## 杭州电子科技大学创新实践实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学院 | 计算机学院 | 班级 |  | 学号 |  |
| 姓名 |  | 日期 | 2024.03.11 | 成绩 |  |
| 实验题目 | 环境安装 | | | | |
|  | | | | | |
| 实验目的 | ①检查硬件，认识MSP432p401r电路板  ②使用 CCS 在 LaunchPad 上运行代码  ③完成CCS开发环境安装，完成MSP432p401r电路板的上电测试，观察亮灯情况 | | | | |
| 硬件原理 | 嵌入式系统介绍：  1.包含一个微控制器，例如 MSP432  2.用于执行某个特定的应用  3.软件程序被烧写到片上存储器中  4.该设备的用户无法对设备中的软件进行读取或写入  5.设备中的软件功能仅限于解决有限的问题  6.微处理器是嵌入或是隐藏在系统当中的  嵌入式系统的特征：  1.持续不断地针对现实世界做出反应 和交互  • 接收输入  • 进行运算  • 产生输出  2.必须在一定的时间内完成输入/运算/输出的过程  系统包括：  • 子系统  • 子系统之间的接口 | | | | |
| 关键代码及注释 | **#include** <stdint.h>  **#include** "msp.h"  **#define** SW1 0x02 // on the left side of the LaunchPad board  **#define** SW2 0x10 // on the right side of the LaunchPad board  **#define** RED 0x01  **#define** GREEN 0x02  **#define** BLUE 0x04  // Version 1 is unfriendly  **void** **Port1\_Init**(**void**){  P1->SEL0 = 0x00;  P1->SEL1 = 0x00; // configure P1.4 and P1.1 as GPIO  P1->DIR = 0x01; // make P1.4 and P1.1 in, P1.0 output  P1->REN = 0x12; // enable pull resistors on P1.4 and P1.1  P1->OUT = 0x12; // P1.4 and P1.1 are pull-up  }  uint8\_t **Port1\_Input**(**void**){  **return** (P1->IN&0x12); // read P1.4,P1.1 inputs  }  **void** **Port2\_Init**(**void**){  P2->SEL0 = 0x00;  P2->SEL1 = 0x00; // configure P2.2-P2.0 as GPIO  P2->DS = 0x07; // make P2.2-P2.0 high drive strength  P2->DIR = 0x07; // make P2.2-P2.0 out  P2->OUT = 0x00; // all LEDs off  }  **void** **Port1\_Output**(uint8\_t data){ // write all of P1.0 outputs  P1->OUT = (P1->OUT&0xFE)|data;  }  **void** **Port2\_Output**(uint8\_t data){ // write all of P2 outputs  P2->OUT = data;  }  **int** **main**(**void**){ uint8\_t status;  Port1\_Init(); // initialize P1.1 and P1.4 and make them inputs (P1.1 and P1.4 built-in buttons)  // initialize P1.0 as output to red LED  Port2\_Init(); // initialize P2.2-P2.0 and make them outputs (P2.2-P2.0 built-in LEDs)  **while**(1){  status = Port1\_Input();  **switch**(status){ // switches are negative logic on P1.1 and P1.4  **case** 0x10: // SW1 pressed  Port2\_Output(BLUE);  Port1\_Output(1);  **break**;  **case** 0x02: // SW2 pressed  Port2\_Output(RED);  Port1\_Output(1);  **break**;  **case** 0x00: // both switches pressed  Port2\_Output(BLUE+RED);  Port1\_Output(1);  **break**;  **case** 0x12: // neither switch pressed  Port2\_Output(0);  Port1\_Output(0);  **break**;  }  }  }  // Version 2 version is friendly  **void** **Port1\_Init2**(**void**){  P1->SEL0 &= ~0x13;  P1->SEL1 &= ~0x13; // 1) configure P1.4 P1.1 P1.0 as GPIO  P1->DIR &= ~0x12; // 2) make P1.4 and P1.1 in  P1->DIR |= 0x01; // 2) make P1.0 out  P1->REN |= 0x12; // 3) enable pull resistors on P1.4 and P1.1  P1->OUT |= 0x12; // P1.4 and P1.1 are pull-up  }  uint8\_t **Port1\_Input2**(**void**){  **return** (P1->IN&0x12); // read P1.4,P1.1 inputs  }  **void** **Port1\_Output2**(uint8\_t data){ // write output to P1.0  P1->OUT = (P1->OUT&0xFE)|data;  }  **void** **Port2\_Init2**(**void**){  P2->SEL0 &= ~0x07;  P2->SEL1 &= ~0x07; // 1) configure P2.2-P2.0 as GPIO  P2->DIR |= 0x07; // 2) make P2.2-P2.0 out  P2->DS |= 0x07; // 3) activate increased drive strength  P2->OUT &= ~0x07; // all LEDs off  }  **void** **Port2\_Output2**(uint8\_t data){ // write three outputs bits of P2  P2->OUT = (P2->OUT&0xF8)|data;  }  **int** **main2**(**void**){ uint8\_t status;  Port1\_Init2(); // initialize P1.1 and P1.4 and make them inputs (P1.1 and P1.4 built-in buttons)  // initialize P1.0 as output to red LED  Port2\_Init2(); // initialize P2.2-P2.0 and make them outputs (P2.2-P2.0 built-in LEDs)  **while**(1){  status = Port1\_Input2();  **switch**(status){ // switches are negative logic on P1.1 and P1.4  **case** 0x10: // SW1 pressed  Port2\_Output2(BLUE);  Port1\_Output2(1);  **break**;  **case** 0x02: // SW2 pressed  Port2\_Output2(RED);  Port1\_Output2(1);  **break**;  **case** 0x00: // both switches pressed  Port2\_Output2(BLUE+RED);  Port1\_Output2(1);  **break**;  **case** 0x12: // neither switch pressed  Port2\_Output2(0);  Port1\_Output2(0);  **break**;  }  }  }  // Version 3 provides for abstraction  **void** **Switch\_Init**(**void**){  P1->SEL0 &= ~0x12;  P1->SEL1 &= ~0x12; // 1) configure P1.4 and P1.1 as GPIO  P1->DIR &= ~0x12; // 2) make P1.4 and P1.1 in  P1->REN |= 0x12; // 3) enable pull resistors on P1.4 and P1.1  P1->OUT |= 0x12; // P1.4 and P1.1 are pull-up  }  // bit-banded addresses, positive logic  **#define** SW2IN ((\*((**volatile** uint8\_t \*)(0x42098010)))^1)  **#define** SW1IN ((\*((**volatile** uint8\_t \*)(0x42098004)))^1)  **void** **RedLED\_Init**(**void**){  P1->SEL0 &= ~0x01;  P1->SEL1 &= ~0x01; // 1) configure P1.0 as GPIO  P1->DIR |= 0x01; // 2) make P1.0 out  }  // bit-banded address  **#define** REDLED (\*((**volatile** uint8\_t \*)(0x42098040)))  **void** **ColorLED\_Init**(**void**){  P2->SEL0 &= ~0x07;  P2->SEL1 &= ~0x07; // 1) configure P2.2-P2.0 as GPIO  P2->DIR |= 0x07; // 2) make P2.2-P2.0 out  P2->DS |= 0x07; // 3) activate increased drive strength  P2->OUT &= ~0x07; // all LEDs off  }  // bit-banded addresses  **#define** BLUEOUT (\*((**volatile** uint8\_t \*)(0x42098068)))  **#define** GREENOUT (\*((**volatile** uint8\_t \*)(0x42098064)))  **#define** REDOUT (\*((**volatile** uint8\_t \*)(0x42098060)))  **int** **main3**(**void**){  Switch\_Init();  ColorLED\_Init();  RedLED\_Init();  GREENOUT = 0;  **while**(1){  **if**(SW1IN||SW2IN){ // Single Red on if either is pressed  REDLED=1;  }**else**{  REDLED=0;  }  **if**(SW1IN){ // Color=Blue if SW1 is pressed  BLUEOUT = 1;  }**else**{  BLUEOUT = 0;  }  **if**(SW2IN){ // Color=Red if SW2 is pressed  REDOUT = 1;  }**else**{  REDOUT = 0;  }  }  } | | | | |
| 实验步骤 | 下载安装CCS9.3.0版本，按照TI-RSLK 模块1的教程完成工程导入，把MSP432p401r电路板接上电脑电源，按照TI-RSLK 模块1的实验视频运行InputOutput工程，在MSP432p401r电路板上进行调试，观察亮灯情况 | | | | |
| 实验结果 | 小车接上电源绿灯亮起，InputOutput工程运行时小车的红灯亮起，但是按下S1或S2按钮其他LED灯都不亮 | | | | |
| 思考与反馈 | 运行工程时发现程序报出警告，提示电路板在低电量模式下无法进入断点，需要暂停程序查看电路板被停止在哪个断点 | | | | |