

Department of Electrical Engineering

University of Arkansas



ELEG3923 Microprocessor

Ch.9 Timer Programming

Dr. Jingxian Wu

wuj@uark.edu

OUTLINE

- **Programming 8051 Timers**
- **Counter programming**
- **Timer programming in C**

TIMERS

- **Timers**

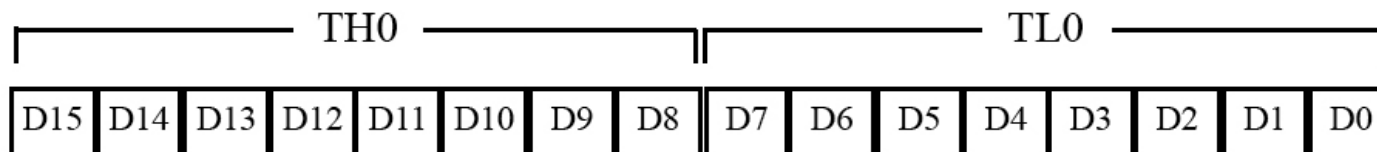
- 8051 has 2 built-in timers: timer 0 and timer 1
 - They can be used to generate a specific time delay (more accurate and easier than loops)
 - Or, can be used as counters to count events happening outside the uC.
- Both timers are 16 bits wide.

- **Timer registers**

- The contents of each timer is stored in two 8-bit registers
- Timer 0: TH0 (higher byte), TL0 (lower byte)
- Timer 1: TH1 (higher byte), TL1 (lower byte)
- E.g.

```
MOV TL0, #4FH
```

```
MOV R5, TH1
```

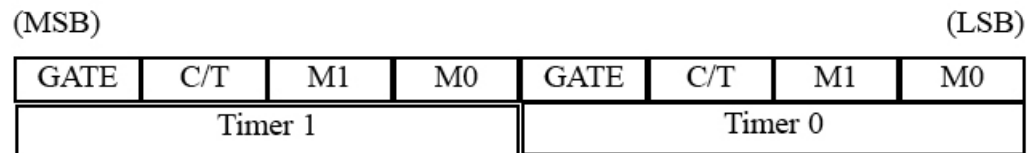


TIMERS: MODE

- **TMOD (timer mode) register**

- TMOD: 8-bit register, used to set the operation modes of timer

- higher nibble: timer 1; lower nibble: timer 0



- M1, M0: the combination of the two bits specify the operation modes of the timer
 - (M1,M0)= 00: mode 0; 01: mode 1; 10: mode 2; 11: mode 3.
- C/T (counter/timer)
 - Whether the timer will be used as a delay generator (timer) or event counter
 - C/T = 0: used as timer for time delay generation
 - C/T = 1: event counter
- Gate
 - Gate = 0: the timer is started or stopped by software
 - Gate = 1: the timer is started or stopped by external hardware

TIMERS: CLOCK SOURCE

- **Clock source for timer**

- Every timer needs a clock pulse to tick.
- When C/T = 0 (timer)
 - The timer use XTAL oscillator as clock source
 - The frequency of the timer is XTAL frequency /12
 - Example:
 - Find the timer's clock frequency is the XTAL frequency is 11.0592MHz

$$f = 11.0592\text{MHz}/12 = 921.6 \text{ KHz}$$

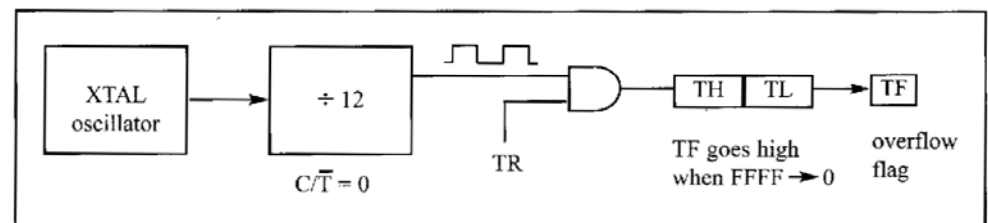
$$T = 1/f = 1/921.6\text{KHz} = 1.085 \text{ uS}$$

TIMERS: MODE 1

• Mode 1

- 16-bit timer: all 16 bits of timer are used.
- Steps: (we use timer0 as example)
 - 1. **Load TMOD value** (set the operation mode of timer)
 - E.g. `MOV TMOD, #01H`; timer 0, C/T=0, GATE=0, mode = 1
 - 2. **Load TL and TH with initial count values**
 - E.g. `MOV TL0, #0F2H` ; load init value to timer 0
`MOV TH0, #0FFH` ; init value: FFF2
 - 3. **Start timer**
 - `SETB TR0` ; start timer 0
 - » **TR0** (timer start 0): when set to 1, timer 0 will start
 - Once timer is started, values in (TH0, TL0) will automatically increase by 1 for every timer tick (every 1.085us if 11.0592 MHz XTAL)
 - when (TH0, TL0) = FFFFH, it will overflow in the next clock period
 - » FFFFH → 0000H,
 - » When overflow, **TF0** (timer flag for timer 0) will automatically set to 1.
 - » By monitoring TF0, we will know when timer expires

(MSB)				(LSB)			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			



TIMERS: MODE 1

- **Mode 1**

- Steps (Cont'd)
 - **4. Monitoring timer flag**
 - HERE: JNB TF0, HERE ; exit loop when TF0 = 1 (timer expires)
 - **5. When timer overflows, stop timer**
 - CLR TR0 ; set TR0 = 0 will stop timer 0
 - The duration of timer is determined by the initial value of timer
 - **6. Clear timer flag for next round**
 - CLR TF0
- The duration of timer is determined by the initial value in timer and timer clock
 - E.g. before timer starts, (TH0)=FFH, (TL0)=F2H. XTAL = 11.0592MHz.
What is the duration of the timer (how long it takes for timer to overflow)?

$(FFFFH - YYXXH + 1) * \text{clock period}$

$(65535 - nnnn + 1) * \text{clock period}$

TIMERS: MODE 1

- **Example**

- Find the delay generated by timer 0 in the following code. Do not include the overhead due to instructions

```

                CLR P2.3
                MOV TMOD, #01                ; 1. timer 0, mode 1 (16-bit)
HERE:          MOV TL0, #3EH                ; 2. load init value
                MOV TH0, #0B8H
                SETB P2.3                    ; set P2.3 high
                SETB TR0                     ; 3. start timer
AGAIN:         JNB TF0, AGAIN               ; 4. monitor TF0 until timer overflows
                CLR TR0                      ; 5. stop timer
                CLR TF0                      ; 6. clear flag

                CLR P2.3

```

- Find the maximum delay that can be incurred by a 16-bit timer

- To generate larger delay, use loops to repeat the timer (example 9-13)

TIMERS: MODE 1

- **Example**

- Write a program generate a square waveform on P1.5. The waveform has 50% duty cycle with period 2 ms.

```

                                MOV TMOD, #10                ; timer 1, mode 1
HERE:                            SETB P1.5
                                ACALL DELAY
                                CLR P1.5
                                ACALL DELAY
                                SJMP HERE

                                ;-----delay subroutine-----

DELAY:                            MOV TL1, #YYH                ; load initial value
                                MOV TH1, #XXH
                                SETB TR1                ; start timer
AGAIN:                            JNB TF1, AGAIN            ; monitor flag
                                CLR TR1                ; stop timer
                                CLR TF1                ; clear flag
                                RET

```

Self study: Examples 9-5 ~ 9-13

TIMER: MODE 0 AND MODE 2

- **Mode 0: 13-bit timer**

- The timer will set $TF = 1$ when it reaches 1FFFH
- `MOV TMOD, #00H` (mode 0)
- The remaining operation is the same as mode 1

- **Mode 2: auto-reload 8-bit timer**

- Timer range: 00H - FFH
- Operations (use timer 0 as example)
 - 1. Set timer 0 to mode 2: `MOV TMOD, #02H` (timer 0, mode 2)
 - 2. Load 8-bit init value to TH0: `MOV TH0, #32H`
 - 3. **8051 automatically copies the init value to TL0**
 - 4. Start timer: `SETB TR0`
 - TL0 starts counting until it reaches FFH
 - 5. Monitor TF0
 - When TL0 overflows from FFH to 00H, it sets $TF = 1$
 - 6. When TL0 overflows to 00H and $TF = 1$, **the init value is automatically reloaded from TH0 to TL0.**
 - 7. Clear TF. The timer will continue to run (go back to step 5)

TIMER: MODE 2

- **Example**

- (1) Find the frequency of the square wave; (2) the smallest frequency achievable in this program and the TH value to do that

```

                MOV TMOD, #20H           ; timer 1, mode 2
                MOV TH1, #5; init value
                SETB TR1                 ; start timer
BACK:          JNB TF1, BACK            ; monitor TF1
                CPL P1.0
                CLR TF1                 ; clear TF0
                SJMP BACK               ; repeat

```

- Find the delay of the following timer

```

                MOV TMOD, #2H           ; timer 0, mode 2
                MOV TH0, #-150         ; TH0 = -150 = 6AH (2's complement of -150)
AGAIN:         SETB P1.3
                ACALL DELAY
                CLR P1.3
                ACALL DELAY
                SJMP AGAIN

DELAY:         SETB TR0                 ; start timer
BACK:         JNB TF0, BACK            ; monitor TF0
                CLR TR0                 ; stop timer (the init value has already been reloaded)
                CLR TF0
                RET

```

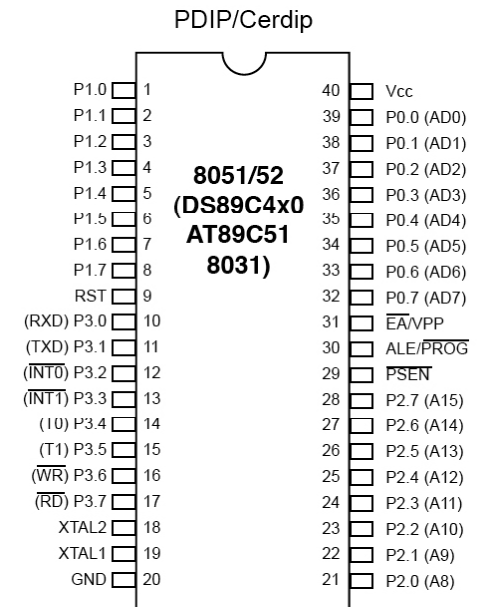
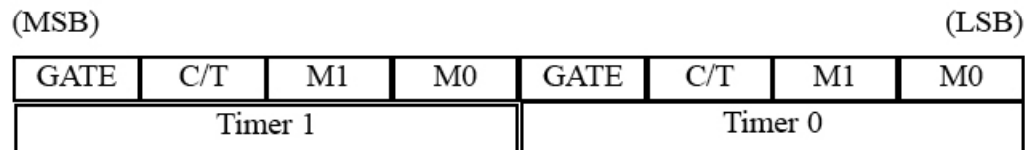
OUTLINE

- Programming 8051 Timers
- **Counter programming**
- Timer programming in C

COUNTER

- **Counter (C/T=1)**

- Counting the events happening **outside of 8051**.
- Counter v.s. timer
 - If C/T=0, timer. The clock source is XTAL/12
 - The value of TH0, TL0 increases by 1 for each clock
 - If C/T = 1, counter. The clock source is an outside pulse to be counted
 - The value of TH0, TL0 increases by 1 for each outside pulse
 - We can use the counter to count the # of pulses
- The remaining operations are the same for counter and timer (e.g. mode 0, 1, 2)
- Connection
 - External pulses are connected through P3.4 (T0) or P3.5 (T1)



COUNTER

• Example

- Assume a 1 Hz frequency pulse is connected to input pin P3.5 (T1). Use mode 2 of counter 1 to count the pulses and display results on P2

```

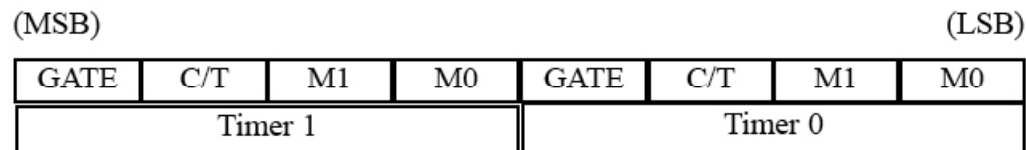
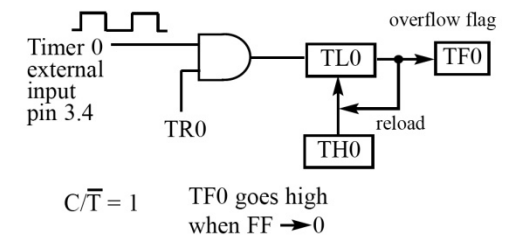
MOV TMOD, #01100000B      ; counter 1, mode 2, C/T = 1

MOV TH1, #0               ; counting from 0
SETB P3.5                 ; make T1 input mode

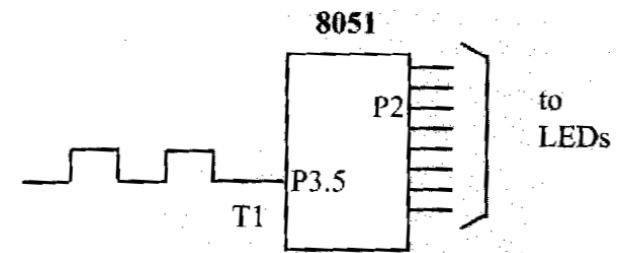
AGAIN: SETB TR1           ; start timer 1
BACK:  MOV A, TL1         ; get copy of count from TL1 (mode 2!)
      MOV P2, A          ; send result to P2 (LCD)
      JNB TF1, BACK      ; keep doing it if TF=0 (no overflow)

CLR TR1                   ; stop timer
CLR TF1                   ; clear TF
SJMP AGAIN

```



Self study: example 9-19



COUNTER: TCON AND TMOD

- **TCON register**

- 8-bit register



- TR0, TF0, TR1, TF1

- SETB TR0 is the same as SETB TCON.4

- IE1, IT1, IE0, IT0 are used by interrupt (will be discussed in Ch. 11)

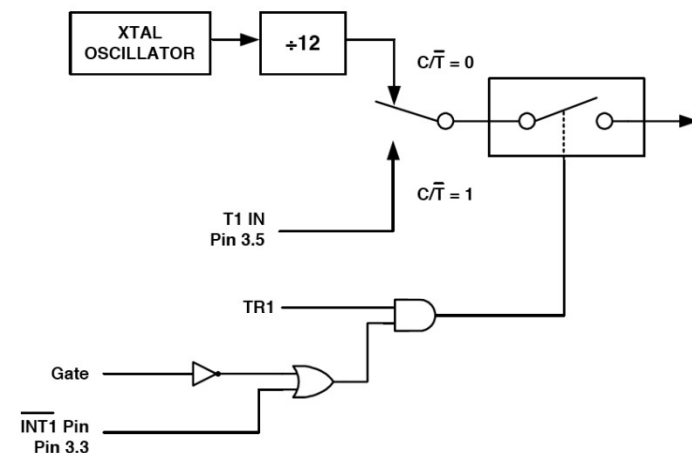
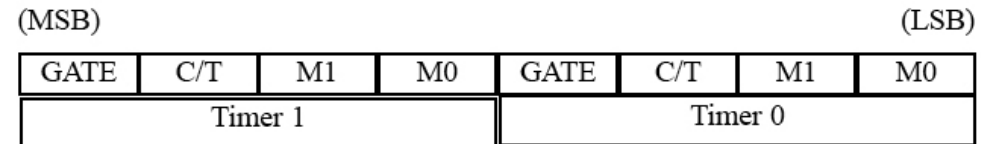
- **Gate bit of TMOD**

- Gate = 0

- Start timer: SETB TR0; stop timer: CLR TR0

- Gate = 1

- The start and stop of timer is done externally through P3.2 for timer 0 or P3.3 for timer 1 via a simple switch (e.g. stop watch)



OUTLINE

- Programming 8051 Timers
- Counter programming
- **Timer programming in C**

PROGRAMMING IN C

- **Example**

- Write a program to toggle P1.5 every 250 ms. Use timer 0, mode 2 (8-bit auto reload)

```
#include <reg51.h>
void TOM2delay25us(void);
sbit mybit = P1^5;
void main(void)
{
    unsigned int x;
    while(1)
    {
        mybit = ~mybit;
        for (x = 0; x < 10000; x++)
            TOM2delay25us();
    }
}
void TOM2delay25us(void)
{
    TMOD = 0x02;           // timer 0, mode 2
    TH0 = -23;            // count 23 times, then overflow. 23*1.085 = 25 us
    TR0 = 1;
    while (TF0 == 0);
    TR0 = 0;
    TF0=0;
}
```

PROGRAMMING IN C

- **Example**

- Assume a 1 Hz external clock is being fed into pin T0 (P3.4). Write a C program for counter T0 in mode 1 (16-bit) to display TH0 and TL0 on P2 and P1, respectively.

```

#include <reg51.h>
void main()
{
    T0 = 1;                //(make T0 an input)
    TMOD = 0x05;          // 0000 0101 (C/T = 1, mode 1)
    TL0 = 0;
    TH0 = 0;              //clear counters
    while(1)
    {
        do
        {
            TR0=1;        //start timer
            P1 = TL0;
            P2 = TH0;
        }while(TF0 == 0);
        TR0 = 0;          //stop timer
        TF0 = 0;         //clear TF
    }
}

```