Experiment 7

Digital Input/Output Interfacing and Programming

Objective

The objective of this lab is to give you a first foot in the door exposure to the programming of I/O, which when executed by the microcontroller (TI LM4F120, an ARM Cortex-M4) simply blinks LED located on the development board.

Introduction to GPIO

A GPIO pin is a pin that can be configured through software to be either a digital input or a digital output. GPIO outputs let you translate logical values within your program to voltage values on output pins and it is these voltage outputs that help your microcontroller exert control over the system into which it is embedded.

Configuring Peripherals

The fundamental initialization steps required to utilize any of the peripheral are:

1. Enable clocks to the peripheral
2. Configure pins required by the peripheral
3. Configure peripheral hardware

Structure of the Program

LED flashing code is implemented using an infinite loop which toggles bit 28 of the address 0x2009C020 after a certain delay. The delay is implemented using the loop that just increments the loop counter until the condition is satisfied. Figure – shows the two different approaches. The shorter flowchart leads to smaller code size and the longer flowchart translates to an inefficient code.

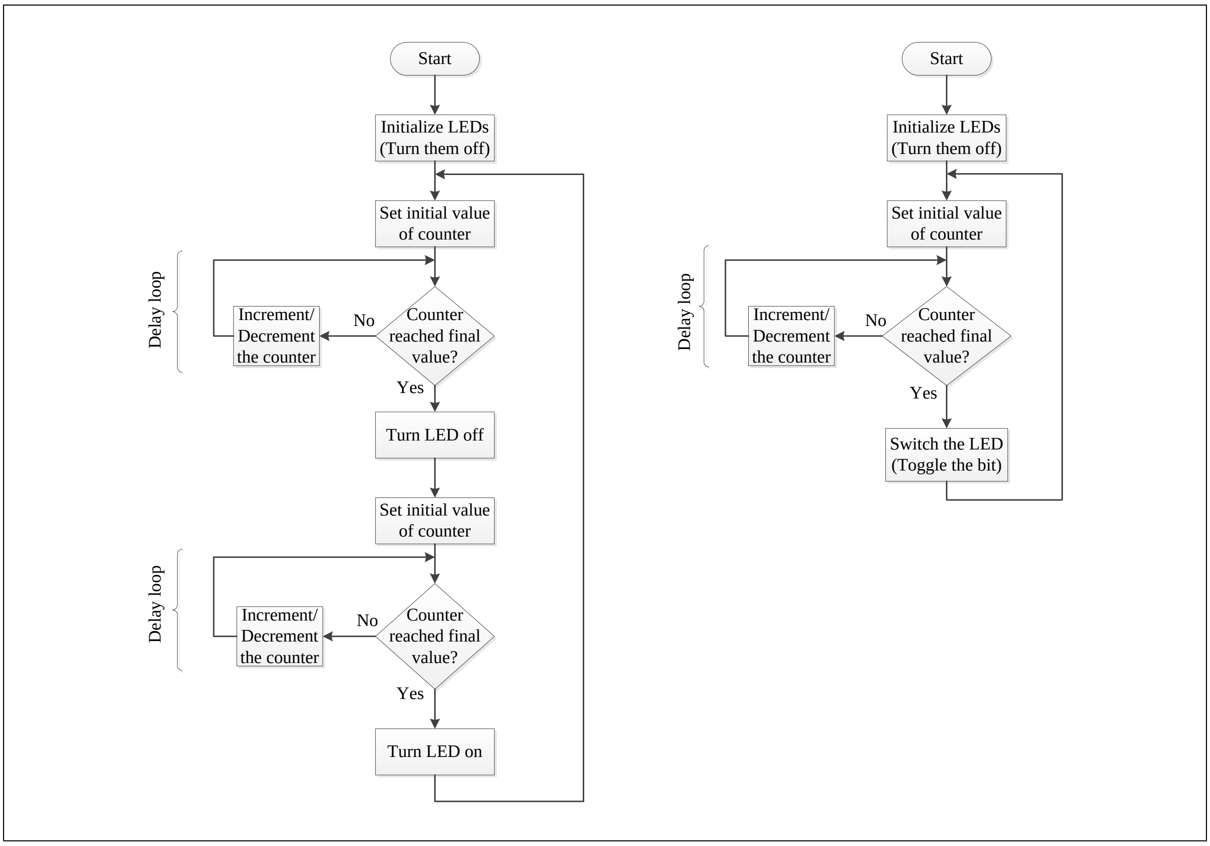

Start
Initialize LEDs (Turn them off)
Set initial value of counter
Delay loop
Increment/ Decrement the counter
No
Counter reached final value?
Yes
Turn LED off
Set initial value of counter
Delay loop
Increment/ Decrement the counter
No
Counter reached final value?
Yes
Turn LED on
Start
Initialize LEDs (Turn them off)
Set initial value of counter
Delay loop
Increment/ Decrement the counter
No
Counter reached final value?
Yes
Switch the LED (Toggle the bit)

Figure 7.1: Flowchart for blinking the LED(s)

The overall structure of the program is shown in listing below. The program begins by defining macros for the relevant register addresses. The main routine follows the initialization steps described above then enters a loop in which it toggles an LED and waits for sometime.

```
#include ”lm4f120h5qr.h”

int main(void) {
  //Enable peripherals
  ... (1) ...
  //Configure pins
  ... (2) ...
  while(1) {
    //Turn ON LED
    ... (3)...
    //Delay for a bit
    ... (4) ...
    //Turn OFF LED
    ... (5) ...
  }
}
```

Where is LED?

The Stellaris LaunchPad comes with an RGB LED. The LED can be configured for use in any custom application. The following table 6.1 shows how the LED is connected to the pins on the microcontroller. Figure 6.1 shows the physical connection of the LED.

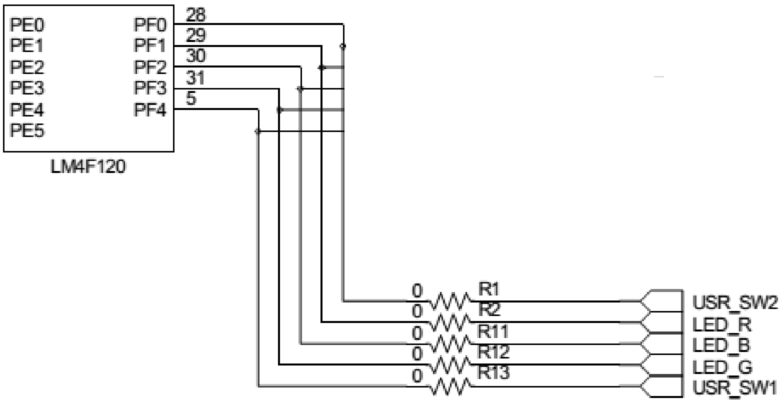


Figure 7.2: LED Schematic

GPIO pin	Pin Function	USB Device
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGB LED (Green)

Table 6.1: RGB LED Signals

LED Configuration

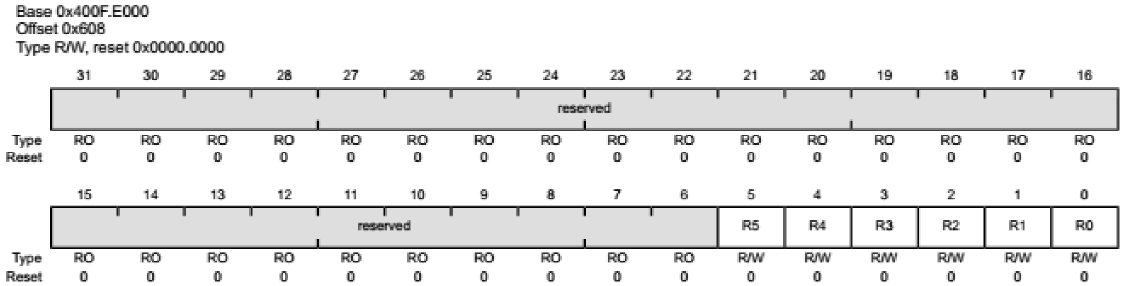
We will follow the steps stated above to configure the on-board LED.

Enabling the Clock

The RCGCGPIO register provides software the capability to enable and disable GPIO modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register is shown in Figure 6.2. The clock can be enabled for the GPIO port F by asserting the 6th bit of RCGGPIO register.

Following command can be used to enable clock signal for GPIO port F

```
SYSCTL_RCGCGPIO_R = 0x20;        // (1)
```



Base 0x400F.E000
Offset 0x608
Type R/W, reset 0x0000.0000

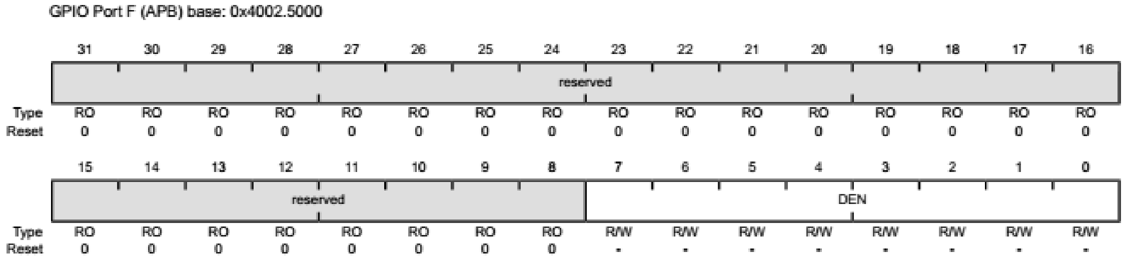
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.3: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Configuring the Pin as Output

After enabling the clocks, it is necessary to configure any required pins. In this case, a single pin (PF3) must be configured as an output. To use the pin as a digital input or output, the corresponding bit in the GPIODEN register must be set and then setting a bit in the GPIODIR register configures the corresponding pin to be an output.

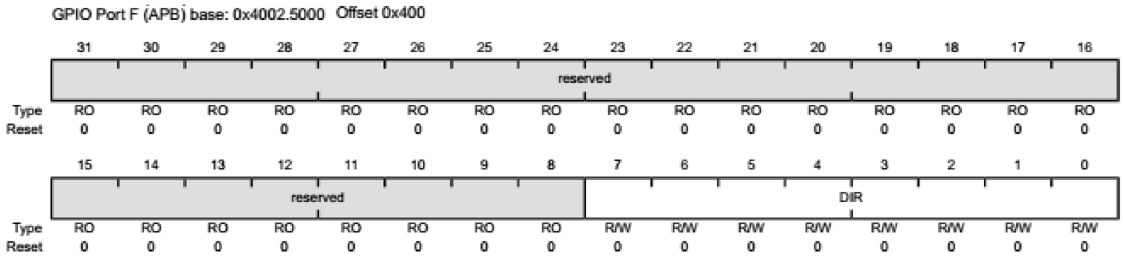


GPIO Port F (APB) base: 0x4002.5000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Figure 7.4: GPIO Digital Enable (GPIODEN)



GPIO Port F (APB) base: 0x4002.5000 Offset 0x400

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

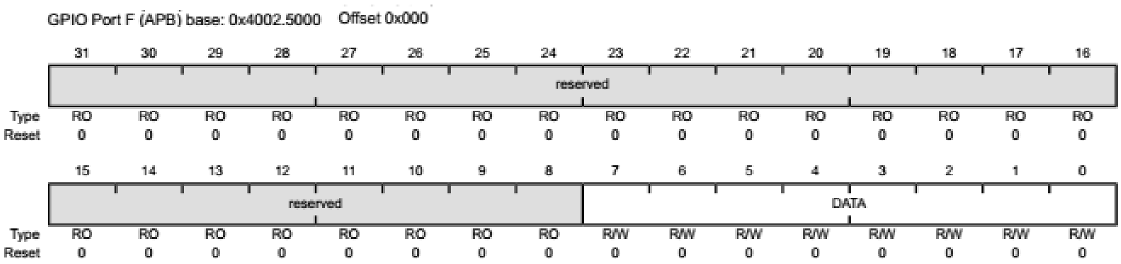
Figure 7.5: GPIO Direction (GPIODIR)

The commands used to set the corresponding bits in GPIODEN and GPIODIR registers are given as follows

```
GPIO_PORTF_DIR_R = 0x08;        // (2)
GPIO_PORTF_DEN_R = 0x08;
```

Toggle the LED

After configuring the LED as an output, now we want to toggle it after regular intervals. LED can be turned ON and OFF by setting and resetting the corresponding bits in the GPIODATA register.



GPIO Port F (APB) base: 0x4002.5000 Offset 0x000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.6: GPIO Data (GPIODATA)

The commands for toggling LED are as follows

```
GPIO_PORTF_DATA_R = 0x08;        // (3)
GPIO_PORTF_DATA_R = 0x00;        // (5)
```

Introducing a Delay

We cannot observe the toggling of LED because of very high frequency. We introduce a delay loop in order to observe the toggle sequence of the LED. The syntax for the loop is shown in the following figure

```
int counter = 0;
    while(counter < 200000){   // (4)
          ++counter;
    }
```

Source Code

The complete C and assembly language code for the program is given as follows

C language Code

```
#define SYSCTL_RCGCGPIO_R        (*((volatile unsigned long *)0x400FE608))

#define GPIO_PORTF_DATA_R        (*((volatile unsigned long *)0x400253FC))
```

```
#define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))

#define GPIO_PORTF_CLK_EN       0x20
#define GPIO_PORTF_PIN3_EN      0x08
#define LED_ON                  0x08
#define LED_OFF                 ~(0x08)
#define DELAY                   200000


int main(void)
{
    volatile unsigned long ulLoop;

    // Enable the GPIO port that is used for the on-board LED.

    SYSCTL_RCGCGPIO_R |= GPIO_PORTF_CLK_EN;

    // Do a dummy read to insert a few cycles after enabling the peripheral.

    ulLoop = SYSCTL_RCGCGPIO_R;

    /* Enable the GPIO pin for the LED (PF3).  Set the direction as output and enable the GPIO pin for digital function.*/

    GPIO_PORTF_DIR_R = GPIO_PORTF_PIN3_EN;
    GPIO_PORTF_DEN_R = GPIO_PORTF_PIN3_EN;

    // Loop forever.

    while(1)
    {

        // Turn on the LED.

        GPIO_PORTF_DATA_R |= LED_ON;

        // Delay for a bit.

        for(ulLoop = 0; ulLoop < DELAY; ulLoop++)
        {
        }

        // Turn off the LED.

        GPIO_PORTF_DATA_R &= LED_OFF;

        // Delay for a bit.

        for(ulLoop = 0; ulLoop < DELAY; ulLoop++)
        {
        }
    }
}
```

Assembly Language Code

```
; Directives
        PRESERVE8
        THUMB                          ; Marks the THUMB mode of operation

;************ Data Variables are declared in DATA AREA  ****************;
    AREA     const_data, DATA, READONLY

; Initializing some constants
SYSCTL_RCGCGPIO_R       EQU     0x400FE608

GPIO_PORTF_AFSEL_R    EQU     0x40025420
GPIO_PORTF_DIR_R      EQU     0x40025400
GPIO_PORTF_DEN_R      EQU     0x4002551C
GPIO_PORTF_DATA_R     EQU     0x400253FC

GPIO_PORTF_CLK_EN     EQU     0x20
GPIO_PORTF_PIN3_EN    EQU     0x08
DELAY                 EQU     200000
LED_ON                EQU     0x08
LED_OFF               EQU     0xF7


;*** The user code (program) is placed in CODE AREA ***;

      AREA     |.text|, CODE, READONLY, ALIGN=2

      ENTRY        ; ENTRY marks the starting point of
                     ; the code execution
      EXPORT __main

__main

; User Code Starts from the next line
; Enable clock for PORT F
      LDR      R1, =SYSCTL_RCGCGPIO_R
      LDR      R0, [R1]
      ORR      R0, #GPIO_PORTF_CLK_EN
      STR      R0, [R1]
      NOP                                ; No operation for 3 cycles
      NOP
```

```
        NOP

; Set the direction for PORT F
        LDR     R1, =GPIO_PORTF_DIR_R
        LDR     R0, [R1]
        ORR     R0, #GPIO_PORTF_PIN3_EN
        STR     R0, [R1]

; Digital enable for PORT F
        LDR     R1, =GPIO_PORTF_DEN_R
        LDR     R0, [R1]
        ORR     R0, #GPIO_PORTF_PIN3_EN
        STR     R0, [R1]

; Infinite loop LED_flash
LED_flash

; Set the data for PORT F to turn LED on
        LDR     R1, =GPIO_PORTF_DATA_R
        LDR     R0, [R1]
        ORR     R0, R0, #LED_ON
        STR     R0, [R1]

; Delay loop
        LDR     R5, =DELAY
delay1

        SUBS     R5,#1
        BNE     delay1

; Set the data for PORT F to turn LED off
        LDR     R1, =GPIO_PORTF_DATA_R
        LDR     R0, [R1]
        AND     R0, R0, #LED_OFF
        STR     R0, [R1]

; Delay loop
        LDR     R5, =DELAY
delay2

        SUBS     R5,#1
        BNE     delay2

        B       LED_flash

        ALIGN

        END             ; End of the program, matched with ENTRY keyword
```

Exercises