MAP Lab

LAB 2 : Get To Know Basic GPIO & LED Blink

AIM

Here we will explain **how to initialize a GPIO** (General-purpose input-output) pins of the TM4C123GH6PM microcontroller with the Tiva C series launchpad. In order to use the GPIO pin of a TM4C123G evaluation kit, we must **first initialize the registers related to GPIO pins.** At the end of the tutorial, the **built-in LED of the TIVA board is blinked using the associated GPIO port with it.**

Steps

- Initialization of the main clock of the board
- Enabling specific port registers of which the GPIO pins will be used
- Initialization of GPIO port pins to either digital output or digital input .i.e. setting the direction pins
- After that, the functionality of the port is selected if it is to be alternative or not (in this tutorial we are not using alternative functionality so skip it at this moment)

TM4C123GH6PM Microcontroller GPIO pins

TM4C123GH6PM belongs to the ARM Cortex M4 microcontroller series. It has six GPIO ports such as PORTA, PORTB, PORTC, PORTD, and PORTE. Each port has a different number of pins as given in this table.

GPIO Ports	Pins
PORTA	PA0 – PA7
PORTB	PB0- PB7
PORTC	PC0 – PC7
PORTD	PD0 – PD7
PORTE	PE0 – PE5
PORTF	PF0-PF7

TM4C123GH6PM Microcontroller GPIO pins

All the address range of port A to port F are given in the table below. As you can see from this table, a total of 4K bytes of memory is allocated for each PORT. The reseason for this larger amount of memory for each port is due to many functions and special functions registers associated with each port of TM4C123G.

Port name	Starting address	Ending address
PortA	0x40004000	0x40004FFF
PortB	0x40005000	0x40005FFF
PortC	0x40006000	0x40006FFF
PortD	0x40007000	0x40007FFF
PortE	0x40024000	0x40024FFF
PortF	0x40025000	0x40025FFF

Revise Steps

- Initialization of the main clock of the board
- Enabling specific port registers of which the GPIO pins will be used
- Initialization of GPIO port pins to either digital output or digital input .i.e. setting the direction pins
- After that, the functionality of the port is selected if it is to be alternative or not (in this tutorial we are not using alternative functionality so skip it at this moment)

few additional steps might be required. For instance, some of the GPIO pins are assigned special functionality by default, and to configure them as general-purpose input-output pins, we need to unlock or unmask the pin. It is possible to configure a GPIO pin as an alternative functionality multiplexing,

Basic steps for TM4C123GH6PM launchpad configuration as a GPIO are listed below

1 - Clock configuration

2 - Data control configuration

3 - Mode control configuration

4 - Pad control configuration

GPIO Pins Clock Enable Register

The first step in GPIO configuration is to enable the clock for a particular peripheral you want to enable. A particular port can be enabled by setting an appropriate bit field for the required GPIO port in the RCGCGPIO register.

RCGCGPIO Register is on pg. No. 340 of datasheet

```
RCGCGPIO |= 0x01 //Enable clock for PORTA
RCGCGPIO |= 0x02 //Enable clock for PORTB
RCGCGPIO |= 0x04 //Enable clock for PORTC
RCGCGPIO |= 0x08 //Enable clock for PORTD
RCGCGPIO |= 0x01 //Enable clock for PORTE
RCGCGPIO |= 0x02 //Enable clock for PORTF
```

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

Base 0x400F.E000
Offset 0x608
Type RW, 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	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	GPIO Port F Run Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in Run mode.
4	R4	RW	0	GPIO Port E Run Mode Clock Gating Control Value Description 0 GPIO Port E is disabled. 1 Enable and provide a clock to GPIO Port E in Run mode.

TM4C123G GPIODATA Register

The GPIO port pins of TM4C123G are multiplexed with different peripherals such as digital I/O, PWM, serial communication, etc. But each pin can be used for only one functionality at a time. GPIODEN register is used to enable GPIO pins as digital input-output pins.

When the port pin is configured as GPIO pins, the GPIODATA register is used to read and write data on the registers. If the pin is configured as a digital output pin, the data written to the GPIODATA register reflects on the corresponding output pin.

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x51C
Type RW, reset -

	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	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	RW	-	Digital Enable Value Description 0 The digital functions for the corresponding pin are disabled. 1 The digital functions for the corresponding pin are enabled. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Pg 654

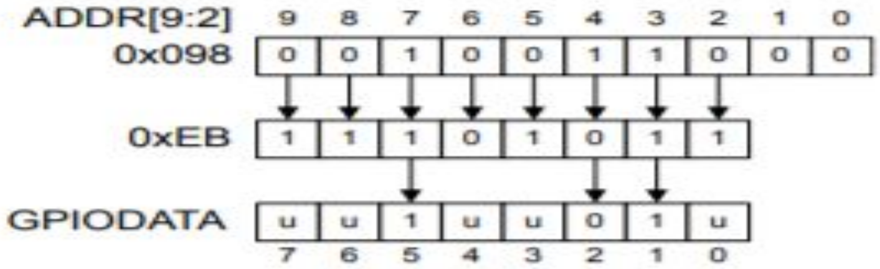
10.2.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 662) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

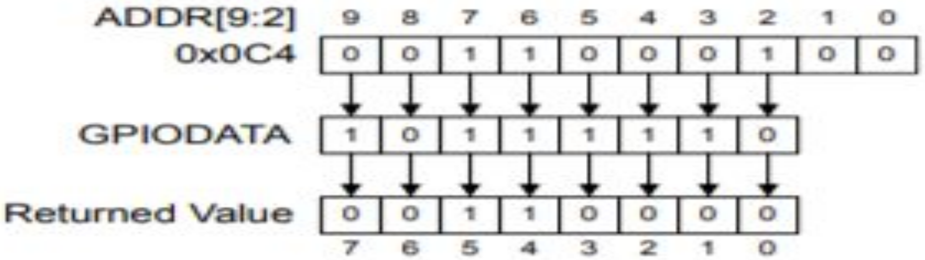
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where u indicates that data is unchanged by the write. This example demonstrates how **GPIODATA** bits 5, 2, and 1 are written.

Figure 10-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 10-4. This example shows how to read **GPIODATA** bits 5, 4, and 0.

Figure 10-4. GPIODATA Read Example



	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	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 654 for examples of reads and writes.

TM4C123G GPIODIR Direction Control Register

A GPIODIR register decides, which pins of the PORT will be configured either as a digital input or a digital output. The individual configuration capability of each GPIO is applicable to other registers of the GPIO port pins.

If we want to enable a pin of a port as digital input-output the corresponding bit on the GPIODIR register should be set or reset. If we want to configure a particular pin of any port as a digital input pin, the corresponding e data direction bit should be cleared. Similarly, if we want to configure a particular pin of any port as a digital output pin, the corresponding data direction bit should be set to one.

	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	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	RW	0x00	GPIO Data Direction

Value	Description
0	Corresponding pin is an input.
1	Corresponding pins is an output.

GPIO PORTF base address = 0x4002 5000
GPIODEN Register offset address = 0x51c // page number 682 TM4C123GH6PM datasheet
physical address = 0x40025000+0x51C = 0x4002551C

```
#define SYSCTL_RCGCGPIO_R (*(( volatile unsigned long *)0x400FE608 ) )
#define GPIO_PORTF_DATA_R (*(( volatile unsigned long *)0x40025038 ) )
#define GPIO_PORTF_DIR_R  (*(( volatile unsigned long *)0x40025400 ) )
#define GPIO_PORTF_DEN_R  (*(( volatile unsigned long *)0x4002551C ) )


int main ( void )
{
SYSCTL_RCGCGPIO_R |= 0x20; // Enable clock for PORTF
GPIO_PORTF_DEN_R  = 0x0E;  // Enable PORTF Pin1, 2 and 3 as a digital pins
GPIO_PORTF_DIR_R  = 0x0E;  // Configure ORTF Pin1, 2 and 3 digital output pins

    while (1)
        {
        GPIO_PORTF_DATA_R |= 0x02; // turn on red LED
        }
}
```

Base address of PORTF = 0x40025000
Offset address of GPIODIR = 0x400 // // page number 663 TM4C123GH6PM datasheet
GPIODIR Physical address = 0x40025000+0x400 = 0x40025400

PORTF Base Address = 0x40025000
GPIODATA Registe offset address = is 0x000
GPIODATA Physical address = 0x40025000+0x608 = 0x40025000

```
#include "TM4C123.h" // header files contains memory
addresses listing
int main ( void )
{
SYSCTL->RCGCGPIO|= 0x20;
GPIOF->DIR = 0x02;
GPIOF->DEN = 0x02;

    while (1){
    GPIOF->DATA = 0x02; // turn on red LED

   }
}
```

```
int main(void)
{

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN)
;
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
   GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

   while(1)
   {
      GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0x02);
      SysCtlDelay(5000000);
      GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0x04);
      SysCtlDelay(5000000);
      GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0x08);
      SysCtlDelay(5000000);

   }


}
```

Assignment 1 :

1. What is Embedded C Programming? How is Embedded C different from C language?
2. What is ISR?
3. What is Void Pointer in Embedded C and why is it used?
4. Why do we use the volatile keyword?
5. How will you use a variable defined in source file1 inside source file2?

Assignment 2 :

1. Go through all registers of GPIO
2. Read following functions:**SysCtlPeripheralEnable,GPIOPinWrite,SysCtlDelay,SysCtlClockSet,GPIOPinTypeGPIOOutput**
3. **Compute XOR from 1 to n**

```
Input : n = 6
Output : 7
// 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6  = 7

Input : n = 7
Output : 0
  // 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 = 0
```