**CC2530 LED点灯（通用IO）实验——实验报告**

**（项目编号：07012023**  **学时：2）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课 程** | 物联网传输技术 | **实验项目** | CC2530 LED点灯实验 | **成 绩** |  |
| **专业班级** | 14级物联网工程 | **学 号** |  | **批阅日期** |  |
| **姓 名** |  | **实验日期** |  | **指导教师** |  |

**一【实验目的】**

1. 熟悉CC2530开发板及外围电池板电路
2. 熟悉IAR集成开发环境

3、建立工程并下载调试cc2530芯片， 通过cc2530控制电池板上的led灯；掌握CC2530基本控制开发。

**二【实验内容】**

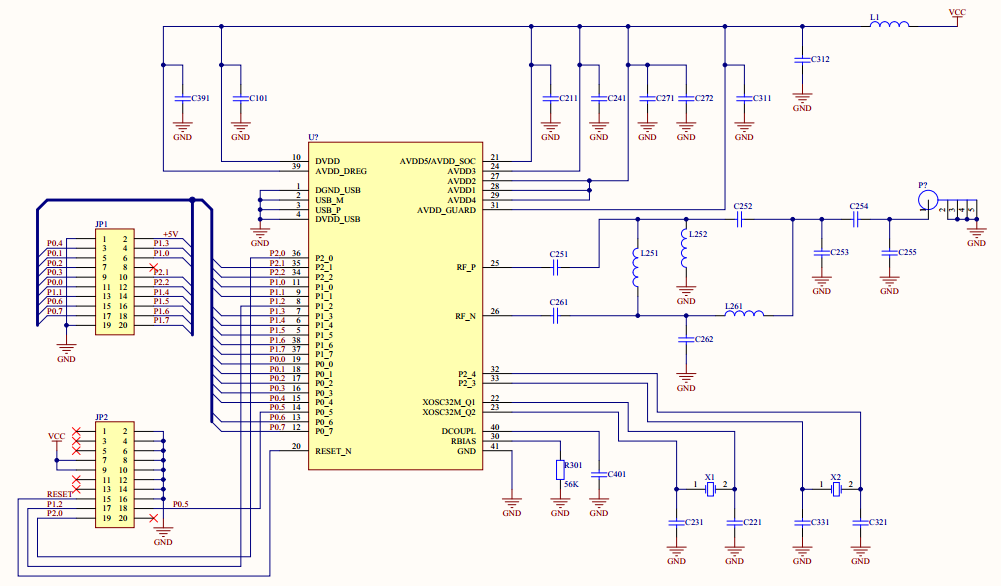
**1、****工具/原料**

* IAR Embedded Workbench for MCS-51
* CC2530 Zigbee开发套件
* CCDebuger调试器

**2、方法/步骤**

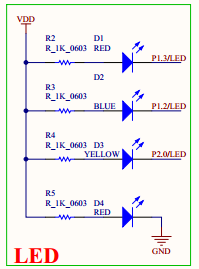
**2.1新建工程**

新建工程，不知道如何配置的可以查看《IAR如何建立工程》学习如何建立、配置、编译、调试嵌入式系统。通过编程实现LED灯控制，其中CC2530节点板电路如图1所示：



**图1**

电池底板上LED电路如图2所示：



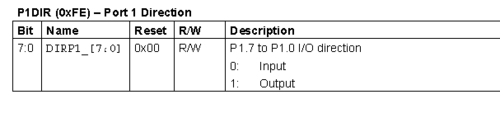
**图2**

步骤阅读

**2.2掌握CC2530**

在学习cc2530时，常需要查看其芯片说明书，**cc2530 datasheet文件**作为最全面的开发指南，快速查阅说明书学习需要的知识并解决问题。在这里需要了解cc2530的I/O口控制相关说明。

步骤阅读



芯片复位状态

在说明书中可以看到，芯片复位后所有I/O口均初始化为**输入**引脚，如下所示：

步骤阅读



1. 4

**引脚方向控制**

对于cc2530而言，其I/O口可配置为输入或是输出，其控制寄存器为：**PxDIR**,设置其对应位为**0**则为**输入**，置位**1**则为**输出**，芯片复位时初始化为输入状态。为了控制LED，需要将其设置为输出引脚：

void IO\_Init(void)

{

  P1DIR |= 0x04;     //P1\_3定义为输出

}

1. 5

**电平设置**

在分析电路图知道，对于D1，当P1\_3为高电平时，灯不亮；当P1\_3为低电平时，灯会亮，那么在初始化时只要控制P1\_3的电平就可以控制LED，如下所示，设置P1\_3为低电平点亮LED：

P1\_3 = 0；

1. 6

**2.3完整代码**

#include <ioCC2530.h>

#define LED1 P1\_3 //定义P1\_3口为D1控制端(红)

#define LED2 P1\_2 //定义P1\_2口为D2控制端（蓝）

#define LED3 P2\_0 //定义P1\_2口为D3控制端（黄）

void IO\_Init(void)

{

P1DIR |= 0x08; //P1\_3定义为输出

P1DIR |= 0x04; //P1\_2定义为输出

//P1DIR |= 0x0C; //此语句等同于以上两句

P2DIR |= 0x01; //P2\_0定义为输出

}

void main(void)

{

IO\_Init(); //调用初始化程序

LED1=0; //点亮D1（红）

LED2=0; //点亮D2（蓝）

LED3=0; //点亮D2（黄）

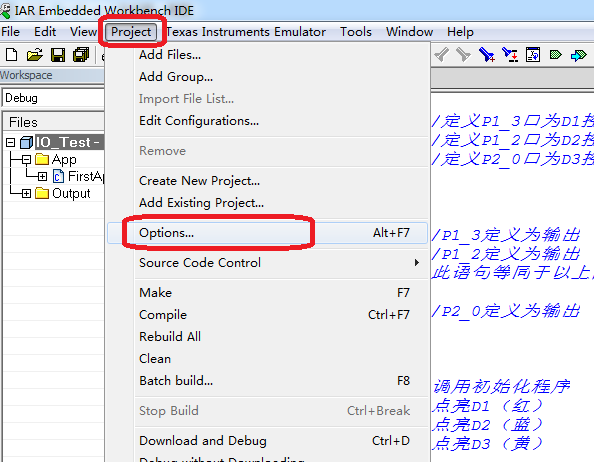
while(1);

}

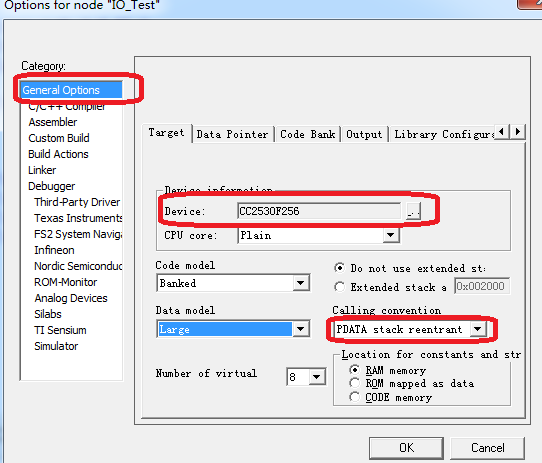
**2.4下载调试**

**注意：**项目配置

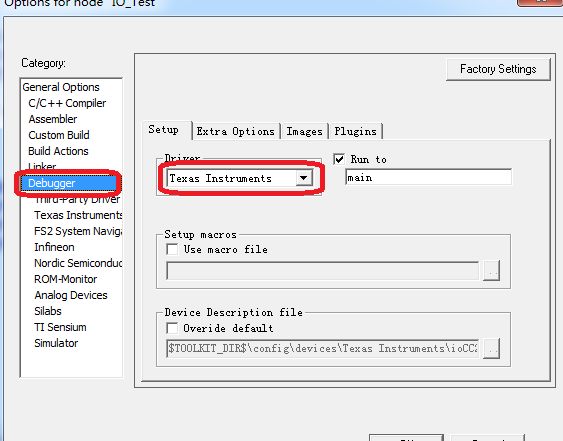
点击“project”菜单下的“options”选项配置工程相关属性，如图3，图4，图5：



**图3**



**图4**



**图5**

配置完成，选择对应的调试模块（按键选择），下载调试（调试器灯为红色，调试器工作正常，可按“reset”键重置；若调试器工作不正常，请查看电脑“设备管理”，卸载并重新安装调试器驱动）。

运行程序，完成实验要求。

**总 结：**

通过以上实验，了解并熟悉CC2530开发板及外围电池板电路；熟悉了IAR集成开发环境；掌握CC2530基本控制开发方法和步骤，能通过CC2530I/O口控制外围电池底板电路的LED灯亮。

**思考：**可扩展此实验，用自定义delay()函数控制LED间隔闪烁!

void delay(void)

{

unsigned int i;

unsigned char j;

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

{

for(j = 0; j < 240; j++)

{

asm("NOP"); //nop是内嵌汇编做空指令 延时用

asm("NOP"); //是一个空等待汇编指令，这个指令执行时，单片机是什么也不做，仅仅起一个时间延时作用。

asm("NOP"); //是指函数内部的参数翻译为汇编指令，其作用是在C语言环境下直接使用汇编指令执行

}

}

}