Principle and Interface Techniques of Microcontroller

--8051 Microcontroller and Embedded Systems Using Assembly and C

LI, Guang (李光) **Prof. PhD, DIC, MIET**

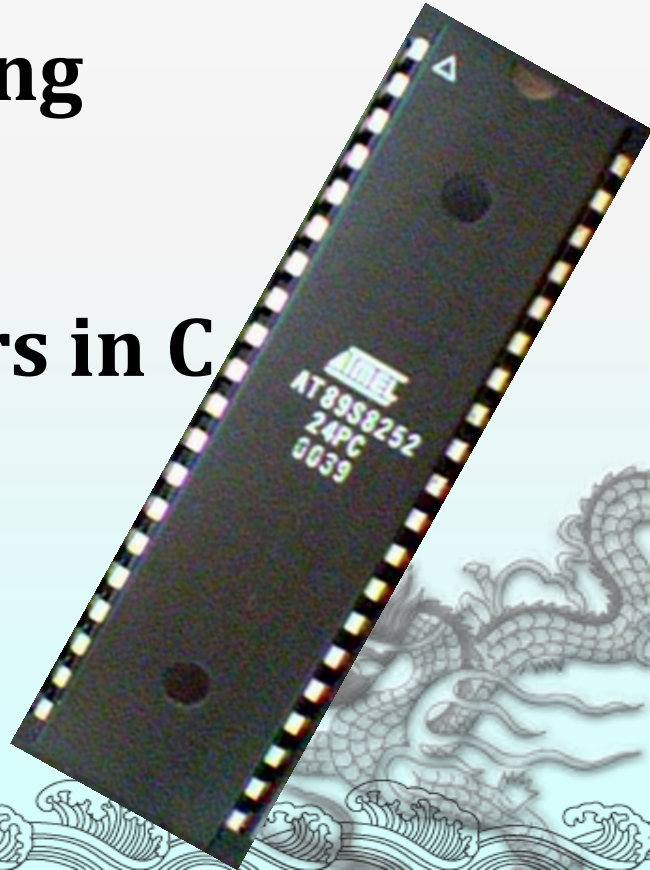
WANG, You (王酉) **PhD, MIET**

杭州 • 浙江大学 • 2022

Chapter 10
Timer Programming

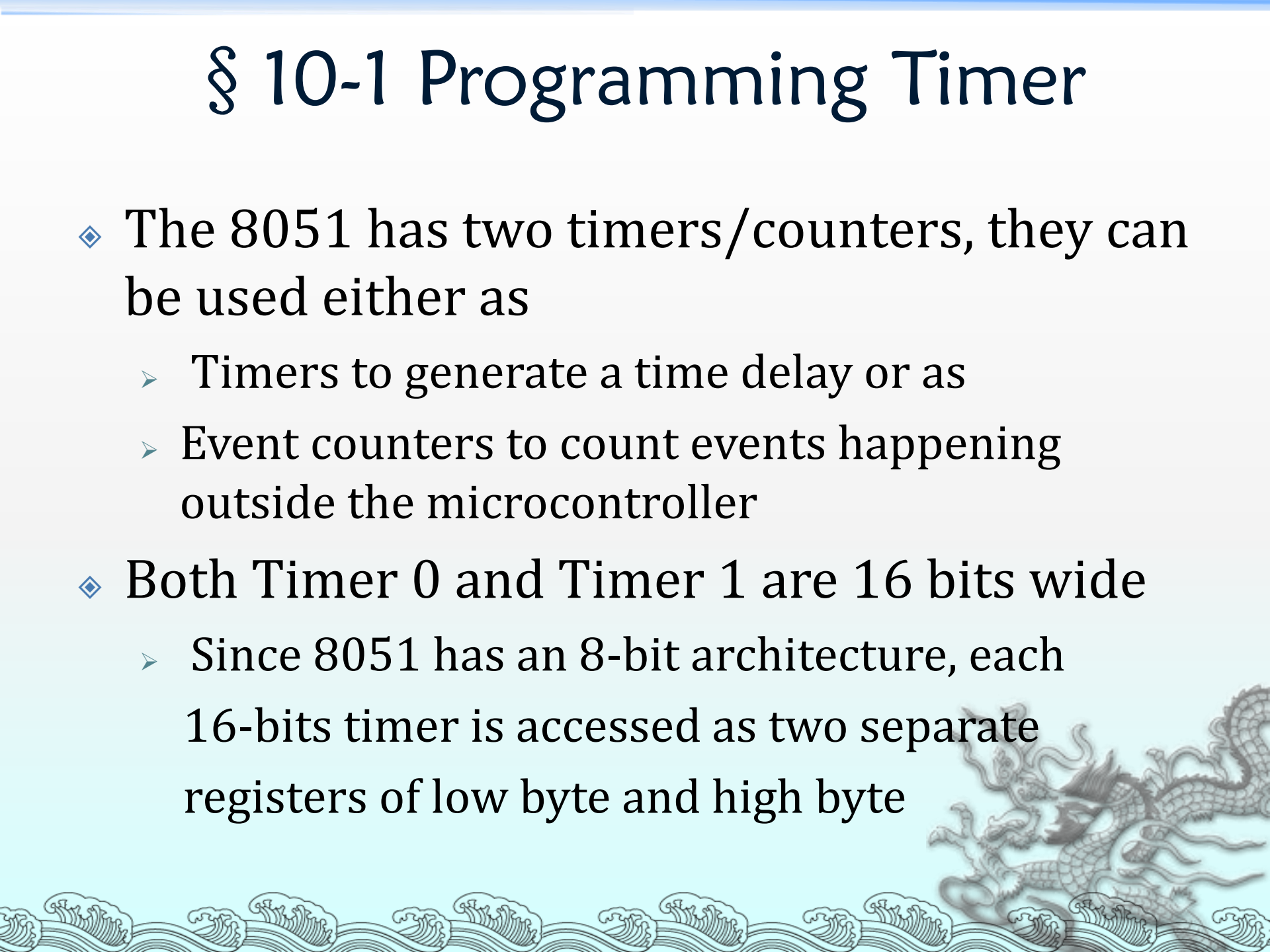
Outline

- § 10-1 Programming Timer
- § 10-2 Counter Programming
- § 10-3 Programming Timers in C
- § 10-4 New Timer

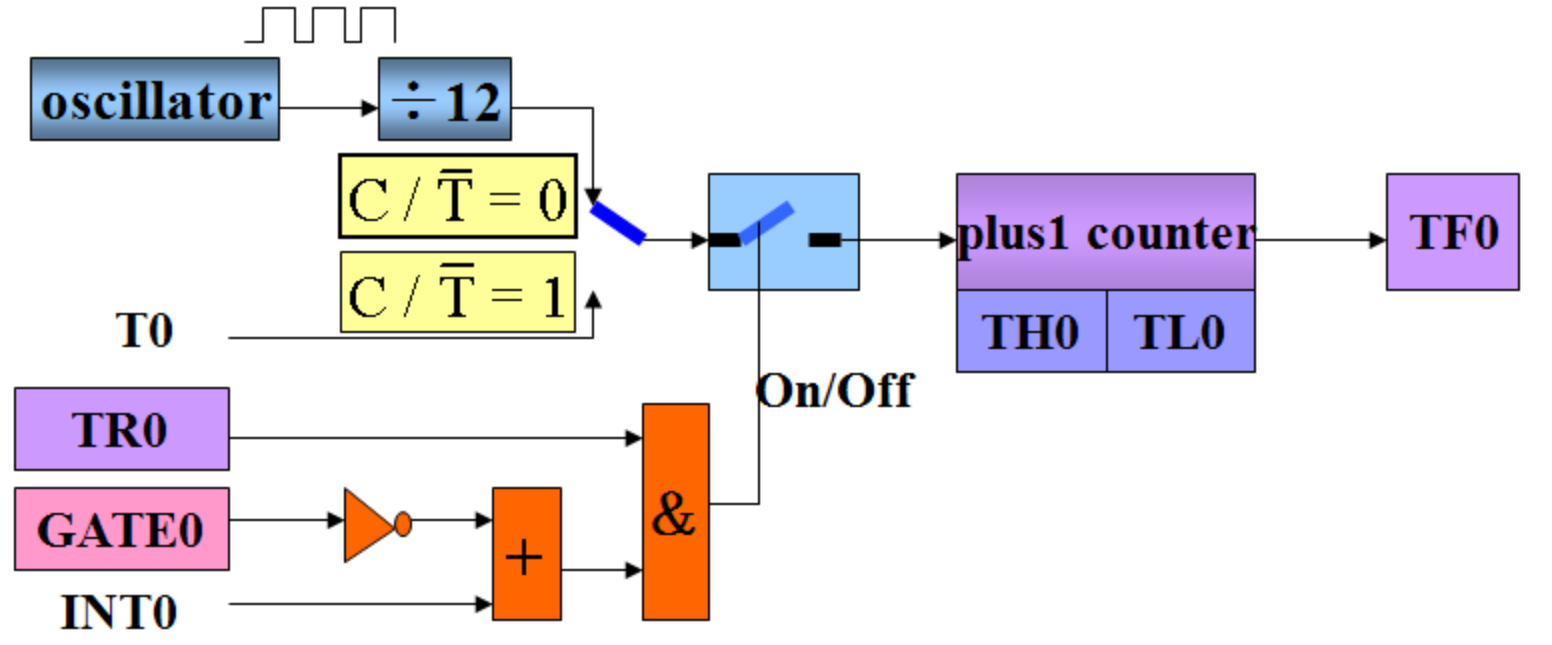


§ 10-1 Programming Timer

- The 8051 has two timers/counters, they can be used either as
  - Timers to generate a time delay or as
  - Event counters to count events happening outside the microcontroller
- Both Timer 0 and Timer 1 are 16 bits wide
  - Since 8051 has an 8-bit architecture, each 16-bits timer is accessed as two separate registers of low byte and high byte

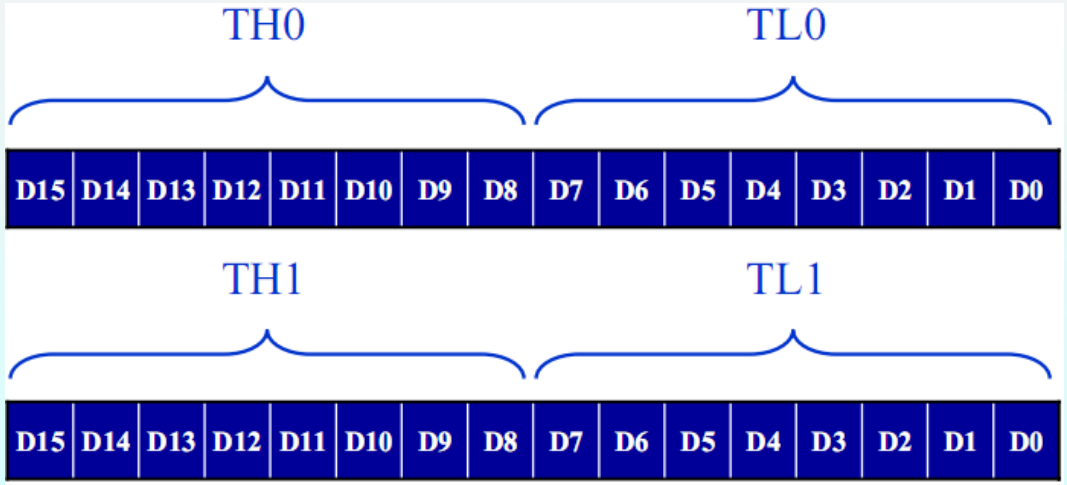


Structure of Timer0



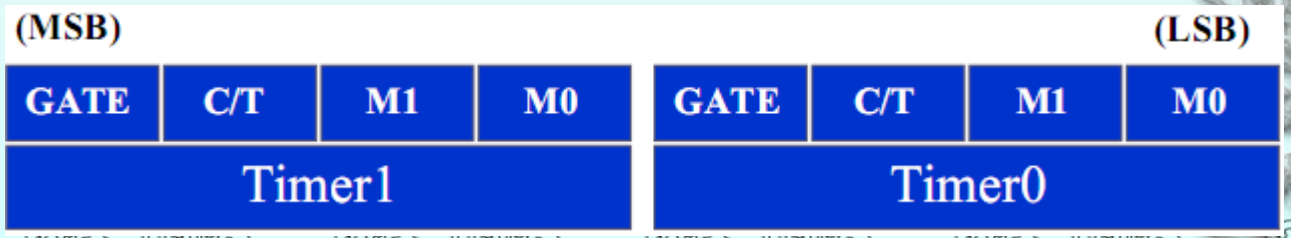
Timer 0 & 1 Registers

- Accessed as low byte and high byte
  - The low byte register is called TL0/TL1 and
  - The high byte register is called TH0/TH1
  - Accessed like any other register
    - ✓ MOV TL0,#4FH
    - ✓ MOV R5,TH0



TMOD Register

- Both timers 0 and 1 use the same register, called TMOD (timer mode), to set the various timer operation modes TMOD is a 8-bit register
  - The lower 4 bits are for Timer 0
  - The upper 4 bits are for Timer 1
  - In each case,
    - ✓ The lower 2 bits are used to set the timer mode
    - ✓ The upper 2 bits to specify the operation



TMOD Register

Timer or counter selected
Cleared for timer operation (input from internal system clock)
Set for counter operation (input from Tx input pin)

Gating control when set.
Timer/counter is enable only while the INTx pin is high and the TRx control bit is set
When cleared, the timer is enabled when the TRx control bit is set

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overfolws
1	1	3	**Split timer mode**

TMOD Register

Example 10-1

Indicate which mode and which timer are selected for each of the following.

(a) MOV TMOD, #01H (b) MOV TMOD, #20H (c) MOV TMOD, #12H

Solution:

We convert the value from hex to binary. From Figure 9-3 we have:

(a) TMOD = 00000001, mode 1 of timer 0 is selected.

(b) TMOD = 00100000, mode 2 of timer 1 is selected.

(c) TMOD = 00010010, mode 2 of timer 0, and mode 1 of timer 1 are selected.

Example 10-2

Find the timer's clock frequency and its period for various 8051-based system, with the crystal frequency 11.0592 MHz when C/T bit of TMOD is 0.

Solution:

1/12 × 11.0529 MHz = 921.6 MHz;

T = 1/921.6 kHz = 1.085 us

If C/T = 0, it is used as a timer for time delay generation.
The clock source for the time delay is the crystal frequency of the 8051

TMOD Register

- Timers of 8051 do starting and stopping by either software or hardware control
  - In using software to start and stop the timer where GATE=0
    - ✓ The start and stop of the timer are controlled by way of software by the TR (timer start) bits TR0 and TR1
      - – The SETB instruction starts it, and it is stopped by the CLR instruction
      - – These instructions start and stop the timers as l
      TMOD register
  - The hardware way of starting and stopping the t
    source is achieved by making GATE=1 in the TM

• Timer 0, mode 2
• C/T = 0 to use XTAL clock source
• gate = 0 to use internal (software) start and stop method.

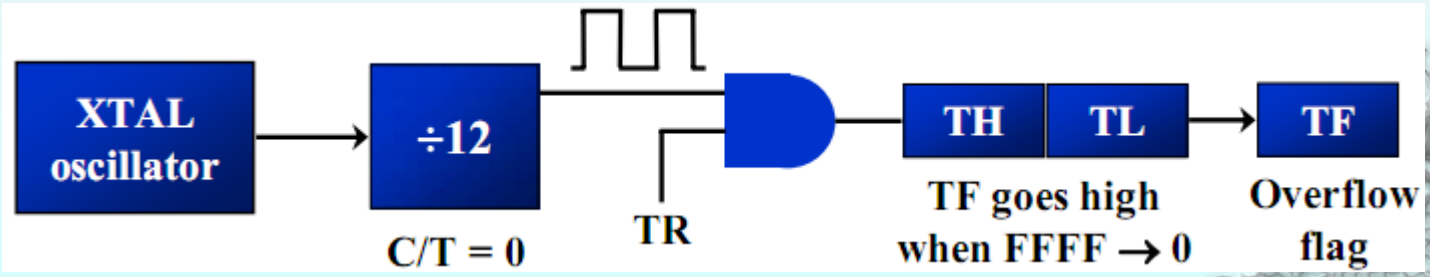
Find the value for TMOD if we want to program timer 0 in mode 2, use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

TMOD = 0000 0010

Mode 1 Programming

- The following are the characteristics and operations of mode1:

1. It is a 16-bit timer; therefore, it allows value of 0000 to FFFFH to be loaded into the timer's register TL and TH
2. After TH and TL are loaded with a 16-bit initial value, the timer must be started
   - This is done by SETB TR0 for timer 0 and SETB TR1 for timer 1
3. After the timer is started, it starts to count up
   - It counts up until it reaches its limit of FFFFH

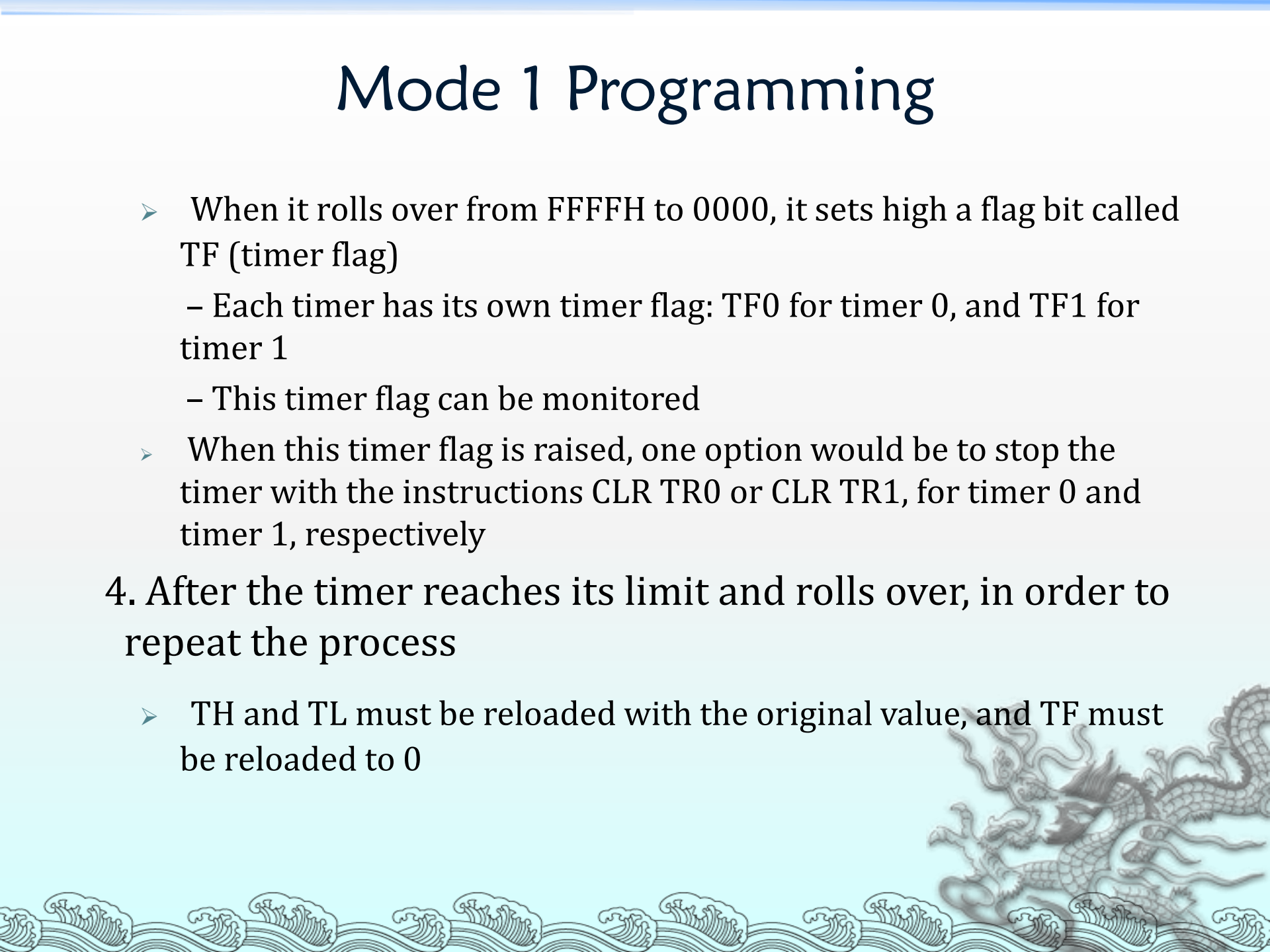


Mode 1 Programming

- When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag)
  – Each timer has its own timer flag: TF0 for timer 0, and TF1 for timer 1
  – This timer flag can be monitored
- When this timer flag is raised, one option would be to stop the timer with the instructions CLR TR0 or CLR TR1, for timer 0 and timer 1, respectively

4. After the timer reaches its limit and rolls over, in order to repeat the process

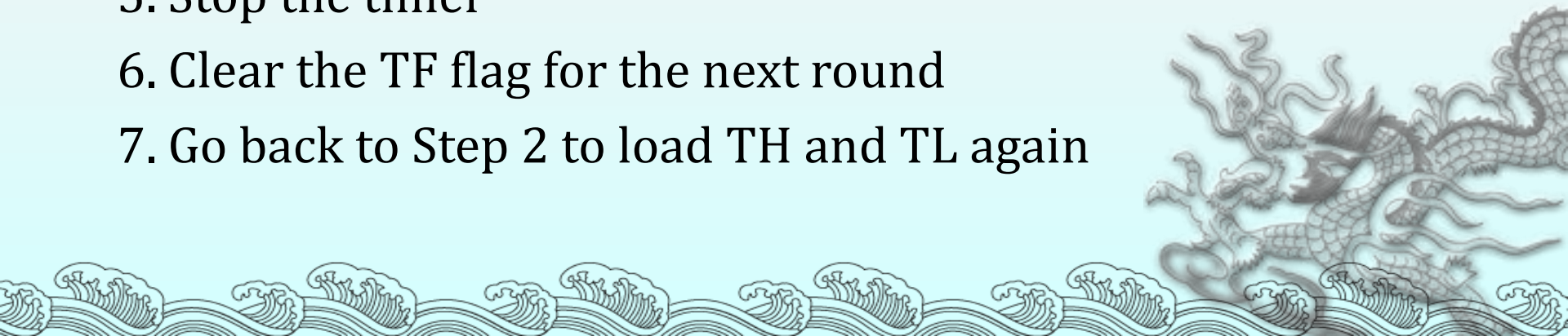
- TH and TL must be reloaded with the original value, and TF must be reloaded to 0



Steps to Mode 1 Program

To generate a time delay

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected
2. Load registers TL and TH with initial count value
3. Start the timer
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see if it is raised
   - Get out of the loop when TF becomes high
5. Stop the timer
6. Clear the TF flag for the next round
7. Go back to Step 2 to load TH and TL again



Example 10-3

In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program

```
        MOV  TMOD,#01 ;Timer 0, mode 1(16-bit mode)
HERE:   MOV  TL0,#0F2H ;TL0=F2H, the low byte
        MOV  TH0,#0FFH ;TH0=FFH, the high byte
        CPL  P1.5 ;toggle P1.5
        ACALL DELAY
        SJMP HERE
```

In the above program notice the following step.

1. TMOD is loaded.
2. FFF2H is loaded into TH0-TL0.
3. P1.5 is toggled for the high and low portions of the pulse.

```
DELAY:
        SETB TR0          ;start the timer 0
AGAIN:  JNB  TF0,AGAIN    ;monitor timer flag 0 until it rolls over
        CLR  TR0          ;stop timer 0
        CLR  TF0          ;clear timer 0 flag
        RET
```

4. The DELAY subroutine using the timer is called.
5. In the DELAY subroutine, timer 0 is started by the SETB TR0 instruction.
6. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFFH. One more clock rolls it to 0, raising the timer flag (TF0=1). At that point, the JNB instruction falls through.

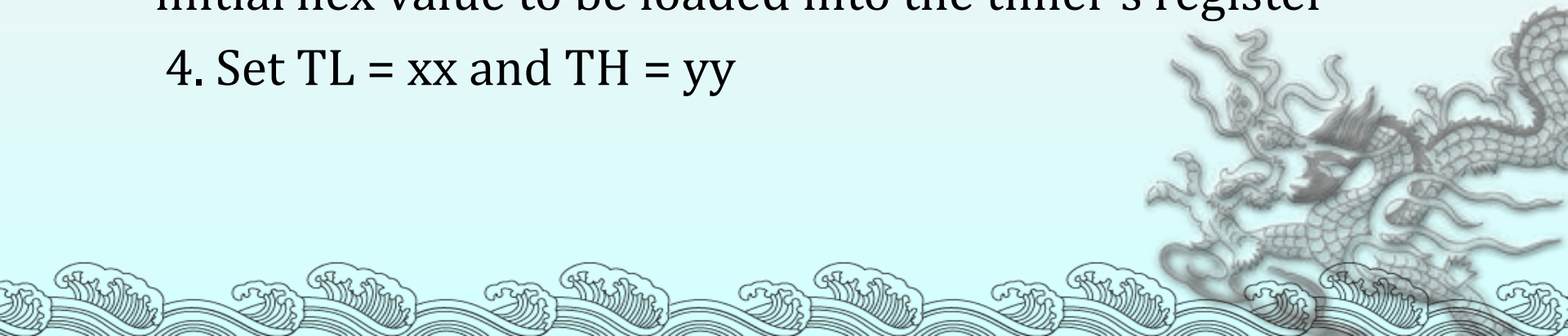
7. Timer 0 is stopped by the instruction CLR TR0. The DELAY subroutine ends, and the process is repeated.

Notice that to repeat the process, we must reload the TL and TH registers, and start the process is repeated

Finding the Loaded Timer Values

- To calculate the values to be loaded into the TL and TH registers, look at the following example
  - Assume XTAL = 11.0592 MHz, we can use the following steps for finding the TH, TL registers' values

1. Divide the desired time delay by 1.085 us
2. Perform 65536 – n, where n is the decimal value we got in Step1
3. Convert the result of Step2 to hex, where yyxx is the initial hex value to be loaded into the timer's register
4. Set TL = xx and TH = yy



Mode 2 Programming

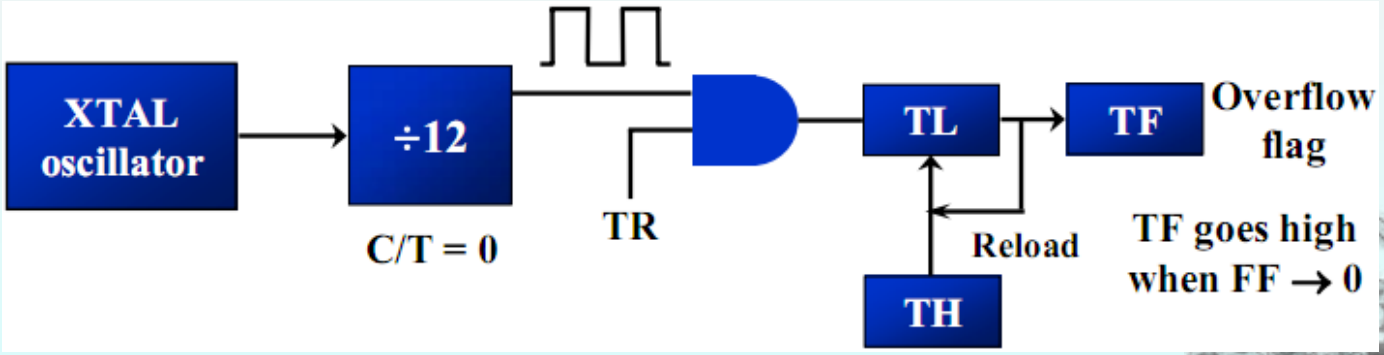
- The following are the characteristics and operations of mode 2:

1. It is an 8-bit timer; therefore, it allows only values of 00 to FFH to be loaded into the timer's register TH
2. After TH is loaded with the 8-bit value, the 8051 gives a copy of it to TL
   - Then the timer must be started
   - This is done by the instruction SETB TR0 for timer 0 and SETB TR1 for timer 1
3. After the timer is started, it starts to count up by incrementing the TL register
   - It counts up until it reaches its limit of FFH
   - When it rolls over from FFH to 00, it sets high the TF (timer flag)

Mode 2 Programming

4. When the TL register rolls from FFH to 0 and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register

- To repeat the process, we must simply clear TF and let it go without any need by the programmer to reload the original value
- This makes mode 2 an auto-reload, in contrast with mode 1 in which the programmer has to reload TH and TL



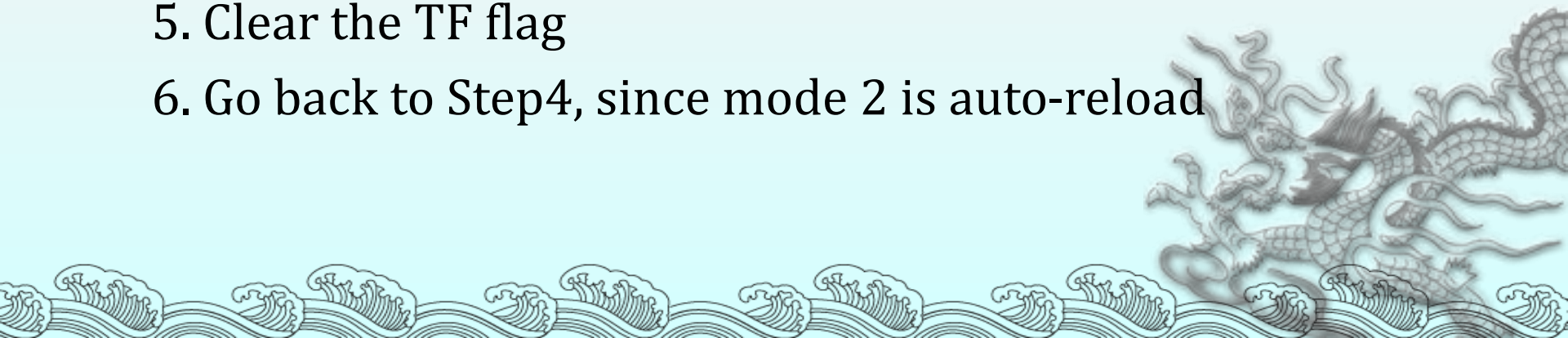
Steps to Mode 2 Program

- To generate a time delay

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used, and the timer mode (mode 2) is selected
2. Load the TH registers with the initial count value
3. Start timer
4. Keep monitoring the timer flag (TF) with the

   JNB TFx , target instruction to see whether it is raised

   - Get out of the loop when TF goes high
5. Clear the TF flag
6. Go back to Step4, since mode 2 is auto-reload



Example 10-13

Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following program

```
        MOV   TMOD,#20H ;T1/8-bit/auto reload
        MOV   TH1,#5 ;TH1 = 5
        SETB  TR1 ;start the timer 1
BACK:   JNB   TF1,BACK  ;till timer rolls over
        CPL   P1.0 ;P1.0 to hi, lo
        CLR   TF1 ;clear Timer 1 flag
        SJMP  BACK ;mode 2 is auto-reload
```

Solution:

First notice the target address of SJMP. In mode 2 we do not need to reload TH since it is auto-reload. Now (256 - 05)$\times$1.085 us = 251$\times$1.085 us = 272.33us is the high portion of the pulse. Since it is a 50% duty cycle square wave, the period T is twice that; as a result T = 2$\times$272.33 us = 544.67us and the frequency = 1.83597 kHz

Example 10-14

Find the frequency of a square wave generated on pin P1.0.

Solution:

```
        MOV   TMOD,#2H  ;Timer 0, mod 2 (8-bit, auto reload)
        MOV   TH0,#0
AGAIN:  MOV   R5,#250    ;multiple delay count
        ACALL DELAY
        CPL   P1.0
        SJMP  AGAIN
DELAY:  SETB TR0 ;start the timer 0
BACK:   JNB   TF0,BACK  ;stay timer rolls over
        CLR   TR0 ;stop timer
        CLR   TF0 ;clear TF for next round
        DJNZ  R5,DELAY
        RET
```

T = 2 (250 $\times$ 256 $\times$ 1.085 us) = 138.88ms, and frequency = 72 Hz

Example 10-15

Assuming that we are programming the timers for mode 2, find the value (in hex) loaded into TH for each of the following cases.

(a) MOV TH1,#-200 (b) MOV TH0,#-60
(c) MOV TH1,#-3 (d) MOV TH1,#-12
(e) MOV TH0,#-48

Solution:

You can use the Windows scientific calculator to verify the result provided by the assembler. In Windows calculator, select decimal and enter 200. Then select hex, then +/- to get the TH value. Remember that we only use the right two digits and ignore the rest since our data is an 8-bit data.

Decimal	2's complement (TH value)
-3	FDH
-12	F4H
-48	D0H
-60	C4H
-200	38H

The number 200 is the timer count till the TF is set to 1

The advantage of using negative values is that you don't need to calculate the value loaded to THx

§ 10-2 Counter Programming

- Timers can also be used as counters counting events happening outside the 8051
  - When it is used as a counter, it is a pulse outside of the 8051 that increments the TH, TL registers
  - TMOD and TH, TL registers are the same as for the timer discussed previously
- Programming the timer in the last section also applies to programming it as a counter
  - Except the source of the frequency

C/T Bit in TMOD Register

- The C/T bit in the TMOD registers decides the source of the clock for the timer
  - When C/T = 1, the timer is used as a counter and gets its pulses from outside the 8051
    - ✓ The counter counts up as pulses are fed from pins 14 and 15, these pins are called T0 (timer 0 input) and T1 (timer 1 input)

Port 3 pins used for Timers 0 and 1

Pin	Port Pin	Function	Description
14	P3.4	T0	Timer/counter 0 external input
15	P3.5	T1	Timer/counter 1 external input

Example 10-16

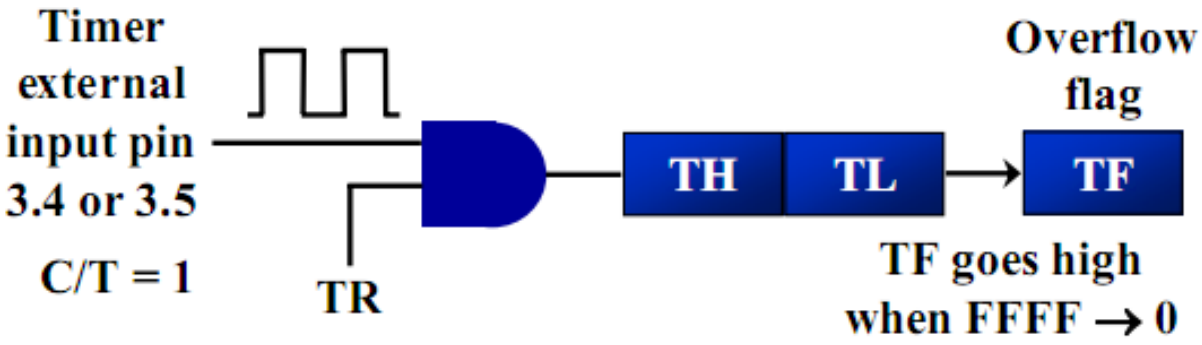
Assuming that clock pulses are fed into pin T1, write a program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P2, which connects to 8 LEDs.

Solution:

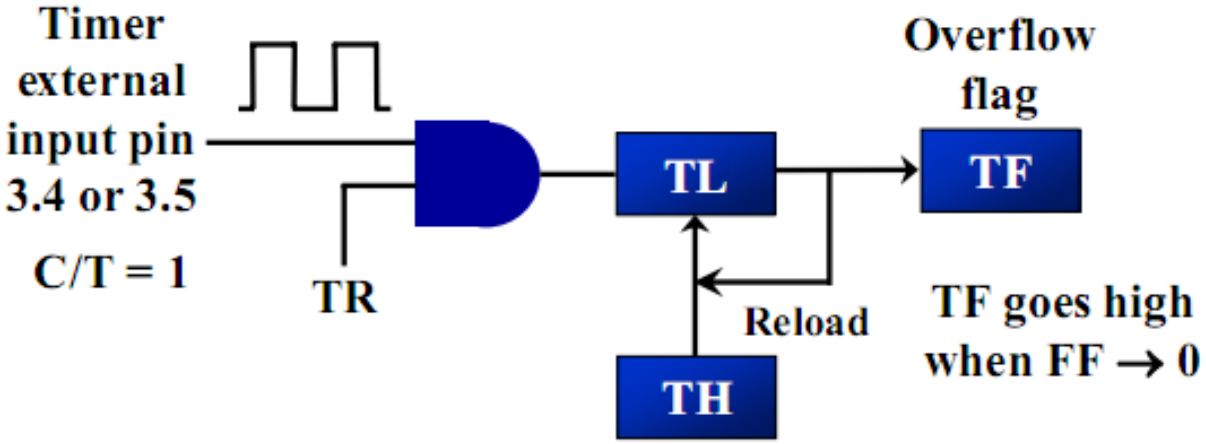
```
        MOV   TM0D,#01100000B   ;counter 1, mode 2,
                                ;C/T=1 external pulses
        MOV   TH1,#0            ;clear TH1
        SETB  P3.5              ;make T1 input
AGAIN:  SETB  TR1               ;start the counter
BACK:   MOV   A,TL1             ;get copy of TL
        MOV   P2,A              ;display it on port 2
        JNB   TF1,Back          ;keep doing, if TF = 0
        CLR   TR1               ;stop the counter 1
        CLR   TF1               ;make TF=0
        SJMP  AGAIN             ;keep doing it
```

Notice in the above program the role of the instruction SETB P3.5. Since ports are set up for output when the 8051 is powered up, we make P3.5 an input port by making it high. In other words, we must configure (set high) the T1 pin (pin P3.5) to allow pulses to be fed into it.

Timer with external input (Mode 1)



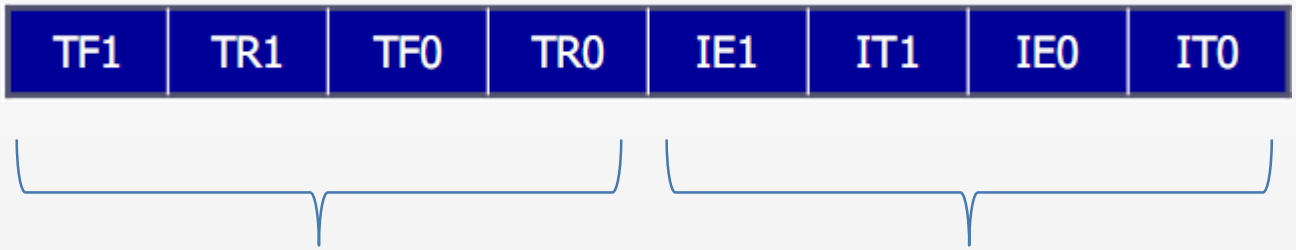
Timer with external input (Mode 2)



TCON Register

- TCON (timer control) register is an 8-bit register

TCON: Timer/Counter Control Register



The upper four bits are used to store the TF and TR bits of both timer 0 and 1

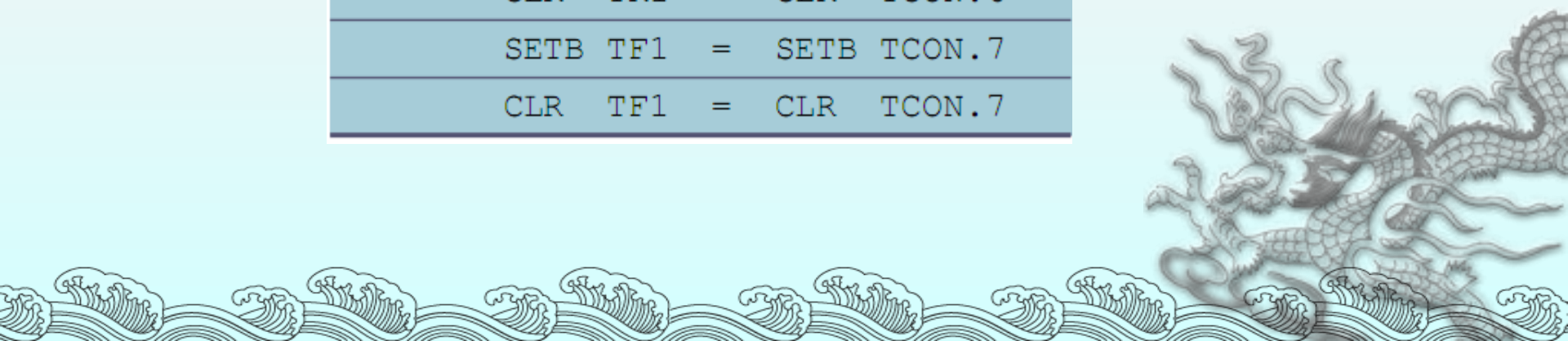
The lower 4 bits are set aside for controlling the interrupt bits

TCON Register

- TCON register is a bit-addressable register

Equivalent instruction for the Timer Control Register

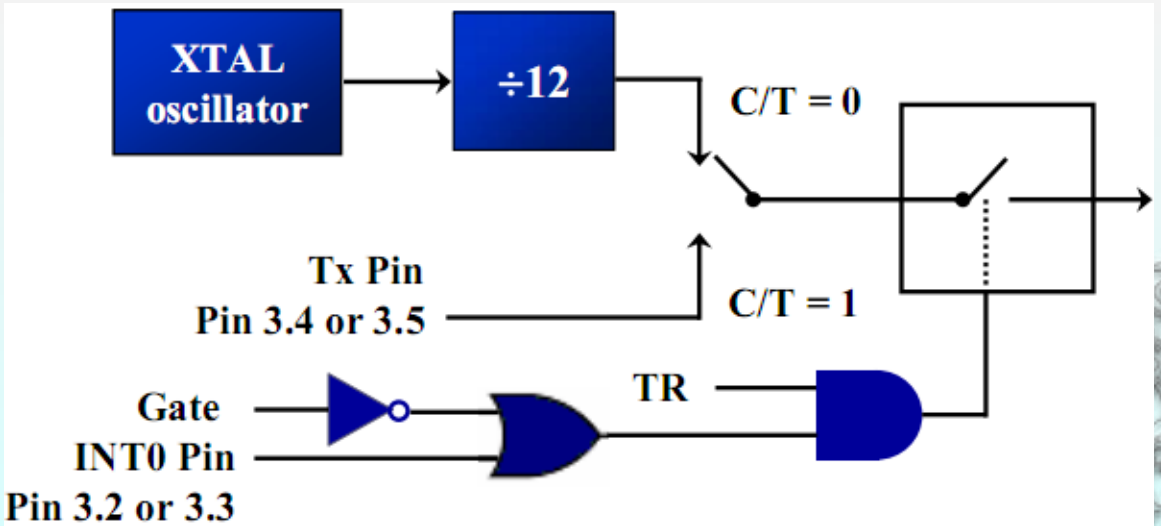
For timer 0			
SETB	TR0	=	SETB TCON.4
CLR	TR0	=	CLR TCON.4
SETB	TF0	=	SETB TCON.5
CLR	TF0	=	CLR TCON.5
For timer 1			
SETB	TR1	=	SETB TCON.6
CLR	TR1	=	CLR TCON.6
SETB	TF1	=	SETB TCON.7
CLR	TF1	=	CLR TCON.7



TCON Register

Case of GATE = 1

- If GATE = 1, the start and stop of the timer are done externally through pins P3.2 and P3.3 for timers 0 and 1, respectively
  - This hardware way allows to start or stop the timer externally at any time via a simple switch



§ 10-3 Programming Timers in C

Accessing Timer Registers

Example 10-17

Write an 8051 C program to toggle all the bits of port P1 continuously with some delay in between. Use Timer 0, 16-bit mode to generate the delay.

Solution:

```
#include <reg51.h>
void T0Delay(void);
void main(void){
   while (1) {
      P1=0x55;
      T0Delay();
      P1=0xAA;
      T0Delay();
   }
}

void T0Delay(){
   TMOD=0x01;
   TL0=0x00;
   TH0=0x35;
   TR0=1;
   while (TF0==0);
   TR0=0;
   TF0=0;
}
```

FFFFH – 3500H = CAFFH
= 51967 + 1 = 51968
51968 × 1.085μs = 56.384ms
is the approximate delay

Calculating Delay Length Using Timers

- To speed up the 8051, many recent versions of the 8051 have reduced the number of clocks per machine cycle from 12 to four, or even one

- The frequency for the timer is always $1/12^{th}$ the frequency of the crystal attached to the 8051, regardless of the 8051 version

Times 0/1 Delay Using Mode 1 (16-bit Non Auto-reload)

Example 10-18

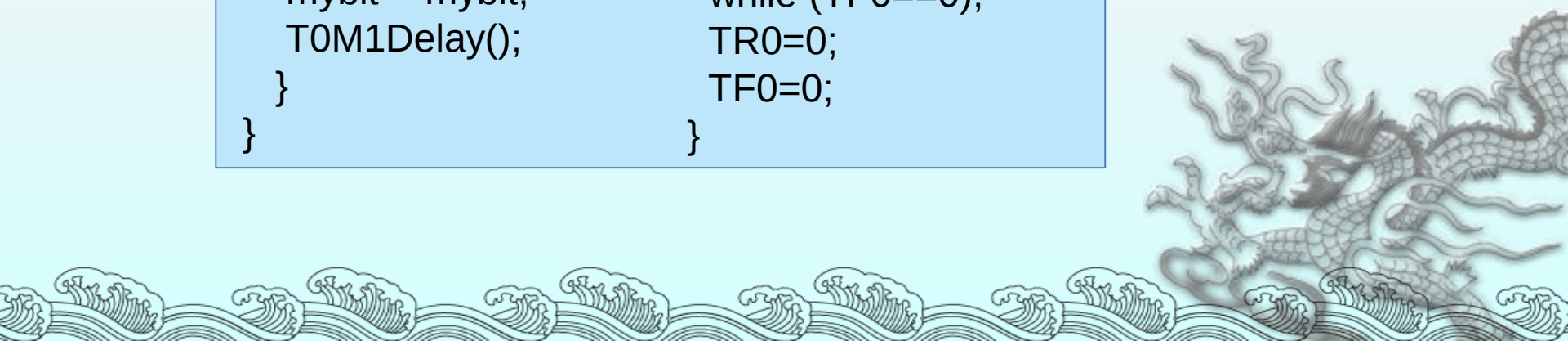
Write an 8051 C program to toggle only bit P1.5 continuously every 50ms. Use Timer 0, mode 1 (16-bit) to create the delay. Test the program on the (a) AT89C51 and (b) DS89C420.

Solution:

```
#include <reg51.h>
void T0M1Delay(void);
sbit mybit=P1^5;
void main(void)
{
  while (1)
   {
     mybit=~mybit;
     T0M1Delay();
    }
 }
```

```
void T0M1Delay(void)
{
   TMOD=0x01;
   TL0=0xFD;
   TH0=0x4B;
   TR0=1;
   while (TF0==0);
   TR0=0;
   TF0=0;
 }
```

FFFFH – 4BFDH = B402H
= 46082 + 1 = 46083
46083 × 1.085 μs = 50ms



Times 0/1 Delay Using Mode 1 (16-bit Non Auto-reload)

Example 10-19

Write an 8051 C program to toggle all bits of P2 continuously every 500ms. Use Timer 1, mode 1 to create the delay.

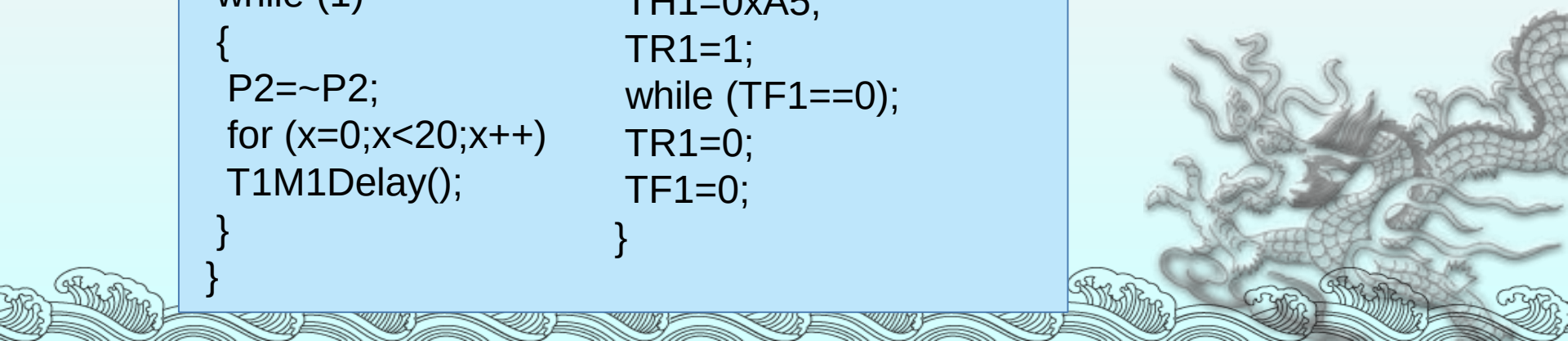
Solution:

//tested for DS89C420, XTAL = 11.0592 MHz

```
#include <reg51.h>
void T1M1Delay(void);
void main(void)
 {
  unsigned char x;
  P2=0x55;
  while (1)
  {
   P2=~P2;
   for (x=0;x<20;x++)
   T1M1Delay();
  }
 }

void T1M1Delay(void)
{
 TMOD=0x10;
 TL1=0xFE;
 TH1=0xA5;
 TR1=1;
 while (TF1==0);
 TR1=0;
 TF1=0;
}
```

A5FEH = 42494 in decimal
65536 – 42494 = 23042
23042 × 1.085μs = 25ms and
20 × 25ms = 500ms



Example 10-22

万年历

Solution:

```
#include <reg51.h>
void Time0Isr(void) interrupt 1
{
   TH0=0x3c;
   TL0=0xb0;
   sec_50ms++;
   if (sec_50ms==_____)
            sec++;
   if (sec==____)
   {
            min++;
            sec=0;
   }
   if (min==____)
   {
            hour++;
            min=0;
   }
   If (hour==_____)
   {
            day++;
            hour=0;
    }
}
```

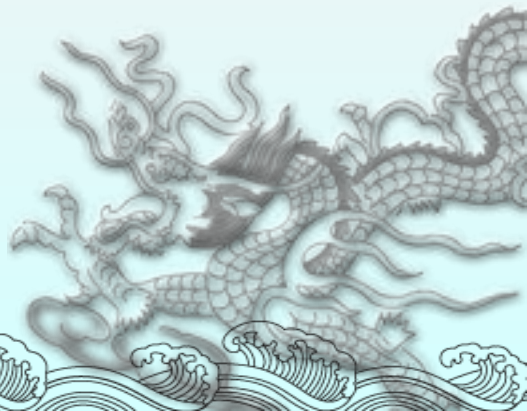
C Programming of Timers as Counters

Example 10-23

Assume that a 1-Hz external clock is being fed into pin T1 (P3.5). Write a C program for counter 1 in mode 2 (8-bit auto reload) to count up and display the state of the TL1 count on P1. Start the count at 0H.

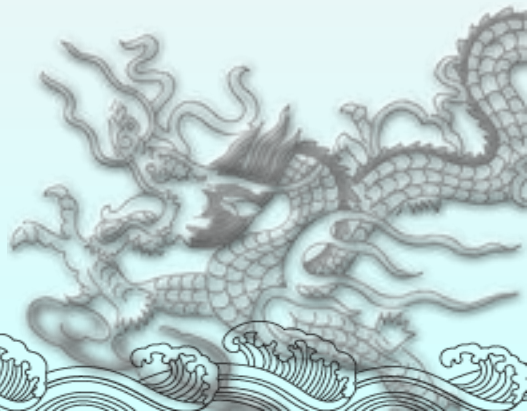
Solution:

```
#include <reg51.h>
sbit T1=P3^5;
void main(void){
   T1=1;
   TMOD=0x60;
   TH1=0;
   while (1) {
     do {
     TR1=1;
     P1=TL1;
    }
   while (TF1==0);
   TR1=0;
   TF1=0;
  }
}
```



Discussion: What's new in Timer

- **More Timer/Counter**
- **More clock source**
- **16 bits reload**
- **PWM output automatically**
- **Up/Down counter**
- **Dynamic trigger level**
- **Watch dog**



- MSP430

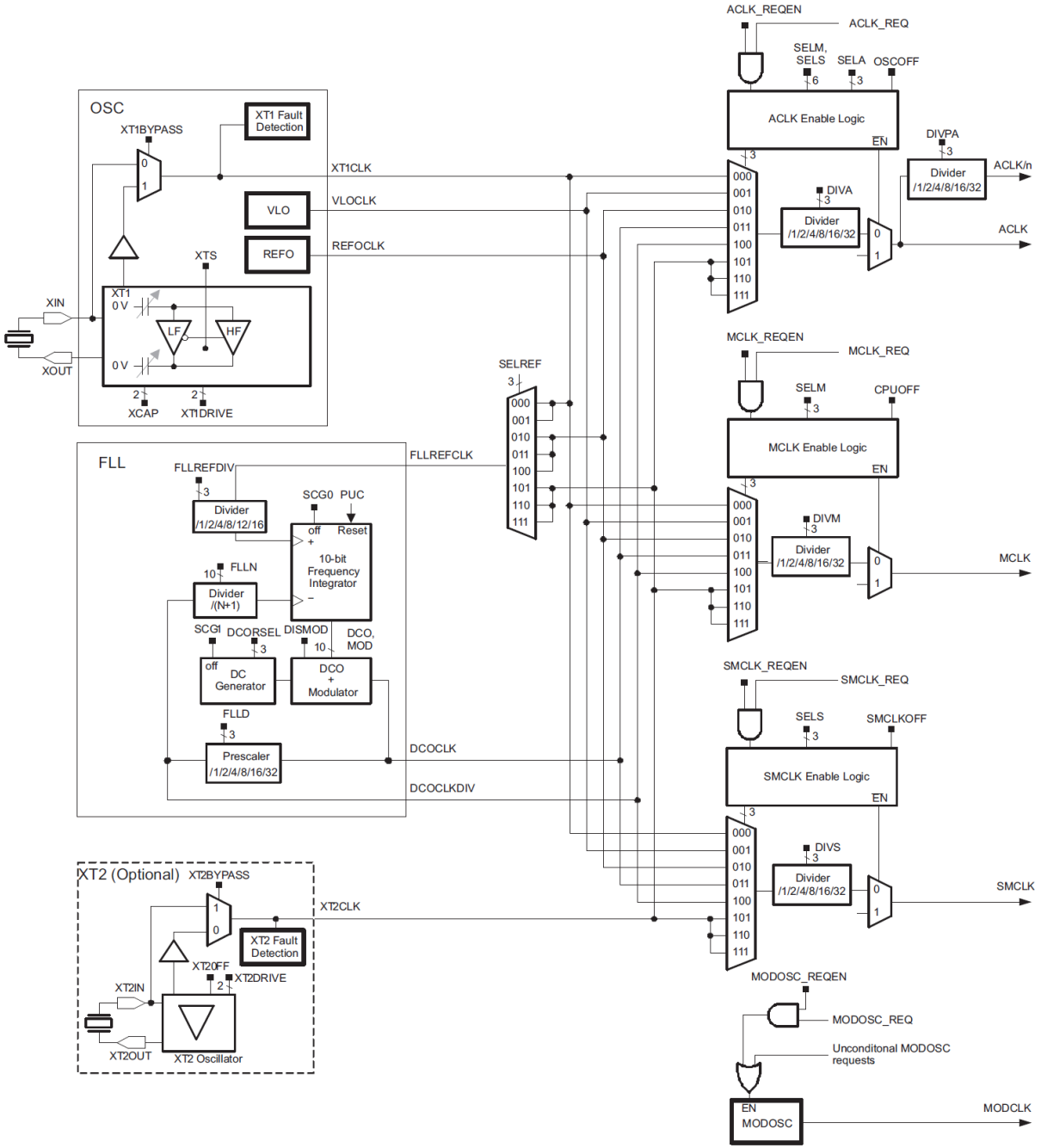


Figure 3-1. Unified Clock System Block Diagram

作业：实时时钟

要求：

1. 编写程序，实现一个实时时钟；
2. 从默认时间点开始计时；
3. 必须使用中断响应程序来处理；
4. 计时结果（年、月、日、时、分、秒）存放在RAM的数组中；
5. 为了调试方便，可以缩短计时时间，实现短时间内计时数天、数月、数年，要在实验报告中说明当前的计时比例；
6. 实验报告中要附上调试结果截图。

THANK YOU!!