目 录

1. 实验目的 1

2. 程序流程示意图 2

3. 代码及实现思路分析 3

3.1 测试项目和方法 3

3.2 测试的资源 3

3.3 测试结果及分析 3

4. 实验结果 4

4.1 对于实验二 4

4.2 对于实验三 4

5. 感想与收获 5

6. 讨论题 6

7. 源码 13

# 实验目的

了解I2C 总线标准及在TM4C1294 芯片的调用方法

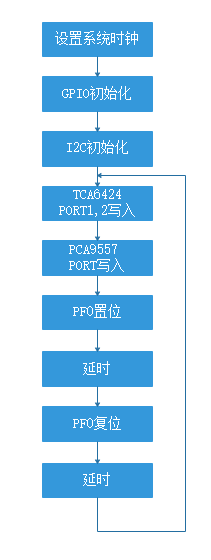
掌握用I2C总线扩展GPIO芯片PCA9557及TCA6424 的方法

能够通过扩展GPIO 来输出点亮LED 及动态数码管

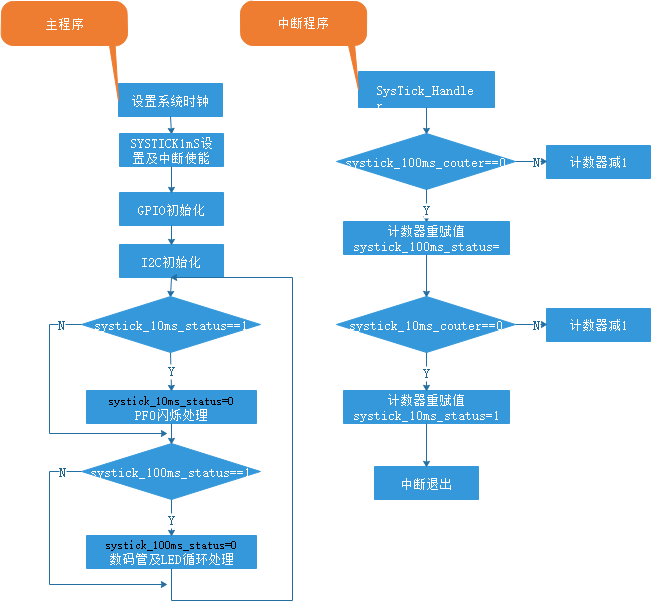
熟悉SYSTICK 中断调用方式，掌握利用软定时器模拟多任务切换的方法

# 程序流程示意图

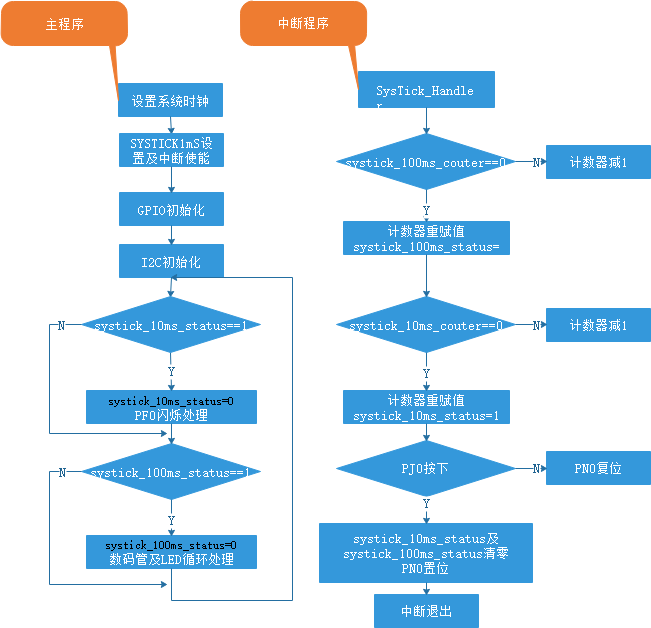
分别对应任务书中实验1、实验4、实验5



图一、实验一



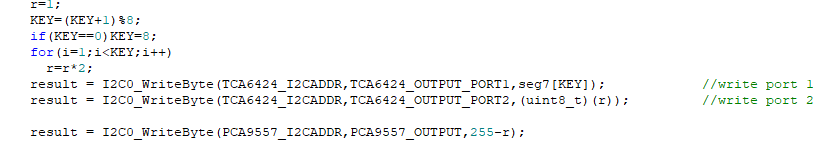
图二、实验四



图三、实验五

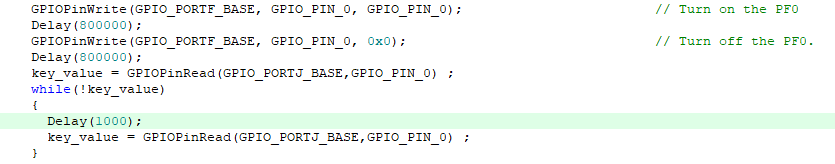
# 代码及实现思路分析

## 实验二



通过循环KEY来控制此时第几位的数码管的状态，对于第i个数码管，对应的数是，所以每次将r乘以2便可以实现换位操作。LED灯的状态与数码管相反（如LED灯第一位对应数码管最后一位），如要实现数码管与LED灯的同步控制，需要在控制第i为数码管()时控制第8-i位LED()。

## 实验三

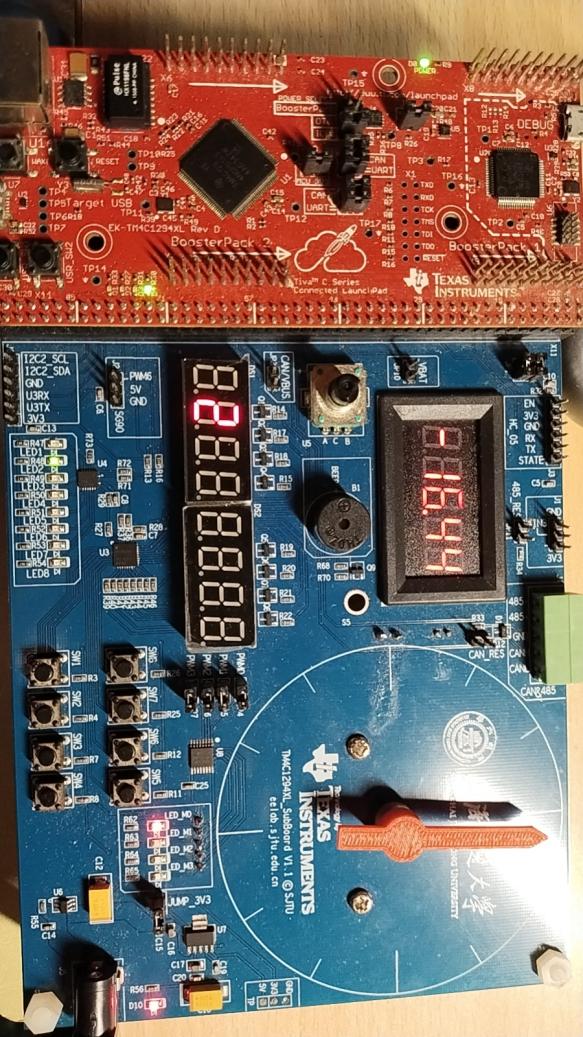


利用key\_value变量存储USR\_SW1的状态，并在检测到被按下后进入循环，循环的第一个操作是系统等待1000单位时间，由此完成：每隔1000单位时间就重新检查一遍此刻USR\_SW1的状态，如果仍然是按下状态，就继续循环；如果处于松开状态，就退出循环保持在走马灯状态（数码管从1到8不断循环，对应位置的LED灯随之亮起与熄灭）。

# 实验结果

## 对于实验二

进行LED的跑马灯实验，当LED在某位点亮时，同时在数码管的某位显示对应的LED管号。如LED跑马灯时，从左到右依次点亮LED1~LED8，此时在数码管上依次显示1~8。



图四、走马灯，同步控制数码管与LED灯

## 对于实验三

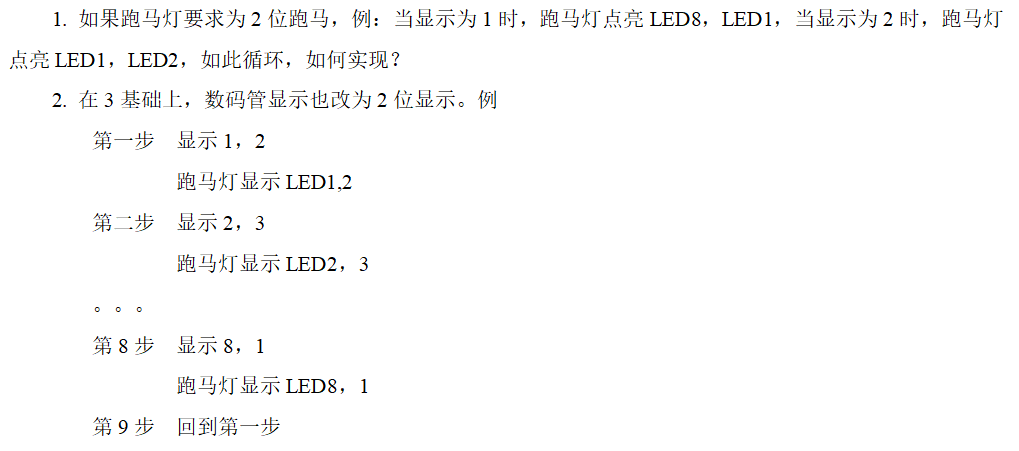
当按键USR\_SW1按下时，停止跑马灯，但LED及数码管显示维持不变；当按键松开后，继续跑马灯。

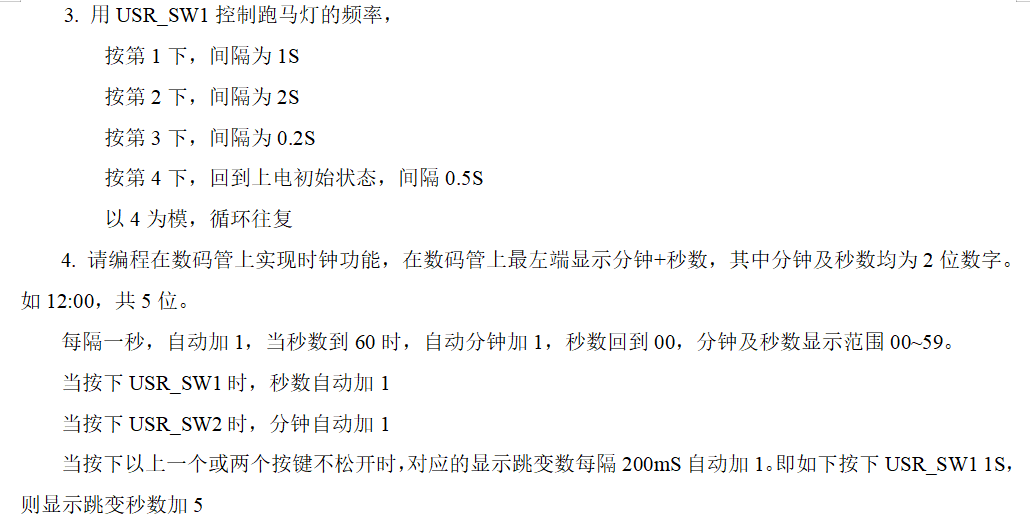
# 感想与收获

在这次实验中，笔者通过扩展GPIO完成了点亮LED与同步控制动态数码管的功能：维护一个变量（2的幂次方），将变量通过I2CO输入到端口中，起到依次在第i位输出数字i的效果，并且对应位置的LED也会同步亮灭。

笔者还通过每隔较短时间对USR\_SW1状态进行检测的方式，来维护延迟较低的暂停功能。将这种思想扩展到整个程序的上层，便可以实现中断操作：即在整个程序逻辑的上层维护一个不断循环的变量，在变量对应的不同状态检测不同按键的状态，并实时针对此刻的状态对整体的逻辑进行修改。

# 讨论题





1. 实现2位跑马灯

假设8个LED，分别编号为LED1, LED2, ..., LED8，我们需要每次点亮两个连续的LED，然后依次循环。

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/sysctl.h"

#include "driverlib/gpio.h"

// Assuming LEDs are connected to PORTF pins

#define LED\_PORT GPIO\_PORTF\_BASE

#define ALL\_LEDS (GPIO\_PIN\_0 | GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_6 | GPIO\_PIN\_7)

void initLEDs() {

// Enable the GPIO port

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// Configure the GPIO pins as output

GPIOPinTypeGPIOOutput(LED\_PORT, ALL\_LEDS);

}

void displayRunningLight(uint8\_t pos) {

// Clear all LEDs

GPIOPinWrite(LED\_PORT, ALL\_LEDS, 0);

// Calculate which two LEDs to light up

uint8\_t led1 = 1 << ((pos - 1) % 8);

uint8\_t led2 = 1 << (pos % 8);

// Set the corresponding LEDs

GPIOPinWrite(LED\_PORT, ALL\_LEDS, led1 | led2);

}

int main() {

uint8\_t position = 1;

initLEDs();

while (1) {

displayRunningLight(position);

position++;

if (position > 8) position = 1;

SysCtlDelay(SysCtlClockGet() / 3); // Adjust delay for your system clock

}

}

2. 数码管显示2位数并与跑马灯同步

假设有一个数码管显示驱动函数 displayDigits(uint8\_t leftDigit, uint8\_t rightDigit)

void displayDigits(uint8\_t leftDigit, uint8\_t rightDigit) {

// Implement your 7-segment display update logic here

}

int main() {

uint8\_t position = 1;

initLEDs();

while (1) {

displayRunningLight(position);

displayDigits(position, position % 8 + 1);

position++;

if (position > 8) position = 1;

SysCtlDelay(SysCtlClockGet() / 3); // Adjust delay for your system clock

}

}

3. 使用USR\_SW1控制跑马灯的频率

假设USR\_SW1连接到一个GPIO引脚并且有一个中断服务函数处理按键事件。

volatile uint8\_t buttonPressCount = 0;

volatile uint32\_t delayTime = SysCtlClockGet() / 2;

void buttonISR(void) {

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

buttonPressCount = (buttonPressCount + 1) % 4;

switch (buttonPressCount) {

case 0:

delayTime = SysCtlClockGet() / 2; // 0.5S

break;

case 1:

delayTime = SysCtlClockGet(); // 1S

break;

case 2:

delayTime = SysCtlClockGet() \* 2; // 2S

break;

case 3:

delayTime = SysCtlClockGet() / 5; // 0.2S

break;

}

}

void initButton() {

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_FALLING\_EDGE);

GPIOIntRegister(GPIO\_PORTF\_BASE, buttonISR);

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

}

int main() {

uint8\_t position = 1;

initLEDs();

initButton();

while (1) {

displayRunningLight(position);

displayDigits(position, position % 8 + 1);

position++;

if (position > 8) position = 1;

SysCtlDelay(delayTime);

}

}

4. 在数码管上实现时钟功能

volatile uint8\_t seconds = 0;

volatile uint8\_t minutes = 0;

void updateClock() {

seconds++;

if (seconds >= 60) {

seconds = 0;

minutes++;

if (minutes >= 60) {

minutes = 0;

}

}

}

void button1ISR(void) {

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

seconds++;

if (seconds >= 60) {

seconds = 0;

}

}

void button2ISR(void) {

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);

minutes++;

if (minutes >= 60) {

minutes = 0;

}

}

void initButtons() {

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

GPIOPinTypeGPIOInput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2);

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_FALLING\_EDGE);

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_FALLING\_EDGE);

GPIOIntRegister(GPIO\_PORTF\_BASE, button1ISR);

GPIOIntRegister(GPIO\_PORTF\_BASE, button2ISR);

GPIOIntEnable(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2);

}

void displayClock() {

// Assuming you have a function to display the time on a 7-segment display

displayDigits(minutes / 10, minutes % 10);

displayDigits(seconds / 10, seconds % 10);

}

int main() {

initLEDs();

initButtons();

while (1) {

displayClock();

SysCtlDelay(SysCtlClockGet()); // 1 second delay

updateClock();

}

}

按键长按处理：可以设置一个标志和一个计时器来检测按键是否按住并进行相应的处理。

volatile bool button1Pressed = false;

volatile bool button2Pressed = false;

volatile uint32\_t button1PressTime = 0;

volatile uint32\_t button2PressTime = 0;

void button1ISR(void) {

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

if (GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_1) == 0) {

button1Pressed = true;

button1PressTime = SysCtlClockGet();

} else {

button1Pressed = false;

}

}

void button2ISR(void) {

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);

if (GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_2) == 0) {

button2Pressed = true;

button2PressTime = SysCtlClockGet();

} else {

button2Pressed = false;

}

}

void checkLongPress() {

if (button1Pressed && (SysCtlClockGet() - button1PressTime) >= (SysCtlClockGet() / 5)) {

seconds++;

if (seconds >= 60) {

seconds = 0;

}

button1PressTime = SysCtlClockGet();

}

if (button2Pressed && (SysCtlClockGet() - button2PressTime) >= (SysCtlClockGet() / 5)) {

minutes++;

if (minutes >= 60) {

minutes = 0;

}

button2PressTime = SysCtlClockGet();

}

}

int main() {

initLEDs();

initButtons();

while (1) {

displayClock();

SysCtlDelay(SysCtlClockGet() / 5); // 200 ms delay

updateClock();

checkLongPress();

}

}

# 源码

#include <stdint.h>

#include <stdbool.h>

#include "hw\_memmap.h"

#include "debug.h"

#include "gpio.h"

#include "hw\_i2c.h"

#include "hw\_types.h"

#include "i2c.h"

#include "pin\_map.h"

#include "sysctl.h"

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

//

//I2C GPIO chip address and resigster define

//

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

#define TCA6424\_I2CADDR                     0x22

#define PCA9557\_I2CADDR                     0x18

#define PCA9557\_INPUT                           0x00

#define PCA9557\_OUTPUT                      0x01

#define PCA9557\_POLINVERT                   0x02

#define PCA9557\_CONFIG                      0x03

#define TCA6424\_CONFIG\_PORT0            0x0c

#define TCA6424\_CONFIG\_PORT1            0x0d

#define TCA6424\_CONFIG\_PORT2            0x0e

#define TCA6424\_INPUT\_PORT0             0x00

#define TCA6424\_INPUT\_PORT1             0x01

#define TCA6424\_INPUT\_PORT2             0x02

#define TCA6424\_OUTPUT\_PORT0            0x04

#define TCA6424\_OUTPUT\_PORT1            0x05

#define TCA6424\_OUTPUT\_PORT2            0x06

void        Delay(uint32\_t value);

void        S800\_GPIO\_Init(void);

uint8\_t     I2C0\_WriteByte(uint8\_t DevAddr, uint8\_t RegAddr, uint8\_t WriteData);

uint8\_t     I2C0\_ReadByte(uint8\_t DevAddr, uint8\_t RegAddr);

void        S800\_I2C0\_Init(void);

volatile uint8\_t result;

uint32\_t ui32SysClock,KEY, key\_value;

uint8\_t seg7[] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x58,0x5e,0x079,0x71,0x5c};

int i,r;

int main(void)

{

    //use internal 16M oscillator, HSI

    ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_16MHZ |SYSCTL\_OSC\_INT |SYSCTL\_USE\_OSC), 16000000);

    S800\_GPIO\_Init();

    S800\_I2C0\_Init();

    KEY=0;

    r=1;

    while (1)

    {

        r=1;

        KEY=(KEY+1)%8;

        if(KEY==0)KEY=8;

        for(i=1;i<KEY;i++)

            r=r\*2;

        result = I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_OUTPUT\_PORT1,seg7[KEY]);                        //write port 1

        result = I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_OUTPUT\_PORT2,(uint8\_t)(r));                 //write port 2

        result = I2C0\_WriteByte(PCA9557\_I2CADDR,PCA9557\_OUTPUT,255-r);

        GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);                                              // Turn on the PF0

        Delay(800000);

        GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, 0x0);                                                             // Turn off the PF0.

        Delay(800000);

        key\_value = GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0) ;

        while(!key\_value)

        {

            Delay(1000);

            key\_value = GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0) ;

        }

    }

}

void Delay(uint32\_t value)

{

    uint32\_t ui32Loop;

    for(ui32Loop = 0; ui32Loop < value; ui32Loop++){};

}

void S800\_GPIO\_Init(void)

{

    SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);                        //Enable PortF

    while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOF));         //Wait for the GPIO moduleF ready

    SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOJ);                        //Enable PortJ

    while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOJ));         //Wait for the GPIO moduleJ ready

  GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);           //Set PF0 as Output pin

    GPIOPinTypeGPIOInput(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0 | GPIO\_PIN\_1);//Set the PJ0,PJ1 as input pin

    GPIOPadConfigSet(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0 | GPIO\_PIN\_1,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU);

}

void S800\_I2C0\_Init(void)

{

    uint8\_t result;

  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_I2C0);

  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOB);

    GPIOPinConfigure(GPIO\_PB2\_I2C0SCL);

  GPIOPinConfigure(GPIO\_PB3\_I2C0SDA);

  GPIOPinTypeI2CSCL(GPIO\_PORTB\_BASE, GPIO\_PIN\_2);//I2C  GPIO\_PIN\_2    SCL

  GPIOPinTypeI2C(GPIO\_PORTB\_BASE, GPIO\_PIN\_3);//I2C  GPIO\_PIN\_3    SDA

    I2CMasterInitExpClk(I2C0\_BASE,ui32SysClock, true);                                      //config I2C0 400k

    I2CMasterEnable(I2C0\_BASE);

    result = I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_CONFIG\_PORT0,0x0ff);        //config port 0 as input

    result = I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_CONFIG\_PORT1,0x0);          //config port 1 as output

    result = I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_CONFIG\_PORT2,0x0);          //config port 2 as output

    result = I2C0\_WriteByte(PCA9557\_I2CADDR,PCA9557\_CONFIG,0x00);                   //config port as output

    result = I2C0\_WriteByte(PCA9557\_I2CADDR,PCA9557\_OUTPUT,0x0ff);              //turn off the LED1-8

}

uint8\_t I2C0\_WriteByte(uint8\_t DevAddr, uint8\_t RegAddr, uint8\_t WriteData)

{

    uint8\_t rop;

    while(I2CMasterBusy(I2C0\_BASE)){};

    I2CMasterSlaveAddrSet(I2C0\_BASE, DevAddr, false);

    I2CMasterDataPut(I2C0\_BASE, RegAddr);

    I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START);

    while(I2CMasterBusy(I2C0\_BASE)){};

    rop = (uint8\_t)I2CMasterErr(I2C0\_BASE);//

    I2CMasterDataPut(I2C0\_BASE, WriteData);

    I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_FINISH);

    while(I2CMasterBusy(I2C0\_BASE)){};

    rop = (uint8\_t)I2CMasterErr(I2C0\_BASE);//

    return rop;//

}

uint8\_t I2C0\_ReadByte(uint8\_t DevAddr, uint8\_t RegAddr)

{

    uint8\_t value,rop;

    while(I2CMasterBusy(I2C0\_BASE)){};

    I2CMasterSlaveAddrSet(I2C0\_BASE, DevAddr, false);

    I2CMasterDataPut(I2C0\_BASE, RegAddr);

//  I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START);

    I2CMasterControl(I2C0\_BASE,I2C\_MASTER\_CMD\_SINGLE\_SEND);//

    while(I2CMasterBusBusy(I2C0\_BASE));

    rop = (uint8\_t)I2CMasterErr(I2C0\_BASE);

    Delay(1);

    //receive data

    I2CMasterSlaveAddrSet(I2C0\_BASE, DevAddr, true);//

    I2CMasterControl(I2C0\_BASE,I2C\_MASTER\_CMD\_SINGLE\_RECEIVE);//

    while(I2CMasterBusBusy(I2C0\_BASE));

    value=I2CMasterDataGet(I2C0\_BASE);//

        Delay(1);

    return value;

}