EL 6483 Real-time Embedded system

HW 2 Due 4th Mar. 2016

Shuaiyu Liang (sl5352)

1.

1. GPIOD_MODER: 0x 40020C00 ~ 0x 40020C20

2. The address computed from stm32f4xx.h

```
#define GPIOD_BASE            (AHB1PERIPH_BASE + 0x0C00)
#define AHB1PERIPH_BASE       (PERIPH_BASE + 0x00020000)
#define PERIPH_BASE           ((uint32_t)0x40000000)
GPIOD_ address should start at PERIPH_BASE + 0X0020000 + 0x0C00 which is
0x40020C00

They are same as I expected in the first question.
```

/*

3. Modified in main.c following in the function init_LED_pins()
4. As two new function added, using on and BSRRL and BSRRH to set/reset GPIOD using -> and |=

   Also now you know LED is controlled under GPIOD

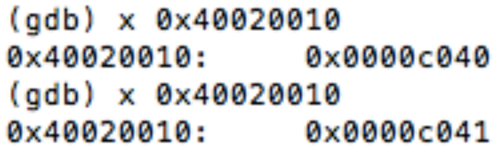
*/

---

2. /* Now you know the button press output to GPIOA */

3.

1. yes. While pressing the butting I got an increment of one.



```
(gdb) x 0x40020010
0x40020010:     0x0000c040
(gdb) x 0x40020010
0x40020010:     0x0000c041
```

2.
3. disassemble

```
74 {
   0x08000544 <+0>: push{r7} // store r7
   0x08000546 <+2>: sub sp, #20 // stack pointer sub by 20
   0x08000548 <+4>: add r7, sp, #0  // r7 change to sp
   0x0800054a <+6>: str r0, [r7, #4]  // store R0 in the address R7+4, in
                    //this case R0 is the parameter i pass into the
                    //function

75     int a = i + 12;
   0x0800054c <+8>: ldr r3, [r7, #4] // load R0 to R3 (i)
   0x0800054e <+10>:adds r3, #12  // R3 = R3 + 12 (a) and update flag
   0x08000550 <+12>:str r3, [r7, #12] // store R3 after the R0's position
// till now, [r7, #4] is i and [r7, # 12] is a

76     GPIOD->BSRRL |= (1 << a);
   0x08000552 <+14>:ldr r2, [pc, #40]    ; (0x800057c <LED_On+56>)
// load from address program counter + 40 to R2
   0x08000554 <+16>:ldr r3, [pc, #36]    ; (0x800057c <LED_On+56>)
// load from address program counter + 36 to R3
   0x08000556 <+18>:ldrh r3, [r3, #24]
// load from address R3 +24 higher 16bits to R3
```

```
   0x08000558 <+20>:uxth r3, r3
// expand 16bit R3 into full 32 bit unsigned
   0x0800055a <+22>:uxth r0, r3
// expand 16bit R3 into R0
   0x0800055c <+24>:movs r1, #1
// move 1 to R1 and update condition flag
   0x0800055e <+26>:ldr  r3, [r7, #12]
// load from address R7 +12 to R3, which is a
   0x08000560 <+28>:lsl.w    r3, r1, r3
// left shift R1 by R3 bit and store the result into R3
   0x08000564 <+32>:uxth r3, r3
// expand R3
   0x08000566 <+34>:mov  r1, r0
// move R0 to R1
```

Appendix: Code:

Assembly for LED_On, LED_Off, read_button, max

```
 .section .text
 .syntax unified
 .global LED_Off, LED_On, read_button, max
 .weak LED_On, LED_Off, read_button, max

LED_On:
         PUSH {LR}
         LDR R1, = #0x40020c18
         LDR R2, [R1]
         MOV R3, R0
         ADD R4, R3, #12
         MOV R3, #1
         LSL R5, R3, R4
         ORR R4, R5, R2
         STR R4, [R1]
         POP {LR}
        BX LR

LED_Off:
        PUSH {LR}
        LDR R1, = #0x40020c18
        LDR R2, [R1]
        MOV R3, R0
        ADD R4, R3, #28
        MOV R3, #1
        LSL R5, R3, R4
        ORR R4, R5, R2
        STR R4, [R1]
        POP {LR}
        BX LR

read_button:
        PUSH {LR}
        LDR R1, = #0x40020010
        LDR R2, [R1]
         AND R3, R2, #1
         MOV R0, R3
         POP {LR}
        BX  LR


max:
        PUSH {LR}
        SUB R1, #1
        LDR R3, [R0]
        MOV R2, #0
        MOV R5, R2
        MOV R6, R0

MAX:
        ADD R5, #1
        ADD R6, #4
        LDR R4, [R6]
        CMP R4, R3
        BGT UPDATE
        CMP R5, R1
  BEQ END
        BNE MAX

UPDATE:
        MOV R3, R4
        MOV R2, R5
        B MAX

END:
        MOV R0, R3
        MOV R1, R2
        POP {LR}
        BX LR
```

C Code:

```
void init_LED_pins()
{
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;  // enable clock to GPIOD

  GPIOD->MODER &= ~(0x3 << (2*12)); // clear the 2 bits corresponding to pin 12
  GPIOD->MODER |= (1 << (2*12));    // set pin 12 to be general purpose output
  // to enable the other pins (13,14,15), you will need to write to the appropriate bits of the ~MODER~
register.

  GPIOD->MODER &= ~(0x3 << (2*13)); // clear the 2 bits corresponding to pin 13
  GPIOD->MODER |= (1 << (2*13));    // set pin 12 to be general purpose output

  GPIOD->MODER &= ~(0x3 << (2*14)); // clear the 2 bits corresponding to pin 14
  GPIOD->MODER |= (1 << (2*14));    // set pin 12 to be general purpose output

  GPIOD->MODER &= ~(0x3 << (2*15)); // clear the 2 bits corresponding to pin 15
  GPIOD->MODER |= (1 << (2*15));    // set pin 12 to be general purpose output
}


void init_button()
{
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;  // enable clock to GPIOA

  GPIOA->MODER &= ~(0x3 << (2*0)); // clear the 2 bits corresponding to pin 0
  // if the 2 bits corresponding to pin 0 are 00, then it is in input mode
}

uint32_t read_button(void)
{
    uint32_t key;

    if (GPIOA->IDR & 0x01)
        key = 1;
    else
        key = 0;

    return(key);
}

void LED_On(uint32_t i)
{
    int a = i + 12;
    GPIOD->BSRRL |= (1 << a);
}


void LED_Off(uint32_t i)
{
    int a = i +12;
    GPIOD->BSRRH |= (1 << a);
}

int main(void)
{
  // initialize
  SystemInit();
  init_systick();
  init_LED_pins();
  init_button();
  int i;

  int arr[6] = {1,2,4,8,8,3};
  uint32_t n = 6;
  uint32_t index;
  int y = max(arr, n, &index);

/*
        LED_On(i);
        LED_Off(i);
*/
 while (1)
    {
    }
}
```