**智能硬件设计基础上机报告**

学院（系）： 求实书院（两人/单人每组）

姓 名： 学号：

姓 名： 学号：

成 绩：

**一、实验目的和要求**

1 掌握通过UART实现与SoC之间的处理原理

2 掌握寄存器方式访问寄存器中值的必要性

**二、实验原理和内容**

1 安装HiSparkStudio、下载并安装WS63 SDK

该步骤已经在第一次实验中进行完毕，此处不再介绍相关内容。

2 通过UART转串口接收串口数据

UART是计算机与人类进行交互的主要通道，HiSpark EP实践平台通过UART转串口，将UART协议的3.3V电平，转换为USB电平。

相关的硬件已经在硬件电路的原理图中进行了规范，如图所示。

日历

AI 生成的内容可能不正确。

图示, 示意图

AI 生成的内容可能不正确。

通过该电路，用户可以实现从SoC到计算机串口端的双向的数据的传输，包括：从SoC端向计算机的硬件的传输机制和反向的传输机制。

3 通过寄存器访问外部设备

开发板上共计设置了六个WS2812B类型的全色型的LED发光二极管(又被称为三色LED)，每个灯可以发出R、G、B等颜色，并且每种颜色均支持256种颜色配置，共计支持16777216种颜色配置。

图示

AI 生成的内容可能不正确。

(1) WS2812B的基本特性

系统中，主要采用了WS2812B芯片完成光电转换控制

|  |  |  |  |
| --- | --- | --- | --- |
| 图示, 工程绘图  AI 生成的内容可能不正确。 | 引脚 | 符号 | 功能描述 |
| 1 | VDD | Power supply LED |
| 2 | DOUT | Control data signal output |
| 3 | VSS | Ground |
| 4 | DIN | Control data signal input |

(2) WS2812B的传输

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 发光色 | 波长 | 可见光强度 | 电流 | 工作电压 |
| RED | 620-630 | 550-700 | 16 | 1.8-2.2 |
| GREEN | 510-530 | 1100-1400 | 16 | 2.8-3.1 |
| BLUE | 465-475 | 200-400 | 16 | 3.0-3.4 |

(3)WS2812B的时序

图示

AI 生成的内容可能不正确。图示, 示意图

AI 生成的内容可能不正确。

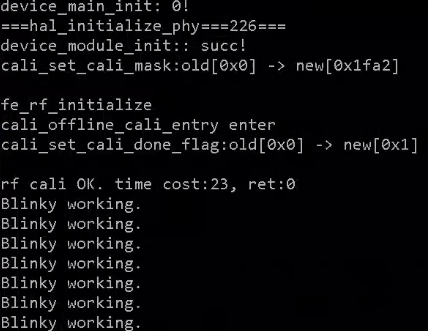
其中，低电平、高电平和RESET的时隙如下。

|  |  |  |  |
| --- | --- | --- | --- |
| T0H | 0 code，High Voltage | 0.4us | ±150ns |
| T1H | 1 code，High Voltage | 0.85us | ±150ns |
| T0L | 0 code，Low Voltage | 0.85us | ±150ns |
| T0H | 1 code，Low Voltage | 0.4us | ±150ns |
| RES | Low Voltage Time | >50us | - |

**三、实验步骤与操作方法**

1 UART的双向传输实验

通过Type-C口连接硬件开发板，构建传输信号的硬件通道。连接成功后，即可通过该硬件通道实现对开发板上信息的获取工作，接收到开发板上发来的各式信息，如图所示。



双向传输要求的详细说明如下：

(1) 默认，接受全部通过USB-Serial转发出来的信息，并可以通过osal-printf函数向串口增加数据，上述步骤可以在BLINKY基础之上实现。

(2) 增加，增加串口接收功能。主要涉及到如下基本步骤，需要自行完善。

① 设置串口的基本功能

#define CONFIG\_I2C\_SCL\_MASTER\_PIN 15

#define CONFIG\_I2C\_SDA\_MASTER\_PIN 16

#define CONFIG\_I2C\_MASTER\_PIN\_MODE 2

uapi\_pin\_set\_mode(CONFIG\_I2C\_SCL\_MASTER\_PIN,

CONFIG\_I2C\_MASTER\_PIN\_MODE);

uapi\_pin\_set\_mode(CONFIG\_I2C\_SDA\_MASTER\_PIN,

CONFIG\_I2C\_MASTER\_PIN\_MODE);

② 设置串口模式

#define UART\_BAUDRATE 115200

#define UART\_DATA\_BITS 3

#define UART\_STOP\_BITS 1

#define UART\_PARITY\_BIT 0

#define UART\_TRANSFER\_SIZE 20

#define CONFIG\_UART\_INT\_WAIT\_MS 5

#define UART\_TASK\_DURATION\_MS 500

#define UART0\_BUS\_ID 0

#define UART0\_TXD\_PIN 17

#define UART0\_RXD\_PIN 18

#define UART0\_PIN\_MODE 1

static uint8\_t g\_app\_uart\_rx\_buff[UART\_TRANSFER\_SIZE] = { 0 };

static uint8\_t g\_app\_uart\_int\_rx\_flag = 0;

static uart\_buffer\_config\_t g\_app\_uart\_buffer\_config = {

.rx\_buffer = g\_app\_uart\_rx\_buff,

.rx\_buffer\_size = UART\_TRANSFER\_SIZE

};

uapi\_pin\_set\_mode(UART0\_TXD\_PIN, UART0\_PIN\_MODE);

uapi\_pin\_set\_mode(UART0\_RXD\_PIN, UART0\_PIN\_MODE);

static void app\_uart\_init\_config(void) {

uart\_attr\_t attr = {

.baud\_rate = UART\_BAUDRATE,

.data\_bits = UART\_DATA\_BITS,

.stop\_bits = UART\_STOP\_BITS,

.parity = UART\_PARITY\_BIT

};

uart\_pin\_config\_t pin\_config = {

.tx\_pin = S\_MGPIO0,

.rx\_pin = S\_MGPIO1,

.cts\_pin = PIN\_NONE,

.rts\_pin = PIN\_NONE

};

uapi\_uart\_init(UART0\_BUS\_ID, &pin\_config, &attr, NULL,

&g\_app\_uart\_buffer\_config);

}

③ 设置两个UART的中断，发送中断和接收中断

static void app\_uart\_read\_int\_handler(const void \*buffer, uint16\_t length, bool error) {

unused(error);

if (buffer == NULL || length == 0) {

osal\_printk("uart%d int mode transfer illegal data!\r\n",

UART0\_BUS\_ID);

return;

}

uint8\_t \*buff = (uint8\_t \*)buffer;

if (memcpy\_s(g\_app\_uart\_rx\_buff, length, buff, length) != EOK) {

osal\_printk("uart%d int mode data copy fail!\r\n", UART0\_BUS\_ID);

return;

}

//此处，处理g\_app\_uart\_rx\_buff中的数据，自行添加代码段内容

g\_app\_uart\_int\_rx\_flag = 1;

}

static void app\_uart\_write\_int\_handler(const void \*buffer, uint32\_t length, const void \*params) {

unused(params);

uint8\_t \*buff = (void \*)buffer;

for (uint8\_t i = 0; i < length; i++) {

osal\_printk("uart%d write data[%d] = %d\r\n",

UART0\_BUS\_ID, i, buff[i]);

}

}

④ 设置主程序内容

void OledTask(void) {

app\_uart\_init\_pin();

app\_uart\_init\_config();

while(1){

if (uapi\_uart\_register\_rx\_callback(UART0\_BUS\_ID,

UART\_RX\_CONDITION\_FULL\_OR\_SUFFICIENT\_DATA\_OR\_IDLE, 1,

app\_uart\_read\_int\_handler) == ERRCODE\_SUCC) {

osal\_printk("uart%d int mode register receive callback succ!\r\n",

UART0\_BUS\_ID);

}

while (g\_app\_uart\_int\_rx\_flag != 1) {

osal\_msleep(CONFIG\_UART\_INT\_WAIT\_MS);

}

g\_app\_uart\_int\_rx\_flag = 0;

osal\_printk("uart%d int mode send back!\r\n", UART0\_BUS\_ID);

if (uapi\_uart\_write\_int(UART0\_BUS\_ID, g\_app\_uart\_rx\_buff,

UART\_TRANSFER\_SIZE, 0,

app\_uart\_write\_int\_handler) == ERRCODE\_SUCC) {

osal\_printk("uart%d int mode send back succ!\r\n",

UART0\_BUS\_ID);

}

}

}

2 寄存器方式访问寄存器

通过寄存器方式，可以实现比单独调用命令更快的访问方法，相比于传统方法，能够更快的获取速度，实现达到运算速度要求的操作速度。

①通过寄存器方式访问

#include "pinctrl.h"

#include "watchdog.h"

#include "tcxo.h"

#include "gpio.h"

#include "soc\_osal.h"

#include "app\_init.h"

#include "watchdog.h"

#include "ws2812b.h"

uint32\_t preg\_placeholder = 0;

static \_\_inline\_\_ void one\_num(void){

uapi\_reg\_setbit(0x44028030, 5);

for (int i = 0; i < 20; i++) {

uapi\_reg\_read32(0x44028030, preg\_placeholder);

}

uapi\_reg\_setbit(0x44028034, 5);

for (int i = 0; i < 4; i++) {

uapi\_reg\_read32(0x44028030, preg\_placeholder);

}

}

static \_\_inline\_\_ void zero\_num(void){

uapi\_reg\_setbit(0x44028030, 5);

for (int i = 0; i < 4; i++) {

uapi\_reg\_read32(0x44028030, preg\_placeholder);

}

uapi\_reg\_setbit(0x44028034, 5);

for (int i = 0; i < 18; i++) {

uapi\_reg\_read32(0x44028030, preg\_placeholder);

}

}

②整合后的RGB颜色输出实验

void rgb\_display(uint8\_t red, uint8\_t green, uint8\_t blue, uint8\_t brightness){

red = ((uint32\_t)red) \* brightness / 255;

green = ((uint32\_t)green) \* brightness / 255;

blue = ((uint32\_t)blue) \* brightness / 255;

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

if(red & 0x80) one\_num();

else zero\_num();

red <<= 1;

}

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

if(green & 0x80) one\_num();

else zero\_num();

green <<= 1;

}

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

if(blue & 0x80) one\_num();

else zero\_num();

blue <<= 1;

}

}

**四、实验结果与分析**

1 实验项目“UART的双向传输实验”的运行结果

说明：需要填补进去代码中间需要填补的空白，实现数据的回传功能

2 实验项目“寄存器方式访问寄存器”的运行结果

说明：需要自行添加调用该函数的功能void rgb\_display(uint8\_t red, uint8\_t green, uint8\_t blue, uint8\_t brightness)

**五、讨论和建议**