嵌入式系统与接口技术
（信工电科教学班）
实验报告

实验 3　UART 串行通讯口及中断优先级实验

成员姓名：强陶

完成时间：2024 年 5 月 28 日

上海交通大学 电子信息与电气工程学院

目　　录

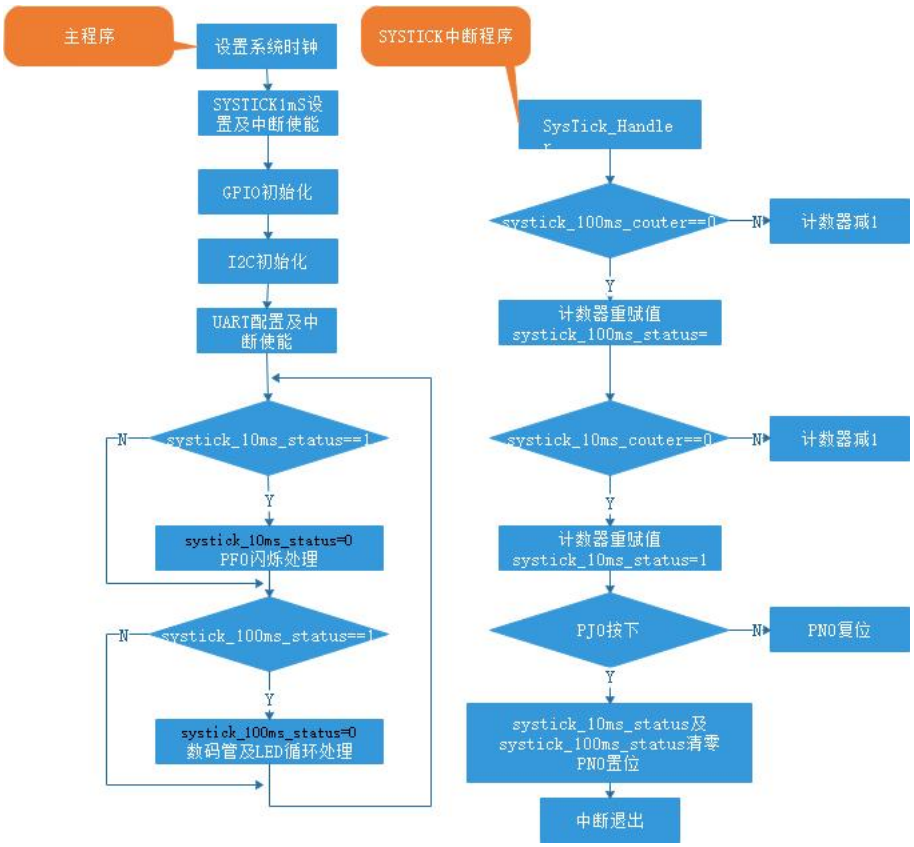
1. 实验目的

了解 UART 串行通讯的工作原理

掌握在 PC 端通过串口调试工具与实验板通过 UART 通讯的方法

掌握 UART 的堵塞式与非堵塞式通讯方法

2. 程序流程示意图



图一、主程序与 SYSTICK 中断程序逻辑

图二、UARTO 中断程序逻辑

3. 代码及实现思路分析

```
void UART0_Handler(void)
{ uint8_t cnt = 0,flag1=1,flag2=1;
  uint32_t ulStatus;
  ulStatus = UARTIntStatus(UART0_BASE, true);
  UARTIntClear(UART0_BASE, ulStatus);

  while( UARTCharsAvail(UART0_BASE) )  {
    RxBuf[cnt++]= UARTCharGetNonBlocking(UART0_BASE);
  }
   RxBuf[cnt]='\0';
  if(strcmp(RxBuf, KEY1)==0)UARTStringPut((uint8_t *)"\r\nCLASS2212\n\r");
  if(strcmp(RxBuf, KEY2)==0)UARTStringPut((uint8_t *)"\r\nCODE522031910206\n\r");
}
```

使用 UARTCharsAvail(UART0_BASE)检查 UART 接收 FIFO 是否有可用数据，如果有的话就通过 UARTCharGetNonBlocking 接收并存储到 RxBuf 数组中。当输入全部被接收后，就使用 strcmp 函数比较接收到的字符串与“AT+CLASS”和“AT+STUDENTCODE”是否相同，相同便作出对应的输出。

```
while( UARTCharsAvail(UART0_BASE) )  {
  RxBuf[cnt++]= UARTCharGetNonBlocking(UART0_BASE);
}
 RxBuf[cnt]='\0';
```

使用了非阻塞的方式接收字符串。

```
void S800_UART_Init(void)
{
  SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);            //Enable PortA
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA));     //Wait for the GPIO moduleA ready

  GPIOPinConfigure(GPIO_PA0_U0RX);                        // Set GPIO A0 and A1 as UART pins.
  GPIOPinConfigure(GPIO_PA1_U0TX);

  GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

  // Configure the UART for 115,200, 8-N-1 operation.
  UARTConfigSetExpClk(UART0_BASE, ui32SysClock,115200,(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |UART_CONFIG_PAR_NONE));
  UARTFIFOLevelSet(UART0_BASE,UART_FIFO_RX1_8,UART_FIFO_RX7_8);
  UARTStringPut((uint8_t *)"\r\nHello, world!\r\n");
}
```

在初始化函数中加入 UARTFIFOLevelSet(UART0_BASE,UART_FIFO_RX1_8,UART_FIFO_RX7_8)，便可以成功讲 AT+STUDENTCODE 发送给串口通信模拟器并被接收。

4. 实验结果

按下 RESET 键后，实验板回以 Hello, world!

图三、按下 RESET

当 PC 端发来 AT+CLASS 后，实验板回以 CLASS2212

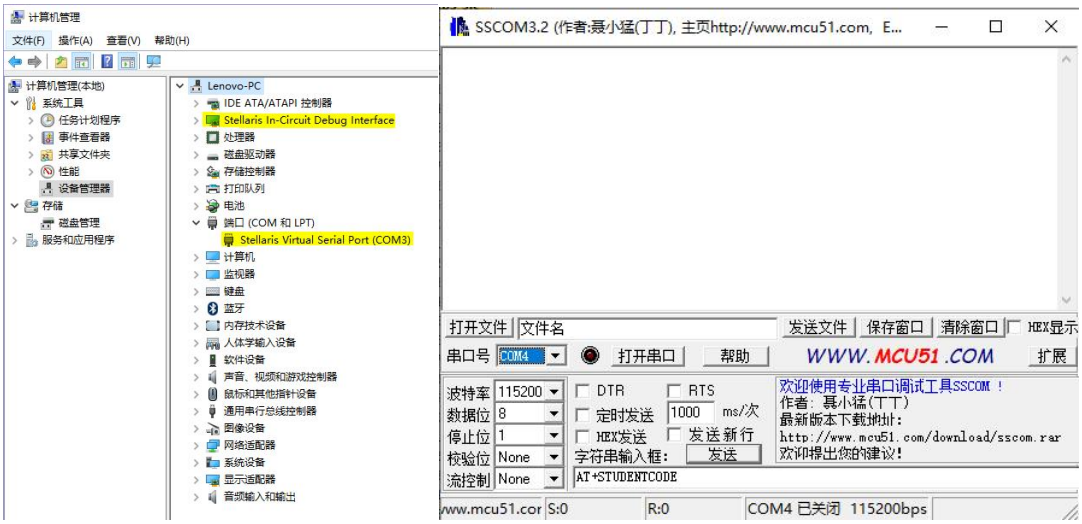
图三、发送 AT+CLASS

当 PC 端发来 AT+STUDENTCODE 后，实验板回以 CODE522031910206

图四、发送 AT+STUDENTCODE

5. 感想与收获

学会了串口通信模拟器的使用方式：



图五、配置虚拟串口通信模拟器

在虚拟机中讲串口号改为 COM4,将波特率改为 115200 便可开始模拟通信。左下方的 S 代表发出的字节数，左下方的 R 代表读取到的字节数。

学会了使用库函数与 UARTFIFOLevelSet 函数来简化代码逻辑，实现性能更强的通信：

将传输的信息通过 SSCOM 的信息输入框发送给主板。在初始化函数中加入 UARTFIFOLevelSet(UART0_BASE,UART_FIFO_RX1_8,UART_FIFO_RX7_8)函数保证可以发送较长的文本。通过非阻塞的方式接收字符串后，利用 C 语言库中的 strcmp 函数来进行字符串的比较。并做出对应的返回。

6. 讨论题

1. 实验 3-2，if (UARTCharsAvail(UART0_BASE))此行程序的作用。如果没有此行，会导致什么问题？
2. 实验 3-3，void UART0_Handler(void)为什么没有在主函数声明？
3. 为什么 3-3 的中断中需要读取中断标志并清除，而 SYSTICK 不需要
4. 请根据上位机的命令，如“MAY+01”，格式为：
其中 MAY 为月份，（JAN,FEB,…DEC）均为三位。
+表示加运算符,-表示减运算符，均为 1 位。
01 表示增加或减少量，均为 2 位。范围 00-11
以上均为 ASCII 码，
MAY+01 应该回之以 JUNE
MAY-06 应该回之以 NOV

5. 请根据上位机的命令，如“14:12+05:06”，格式为：
其中 14:12 为分钟与秒，共 5 位，包括一个“:”。
+表示加运算符,-表示减运算符，均为 1 位。
05:06 为分钟与秒的变化量，共 5 位。包括一个“：”，范围 00:00~23:59
以上均为 ASCII 码，
14:12+05:06 回之以 19:18

1. 作用：
UARTCharsAvail(UART0_BASE)用于检查 UART 接收 FIFO 是否有可用数据。如果没有这行代码，会导致以下问题：
(1).数据读取错误：直接读取可能会发生在没有数据可读的情况下，读取到无效或错误数据。
(2).程序崩溃或陷入死循环：如果读取操作依赖于此检查，缺少它会导致程序读取错误地址或数据，可能崩溃或进入死循环。

2. 没有在主函数声明的原因：
中断处理函数 UART0_Handler 不需要在主函数中声明，因为它是在启动文件或系统中断向量表中声明和定义的。这是一个惯例，中断处理函数由硬件中断向量表直接调用。

3. 3-3 的中断中需要读取中断标志并清除，而 SYSTICK 不需要的原因:
(1).UART 中断：UART 中断标志在触发后需要被清除，以防止中断处理程序反复处理相同的中断事件。通常是通过读取相关寄存器并进行清除操作。

(2).SysTick 中断：SysTick 计时器在每次溢出时自动触发中断，不需要手动清除中断标志。SysTick 定时器自动重装和溢出处理机制不需要用户代码清除标志。

4.

```
#include <stdio.h>
#include <string.h>
const char *months[] = {
   "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"
};

int getMonthIndex(const char *month) {
   for (int i = 0; i < 12; i++) {
      if (strncmp(months[i], month, 3) == 0) {
         return i;
      }
   }
   return -1;  // 错误月份
}

void processMonthCommand(const char *command) {
   char month[4] = {0};
   strncpy(month, command, 3);
   char op = command[3];
   int value = (command[4] - '0') * 10 + (command[5] - '0');

   int monthIndex = getMonthIndex(month);
   if (monthIndex == -1) {
      printf("Invalid month\n");
      return;
   }

   if (op == '+') {
      monthIndex = (monthIndex + value) % 12;
   } else if (op == '-') {
      monthIndex = (monthIndex - value + 12) % 12;
   } else {
```

```
        printf("Invalid operation\n");
        return;
    }

    printf("Result: %s\n", months[monthIndex]);
}

int main() {
    processMonthCommand("MAY+01");  // Expected output: JUNE
    processMonthCommand("MAY-06");  // Expected output: NOV
    return 0;
}
```

5.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 解析时间字符串 "MM:SS"
void parseTime(const char *timeStr, int *minutes, int *seconds) {
    *minutes = (timeStr[0] - '0') * 10 + (timeStr[1] - '0');
    *seconds = (timeStr[3] - '0') * 10 + (timeStr[4] - '0');
}

// 将分钟和秒数转换回字符串 "MM:SS"
void formatTime(int minutes, int seconds, char *timeStr) {
    snprintf(timeStr, 6, "%02d:%02d", minutes, seconds);
}

void processTimeCommand(const char *command) {
    int minutes1, seconds1;
    int minutes2, seconds2;
    char result[6];

    parseTime(command, &minutes1, &seconds1);
    parseTime(command + 6, &minutes2, &seconds2);
```

```
    char op = command[5];

    int totalSeconds1 = minutes1 * 60 + seconds1;
    int totalSeconds2 = minutes2 * 60 + seconds2;
    int resultSeconds;

    if (op == '+') {
        resultSeconds = totalSeconds1 + totalSeconds2;
    } else if (op == '-') {
        resultSeconds = totalSeconds1 - totalSeconds2;
    } else {
        printf("Invalid operation\n");
        return;
    }

    if (resultSeconds < 0) {
        resultSeconds += 24 * 60 * 60;  // 处理负时间
    }

    int resultMinutes = (resultSeconds / 60) % (24 * 60);
    resultSeconds = resultSeconds % 60;

    formatTime(resultMinutes, resultSeconds, result);
    printf("Result: %s\n", result);
}

int main() {
    processTimeCommand("14:12+05:06");  // Expected output: 19:18
    return 0;
}
```

7. 源码

```
#include <stdint.h>
#include <stdbool.h>
#include <string.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"
#include "systick.h"
#include "interrupt.h"
#include "uart.h"
#include "hw_ints.h"

#define SYSTICK_FREQUENCY                1000                //1000hz

#define  I2C_FLASHTIME                          500                          //500mS
#define GPIO_FLASHTIME                  300                          //300mS
//*****************************************************************************
//
//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);
void            S800_UART_Init(void);
//systick software counter define
volatile uint16_t systick_10ms_couter,systick_100ms_couter;
volatile uint8_t   systick_10ms_status,systick_100ms_status;

volatile uint8_t result,cnt,key_value,gpio_status;
volatile uint8_t rightshift = 0x01;
uint32_t ui32SysClock;
uint8_t                                         seg7[]                                  =
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x58,0x5e,0x079,0x71,0x5c};
uint8_t uart_receive_char;
char *uart_receive_string;
char *KEY1="AT+CLASS";
char *KEY2="AT+STUDENTCODE";
char RxBuf[100];

int main(void)
```

```
{
        volatile uint16_t i2c_flash_cnt,gpio_flash_cnt;
        //use internal 16M oscillator, PIOSC
  //ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_16MHZ |SYSCTL_OSC_INT
|SYSCTL_USE_OSC), 16000000);
        //ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_16MHZ |SYSCTL_OSC_INT
|SYSCTL_USE_OSC), 8000000);
        //use external 25M oscillator, MOSC
  //ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |SYSCTL_OSC_MAIN
|SYSCTL_USE_OSC), 25000000);

        //use external 25M oscillator and PLL to 120M
  //ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |SYSCTL_OSC_MAIN |
SYSCTL_USE_PLL |SYSCTL_CFG_VCO_480), 120000000);;
        ui32SysClock = SysCtlClockFreqSet((SYSCTL_OSC_INT | SYSCTL_USE_PLL
|SYSCTL_CFG_VCO_480), 20000000);

  SysTickPeriodSet(ui32SysClock/SYSTICK_FREQUENCY);
        SysTickEnable();
        SysTickIntEnable();
                                                        //Enable Systick interrupt



        S800_GPIO_Init();
        S800_I2C0_Init();
        S800_UART_Init();

        IntEnable(INT_UART0);
  UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);      //Enable UART0 RX,TX
interrupt
  IntMasterEnable();

        while (1)
        {
                if (systick_10ms_status)
                {
```

```
                systick_10ms_status             = 0;
                if (++gpio_flash_cnt    >= GPIO_FLASHTIME/10)
                {
                        gpio_flash_cnt                  = 0;
                        if (gpio_status)
                                GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_0,GPIO_PIN_0 );
                        else
                                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0,0);
                        gpio_status                                     = !gpio_status;

                }
        }
        if (systick_100ms_status)
        {
                systick_100ms_status    = 0;
                if (++i2c_flash_cnt             >= I2C_FLASHTIME/100)
                {
                        i2c_flash_cnt                   = 0;
                        result                                          =
I2C0_WriteByte(TCA6424_I2CADDR,TCA6424_OUTPUT_PORT1,seg7[cnt+1]);  //write port 1

                        result                                          =
I2C0_WriteByte(TCA6424_I2CADDR,TCA6424_OUTPUT_PORT2,rightshift);    //write port 2

                        result                                                  =
I2C0_WriteByte(PCA9557_I2CADDR,PCA9557_OUTPUT,~rightshift);

                        cnt++;
                        rightshift= rightshift<<1;

                        if (cnt             >= 0x8)
                        {
                                rightshift= 0x01;
                                cnt                     = 0;
                        }
```

```
                }
            }
        }


}

void Delay(uint32_t value)
{
        uint32_t ui32Loop;
        for(ui32Loop = 0; ui32Loop < value; ui32Loop++){};
}


void UARTStringPut(uint8_t *cMessage)
{
        while(*cMessage!='\0')
                UARTCharPut(UART0_BASE,*(cMessage++));
}
void UARTStringPutNonBlocking(const char *cMessage)
{
        while(*cMessage!='\0')
                UARTCharPutNonBlocking(UART0_BASE,*(cMessage++));
}

void S800_UART_Init(void)
{
        SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
        //Enable PortA
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA));                    //Wait for the
GPIO moduleA ready
```

```
        GPIOPinConfigure(GPIO_PA0_U0RX);
                                        // Set GPIO A0 and A1 as UART pins.
 GPIOPinConfigure(GPIO_PA1_U0TX);

 GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

        // Configure the UART for 115,200, 8-N-1 operation.
 UARTConfigSetExpClk(UART0_BASE,         ui32SysClock,115200,(UART_CONFIG_WLEN_8         |
UART_CONFIG_STOP_ONE |UART_CONFIG_PAR_NONE));
        UARTFIFOLevelSet(UART0_BASE,UART_FIFO_RX1_8,UART_FIFO_RX7_8);
        UARTStringPut((uint8_t *)"\r\nHello, world!\r\n");
}
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
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
        //Enable PortN
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPION));                    //Wait for the
GPIO moduleN ready

 GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);                  //Set PF0 as Output
pin
 GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0);                  //Set PN0 as Output
pin
 GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);          //Set PN1 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);
 GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

        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;
}

/*
```

```
        Corresponding to the startup_TM4C129.s vector table systick interrupt program name
*/
void SysTick_Handler(void)
{
        if (systick_100ms_couter != 0)
                systick_100ms_couter--;
        else
        {
                systick_100ms_couter    = SYSTICK_FREQUENCY/10;
                systick_100ms_status    = 1;
        }

        if (systick_10ms_couter  != 0)
                systick_10ms_couter--;
        else
        {
                systick_10ms_couter             = SYSTICK_FREQUENCY/100;
                systick_10ms_status     = 1;
        }
        if (GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0) == 0)
        {
                systick_100ms_status    = systick_10ms_status = 0;
                GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,GPIO_PIN_0);
        }
        else
                GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,0);
}

/*
        Corresponding to the startup_TM4C129.s vector table UART0_Handler interrupt program name
*/

/*
void UART0_Handler(void)
{
        int32_t uart0_int_status;
```

```
 uart0_int_status                = UARTIntStatus(UART0_BASE, true);          // Get the interrrupt
status.

 UARTIntClear(UART0_BASE, uart0_int_status);
                //Clear the asserted interrupts

 while(UARTCharsAvail(UART0_BASE))
                                // Loop while there are characters in the receive FIFO.
 {
                ///Read the next character from the UART and write it back to the UART.
  //UARTCharPutNonBlocking(UART0_BASE,UARTCharGetNonBlocking(UART0_BASE));
                //GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1,GPIO_PIN_1 );
//              Delay(1000);

        }
        //GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1,0 );
}
*/
void UARTStringGet(uint32_t ui32Base,char *cMessage,const char Iden)
{
  while(1)
        {
         *cMessage=UARTCharGet(ui32Base);
         if(*cMessage!=Iden)
         {
                cMessage=cMessage+1;
         }
         else
         {
                *cMessage='\0';
                 break;
         }
        }
}
```

```
void UART0_Handler(void)
{       uint8_t cnt = 0,flag1=1,flag2=1;
        uint32_t ulStatus;
        ulStatus = UARTIntStatus(UART0_BASE, true);
        UARTIntClear(UART0_BASE, ulStatus);

        while( UARTCharsAvail(UART0_BASE) )  {
                RxBuf[cnt++]= UARTCharGetNonBlocking(UART0_BASE);
 }
         RxBuf[cnt]='\0';
        if(strcmp(RxBuf, KEY1)==0)UARTStringPut((uint8_t *)"\r\nCLASS2212\n\r");
        if(strcmp(RxBuf, KEY2)==0)UARTStringPut((uint8_t *)"\r\nCODE522031910206\n\r");
}
```