```c
//Kunal Mukherjee;
//3/7/2019
//Assignment 5: I2C Master Out

#include "stm32l432.h"
#include "string.h"

void GPIO_Init(void);

void I2C1_Initialize(void);
void I2C1_Start(unsigned long SlaveAddress,
                char size,
                char direction);

void I2C1_Stop(void);
void I2C1_WaitLineIdle(void);

int I2C1_SendData(char SlaveAddress,
                  char size,
                  char * pData);

int main()
{
  char * data = "55";

  unsigned char slaveAddress = 0x53;
  int i;

  GPIO_Init();
  I2C1_Initialize();

  while(1)
  {
    I2C1_SendData(slaveAddress, sizeof(data), data);

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

void GPIO_Init(void)
{
  RCC_AHB2ENR  |= (1 << 0);    //GPIOA clk enable

  //set up PA9 as alternate func I2C1_SCL
  //set up PA10 as alternate func I2C1_SDA

  GPIOA_MODER   &= ~(3 << (2 * 9)); //clear the GPIOA mode bits
  GPIOA_MODER   |=  (2 << (2 * 9)); //set port PA9 is alternate 10

  GPIOA_MODER   &= ~(3 << (2 * 10)); //clear the GPIOA mode bits
  GPIOA_MODER   |=  (2 << (2 * 10)); //set port PA10 is alternate 10

  GPIOA_AFRH    |=  (4 << (4 * 1)); //alt func 4, port pin 1,
                                    //control bit are 4 bit wide

  GPIOA_AFRH    |=  (4 << (4 * 2)); //alt func 4, port pin 1,
                                    //control bit are 4 bit wide

  GPIOA_OTYPER |= (1 << 9);  //pa9 as open-drain
  GPIOA_OTYPER |= (1 << 10); //pa10 as open-drain

  GPIOA_PUPDR  &= ~(3 << (2 * 9)); //no pull-up/pull-dn
  GPIOA_PUPDR  &= ~(3 << (2 * 10)); //no pull-up/pull-dn
}

void I2C1_Initialize(void)
{
  RCC_APB1ENR1 |= 1 << 21; //I2C1 clock enable
  I2C1_CR1 &= ~I2C_CR1_PE;

  I2C1_TIMINGR |= 0 << 28; //Fpresc = Fi2c/(0+1) = 4 MHz
  I2C1_TIMINGR |= 5 << 20; //SCLDEL = 5 -> 1.5 us
```

```
  I2C1_TIMINGR |= 5 << 16; //SDADEL = 5 -> 1.5 us
  I2C1_TIMINGR |= 19 << 8; //SCLH = 19 -> Thigh = 5 us
  I2C1_TIMINGR |= 19; //SCLL = 19 -> Tlow = 5 us

  I2C1_CR1 |= I2C_CR1_PE; //peripheral enbale
}

void I2C1_Start(unsigned long SlaveAddress,
                char size,
                char direction)
{
  unsigned long temp = I2C1_CR2;

  temp &= 0xFC009800; //clear SADD, NBYTES, RELOAD, AUTOEND, RD_WRN, START, STOP

  if(direction == READ)
  {
    temp |= I2C_CR2_WRN;
  }
  else
  {
      temp &= ~(unsigned long)I2C_CR2_WRN;
  }

  temp |= SlaveAddress << 1; //why the shift of one
  temp |= (unsigned long) size << 16;
  temp |= I2C_CR2_START;

  I2C1_CR2 = temp;
}

void I2C1_Stop(void)
{
  //generate the STOP bit after the current byte has been transferred
  I2C1_CR2 |= I2C_CR2_STOP;

  //wait while stopf flag is reset
  while ((I2C1_ISR & I2C_ISR_STOPF) == 0);

  I2C1_ICR |= I2C_ICR_STOPCF; //write 1 to clear STOPF flag
}

void I2C1_WaitLineIdle(void)
{
  while ((I2C1_ISR & I2C_ISR_BUSY) == I2C_ISR_BUSY); //if busy wait
}

int I2C1_SendData(char SlaveAddress,
                  char size,
                  char * pData)
{
  int i;
  if (size <= 0 || pData == NULL) return -1;

  //wait until the line is idle
  I2C1_WaitLineIdle();

  I2C1_Start(SlaveAddress, size, WRITE);

  for (i = 0; i < size; i++)
  {
    //wait for Tx to be empty
    while((I2C1_ISR & I2C_ISR_TXIS) == 0);
    //TXIS is cleared by writing to the TXDR reg
    I2C1_TXDR = pData[i];
  }

  //wait intil TC flag is set
  while ((I2C1_ISR & I2C_ISR_TC) == 0 &&
          (I2C1_ISR & I2C_ISR_NACKF) == 0);

  if (( I2C1_ISR & I2C_ISR_NACKF) != 0)
```

```
145         return -1;
146 
147       I2C1_Stop();
148 
149       return 0;
150     }
151 
152 
```

```
1   //Kunal Mukherjee
2   //3/15/2019
3
4   #define READ 1
5   #define WRITE 0
6   #define I2C_CR1_PE      (1 << 0)
7   #define I2C_CR2_WRN     (1 << 10)
8   #define I2C_CR2_START   (1 << 13)
9   #define I2C_CR2_STOP    (1 << 14)
10  #define I2C_ISR_STOPF   (1 << 05)
11  #define I2C_ICR_STOPCF  (1 << 05)
12  #define I2C_ISR_BUSY    (1 << 15)
13  #define I2C_ISR_TXIS    (1 << 01)
14  #define I2C_ISR_TC      (1 << 06)
15  #define I2C_ISR_NACKF   (1 << 04)
16  #define I2C_ISR_RXNE    (1 << 02)
17  #define I2C_ISR_ADDCODE (127 << 17)
18  #define I2C_ISR_DIR       (1 << 16)
19
20  //RCC starts at 0x4002 1000
21  #define RCC_AHB2ENR  (*((volatile unsigned long *) 0x4002104C)) //AHB2 peripheral clock enable register
22  #define RCC_APB1ENR1 (*((volatile unsigned long *) 0x40021058)) //APB1 peripheral clock enable register 1
23
24  //GPIOA start 0x4800 0000
25  #define GPIOA_MODER   (*((volatile unsigned long *) 0x48000000)) //GPIO A Mode register
26  #define GPIOA_OTYPER  (*((volatile unsigned long *) 0x48000004)) //GPIO A Output type reg
27  #define GPIOA_OSPEEDR (*((volatile unsigned long *) 0x48000008)) //GPIO A Output speed register
28  #define GPIOA_PUPDR   (*((volatile unsigned long *) 0x4800000C)) //GPIO A Pudr register
29  #define GPIOA_AFRL    (*((volatile unsigned long *) 0x48000020)) //GPIO A Alternate func register low
30  #define GPIOA_AFRH    (*((volatile unsigned long *) 0x48000024)) //GPIO A Alternate func register high
31  #define GPIOA_ODR     (*((volatile unsigned long *) 0x48000014)) //GPIO A Output data reg
32
33  //GPIOB start 0x4800 0400
34  #define GPIOB_MODER (*((volatile unsigned long *) 0x48000400)) //GPIO B Mode register
35  #define GPIOB_PUPDR (*((volatile unsigned long *) 0x4800040C)) //GPIO B Pudr register
36  #define GPIOB_BSRR  (*((volatile unsigned long *) 0x48000418)) //GPIO B Output Bit set/reset register
37  #define GPIOB_AFRL  (*((volatile unsigned long *) 0x48000420)) //GPIO B Alternate func register
38  #define GPIOB_ODR   (*((volatile unsigned long *) 0x48000414)) //GPIO B Output data reg
39
40  //ADC start 0x5004 0000
41  #define ADC_ISR     (*((volatile unsigned long *) 0x50040000)) //ADC interrupt and status register
42  #define ADC_IER     (*((volatile unsigned long *) 0x50040004)) //ADC interrupt enable register
43  #define ADC_CR      (*((volatile unsigned long *) 0x50040008)) //ADC control register
44  #define ADC_SQR1    (*((volatile unsigned long *) 0x50040030)) //ADC regular sequence register
45  #define ADC_DR      (*((volatile unsigned long *) 0x50040040)) //ADC data register
46  #define ADC_CCR     (*((volatile unsigned long *) 0x50040308)) //ADC common control register
47  #define ADC_CFGR    (*((volatile unsigned long *) 0x5004000C)) //ADC configuration register
48
49  //TIM2 start 0x4000 0000
50  #define TIM2_CR1   (*((volatile unsigned long *) 0x40000000)) //TIM2 control register
51  #define TIM2_EGR   (*((volatile unsigned long *) 0x40000014)) //TIM2 event generation register
52  #define TIM2_CCMR1 (*((volatile unsigned long *) 0x40000018)) //TIM2 capture/compare mode register
53  #define TIM2_CCMR2 (*((volatile unsigned long *) 0x4000001C)) //TIM2 capture/compare mode register
54  #define TIM2_PSC   (*((volatile unsigned long *) 0x40000028)) //TIM2 event generation register
55  #define TIM2_ARR   (*((volatile unsigned long *) 0x4000002C)) //TIM2 auto-reload register
56  #define TIM2_CCR1  (*((volatile unsigned long *) 0x40000034)) //TIM2 capture/compare register
57  #define TIM2_CCR2  (*((volatile unsigned long *) 0x40000038)) //TIM2 capture/compare register
58  #define TIM2_CCR3  (*((volatile unsigned long *) 0x4000003C)) //TIM2 capture/compare register
59  #define TIM2_CCR4  (*((volatile unsigned long *) 0x40000040)) //TIM2 capture/compare register
60  #define TIM2_CCER  (*((volatile unsigned long *) 0x40000020)) //TIM2 capture/compare enable register
61  #define TIM2_DIER  (*((volatile unsigned long *) 0x4000000C)) //TIM2 interrupt enable register
62  #define TIM2_SR    (*((volatile unsigned long *) 0x40000010)) //TIM2 status register
63
64  //I2C1 starts 0x4000 5400
65  #define I2C1_CR1     (*((volatile unsigned long *) 0x40005400)) //I2C1 status register
66  #define I2C1_CR2     (*((volatile unsigned long *) 0x40005404)) //I2C1 status register
67  #define I2C1_TIMINGR (*((volatile unsigned long *) 0x40005410)) //I2C1 timing register
68  #define I2C1_ISR     (*((volatile unsigned long *) 0x40005418)) //I2C1 interrupt and starts register
69  #define I2C1_ICR     (*((volatile unsigned long *) 0x4000541C)) //I2C1 interrupt control register
70  #define I2C1_TXDR    (*((volatile unsigned long *) 0x40005428)) //I2C1 tranfer data register
71  #define I2C1_RXDR    (*((volatile unsigned long *) 0x40005424)) //I2C1 receive data register
72
```

```
73  //I2C3 starts 0x4000 5C00
74  #define I2C3_CR1     (*((volatile unsigned long *) 0x40005C00)) //I2C3 status register
75  #define I2C3_CR2     (*((volatile unsigned long *) 0x40005C04)) //I2C3 status register
76  #define I2C3_TIMINGR (*((volatile unsigned long *) 0x40005C10)) //I2C3 timing register
77  #define I2C3_ISR     (*((volatile unsigned long *) 0x40005C18)) //I2C3 interrupt and starts register
78  #define I2C3_ICR     (*((volatile unsigned long *) 0x40005C1C)) //I2C3 interrupt control register
79  #define I2C3_TXDR    (*((volatile unsigned long *) 0x40005C28)) //I2C3 tranfer data register
80  #define I2C3_RXDR    (*((volatile unsigned long *) 0x40005C24)) //I2C3 receive data register
81
82  //NVIC 0xE000 E100 programmer maunal
83  #define NVIC_ISER0        (*((volatile unsigned long *) 0xE000E100)) ////ISER0
84
85  //SYSCFG 0x4001 0000
86  #define SYSCFG_EXTICR1    (*((volatile unsigned long *) 0x40010008)) //SYSCFG
87
88  //EXTI  0x4001 0400
89  #define EXTI_IMR1         (*((volatile unsigned long*) 0x40010400)) //EXTI_IMR1
90  #define EXTI_RTSR1        (*((volatile unsigned long*) 0x40010408)) //EXTI_RTSR1
91  #define EXTI_PR1          (*((volatile unsigned long*) 0x40010414)) //EXTI_PR1
92
93
94
95
96
```