```
#define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R      (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R      (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R       (*((volatile unsigned long *)0x4002552C))
#define PF4                     (*((volatile unsigned long *)0x40025040))
#define PF3                     (*((volatile unsigned long *)0x40025020))
#define PF2                     (*((volatile unsigned long *)0x40025010))
#define PF1                     (*((volatile unsigned long *)0x40025008))
#define PF0                     (*((volatile unsigned long *)0x40025004))
#define GPIO_LOCK_KEY           0x4C4F434B  // Unlocks the GPIO_CR register
#define SYSCTL_RCGCGPIO_R       (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_PRGPIO_R         (*((volatile unsigned long *)0x400FEA08))
#define RED       0x02
#define BLUE      0x04
#define GREEN     0x08
#define PF123_mask              0x0E
#define PF04_mask               0x11
#define PF_mask                 0x20
```

```
;Q1. Write an assembly function that initializes port F pins 1, 2, and 3 as Digital Output with initial zero values


RGB_LED_Init    LDR R1, =SYSCTL_RCGCGPIO_R                      ;activating the clock of PORT F
                LDR R0, [R1]
                ORR R0, R0, #0x20
                STR R0, [R1]
                NOP                                             ;waiting for activation to complete
                NOP

                LDR R1, =GPIO_PORTF_LOCK_R                      ;unlocking PORT F
                LDR R0, [R1]
                LDR R0, =0x4C4F434B
                STR R0, [R1]

                LDR R1, =GPIO_PORTF_CR_R                        ;unlocking PORT F pins no. 1 2 3
                LDR R0, [R1]
                ORR     R0, R0, #0x0E
                STR R0, [R1]

                LDR R1, =GPIO_PORTF_DIR_R                       ;setting pins 1,2,3 of port F to be output
                LDR R0, [R1]
                ORR     R0, R0, #0x0E
                STR R0, [R1]

                LDR R1, =GPIO_PORTF_DEN_R                       ;setting pins 1,2,3 of port F to be digital
                LDR R0, [R1]
                ORR     R0, R0, #0x0E
                STR R0, [R1]
```

```
LDR R1, =GPIO_PORTF_AMSEL_R                    ;disable pins 1,2,3 of port F to be analog
LDR R0, [R1]
BIC     R0, R0, #0x0E
STR R0, [R1]

LDR R1, =GPIO_PORTF_AFSEL_R                    ;disable pins 1,2,3 of port F to be alternative
LDR R0, [R1]
BIC     R0, R0, #0x0E
STR R0, [R1]

LDR R1, =GPIO_PORTF_PCTL_R                     ;disable pins 1,2,3 of port F to be analog
LDR R0, [R1]
LDR R2, =0xFFF0
BIC     R0, R0, R2
STR R0, [R1]

LDR R1, =GPIO_PORTF_DATA_R                     ;initializing the pins with zero values
LDR R0, [R1]
BIC R0, R0, #0x0E

BX LR
```

```c
//Q1. Write a C function that initializes port F pins 1, 2, and 3 as Digital Output with initial zero values


#include "IO.h"

void RGBLED_Init(void)
{
                SYSCTL_RCGCGPIO_R       |= 0x20;                                  //PORT F Clock enable

                while ( (SYSCTL_PRGPIO_R&0x20) == 0) {}

                GPIO_PORTF_LOCK_R |= 0x4C4F434B;                                //Unlock PORT F
                GPIO_PORTF_CR_R |= 0x0E;                                        //Allow changes to pins 1,2,3
                GPIO_PORTF_DIR_R |= 0x0E;                                       //SET pins 1,2,3 to be output
                GPIO_PORTF_DEN_R |= 0x0E;                                       //SET pins 1,2,3 to be digital
                GPIO_PORTF_AMSEL_R &= ~0x0E;                                    //Disable Analog function
                GPIO_PORTF_AFSEL_R &= ~0x0E;                                    //No alternative function
                GPIO_PORTF_PCTL_R &= ~0xFFF0;                                   //GPIO clear bit PCTL
                GPIO_PORTF_DATA_R &= ~0x0E;                                     //Initialize LEDs to be off
}
```

```
;Q2. Write an assembly function that initializes port F pin 4 as Digital Input that will be connected to a switch.

SW1_Init                        LDR R1, =SYSCTL_RCGCGPIO_R
                                LDR R0, [R1]
                                ORR R0, R0, #0x20
                                STR R0, [R1]
                                NOP
                                NOP

                                LDR R1, =GPIO_PORTF_LOCK_R
                                LDR R0, =0x4C4F434B
                                STR R0, [R1]

                                LDR R1, =GPIO_PORTF_CR_R
                                LDR R0, [R1]
                                ORR R0, R0, #0x10
                                STR R0, [R1]

                                LDR R1, =GPIO_PORTF_DIR_R
                                LDR     R0, [R1]
                                BIC R0, R0, #0x10
                                STR R0, [R1]

                                LDR R1, =GPIO_PORTF_DEN_R
                                LDR R0, [R1]
                                ORR R0, R0, #0x10
                                STR R0, [R1]

                                LDR R1, =GPIO_PORTF_AMSEL_R
                                LDR R0, [R1]
                                BIC R0, R0, #0x10
                                STR R0, [R1]
```

```
LDR R1, =GPIO_PORTF_AFSEL_R
LDR R0, [R1]
BIC R0, R0, #0x10
STR R0, [R1]

LDR R1, =GPIO_PORTF_PCTL_R
LDR R1, [R0]
LDR R2, =0xF0000
BIC R0, R0, R2
STR R0, [R1]

LDR R1, =GPIO_PORTF_PUR_R
LDR R0, [R1]
ORR R0, R0, #0x10
STR R0, [R1]

BX LR
```

```c
//Q2. Write a C function that initializes port F pin 4 as Digital Input that will be connected to a switch.


void SW1_Init   (void)
{
                SYSCTL_RCGCGPIO_R       |= 0x20;

                while ( (SYSCTL_PRGPIO_R&0x20) == 0) {}

                GPIO_PORTF_LOCK_R |= 0x4C4F434B;
                GPIO_PORTF_CR_R |= 0x10;



                GPIO_PORTF_DIR_R &= ~0x10;
                GPIO_PORTF_DEN_R |= 0x10;
                GPIO_PORTF_AMSEL_R &= ~0x10;
                GPIO_PORTF_AFSEL_R &= ~0x10;
                GPIO_PORTF_PCTL_R &= ~0xF000;
                GPIO_PORTF_PUR_R |= 0x10;
}
```

```
;Q3. Write an assembly function that reads PORTF pin4.

SW1_Input                       LDR R1, =GPIO_PORTF_DATA_R
                                LDR R0, [R1]                                    ;imports all contents of port f
                                AND R0, R0, #0x10                               ;reads content of pin 4 only
                                STR R0, [R1]
                                BX LR
```

```c
//Q3. Write a C function that reads PORTF pin4.



unsigned char SW1_Input (void)
{

                        return GPIO_PORTF_DATA_R & 0x10;

}
```

```
;Q4. Write an assembly function that clears pin1, pin2, and pin3 then update the mentioned pins with new values of data in PORTF.

RGB_output                  LDR R3, =data
                            LDR R0, [R3]

                            LDR R1, =GPIO_PORTF_DATA_R
                            LDR R2, [R1]

                            BIC R2, R2, #0x0E                       ;clearing pins 1,2,3

                            ORR R2, R2, R0                          ;updating the new values of data

                            STR R2, [R1]

                            BX LR
```

```c
//Q4. Write a C function that clears pin1, pin2, and pin3 then update the mentioned pins with new values of data in PORTF.


void RGB_Output (unsigned char data)
{
                        GPIO_PORTF_DATA_R &= ~0x0E;
                        GPIO_PORTF_DATA_R |= data;

}
```

```
;Q5. In Tiva C, PF4 is connected to a push button and PF1, PF2, and PF3 are connected to an RGB LED.
;PF1 is red, PF2 is blue, and PF3 is green. Write assembly application that uses the init functions developed in
;previous questions that reads input from the switch and when it is pressed for the first time the red LED
;should be turned on then when pressed a second time turn off the red LED and turn on the blue LED then
;when pressed a third time turn off the blue LED and turn on the green LED then when pressed a fourth time
;turn off the green LED and turn on again the red LED and then repeat the cycle.


                        BL RGB_LED_Init
                        BL SW1_Init

___main                 MOV R3, #0x02
SuperLoop               CMP R3, #0x10
                        BNE read_SW1
                        MOV R3, #0x02

read_SW1                BL SW1_Input
                        CMP R0, #0x10
                        BEQ end_if
                        MOV R0,R3
                        BL RGB_Output
                        LSL R3, R3, #1
end_if                  B SuperLoop
```

```c
//Q5. In Tiva C, PF4 is connected to a push button and PF1, PF2, and PF3 are connected to an RGB LED.
//PF1 is red, PF2 is blue, and PF3 is green. Write C application that uses the init functions developed in
//previous questions that reads input from the switch and when it is pressed for the first time the red LED
//should be turned on then when pressed a second time turn off the red LED and turn on the blue LED then
//when pressed a third time turn off the blue LED and turn on the green LED then when pressed a fourth time
//turn off the green LED and turn on again the red LED and then repeat the cycle.


unsigned char LED_OUT = 0x02;             //initalize LED TO RED  0000 0010
unsigned char Button_In;

void main(void)
{

         RGB_LED_INIT();
         SW1_init;

                 while(1){
                 if ( LED_OUT == 0x10 ){             // 0x10 --> 0001 0000 Which means that All LEDs are OFF and SW1 OFF
                 LED_OUT = 0x02;                 // Reset LED_OUT TO RED AFTER GREEN
        }

                 Button_In = SW1_Input();         // CHECK ON STATUS OF SW1 IF PRESSED (0) OR NOT (1)

                if (Button_In != 0x10)
                {         // SW1 is pressed

                RGB_Output(LED_OUT);
                LED_OUT = LED_OUT<<1;       //Shift Left to next color in pin
                }
        }



}
```

Q7. What is a direction register? Why does the microcontroller have direction registers?

→ it determines whether the pins operate as inputs or outputs "GPIO_PORTF_DIR_R"
& Using the Concept of direction registers to make the micro Controller more marketable.

Q8. What is the alternative function register?

→ Individual port pins can be general purpose I/O "GPIO" or have an alternative function. We will set bits in the alternative function register "GPIO_PORTF_AFSEL_R" when we wish to activate the alternative function listed in tables

Juba