嵌入式系统与接口技术
（信工电科教学班）
实验报告

实验 2　I2C GPIO 扩展及 SYSTICK 中断实验

成员姓名：强陶

完成时间：2024 年 5 月 27 日

上海交通大学　电子信息与电气工程学院

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 4

3.1 测试项目和方法 .... **错误！未定义书签。**

3.2 测试的资源 .... **错误！未定义书签。**

3.3 测试结果及分析 .... **错误！未定义书签。**

4. 实验结果 5

4.1 对于实验二 5

4.2 对于实验三 5

5. 感想与收获 6

6. 讨论题 7

7. 源码 14

1. 实验目的

了解 I2C 总线标准及在 TM4C1294 芯片的调用方法

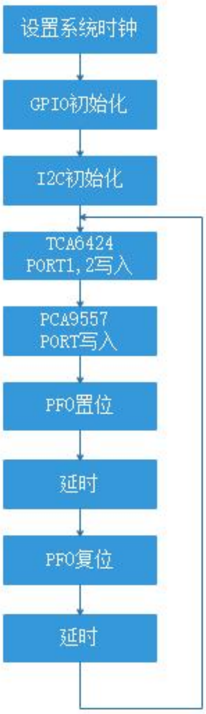
掌握用 I2C 总线扩展 GPIO 芯片 PCA9557 及 TCA6424 的方法

能够通过扩展 GPIO 来输出点亮 LED 及动态数码管

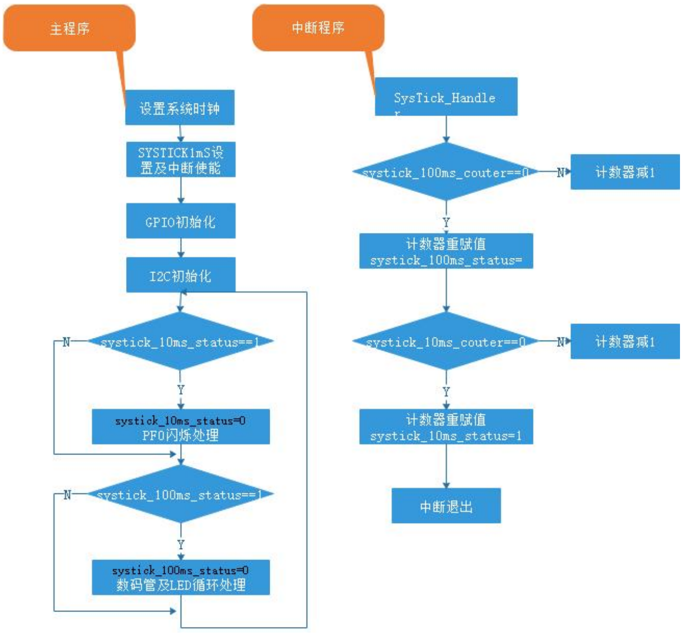
熟悉 SYSTICK 中断调用方式，掌握利用软定时器模拟多任务切换的方法

2. 程序流程示意图

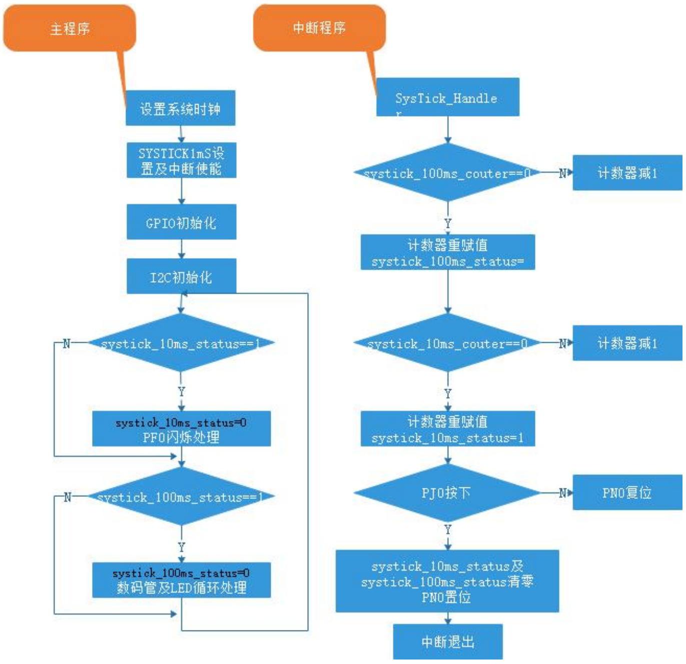
分别对应任务书中实验 1、实验 4、实验 5



图一、实验一



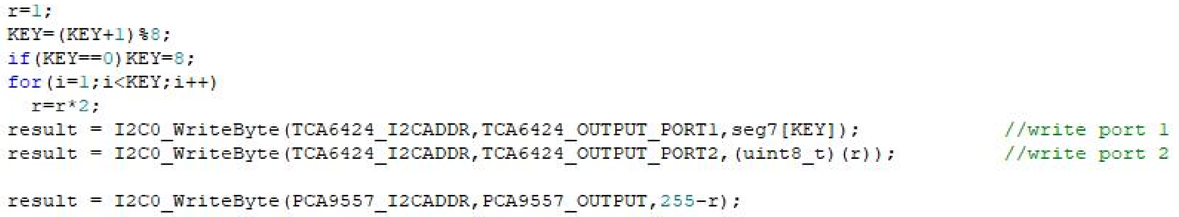
图二、实验四


主程序
设置系统时钟
SYSTICK1mS设置及中断使能
GPIO初始化
I2C初始化
systick_10ms_status==1
N
Y
systick_10ms_status=0
PF0闪烁处理
systick_100ms_status==1
N
Y
systick_100ms_status=0
数码管及LED循环处理
中断程序
SysTick_Handler
systick_100ms_couter==0
N
计数器减1
Y
计数器重赋值
systick_100ms_status=
systick_10ms_couter==0
N
计数器减1
Y
计数器重赋值
systick_10ms_status=1
PJ0按下
N
PN0复位
Y
systick_10ms_status及
systick_100ms_status清零
PN0置位
中断退出

图三、实验五

3. 代码及实现思路分析

3.1 实验二

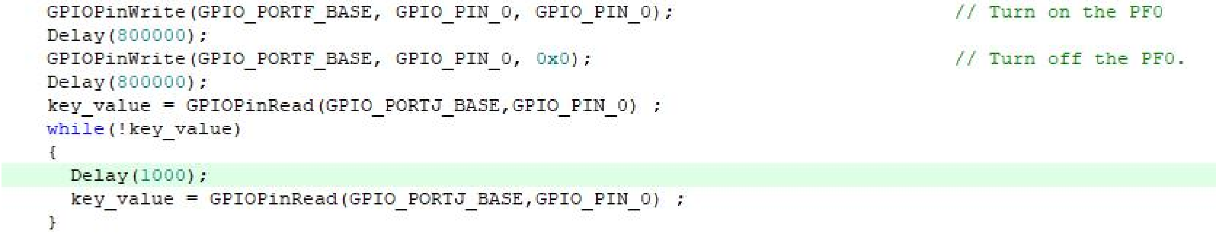


```
r=1;
KEY=(KEY+1)%8;
if(KEY==0)KEY=8;
for(i=1;i<KEY;i++)
  r=r*2;
result = I2C0_WriteByte(TCA6424_I2CADDR,TCA6424_OUTPUT_PORT1,seg7[KEY]);            //write port 1
result = I2C0_WriteByte(TCA6424_I2CADDR,TCA6424_OUTPUT_PORT2,(uint8_t)(r));         //write port 2

result = I2C0_WriteByte(PCA9557_I2CADDR,PCA9557_OUTPUT,255-r);
```

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2^i，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(2^i)时控制第 8-i 位 LED$(255-2^i)$。

3.2 实验三



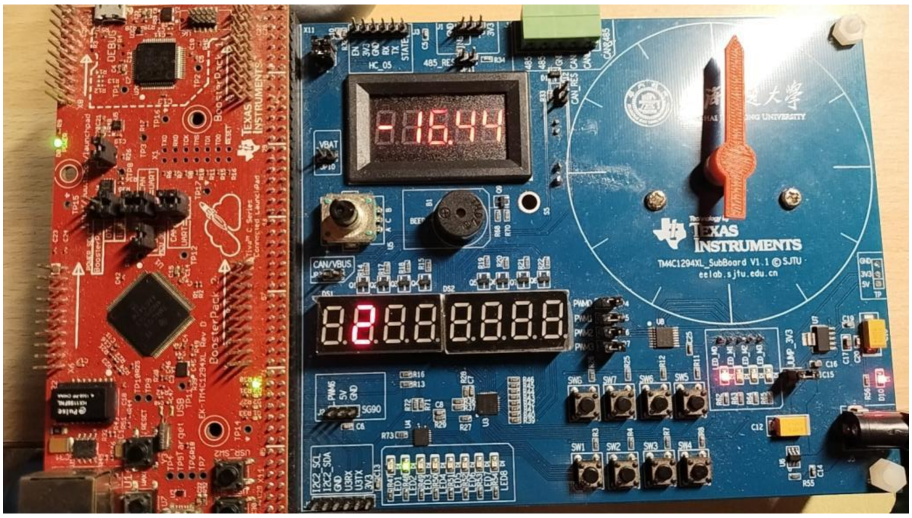
```
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);                        // Turn on the PF0
Delay(800000);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);                               // Turn off the PF0.
Delay(800000);
key_value = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0) ;
while(!key_value)
{
  Delay(1000);
  key_value = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0) ;
}
```

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）。

4. 实验结果

4.1 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

4.2 对于实验三

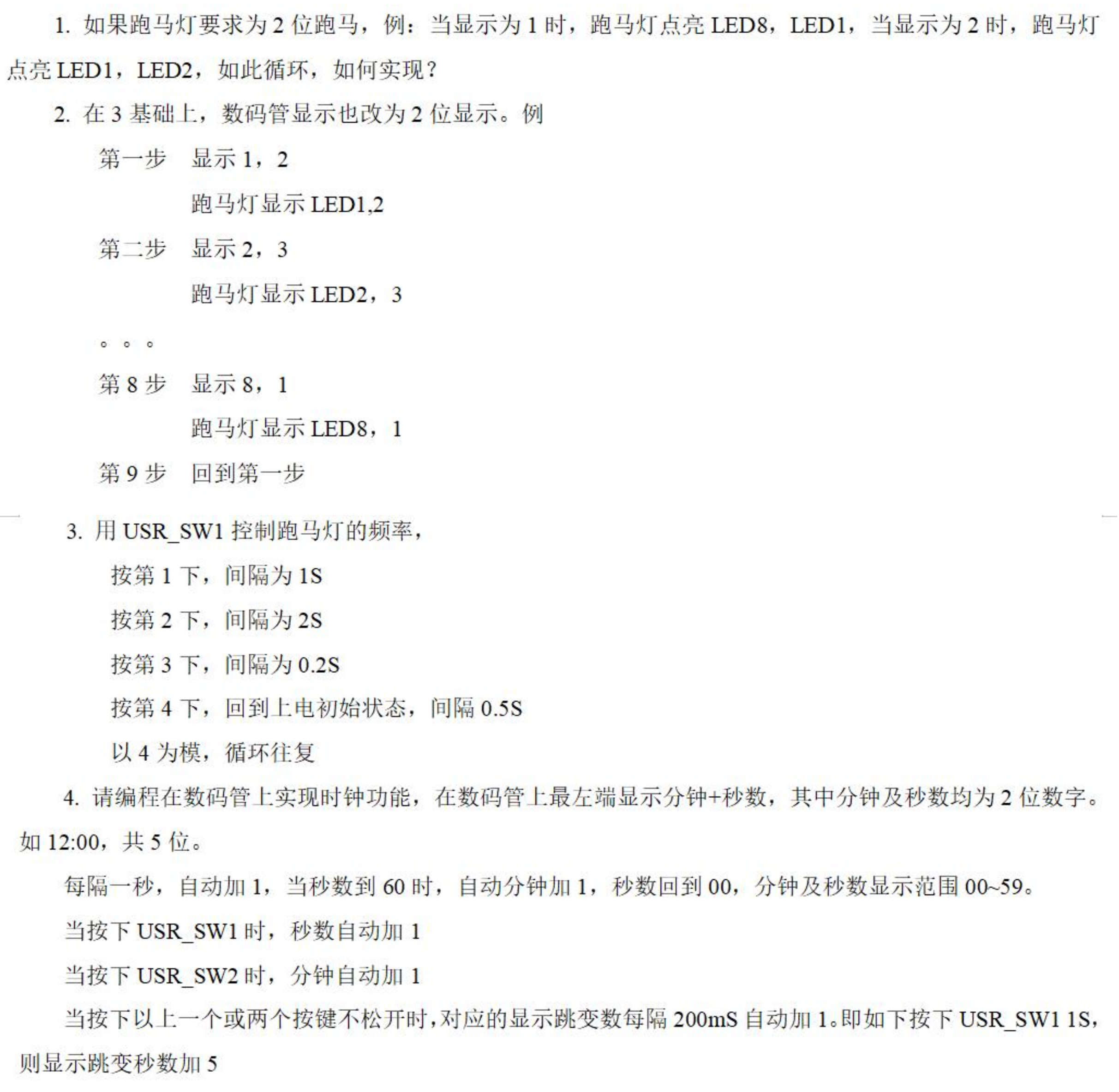
当按键 USR_SW1 按下时，停止跑马灯，但 LED 及数码管显示维持不变；当按键松开后，继续跑马灯。

5. 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。

6. 讨论题



1. 如果跑马灯要求为 2 位跑马，例：当显示为 1 时，跑马灯点亮 LED8，LED1，当显示为 2 时，跑马灯点亮 LED1，LED2，如此循环，如何实现？

2. 在 3 基础上，数码管显示也改为 2 位显示。例

第一步　显示 1，2
　　　　跑马灯显示 LED1,2
第二步　显示 2，3
　　　　跑马灯显示 LED2，3
。。。
第 8 步　显示 8，1
　　　　跑马灯显示 LED8，1
第 9 步　回到第一步

3. 用 USR_SW1 控制跑马灯的频率，

按第 1 下，间隔为 1S
按第 2 下，间隔为 2S
按第 3 下，间隔为 0.2S
按第 4 下，回到上电初始状态，间隔 0.5S
以 4 为模，循环往复

4. 请编程在数码管上实现时钟功能，在数码管上最左端显示分钟+秒数，其中分钟及秒数均为 2 位数字。如 12:00，共 5 位。

每隔一秒，自动加 1，当秒数到 60 时，自动分钟加 1，秒数回到 00，分钟及秒数显示范围 00~59。

当按下 USR_SW1 时，秒数自动加 1

当按下 USR_SW2 时，分钟自动加 1

当按下以上一个或两个按键不松开时，对应的显示跳变数每隔 200mS 自动加 1。即如下按下 USR_SW1 1S，则显示跳变秒数加 5

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();
    }
}
```

7. 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;
}
```