCD\_Init();

//清屏

LCD\_Clear();

//显示信息

LCD\_WriteString(0,1,"+----------------+");

LCD\_WriteString(0,2,"| Welcome |");

LCD\_WriteString(0,3,"| |");

LCD\_WriteString(0,4,"| Writer:LJJ |");

LCD\_WriteString(0,5,"| |");

LCD\_WriteString(0,6,"|2019.06.01 00:00|");

LCD\_WriteString(0,7,"+----------------+");

UART0SendByte('\n');

while(1)

{

if(key\_num==0)

{

}

else if(key\_num%2==0)

{

button\_LCD\_Num(key\_num);

key\_1();

}

else if(key\_num%2==1)

{

button\_LCD\_Num(key\_num);

key\_2();

}

}

}

//#pragma optimize=none//优化函数 放在被优化函数之前

#pragma vector=P0INT\_VECTOR

\_\_interrupt void love()

{

if(P0IFG & 0X02)//0000 0010 0000 0000

{//P0组第2位P0\_1有没有引发外部中断 引发了外部中断 P0IFG对应的位就会被置1

delay();

if(1==P0\_1)

{//如果按键确实被按下 那么确实是连接在P0\_1的按键触发了外部中断

key\_num++;

UART0SendString(LED\_num\_judge(7));

}

}

P0\_1=0;

P0IFG=0;//P0中断状态标志

P0IF=0;//端口0中断标志 1表示中断挂起 0表示中断未挂起

}

void key\_1()

{

while(1)

{

running2();

delay();

if(key\_num%2!=0)

{

P0\_1=0;

break;

}

}

}

void key\_2()

{

while(1)

{

running();

delay();

if(key\_num%2!=1)

{

P0\_1=0;

break;

}

}

}

1. LCD.c文件

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 文件名称：hal\_led.c

\* 功 能：OLED驱动

\* 使用硬件SPI总线驱动128\*64点阵OLED液晶

\* 硬件连接：液晶模块与CC2530的硬件连接关系如下：

\* 液晶模块 CC2530

\* CS P1.2

\* SDA P1.6

\* SCK P1.5

\* RESET P0.0

\* D/C# P2.2

\* 字库芯片

\* CS# P2.1

\* SCLK P1.5

\* SI P1.6

\* SO P1.7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 包含头文件 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "ioCC2530.h"

//#include "hal\_board\_cfg.h"

#include "LCD.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 本地变量 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 X\_Witch = 6;

uint8 Y\_Witch = 1;

uint8 X\_Witch\_cn = 16;

uint8 Y\_Witch\_cn = 16;

uint8 Dis\_Zero = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define FUNCTION\_SET(options,OLED\_DC) halOLED\_control(options,OLED\_DC)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_TimeDelay

\* 功 能：延时函数

\* 入口参数：Timers 延时时间参数

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_TimeDelay(uint16 Timers)

{

uint16 i;

while(Timers)

{

Timers--;

for(i = 0; i < 100; i++)NOP();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_SPISSSet

\* 功 能：置SS线状态

\* 入口参数：Status SS线状态

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_SPISSSet(uint8 Status)

{

NOP();NOP();NOP();

if(Status)

H\_LCD\_CSn();

else

L\_LCD\_CSn();

NOP();NOP();NOP();NOP();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_SPI\_DC

\* 功 能：置DATA/COMMAND线状态

\* 入口参数：Status SS线状态

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_SPI\_DC(uint8 Status)

{

NOP();NOP();NOP();

if(Status)

LCD\_DATA();

else

LCD\_COMMAND();

NOP();NOP();NOP();NOP();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：GT20L\_SPI

\* 功 能：片选GT20

\* 入口参数：Status SS线状态

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GT20L\_SPI(uint8 Status)

{

NOP();NOP();NOP();

if(Status)

L\_GT20\_CSn();

else

H\_GT20\_CSn();

NOP();NOP();NOP();NOP();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_SPISendData

\* 功 能：通过串行SPI口输送一个byte的数据置模组

\* 入口参数：Data 要传送的数据

\* 出口参数：temp

\* 返 回 值：SPI接到数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 LCD\_SPISendData(uint8 Data)

{

uint8 i=0;

uint8 temp=0;

for(i = 0; i < 8; i++)

{

NOP();NOP();NOP();NOP();NOP();NOP(); // 适当插入一些空操作以保证SPI时钟速度小于2MHz

L\_LCD\_SCK();

if(Data&0x80)

H\_LCD\_SDA();

else

L\_LCD\_SDA();

NOP();NOP();NOP();NOP();NOP(); //适当插入一些空操作以保证SPI时钟速度小于2MHz

NOP();NOP();NOP();NOP();NOP();

H\_LCD\_SCK();

if(GT20\_MISO==1)

{

temp=temp | BV(7-i);

}

NOP();NOP();NOP();NOP();NOP();

Data = Data << 1; //数据左移一位

}

NOP();NOP();NOP();NOP();NOP();NOP();NOP();NOP();

return temp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：halOLED\_control

\* 功 能：通过串行SPI口输送一个byte的数据置模组

\* 入口参数：数据cmd, 命令控制 OLED\_DC

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void halOLED\_control(uint8 cmd, uint8 OLED\_DC)

{

LCD\_SPISSSet(0); //SS置低电平

if(OLED\_DC==COMMAND)

{

LCD\_SPI\_DC(0);

}

else

{

LCD\_SPI\_DC(1);

}

LCD\_SPISendData(cmd); //送指令0x80

LCD\_SPI\_DC(1);

LCD\_SPISSSet(1); //完成操作置SS高电平

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：HalLed\_SET\_XY

\* 功 能：设置 x y 地址

\* 入口参数：x x地址

\* y y地址

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HalLed\_SET\_XY(unsigned char x,unsigned char y)

{

FUNCTION\_SET(0x22,COMMAND); //传送指令0x22

FUNCTION\_SET(y,COMMAND); //要显示字符的左上角的Y轴位置

FUNCTION\_SET(Dis\_Y\_MAX ,COMMAND); //要显示字符的左上角的Y轴位置

FUNCTION\_SET(0x21,COMMAND); //传送指令0x21

FUNCTION\_SET(x,COMMAND); //要显示字符的左上角的X轴位置

FUNCTION\_SET(Dis\_X\_MAX ,COMMAND); //要显示字符的左上角的Y轴位置

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：HalGT20L\_TX\_CMD

\* 功 能：GT20L硬件写入控制

\* 入口参数：cmd 写入数据

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int8 HalGT20L\_TX\_CMD(uint8 cmd )

{

return LCD\_SPISendData(cmd);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：halGT20L\_HRD\_Font

\* 功 能：GT23L读取字符码值函数

\* 入口参数： Dst 字库地址, no\_bytes 一个字符需要的码值 字节数 , \*buffer 码值

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void halGT20L\_HRD\_Font(uint16 \* Dst, uint8 no\_bytes,uint8 \*buffer)

{

unsigned char i = 0;

GT20L\_SPI(1); // enable device

HalGT20L\_TX\_CMD(0x0B); // read command

HalGT20L\_TX\_CMD(Dst[1]); // send 3 address bytes

HalGT20L\_TX\_CMD(((Dst[0]) >> 8));

HalGT20L\_TX\_CMD(Dst[0] & 0xFF);

HalGT20L\_TX\_CMD(0xFF); //dummy byte

for (i = 0; i < no\_bytes; i++) // read until no\_bytes is reached

{

buffer[i] =HalGT20L\_TX\_CMD(0xFF); // receive byte and store at address 80H - FFH

}

GT20L\_SPI(0); // disable device

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：halASCII\_Searh\_ADDR

\* 功 能：ASCII码字符内码

\* 入口参数：uint8 ASCIICode,uint16 \* CODE\_ADDR

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void halASCII\_Searh\_ADDR(uint8 ASCIICode,uint16 \* CODE\_ADDR)

{

uint16 ASCII\_Code\_Addr = 0;

// ASCII\_Code\_Addr = ((((uint16)ASCIICode) - (uint16)0x20) \* 8) +0x66C0;//切换字体

ASCII\_Code\_Addr = ((((uint16)ASCIICode) - (uint16)0x20) \* 8) +0xbfC0;

CODE\_ADDR[0]=ASCII\_Code\_Addr ;

CODE\_ADDR[1]=0x03;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_WriteChar

\* 功 能：显示ASCII码字符

\* 入口参数：x 要显示的字符x地址

\* y 要显示的字符y地址

\* a 要显示的ASCII码字符值

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_WriteChar(uint8 x, uint8 y, const char a)

{

uint8 i=0;

uint16 ASCII\_CODE\_ADDR[2];

uint8 font\_buffer[8];

halASCII\_Searh\_ADDR(a ,ASCII\_CODE\_ADDR);

halGT20L\_HRD\_Font(ASCII\_CODE\_ADDR,8,font\_buffer);

HalLed\_SET\_XY(x,y);

for(i=0;i<8;i++)

{

FUNCTION\_SET(font\_buffer[i],DATA);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_WriteString

\* 功 能：在x、y为起始坐标处写入一串字符

\* 入口参数：x X轴坐标，取值范围：0 - 127

\* y Y轴坐标，取值范围0-63

\* p 要显示的字符串

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_WriteString(uint8 x, uint8 y, const char \*p)

{

while(\*p != 0)

{

LCD\_WriteChar(x, y, \*p);

x += 7;

if(x > Dis\_X\_MAX)

{

x = Dis\_Zero;

if((Dis\_Y\_MAX - y) < Y\_Witch) break;

else y += 1;

}

p+=1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_SetBackLight

\* 功 能：设置背光亮度等级

\* 入口参数：Deg 亮度等级0~127

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_SetBackLight(uint8 Deg)

{

GT20L\_SPI(1); // enable device

HalGT20L\_TX\_CMD(0x81);

LCD\_SPISendData(Deg); //背光设置亮度值

GT20L\_SPI(0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_Clear

\* 功 能：LCD清屏

\* 入口参数：无

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_Clear(void)

{

uint8 i,j;

//清屏操作

FUNCTION\_SET(0x22,COMMAND);

FUNCTION\_SET(0,COMMAND);

FUNCTION\_SET(7,COMMAND);

FUNCTION\_SET(0x21,COMMAND);

FUNCTION\_SET(0,COMMAND);

FUNCTION\_SET(127,COMMAND);

for(i=0;i<8;i++)

{

for(j=0;j<128;j++)

FUNCTION\_SET(0x00,DATA);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_Init

\* 功 能：LCD初始化

\* 入口参数：无

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_Init(void)

{

P0DIR |= (0x01<<0); // P0.0为输出

P1DIR |= (0x01<<2); // P1.2为输出

P1DIR |= (0x01<<5); // P1.5为输出

P1DIR |= (0x01<<6); // P1.6为输出

P1DIR &= (0X7F); // P1.7为输入

P2DIR |= (0x01<<1); // P2.1为输出

P2DIR |= (0x01<<2); // P2.2为输出

/\* 复位LCD \*/

L\_LCD\_RESETn();

LCD\_TimeDelay(3000);

H\_LCD\_RESETn();

LCD\_TimeDelay(3000);

//Charge Pump Setting

FUNCTION\_SET(0x8d,COMMAND);

FUNCTION\_SET(0x14,COMMAND);

FUNCTION\_SET(0xaf,COMMAND);

//Set Display Clock

FUNCTION\_SET(0xD5,COMMAND);

FUNCTION\_SET(0xF0,COMMAND);

//Set Pre-charge Period

FUNCTION\_SET(0xD9,COMMAND);

FUNCTION\_SET(0x11,COMMAND);

//Set VCOMH Deselect Level

FUNCTION\_SET(0xDb,COMMAND);

FUNCTION\_SET(0x0,COMMAND);

//Set Norma Display

FUNCTION\_SET(0xa6,COMMAND);

//Entire Display ON

//FUNCTION\_SET(0xa5,COMMAND);

//Set Contrast Control

FUNCTION\_SET(0x81,COMMAND);

FUNCTION\_SET(0xff,COMMAND); //1-256

//Set Segment Re-map

FUNCTION\_SET(0xa1,COMMAND);

//Set COM Output Scan Direction

FUNCTION\_SET(0xc8,COMMAND);

//Set Memory Addressing Mode

FUNCTION\_SET(0x20,COMMAND);

FUNCTION\_SET(0x00,COMMAND);

LCD\_SetBackLight(100); // 调节背光亮度

}

1. LCD.h文件

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 文件名称：LCD.h

\* 功 能：LCD驱动

\* 驱动128\*64点阵图形液晶（MzLH04-12864）

\* 硬件连接：液晶模块与CC2530的硬件连接关系如下：

\* 液晶模块 CC2530

\*

\* CS(PIN2) P1.2

\* SDA(PIN3) P1.6

\* SCK(PIN5) P1.5

\* RESET(PIN6) P0.0

\* VDD(PIN1) x

\* NC(PIN4) x

\* VSS(PIN7) x

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef \_LCD\_H\_

#define \_LCD\_H\_

//防止头文件重名

//宏定义

#ifndef BV

#define BV(n) (1 << (n))

#endif

/\* 类型定义 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef signed char int8;

typedef unsigned char uint8;

typedef signed short int16;

typedef unsigned short uint16;

typedef signed long int32;

typedef unsigned long uint32;

typedef unsigned char bool;

typedef uint8 halDataAlign\_t;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 相关引脚定义 \*/

/\*===================================================\*/

#define LCD\_SDA P1\_6

#define LCD\_SCK P1\_5

#define LCD\_CSn P1\_2

#define LCD\_RESETn P0\_0

#define LCD\_DC P2\_2

#define GT20\_MISO P1\_7

#define GT20\_CS P2\_1

/\*===================================================\*/

/\* 相关引脚输出电平定义 \*/

/\*===================================================\*/

#define H\_LCD\_SCK() LCD\_SCK = 1

#define L\_LCD\_SCK() LCD\_SCK = 0

#define H\_LCD\_SDA() LCD\_SDA = 1

#define L\_LCD\_SDA() LCD\_SDA = 0

#define H\_LCD\_CSn() LCD\_CSn = 1

#define L\_LCD\_CSn() LCD\_CSn = 0

#define H\_LCD\_RESETn() LCD\_RESETn = 1

#define L\_LCD\_RESETn() LCD\_RESETn = 0

#define LCD\_DATA() LCD\_DC = 1

#define LCD\_COMMAND() LCD\_DC = 0

#define H\_GT20\_CSn() GT20\_CS = 1

#define L\_GT20\_CSn() GT20\_CS = 0

/\*===================================================\*/

#define NOP() asm("nop")

//屏幕128\*64 可以分成X和Y轴

#define Dis\_X\_MAX 127

#define Dis\_Y\_MAX 7

#define COMMAND 0

#define DATA 1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_Init

\* 功 能：LCD初始化

\* 入口参数：无

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern void LCD\_Init(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_Clear

\* 功 能：LCD清屏

\* 入口参数：无

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern void LCD\_Clear(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_WriteString

\* 功 能：在x、y为起始坐标处写入一串字符

\* 入口参数：x X轴坐标，取值范围：0 - 127

\* y Y轴坐标，取值范围0-63

\* p 要显示的字符串

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern void LCD\_WriteString(uint8 x, uint8 y, const char \*p);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：LCD\_WriteChar

\* 功 能：显示ASCII码字符

\* 入口参数：x 要显示的字符x地址

\* y 要显示的字符y地址

\* a 要显示的ASCII码字符值

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern void LCD\_WriteChar(uint8 x, uint8 y, const char a);

#endif /\* \_LCD\_H\_ \*/

1. UART.c文件

#include "ioCC2530.h"

#include "UART.h"

#include "LCD.h"

#include "LED.h"

char ch;

char buff[20];

/\*

1、指定串口的IO位置；

2、相应IO配置成偏上外设功能；

3、8个数据位、1个停止位、无流控、无校验确立。（我们先最简单的通信）

传送一个UART字节包含一个起始位，8个数为，1个作为可选的第9位数据或校验位，

再加上一个或者2个停止位

4、波特率；（9600或者115200）

5、开CPU中断、对应串口接收中断；

\*/

//串口初始化

void InitUART(void)

{

//1、指定串口的IO位置；P0\_2 3 UART0 第一个位置 把PERCFG的第0位置0

PERCFG &= 0xFE;//1111 1110 选中串口0的备用位置1

//2、相应IO配置成偏上外设功能；

P0SEL |= 0x0C;//0000 1100 P0\_2 P0\_3置1 为偏上外设功能

//3、8个数据位、1个停止位、无流控、无校验确立。（我们先最简单的通信）

U0CSR |= 0xC0;//1100 0000 选择 UART模式 接收器使能置1

//4、波特率；（9600或者115200）

U0GCR=8;

U0BAUD=59;//设置成9600

//5、开CPU中断、对应串口接收中断；

EA=1;//开启总中断

URX0IE=1;//USRAT0中断使能

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：UART0SendByte

\* 功 能：UART0发送一个字节

\* 入口参数：c

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART0SendByte(unsigned char c)

{

U0DBUF = c; // 将要发送的1字节数据写入U0DBUF(串口 0 收发缓冲器)

while (!UTX0IF); // 等待TX中断标志，即U0DBUF就绪

UTX0IF = 0; // 清零TX中断标志

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：UART0SendString

\* 功 能：UART0发送一个字符串

\* 入口参数：无

\* 出口参数：无

\* 返 回 值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART0SendString(char \*str)

{

while(1)

{

if(\*str == '\0') break; // 遇到结束符，退出

UART0SendByte(\*str++); // 发送一字节

}

}

#pragma optimize=none//优化函数 放在被优化函数之前

#pragma vector=URX0\_VECTOR

\_\_interrupt void MYuart()

{

//串口0来数据的标志位，硬件会置1，我们软件要清0

URX0IF=0;

//数据的交换

ch=U0DBUF;//从接受寄存器里取字节存入到变量ch里

if(ch=='0')

{

P1\_0=0;

P1\_1=0;

P1\_4=0;

UART0SendString(LED\_num\_judge(0));

UART0SendString(LED\_num\_judge(5));

}

if(ch=='1')

{

P1\_0=1;

P1\_1=0;

P1\_4=0;

UART0SendString(LED\_num\_judge(1));

UART0SendString(LED\_num\_judge(5));

}

if(ch=='2')

{

P1\_0=0;

P1\_1=1;

P1\_4=0;

UART0SendString(LED\_num\_judge(2));

UART0SendString(LED\_num\_judge(5));

}

if(ch=='3')

{

P1\_0=0;

P1\_1=0;

P1\_4=1;

UART0SendString(LED\_num\_judge(3));

UART0SendString(LED\_num\_judge(5));

}

if(ch=='4')

{

P1\_0=1;

P1\_1=1;

P1\_4=1;

UART0SendString(LED\_num\_judge(4));

UART0SendString(LED\_num\_judge(5));

}

if(ch=='T')

{

UART0SendString(LED\_num\_judge(6));

UART0SendString(LED\_num\_judge(5));

}

U0DBUF=ch;//把变量ch里的值赋给串口0发送寄存器

\*buff = U0DBUF;

//判断发送有没有完成，判断寄存器的位UTX0IF有没有被置1

while(0==UTX0IF);

//手动置0

UTX0IF=0;

UART0SendByte('\n');

}

1. UART.h文件

#ifndef \_UART\_H\_

#define \_UART\_H\_

#include "ioCC2530.h"

void InitUART(void); /\* 串口初始化 \*/

void UART0SendByte(unsigned char c); /\* UART0发送一个字节 \*/

void UART0SendString(char \*str); /\* UART0发送一个字符串 \*/

\_\_interrupt void MYuart(); /\* 串口中断 \*/

#endif /\* \_UART\_H\_ \*/

1. timer.c文件

#include <ioCC2530.h>

#include "timer.h"

//延时程序 63us

void delayus(void)//因为我们CC2530上电默认昿6M高频 机器周期大概昿us

{

char k=63;

while(k--);

}

//初始化32M外部晶振

void Init32M()

{

//1、让2个高频时钟源起振；

SLEEPCMD &= 0xFB;//1111 1011 开启两个高频时钟源

//2、等待目标时钟源振荡稳定；

while(0==(SLEEPSTA & 0x40));//0100 0000 等待32M稳定下来 通过查询SLEEPSTA寄存器的第6位是否为0

//32M晶振是否稳定下来 0不稳定 1稳定

//3、延时一小段时63us；

delayus();

//4、不分频输出；

CLKCONCMD &= 0xF8;//1111 1000

//5、选中目标高频时钟源作为系统主时钟；32M

CLKCONCMD &= 0xBF;//1011 1111 ~0x40

//6、确认一下当前工作的系统时钟是不是所选的高频时钟；

while(CLKCONSTA & 0x40);//0100 0000 //查询CLKCONSTA第六位是否为0 为0 的时候表达式为假 已经选择32M作为系统主时钟

}

//ѓʱ

void delay()

{

int i,j;

for(i=0;i<1000;i++)

for(j=0;j<100;j++);

}

1. timer.h文件

#ifndef \_TIMER\_H\_

#define \_TIMER\_H\_

#include "ioCC2530.h"

void delayus(void); /\* 延时程序 63us 8 \*/

void Init32M(void); /\* 初始匿2M外部晶振 \*/

void delay(void);

#endif /\* \_TIMER\_H\_ \*/

1. LED.c文件

#include <ioCC2530.h>

#include "timer.h"

#include "LCD.h"

#include "UART.h"

char \*LED\_num\_judge(int num)

{

switch(num)

{

case 0: return "[D1:0 D2:0 D3:0]";

case 1: return "[D1:1 D2:0 D3:0]";

case 2: return "[D1:0 D2:1 D3:0]";

case 3: return "[D1:0 D2:0 D3:1]";

case 4: return "[D1:1 D2:1 D3:1]";

case 5: return "Input: ";

case 6: return "\nEnter test2 mode: ";

case 7: return "\nButton2 Press down:\n";

}

return "\nERROR : LED\_num\_judge Functions do not accept 0-4 \n";

}

void running\_LCD\_WriteChar(char a,char b,char c)

{

LCD\_WriteChar(105,3,a);

LCD\_WriteChar(105,4,b);

LCD\_WriteChar(105,5,c);

}

void button\_LCD()

{

LCD\_WriteString(0,2,"| ButtonNum : 1 |");

LCD\_WriteString(0,3,"| LED P1\_0 : 0 |");

LCD\_WriteString(0,4,"| LED P1\_1 : 0 |");

LCD\_WriteString(0,5,"| LED P1\_4 : 0 |");

}

void button\_LCD\_Num(int a)

{

if(a==1)

{

button\_LCD();

}

else

{

LCD\_WriteChar(98,2,a+48);

}

}

void running()

{

running\_LCD\_WriteChar('1','0','0');

UART0SendString(LED\_num\_judge(1));

P1\_0=1;

P1\_1=0;

P1\_4=0;

delay();

delay();

running\_LCD\_WriteChar('0','1','0');

UART0SendString(LED\_num\_judge(2));

P1\_0=0;

P1\_1=1;

P1\_4=0;

delay();

delay();

running\_LCD\_WriteChar('0','0','1');

UART0SendString(LED\_num\_judge(3));

P1\_0=0;

P1\_1=0;

P1\_4=1;

delay();

UART0SendByte('\n');

delay();

}

void running2()

{

running\_LCD\_WriteChar('0','0','0');

UART0SendString(LED\_num\_judge(0));

P1\_0=0;

P1\_1=0;

P1\_4=0;

delay();

delay();

running\_LCD\_WriteChar('1','1','1');

UART0SendString(LED\_num\_judge(4));

P1\_0=1;

P1\_1=1;

P1\_4=1;

delay();

UART0SendByte('\n');

delay();

}

void zijian()

{

P1\_0=0;

P1\_1=0;

P1\_4=0;

delay();

delay();

P1\_0=1;

P1\_1=1;

P1\_4=1;

delay();

delay();

delay();

P1\_0=0;

P1\_1=0;

P1\_4=0;

delay();

delay();

delay();

running();

delay();

delay();

running2();

delay();

delay();

}

void All\_OFF()

{

P1\_0=0;

P1\_1=0;

P1\_4=0;

}

void button\_Init()

{

//让P0\_1处于普通IO口，处于输入模式

P0SEL &= ~(0x01<<1);

P1DIR &= ~(0X01<<1);//1111 1101 0XfD

//让P0\_1处于上拉模式

P0INP &= ~(0x01<<1);//让P0\_1处于上下拉模式

P2INP &= ~(0X01<<5);//让P0\_1(P0）处于上拉模式（端口0）

}

1. LED.h文件

#ifndef \_LED\_H\_

#define \_LED\_H\_

#include "ioCC2530.h"

void running(void); /\* 跑马灿\*/

void running2(void); /\* 闪烁 \*/

void zijian(void); /\* 自检 \*/

void All\_OFF(void);

void button\_LCD(void);

void button\_LCD\_Num(int );

void running\_LCD\_WriteChar(char a,char b,char c);

char \*LED\_num\_judge(int num);

void button\_Init();

#endif /\* \_LED\_H\_ \*/

1. Date.c 文件

#include "ioCC2530.h"

#include "LCD.h"

unsigned int count\_date = 0;

int second\_one=0,second\_two=0; //秒

int branch\_one=0,branch\_two=0; //分

void Init\_Timer1()

{

T1CC0L = 0xd4; //设置最大计数值的低8位

T1CC0H = 0x30; //设置最大计数值的高8位

T1CCTL0 |= 0x04; //开启通道0的输出比较模式

T1IE = 1; //使能定时器1中断

T1OVFIM = 1; //使能定时器1溢出中断

EA = 1; //使能总中断

T1CTL = 0x0e; //分频系数是128,模模式

}

unsigned int count\_branch\_num = 0; //分计时

void branch\_count() //分

{

branch\_one++;

if(branch\_one > 9)

branch\_one = 0;

if(count\_branch\_num %10 == 0) // 十分到

{

branch\_two++;

if(branch\_two > 5)

{

branch\_two = 0;

}

LCD\_WriteChar(83,6,(branch\_two+48));

LCD\_WriteChar(91,6,(branch\_one+48));

}

else

{

LCD\_WriteChar(91,6,(branch\_one+48));

}

}

void second\_count(unsigned int s) //秒计时

{

second\_one++;

if(second\_one > 9)

second\_one = 0;

if(s %100 == 0) // 十秒到

{

second\_two++;

if(second\_two > 5)

{

second\_two = 0;

}

LCD\_WriteChar(105,6,(second\_two+48));

LCD\_WriteChar(112,6,(second\_one+48));

}

else

{

LCD\_WriteChar(112,6,(second\_one+48));

}

}

/\*================定时器1服务函数--日历==================\*/

#pragma vector = T1\_VECTOR

\_\_interrupt void Timer1\_Sevice()

{

T1STAT &= ~0x01; //清除定时器1通道0中断标志

count\_date++;

if(count\_date%10 == 0) //定时1秒到

{

second\_count(count\_date);

if(count\_date%600 == 0) //如果到六十秒

{

count\_branch\_num++; //秒数显示的十位数到达 6 时，分数计数器 +1

branch\_count();

count\_date=0;

if(count\_branch\_num%60 == 0) //如果到60分

count\_branch\_num=0;

}

}

}

1. Date.h文件

#ifndef \_Date\_H\_

#define \_Date\_H\_

#include "ioCC2530.h"

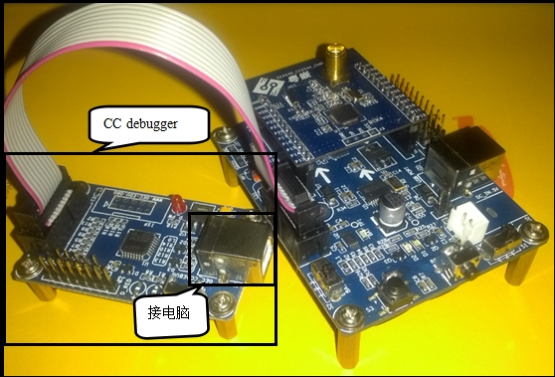
void Init\_Timer1();

\_\_interrupt void Timer1\_Sevice();

#endif /\* \_Date\_H\_ \*/

# 8 、实验步骤和结果

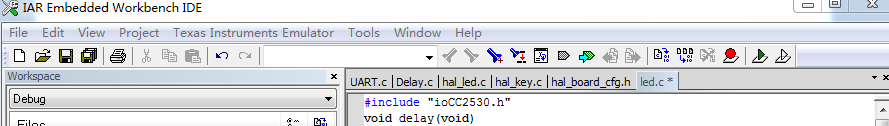
（1）连接好实验环境。将 Debugger下载器连接到电脑上，



（2）打开IAR Embedded Workbench for 8051 7.60，在IAR 中打开实验工程文

件夹666.eww；并对工程进行编译。

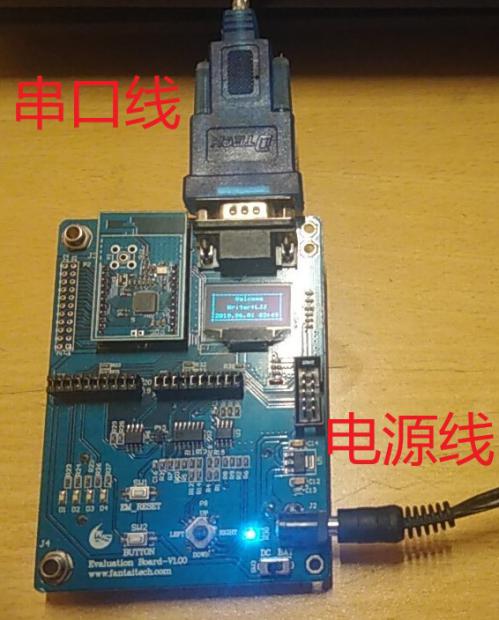
先按左边圈中按钮---编译，再按右边圈中按钮----下载



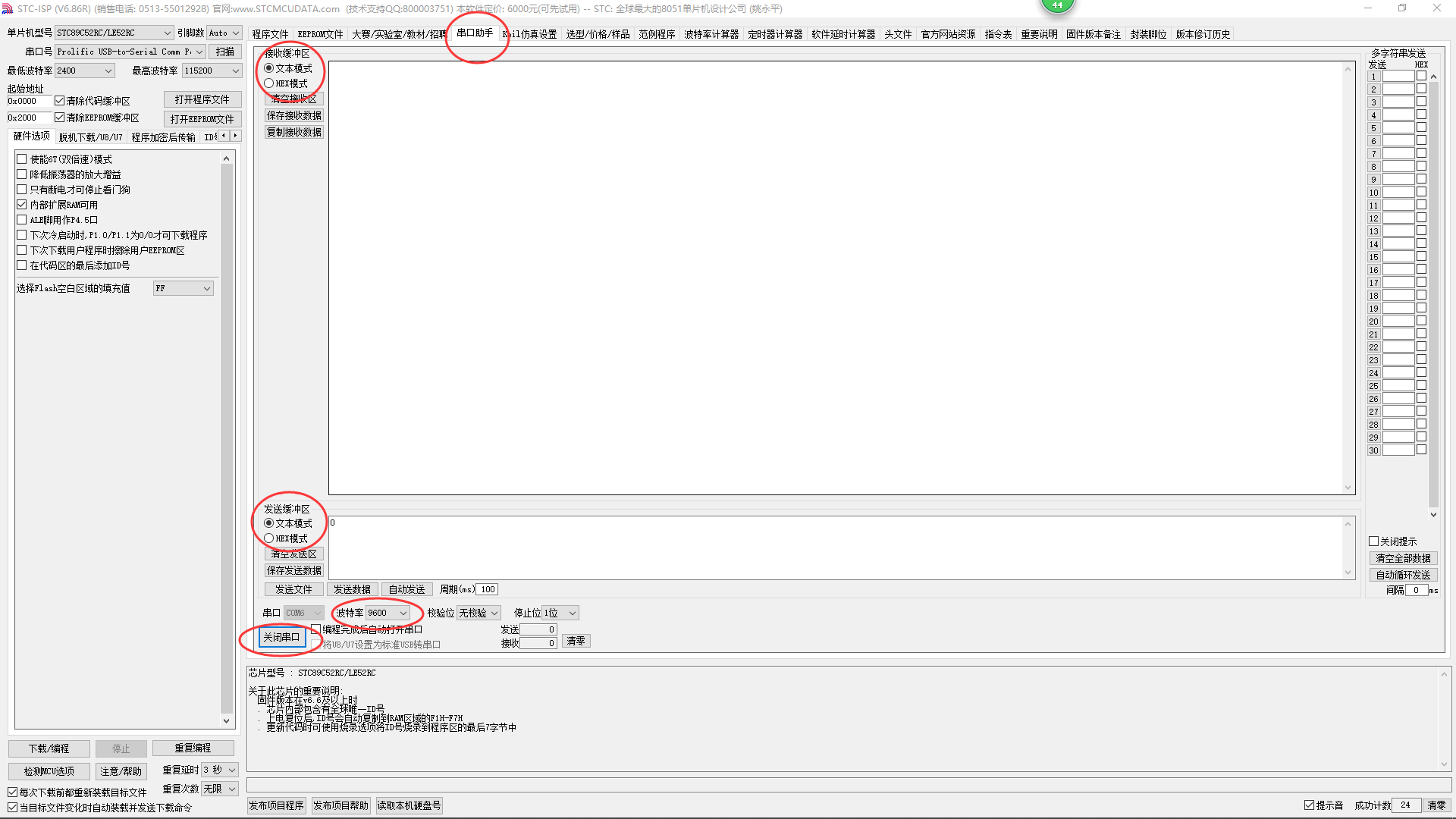
（3）单击IAR的全速运行，然后退出IAR。



1. 断开下载Debugger下载器，连接USB充电线，连接串口线。



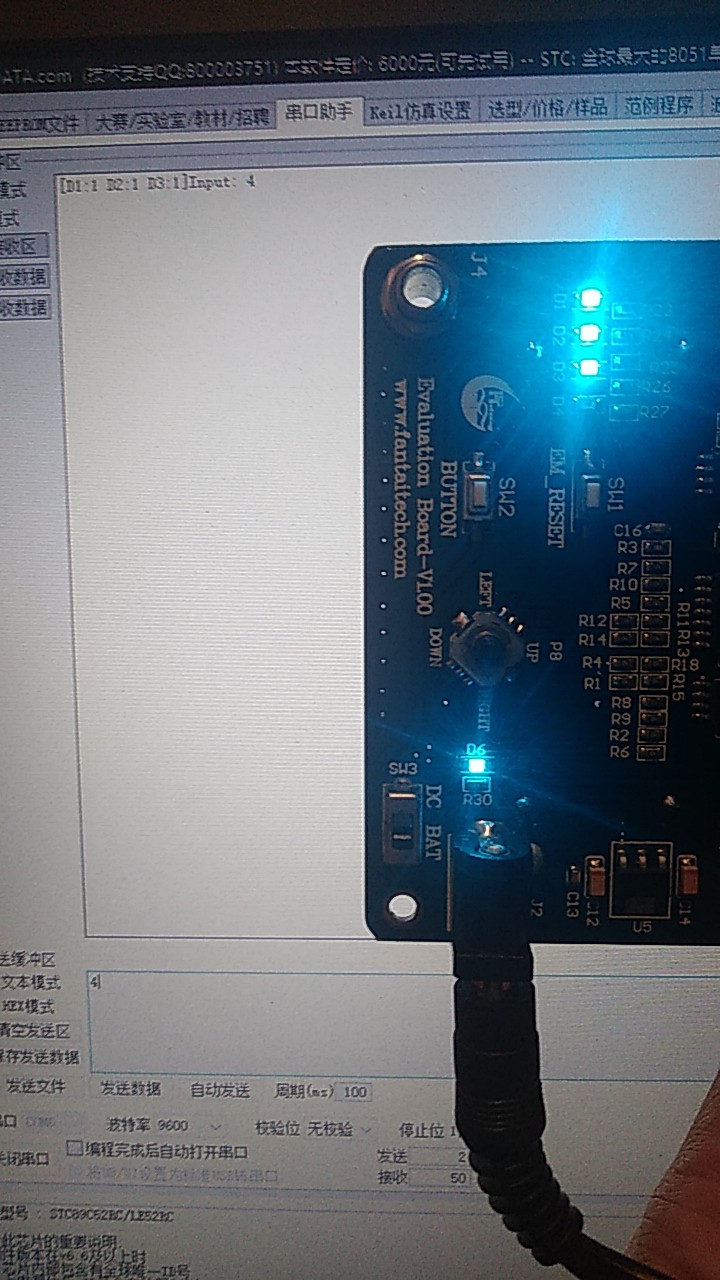
1. 打开stc-isp-15xx-v6.86r.exe 或者 串口调试助手.exe 连接上串口并设置波特率9600，接受区和发送区选择文本模式，打开串口。



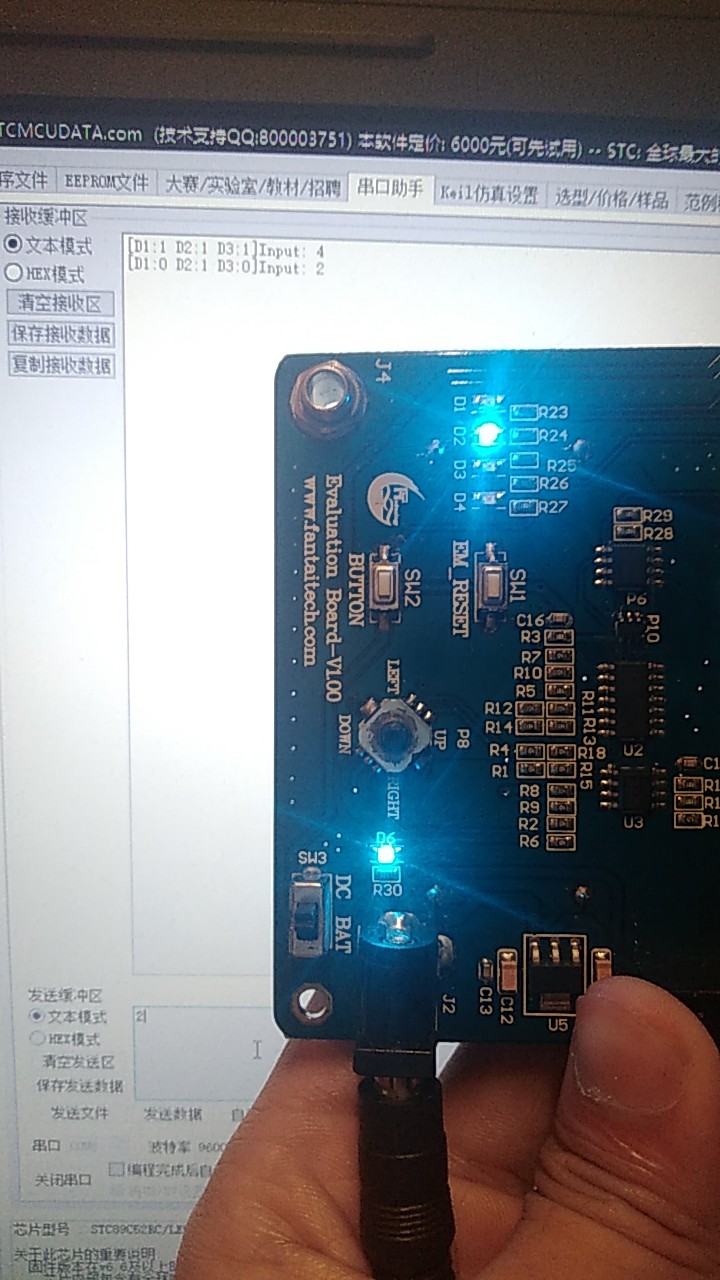
1. 通过串口发送命令控制LED亮灭，同时通过串口回传LED状态信息。

目前包含的命令有 0 灯全灭，1 只亮D1，2 只亮D2，3 只亮D3，4 灯全亮，T 进入计划二模式

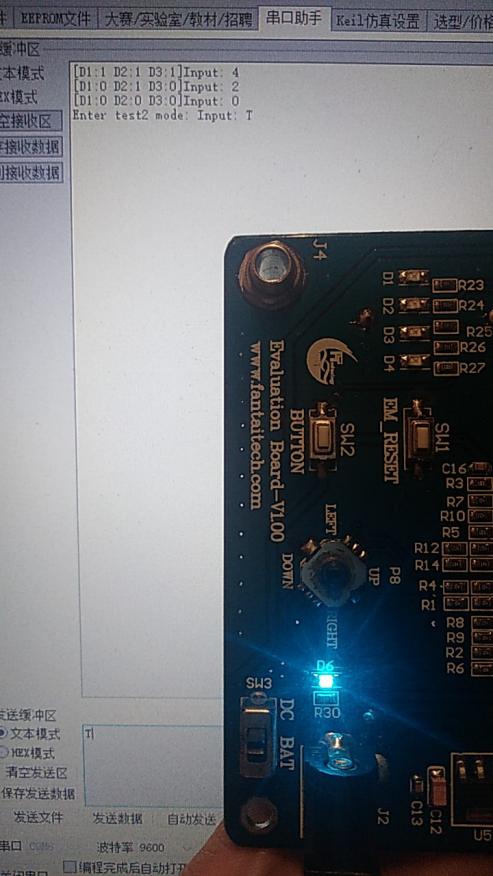
发送区输入 4



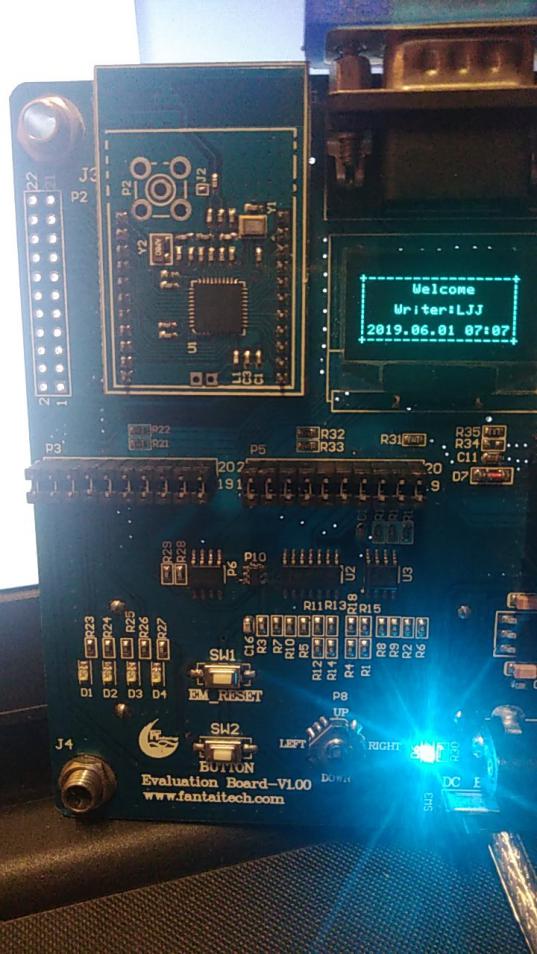
发送区输入 2



发送区输入 T



（7）OLED屏幕显示开机页面，包含作者信息及欢迎标题。四个LED灯默认关闭状态。



（8）按下按钮 BUTTON\_SW2 按下一次时，同时通过串口回传LED状态以及哪个按键被按下信息。实现跑马灯，同时屏幕显示相关内容，通过串口回传LED状态信息。按下第二次时，实现四个灯同时闪烁，同时屏幕显示相关内容，通过串口回传LED状态信息。按下第三次时，实现跑马灯。以此循环。



1. 分秒计时器



**9 、实验感想**

老师讲得好，实验写起来真简单。以现在的知识量能用cc2530Zigbee写的功能很少，希望以后能学到更多知识，实现更多更有意思的功能。

感觉写完这个项目后头有点冷！