

Cortex-M4原理与实践实验报告

题目： 红外遥控接收编程实验

**作者姓名 林宇航**

**指导教师**  **郭方洪**

**专业班级 自动化1901**

**学 院 信息工程学院**

**提交日期** 2021年12月22日

**一、实验目的**

1. 了解红外数据传输的通信原理。

2. 了解红外编码解码的原理

3. 掌握红外通信的编程

**二、实验设备**

1.计算机一台，操作系统为Windos11，装有CCSv11.0软件。

2.EK-TM4C1294XL实验开发板一块。

3.USB连接线一条。

**三、实验原理和流程**

**1. 红外传感器简介**

红外遥控发射芯片采用 PPM 编码方式，当发射器按键按下后，将发射一组 108ms 的编码脉冲。遥控编码脉冲由前导码、16 位地址码（8 位地址码、8 位地址码的反码）和16位操作码（8 位操作码、8 位操作码的反码）组成。通过对用户码的检验，每个遥控器只能控制一个设备动作，这样可以有效地防止多个设备之间的干扰。编码后面还要有编码的反码，用来检验编码接收的正确性，防止误操作，增强系统的可靠性。前导码是一个遥控码的起始部分，由一个 9ms 的高电平（起始码）和一个4.5ms的低电平（结果码）组成，作为接受数 据的准备脉冲。以脉宽为 0.56ms、周期为 1.12ms 的组合表示二进制的“0”；以脉宽为 1.68ms、周期为 2.24ms 的组合表示二进制的“1”。

图示

描述已自动生成

图 1 发送一组完整的编码脉冲

**2. 软件原理**

开始时发射一个特定的同步码头，对于接收端而言就是一个9ms 的低电平,和一个 4.5ms的高电平，这个同步码头可以使程序知道从这个同步码头以后可以开始接收数据。采用脉宽调制的串行码，以脉宽为 0.565ms、间隔 0.56ms、周期为 1.125ms 的组合表示二进制的“0”；以脉宽为 0.565ms、间隔 1.685ms、周期为 2.25ms 的组合表示二进制的“1”。解码的关键是如何识别“0”和“1”，从位的定义我们可以发现“0”、“1”均以 0.56ms 的高电平开始，不同的是低电平的宽度不同，“0”为 0.56ms，“1”为 1.685ms，所以必须根据高电平的宽度区别“0”和 “1”。如果从 0.56ms 低电平过后，开始延时，0.56ms 以后，若读到的电平为低，说明该位为“0”，反之则为“1”，为了可靠起见，延时必须比0.56ms 长些，但又不能超过 1.12ms，否则如果该位为“0”，读到的已是下一位的高电平，因此取（1.12ms+0.56ms）/2=0.84ms 最为可靠，一般取 0.84ms 左右即可。根据红外编码的格式，程序应该等待 9ms 的起始码和 4.5ms 的结果码完成后才能读码。

**流程图：**



**四、实验代码、注释**

**实验代码：**

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_nvic.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/systick.h"

#include "driverlib/sysctl.h"

#include "TFTinit/TFT\_400x240\_OTM4001A\_16bit.h"

//#include "TFTinit/picture.h"

//#include "TOUCHinit/TOUCH\_TSC2046.h"

#include "EPIinit/EPIinit.h"

volatile uint32\_t ui32SysClock;

volatile uint32\_t error= 0;

volatile uint32\_t count= 0;

unsigned char IrData[4];

//红外解码输出，数字0-9对应的关系如下，地址吗为0x00

/\*address 0x00

\* 0->22

\* 1->12

\* 2->24

\* 3->94

\* 4->8

\* 5->28

\* 6->90

\* 7->66

\* 8->82

\* 9->74

\*/

///////////////////////////////////////////米字管

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

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

#define \_NOP() \_nop()

#define I2C0\_MASTER\_BASE 0x40020000

#define I2C0\_SLAVE\_BASE 0x40020000

//U21控制4个米字管和特殊管脚的亮灭

#define I2C0\_ADDR\_TUBE\_SEL 0x30 //00110000

//U22控制米字管7~14管脚对应的码段

#define I2C0\_ADDR\_TUBE\_SEG\_LOW 0x32 //00110010

//U23控制米字管15~18管脚对应的码段

#define I2C0\_ADDR\_TUBE\_SEG\_HIGH 0x34 //00110100

//U24控制LED光柱

//PCA9557内部寄存器，也称子地址

#define PCA9557\_REG\_INPUT 0x00

#define PCA9557\_REG\_OUTPUT 0x01

#define PCA9557\_REG\_PolInver 0x02

#define PCA9557\_REG\_CONFIG 0x03

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

#define NUM 0

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

拉高 SDA 信号

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

void I2C\_Set\_sda\_high\_m( void )

{

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3); //拉高PB3

\_NOP();//空循环一个机器指令的时间，一个机器周期

\_NOP();//空循环一个机器指令的时间，一个机器周期

return;

}

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

拉低SDA 信号

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

void I2C\_Set\_sda\_low\_m ( void )

{

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_3,0X00000000); //拉低PB3

\_NOP();//空循环一个机器指令的时间，一个机器周期

\_NOP();//空循环一个机器指令的时间，一个机器周期

return;

}

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

拉高SCL 信号

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

void I2C\_Set\_scl\_high\_m( void )

{

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_2,GPIO\_PIN\_2); //拉高PB2

\_NOP();//空循环一个机器指令的时间，一个机器周期

\_NOP();//空循环一个机器指令的时间，一个机器周期

return;

}

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

拉低SCL 信号

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

void I2C\_Set\_scl\_low\_m ( void )

{

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_2,0X00000000); //拉低PB2

\_NOP();//空循环一个机器指令的时间，一个机器周期

\_NOP();//空循环一个机器指令的时间，一个机器周期

return;

}

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

IIC 信号结束信号函数

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

void I2C\_STOP\_m(void)//SCL为高，SDA从低变高

{

int i;

I2C\_Set\_sda\_low\_m();//拉低SDA

for(i = NUM;i > 0;i--);

I2C\_Set\_scl\_low\_m();//拉低SCL

for(i = NUM;i > 0;i--);

I2C\_Set\_scl\_high\_m();//拉高SCL

for(i = NUM;i > 0;i--);

I2C\_Set\_sda\_high\_m();//拉高SDA

for(i = NUM+1;i > 0;i--);//一次

return;

}

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

IIC 信号初始化

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

void I2C\_Initial\_m( void )//开始状态

{

I2C\_Set\_scl\_low\_m();//拉低SCL

I2C\_STOP\_m();//停止

return;

}

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

IIC 信号起始信号函数

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

void I2C\_START\_m(void)//当SCL为高时，SDA由高到低

{

int i;

I2C\_Set\_sda\_high\_m();//拉高SDA

for(i = NUM;i > 0;i--);

I2C\_Set\_scl\_high\_m();//拉高SCL

for(i = NUM;i > 0;i--);

I2C\_Set\_sda\_low\_m();//拉低SDA

for(i = NUM;i > 0;i--);

I2C\_Set\_scl\_low\_m();//拉低SCL

return;

}

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

IIC 获取应答函数

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

int I2C\_GetACK\_m(void)

{

int j;

\_NOP();//空循环一个机器指令的时间

\_NOP();//空循环一个机器指令的时间

I2C\_Set\_scl\_low\_m();//拉低SCL

for(j = NUM;j> 0;j--);

I2C\_Set\_scl\_high\_m();//拉高SCL

for(j = NUM;j> 0;j--);

I2C\_Set\_sda\_low\_m();//拉低SDA

for(j = NUM;j > 0;j--);

I2C\_Set\_scl\_low\_m();//拉低SCL

return 1;

}

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

IIC 发送字节函数

参数 1：要发送字节值

return ：无返回

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

void I2C\_TxByte\_m(unsigned char nValue)

{

int i;

int j;

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

{

if(nValue & 0x80)

I2C\_Set\_sda\_high\_m();//拉高SDA

else

I2C\_Set\_sda\_low\_m();//拉低SDA

for(j = NUM;j > 0;j--);

I2C\_Set\_scl\_high\_m();//拉高SCL

nValue <<= 1;

for(j = NUM;j > 0;j--);

I2C\_Set\_scl\_low\_m();//拉低SCL

}

return;

}

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

IIC 发送数组函数

参数 1 num : 发送字节数

2 device\_addr : iic目标地址

3 \*data ：发送数组地址

return ：无返回

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

void i2c\_write\_m(int num, unsigned char device\_addr,unsigned char \*data)

{

int i = 0;

int count = num;

unsigned char \*send\_data = data;

unsigned char write\_addr = device\_addr;

I2C\_Set\_scl\_high\_m();//拉高SCL

for(i = NUM;i > 0;i--);

I2C\_Set\_sda\_high\_m();//拉高SDA

for(i = NUM;i > 0;i--);

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

{

I2C\_START\_m(); //模拟I2C写数据的时序

I2C\_TxByte\_m(write\_addr);//发送地址

I2C\_GetACK\_m();//获取应答

I2C\_TxByte\_m(send\_data[i]);//发送一个字节，子地址

I2C\_GetACK\_m();//获取应答

i++;

I2C\_TxByte\_m(send\_data[i]);//发送一个字节

I2C\_GetACK\_m();//获取应答

I2C\_STOP\_m();//停止，每两个字节一次

}

}

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

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

//\*\*\*\*\*\*配置I2C0模块的IO引脚，\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void I2C0GPIOBEnable\_m(void)

{ // Enable GPIO portB containing the I2C pins (PB2&PB3)

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOB);//使能模块

GPIOPinTypeGPIOOutput(GPIO\_PORTB\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3);//配置PB2\PB3为输出

}

//\*\*\*\*\*\*配置PCA9557芯片中连接米字管的各引脚为输出\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void I2C0DeviceInit(void)

{

unsigned char dataBuf[2] = {PCA9557\_REG\_CONFIG, 0x00};//PCA9557芯片引脚方向寄存器

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEL,dataBuf);//往三个PCA9557芯片PCA9557\_REG\_CONFIG写数据

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEG\_LOW,dataBuf);

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEG\_HIGH,dataBuf);

}

//\*\*\*\*\*\*\*设置米字管的管选信号\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void I2C0TubeSelSet(char data)

{ //选择1、2、3、4、5哪个米字管亮

unsigned char dataBuf[2] = {PCA9557\_REG\_OUTPUT, data};

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEL,dataBuf);

}

//\*\*\*\*\*\*\*点亮米字管的相应码段\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void I2C0TubeLowSet(char data)

{ //点亮7-14管脚对应的码段

unsigned char dataBuf[2] = {PCA9557\_REG\_OUTPUT, data};

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEG\_LOW,dataBuf);

}

void I2C0TubeHighSet(char data)

{ //点亮15-18管脚对应的码段

unsigned char dataBuf[2] = {PCA9557\_REG\_OUTPUT, data};

i2c\_write\_m(2,I2C0\_ADDR\_TUBE\_SEG\_HIGH,dataBuf);

}

//\*\*\*\*\*\*\*\*预设码段值，方便查找\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static const char tubeCodeTable[10][2]=

{ // SegmLow, SegHigh

{ 0x10, 0x3E }, // 0

{ 0x00, 0x18 }, // 1

{ 0x70, 0x2C }, // 2

{ 0x70, 0x26 }, // 3

{ 0x60, 0x32 }, // 4

{ 0x70, 0x16 }, // 5

{ 0x70, 0x1E }, // 6

{ 0x00, 0x26 }, // 7

{ 0x70, 0x3E }, // 8

{ 0x70, 0x36 }, // 9

// { 0x60, 0x3E }, // a

};

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

//////////////////////////////////////米字管结束

void IrOutput()//发送

{

switch(IrData[2])

{

case 12 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x02);break;

case 24 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x01);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x00);break;

case 94 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x00);break;

case 8 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x00);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 28 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 90 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x01);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 66 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 82 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x00);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x01);break;

case 74 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x01);break;

default: GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x11);

}

}

void MyIrOutput()//发送

{

switch(IrData[2])

{

case 12 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x02);break;

case 24 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x01);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x00);break;

case 94 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x00);break;

case 8 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x00);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 28 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 90 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x01);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 66 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x10);break;

case 82 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x00);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x01);break;

case 74 : GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x02);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x01);break;

default: GPIOPinWrite(GPIO\_PORTN\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_0 , 0x03);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0 , 0x11);

}

}

/////////////////////////////////////////////////////////////////////

//GPIO口初始化

void GPIOInitial(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPION);

SysCtlGPIOAHBEnable(SYSCTL\_PERIPH\_GPION);//PN0 PN1

GPIOPinTypeGPIOOutput(GPIO\_PORTN\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

SysCtlGPIOAHBEnable(SYSCTL\_PERIPH\_GPIOF);//PF0 PF1 PF2 PF3 PF4

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4);

}

//////////////////////////////////////

//中断初始化

void GPIOIntInitial(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPION);//

SysCtlGPIOAHBEnable(SYSCTL\_PERIPH\_GPION);//

GPIOPinTypeGPIOInput(GPIO\_PORTN\_BASE,GPIO\_PIN\_4 );//PN4

GPIOIntTypeSet(GPIO\_PORTN\_BASE,GPIO\_PIN\_4,GPIO\_LOW\_LEVEL);//低电平触发

GPIOIntEnable(GPIO\_PORTN\_BASE,GPIO\_INT\_PIN\_4);//中断使能

}

/////////////////////////////////////

//红外解码

bool getdata()

{

unsigned int i,j,IrAddressData,IrValueData;

volatile uint32\_t temp = 0x00;

IrAddressData = 0x00;

IrValueData = 0x00;

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

{

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

{

temp = temp>>1;

while((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x00)//检测到低电平

{

}

SysCtlDelay(8.4\*50000000/30000);//延时7ms

if((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x10)//1

{

temp = temp|0x80;//高位

while((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x10)//检测到高电平

{

count++;

if(count>0x200000)

{

count=0;

return 0;//超时，出错

}

}

}

else//0

{

temp = temp&0x7f;

}

}

IrData[i] = temp;

}

//abc = IrData[2];

IrAddressData = IrData[0]+IrData[1];

IrValueData = IrData[2]+IrData[3];

if((IrAddressData==0xff)&&(IrValueData==0xff))

return 1;//成功拿到数据

else

return 0;//发生错误

}

/////////////////////////////////////////////

//中断

void GPION(void)

{

volatile uint32\_t LowTime,HighTime;

unsigned long Status;

LowTime = 0;

HighTime = 0;

IntDisable(INT\_GPION);//关中断

Status = GPIOIntStatus(GPIO\_PORTN\_BASE,true);

if(Status==GPIO\_INT\_PIN\_4)//四号脚触发中断

{

SysCtlDelay(4\*50000000/3000);//4ms

if((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x00)//低

{

LowTime = 1;

while((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x00)//低

{

}

}

else

LowTime = 0;

SysCtlDelay(3\*50000000/3000);//3ms

if((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x10)//高

{

HighTime = 1;

while((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x10)

{

if(count>0x200000)

{

count=0;

//return 0;

}

}

}

else

HighTime = 0;

////////////////////////////////////////////////////同步码头成功获取后

if((HighTime==1)&&(LowTime==1))

{

if(getdata()==1)//解码

{

//IrOutput();

TFTLCD\_ShowData(100,100,IrData[2],CYAN,LIGHTBLUE);

}

else

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 , 0x02);//灯闪,报错，没有解密成功

SysCtlDelay(500\*(50000000/3000));//2

// for(i=0;i<1000;i++){;}

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 , 0x00);

SysCtlDelay(500\*(50000000/3000));//2

}

}

else

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 , 0x02);//没有拿到同步码头

SysCtlDelay(500\*(50000000/3000));

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 , 0x00);

SysCtlDelay(500\*(50000000/3000));

}

}

SysCtlDelay(50000000/3);

GPIOIntClear(GPIO\_PORTN\_BASE,Status);//清中断

}

uint32\_t g\_ui32SysClock;

int main(void)

{

g\_ui32SysClock=SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ |SYSCTL\_OSC\_MAIN | SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480), 50000000);

GPIOIntInitial();

GPIOInitial();

IntMasterEnable();//使能中断

EPIGPIOinit();

TFT\_400x240\_OTM4001Ainit(g\_ui32SysClock);//液晶屏初始化

I2C0GPIOBEnable\_m();//米字管 配置I2C0模块的IO引脚

I2C0DeviceInit();

// Open BackLight.打开背关

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);

TFTLCD\_ShowString(50,10,"WULAWULA",CYAN,LIGHTBLUE);

while(1)

{

if((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x00)//低电平

{

SysCtlDelay(2\*50000000/3000);//2ms，消抖

if((GPIOPinRead(GPIO\_PORTN\_BASE,GPIO\_PIN\_4)&0x10)==0x00)//低电平

IntEnable(INT\_GPION);//重新开中断

}

int mg=IrData[2];

int a=mg/1000;

mg=mg%1000;

I2C0TubeSelSet(0x1E);//选一

I2C0TubeLowSet(tubeCodeTable[a][0]);//

I2C0TubeHighSet(tubeCodeTable[a][1]);

SysCtlDelay(20000);

I2C0TubeSelSet(0xFF);//全选

I2C0TubeLowSet(0x00);//全暗

I2C0TubeHighSet(0x00);

SysCtlDelay(200);

a=mg/100;

mg=mg%100;

I2C0TubeSelSet(0x3C);//选一

I2C0TubeLowSet(tubeCodeTable[a][0]);//

I2C0TubeHighSet(tubeCodeTable[a][1]);

SysCtlDelay(20000);

I2C0TubeSelSet(0xFF);//全选

I2C0TubeLowSet(0x00);//全暗

I2C0TubeHighSet(0x00);

SysCtlDelay(2000);

a=mg/10;

mg=mg%10;

I2C0TubeSelSet(0x3A);//选一

I2C0TubeLowSet(tubeCodeTable[a][0]);//

I2C0TubeHighSet(tubeCodeTable[a][1]);

SysCtlDelay(20000);

I2C0TubeSelSet(0xFF);//全选

I2C0TubeLowSet(0x00);//全暗

I2C0TubeHighSet(0x00);

SysCtlDelay(2000);

a=mg;

I2C0TubeSelSet(0x36);//选一

I2C0TubeLowSet(tubeCodeTable[a][0]);//

I2C0TubeHighSet(tubeCodeTable[a][1]);

SysCtlDelay(20000);

I2C0TubeSelSet(0xFF);//全选

I2C0TubeLowSet(0x00);//全暗

I2C0TubeHighSet(0x00);

SysCtlDelay(2000);

}

}

**实验现象：**

通过遥控器上不同的按键，控制开发板上触控屏和米字管显示不同的数字。

**五、思考题**

**按下按键的时候有时候会指示错误，如何调整代码，提高精确度。**

在实验过程中，并未出现错误提示，可见原代码识别的精确度已经达到了较高的程度。若出现这种现象，可进行重复码过滤来提高精度。程序读取时间的不合理设置也会造成指示错误。

**六、实验体会与心得**

本次时单片机的最后一次实验，回顾以往实验发现单片机的每一次实验都有十分重要的实际应用价值，比如adc测温度、三轴加速度计的应用、TFT的使用和本次实验的红外收发。虽然在实验中实现起来较为简单，那是在硬件电路和软件都提供情况下，若要自己从头开始做一套红外收发装置可是十分麻烦的，所以单片机实验也仅仅只能起一个引导作用，真正想搞懂还是得课后自己多花时间去探讨。