第15次實習課

學生：林培瑋

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

EXAMPLE 15.4 考題方向

- 已知三個條件，計算出經過的時間(sec)。
  - system clocks ? MHz
  - ?-bit timer
  - match value = ?

```
LDR     r1, [r8]          ; set as 16-bit timer only
ORR     r1, #0x4          ; base 0x40030000
STR     r1, [r8]          ; offset 0, bit[2:0]=0x4

MOVW    r7, #0x30         ; set the match value at 0
MOV     r1, #0            ; since we're counting down
STR     r1, [r8, r7]      ; offset - 0x30
```

GPIO

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

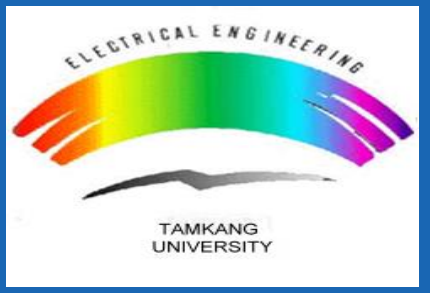


FIGURE 16.11 block diagram

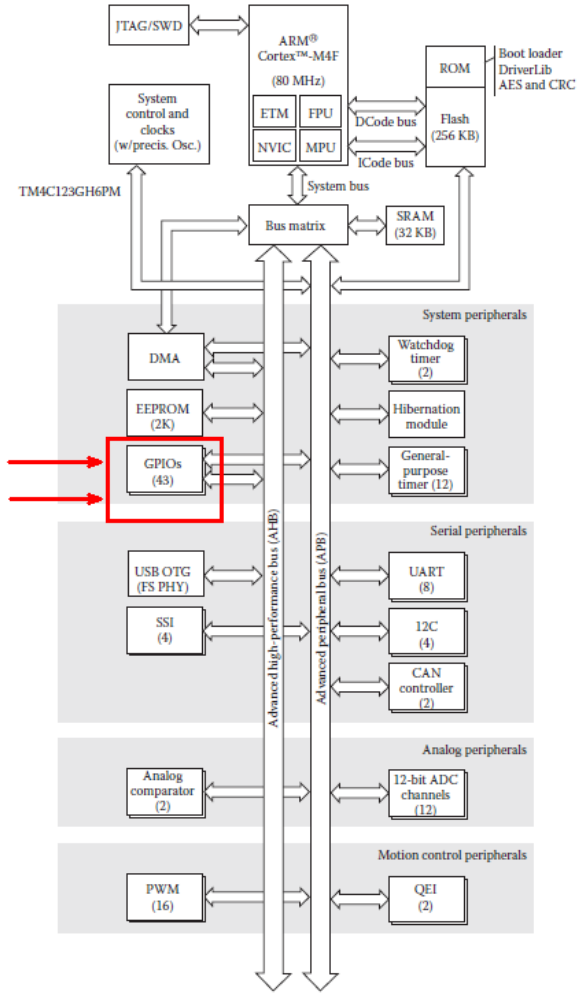


FIGURE 16.11 Tiva™ TM4C123GH6PM microcontroller high-level block diagram.

16.4.5 – 系統設定

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

P362 16.4.5

```
myStart:
    ; Set sysclk to DIV/4, use PLL, XTAL_16 MHz, OSC_MAIN
    ; system control base is 0x400FE000, offset 0x60
    ; bits[26:23]= 0x3
    ; bit[22] = 0x1
    ; bit[13] = 0x0
    ; bit[11] = 0x0
    ; bits[10:6] = 0x15
    ; bits[5:4] = 0x0
    ; bit[0] = 0x0
    ; This all translates to a value of 0x01C00540
    MOVW      r0, #0xE000
    MOVT      r0, #0x400F
    MOVW      r2, #0x60       ; offset 0x60 for this register
    MOVW      r1, #0x0540
    MOVT      r1, #0x01C0
    STR       r1, [r0, r2]    ; write the register's contents
    ; Enable GPIOF
    ; RCGCGPIO (page 339)
    MOVW      r2, #0x608      ; offset for this register
    LDR       r1, [r0, r2]    ; grab the register contents
    ORR       r1, r1, #0x20   ; enable GPIOF clock
    STR       r1, [r0, r2]
```

系統設定

P359 16.4.2 GPIO

```
; Enable GPIOF
; RCGCGPIO (page 339)
MOVW     r2, #0x608        ; offset for this register
LDR      r1, [r0, r2]      ; grab the register contents
ORR      r1, r1, #0x20     ; enable GPIOF clock
STR      r1, [r0, r2]
```

0x20 = 0b0010 0000
FEDCBA

TABLE 16.4
GPIO Port Locations

	Port	Address
APB Bus	GPIO Port A	0x40004000
	GPIO Port B	0x40005000
	GPIO Port C	0x40006000
	GPIO Port D	0x40007000
	GPIO Port E	0x40024000
	GPIO Port F	0x40025000
AHB Bus	GPIO Port A	0x40058000
	GPIO Port B	0x40059000
	GPIO Port C	0x4005A000
	GPIO Port D	0x4005B000
	GPIO Port E	0x4005C000
	GPIO Port F	0x4005D000

16.4.5 – GPIO設定

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

Set the direction of GPIO Port F lines 1, 2, and 3

TM4C123GH6PM. The LED is actually located on APB bus Port F, so our base address will be 0x40025000. We'll set the direction of GPIO Port F lines 1, 2, and 3 to be outputs:

```
; Set the direction using GPIODIR (page 661)
; Base is 0x40025000
MOVW    r0, #0x5000
MOVT    r0, #0x4002
MOVW    r2, #0x400      ; offset for this register
MOV     r1, #0xE
STR     r1, [r0, r2]    ; set 1 or 2 or 3 for output
```

; read
LDR r3, [r0, r2]
ORR r3, r1
; modify

r3 ; write

The GPIO Digital Enable Register

If you are using the Keil tools, the constant 0x40025000 can be loaded into register r0 with an LDR pseudo-instruction from Chapter 6; however, the Code Composer Studio assembler does not support this, so two separate move instructions will do the trick. There's an additional level of gating on the port—the GPIO Digital Enable Register will need to be configured:

```
; set the GPIODEN lines
MOVW    r2, #0x51c      ; offset for this register
STR     r1, [r0, r2]    ; set 1 and 2 and 3 for I/O
```

LDR r3, [r0, r2]
ORR r3, r1
r3

Turning on the LEDs

16.4.4 Turning on the LEDs

The Tiva Launchpad board has a multi-colored LED that is controlled through three GPIO lines on Port F, one for red, one for green, and one for blue. The red LED is attached to line PF1, the green LED is attached to line PF2, and the blue LED is attached to line PF3. Now that Port F has been enabled and the appropriate lines have been configured as outputs, we can light the LEDs by driving a 1 to the GPIO line of our choice. To showcase all three colors, we can create a loop that selects one color at a time, cycling through all three by changing the value being written to the port.

In Chapter 5, the concept of bit-banding allowed individual bits of a memory-mapped register to be accessed using a single read or write operation. It turns out the a similar feature is available on the TM4C123GH6PM microcontroller for accessing GPIO lines, where the address is used as a mask, as shown in Figure 16.12. For example, suppose we wish to alter only bits [7:6] and bit [0] of GPIO Port F. We can use bits [9:2] of the address in our write operation to form a bit mask, so that instead of writing to the base address of Port F (0x40025000), we would store our value to address (0x40025304). Now that our mask is in place, no matter what value we write to the port, such as 0xF6, only bits [7:6] and [0] are altered. Specifically for our LED example, we wish to have bits [9:2] of the address be 0b00111000, or 0x38, since we only want to change the LED lines connected to Port F. This value becomes our offset.

The following code shows our loop, complete with a small delay to give the observer a chance to see the individual colors of the LED. The delay value of 0xF40000 is arbitrary, but at 16 MHz, it gives us about 1 second to view a single color.

Turning on the LEDs

```
        SUB     r7, r7, r7          ; clear out r7
        MOV     r6, #2              ; start with LED = 0b10
mainloop
        ; turn on the LED
        ; if bits [9:2] affect the writes, then the address
        ; is offset by 0x38
        STR     r6, [r0, #0x38]     ; base + 0x38 so [9:2] = 0b111000
        MOVT    r7, #0xF4           ; set counter to 0xF40000
spin
        SUBS    r7, r7, #1
        BNE     spin
        ; change colors
        CMP     r6, #8
        ITE     LT
        LSLLT   r6, r6, #1          ; LED = LED * 2
        MOVGE   r6, #2              ; reset to 2 otherwise
        B       mainloop
```

line 1：red

0b00000010

0b00000100 → line 2：green

0b00001000 → line 3：blue

不加左側程式碼(顏色變化間隔時間太短，肉眼看不出變化) → 加入左側程式碼(製造Delay)

由RED -> GREEN

花費5 instructions

5 clock cycles -> 5/16000000 seconds (16MHz)

Delay = (0xF40000 * 2 clock cycles)/16000000

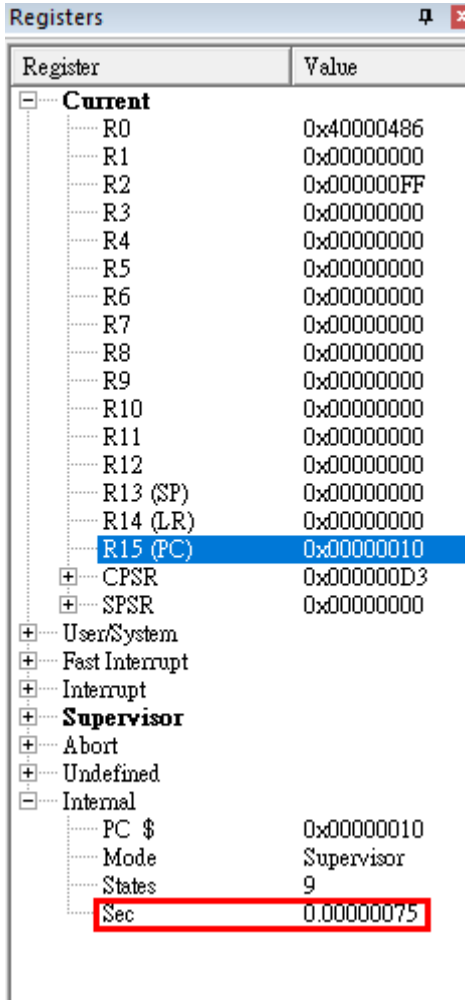
RED -> GREEN -> BLUE -> RED -> GREEN -> BLUE -> ... ...

可有可無，讓編譯程式早點知道後面程式碼的結構。

T →

E →

Turning on the LEDs考題方向(1/2)




```
    MOVT    r7, #0xF4          ; set counter to 0xF40000
spin
    SUBS    r7, r7, #1
    BNE     spin
```

由左側視窗Sec，經由手算得知執行一個指令需要多少秒，
→寫一段程式執行時間5 seconds。
(創造?秒的Delay)
(副程式F10)

Turning on the LEDs考題方向(2/2)

line？：不同顏色

```
    SUB     r7, r7, r7          ; clear out r7
    MOV     r6, #2              ; start with LED=0b10
mainloop
    ; turn on the LED
    ; if bits [9:2] affect the writes, then the address
    ; is offset by 0x38
    STR     r6, [r0, #0x38]     ; base+0x38 so [9:2]=0b111000
    MOVT    r7, #0xF4           ; set counter to 0xF40000
spin
    SUBS    r7, r7, #1
    BNE     spin
    ; change colors
    CMP     r6, #8
    ITE     LT
    LSLLT   r6, r6, #1          ; LED=LED * 2
    MOVGE   r6, #2              ; reset to 2 otherwise
    B       mainloop
```

改成不同bit，亮不同顏色的燈。(改變連接方式) (EX：bit 2、4、6)

(每次向左移2 bits)

改成0x40000038

閃爍順序改變

BIT-BANDED MEMORY

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

P95 5.6 BIT-BANDED MEMORY

❖ 動機：為了節省Read、Modify、Write時間。

5.6 BIT-BANDED MEMORY

With the introduction of the Cortex-M3 and M4 processors, ARM gave programmers the ability to address single bits more efficiently. Imagine that some code wants to access only one particular bit in a memory location, say bit 2 of a 32-bit value held at address 0x40040000. Microcontrollers often use memory-mapped registers in place of registers in the core, especially in industrial microcontrollers where you have ten or twenty peripherals, each with its own set of unique registers. Let's further say that a peripheral such as a Controller Area Network (CAN) controller on the Tiva TM4C123GH6ZRB, which starts at memory address 0x40040000, has individual control bits that are set or cleared to enable different modes, read status information, or transmit data. For example, bit 7 of the CAN Control Register puts the CAN controller in test mode. If we wish to set this bit and only this bit, you could use a read-modify-write operation such as:

因為bit沒有對應的位置。

```
LDR     r3, =0x40040000   ; location of CAN Control Register
LDR     r2, [r3]          ; read the memory-mapped register contents
ORR     r2, #0x80         ; set bit 7
STR     r2, [r3]          ; write the entire register contents back
```

P95 5.6 BIT-BANDED MEMORY

This seems horribly wasteful from a code size and execution time perspective to set just one bit in a memory-mapped register. Imagine then if every bit in a register had its own address—rather than loading an entire register, modifying one bit, then writing it back, an individual bit could be set by just writing to its address. Examining Table 5.1 again, you can see that there are two bit-banded regions of memory: addresses from 0x22000000 to 0x220FFFFF are used specifically for bit-banding the 32KB region from 0x20000000 to 0x20007FFF; and addresses from 0x42000000 to 0x43FFFFFF are used specifically for bit-banding the 1MB region from 0x40000000 to 0x400FFFFF. Figure 5.6 shows the mapping between

P81 TABLE 5.1

TABLE 5.1
Memory Map of the Tiva TM4C123GH6ZRB

Start	End	Description	For Details, See Page...[a]
		Memory	
0x0000.0000	0x0003.FFFF	On-chip flash	553
0x0004.0000	0x00FF.FFFF	Reserved	—
0x0100.0000	0x1FFF.FFFF	Reserved for ROM	538
0x2000.0000	0x2000.7FFF	Bit-banded on-chip SRAM	537
0x2000.8000	0x21FF.FFFF	Reserved	—
0x2200.0000	0x220F.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	537
0x2210.0000	0x3FFF.FFFF	Reserved	—
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	—

P96 FIGURE 5.6

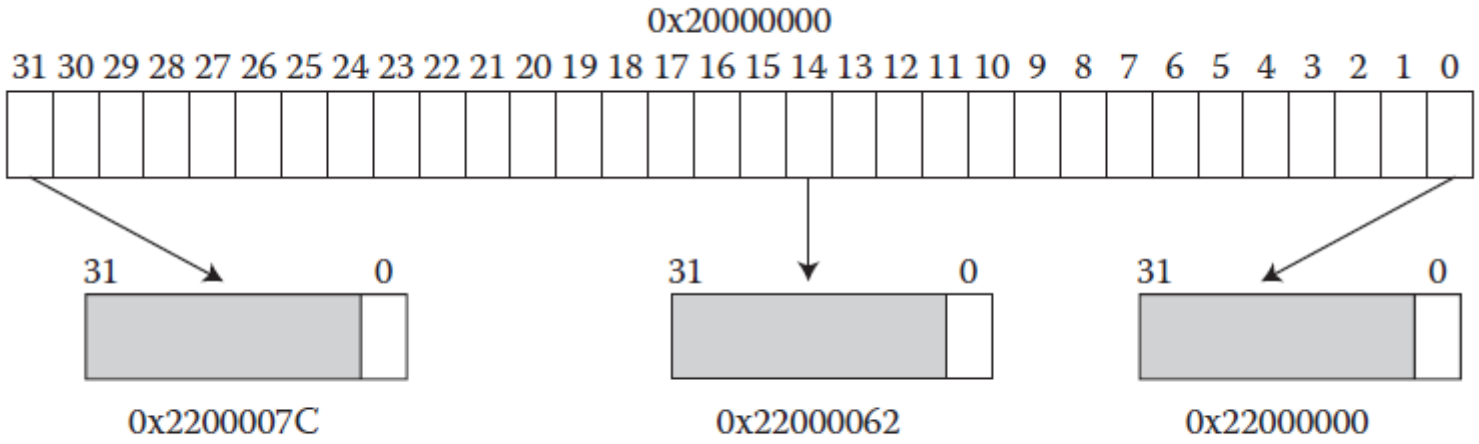


FIGURE 5.6 Mapping bit-banded regions.

Q&A

Thanks for your attention !!