EE 354 **Fall 2016**

Notes on timer operations

The AT89C51CC03 has two general purpose timer counters each of which is 16-bits long. Both count upward and both can be used in either a polling or an interrupt mode. The AT89C51CC03 has an additional timer (Timer 2) which is used for setting baud rate for serial communication.

Figure 5. Clock CPU Generation Diagram

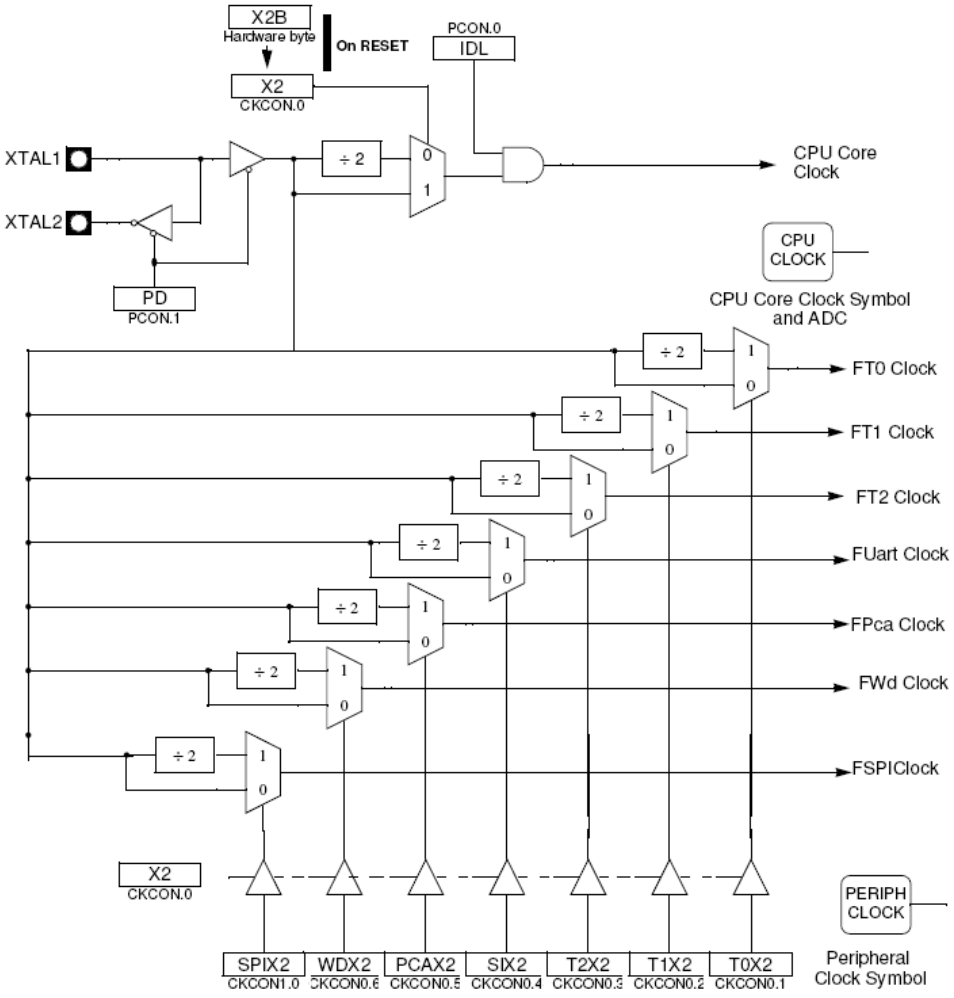

Figure 1
Clock signals for the AT89C51CC03.

The clock labeled CPU Core Clock drives the CPU and it's speed can be changed using bit CKCON.0. If this bit is 1 the CPU is said to be in the X2 mode and the CPU is clocked at the same rate as the crystal frequency. If CKCON.0 = 0, the CPU is in the standard (STD) mode.

There are three clock signals which drive the timers: FT0 Clock, FT1 Clock, and FT2 Clock. In Standard mode (NOT double clocked). Each of these can run at either the clock frequency or at ½ of the clock frequency depending on a bit in CKCON. The timer clock frequency is independent of the CPU core clock frequency.

For the board being used in class there is an external crystal at 28.2076MHz. If CKCON.1 = 1, timer 0 runs at 28.2076MHz/2. If CKCON.1 = 0, timer 0 runs at 28.2076 MHz.

Mode registers

There are four registers associated with each timer/counter. These are the timer control register TCON, the timer mode register TMOD, and the 16 bit counter register which is divided into to 8-bit registers called TLX and THX where X is 0, 1, or 2.

TCON

The bit assignments for TCON are shown in Figure 2. TFX is the timer overflow flag. This is the carry bit that is set when the timer counts past it's register limit and overflows. TRX is the timer run bit. This bit must be set to 1 to get the timer to run.

Table 30. TCON Register
TCON (S:88h)
Timer/Counter Control Register

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Description
7	TF1	**Timer 1 Overflow Flag** Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 1 register overflows.
6	TR1	**Timer 1 Run Control Bit** Clear to turn off Timer/Counter 1. Set to turn on Timer/Counter 1.
5	TF0	**Timer 0 Overflow Flag** Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 0 register overflows.
4	TR0	**Timer 0 Run Control Bit** Clear to turn off Timer/Counter 0. Set to turn on Timer/Counter 0.
3	IE1	**Interrupt 1 Edge Flag** Cleared by hardware when interrupt is processed if edge-triggered (see IT1). Set by hardware when external interrupt is detected on INT1# pin.
2	IT1	**Interrupt 1 Type Control Bit** Clear to select low level active (level triggered) for external interrupt 1 (INT1#). Set to select falling edge active (edge triggered) for external interrupt 1.
1	IE0	**Interrupt 0 Edge Flag** Cleared by hardware when interrupt is processed if edge-triggered (see IT0). Set by hardware when external interrupt is detected on INT0# pin.
0	IT0	**Interrupt 0 Type Control Bit** Clear to select low level active (level triggered) for external interrupt 0 (INT0#). Set to select falling edge active (edge triggered) for external interrupt 0.

Reset Value = 0000 0000b

Figure 2
Timer Control register TCON

TMOD

The timer mode register bits are shown in Figure 3. TMOD is a single register where the upper nibble contains the mode control bits for Timer 1 and the lower nibble contains the mode control bits for Timer 0.

Table 31. TMOD Register

TMOD (S:89h)
Timer/Counter Mode Control Register

7	6	5	4	3	2	1	0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Bit Number	Bit Mnemonic	Description
7	GATE1	**Timer 1 Gating Control Bit** Clear to enable Timer 1 whenever TR1 bit is set. Set to enable Timer 1 only while INT1# pin is high and TR1 bit is set.
6	C/T1#	**Timer 1 Counter/Timer Select Bit** Clear for Timer operation: Timer 1 counts the divided-down system clock. Set for Counter operation: Timer 1 counts negative transitions on external pin T1.
5	M11	**Timer 1 Mode Select Bits** M11 M01 Operating mode 0 0 Mode 0: 8-bit Timer/Counter (TH1) with 5-bit prescaler (TL1). 0 1 Mode 1: 16-bit Timer/Counter. 1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL1) (1) 1 1 Mode 3: Timer 1 halted. Retains count
4	M01	
3	GATE0	**Timer 0 Gating Control Bit** Clear to enable Timer 0 whenever TR0 bit is set. Set to enable Timer/Counter 0 only while INT0# pin is high and TR0 bit is set.
2	C/T0#	**Timer 0 Counter/Timer Select Bit** Clear for Timer operation: Timer 0 counts the divided-down system clock. Set for Counter operation: Timer 0 counts negative transitions on external pin T0.
1	M10	**Timer 0 Mode Select Bit** M10 M00 Operating mode 0 0 Mode 0: 8-bit Timer/Counter (TH0) with 5-bit prescaler (TL0). 0 1 Mode 1: 16-bit Timer/Counter. 1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL0) (2) 1 1 Mode 3: TL0 is an 8-bit Timer/Counter TH0 is an 8-bit Timer using Timer 1's TR0 and TF0 bits.
0	M00	

1. Reloaded from TH1 at overflow.
2. Reloaded from TH0 at overflow.

Reset Value = 0000 0000b

Figure 3
Timer mode register.

For Timer 0 bits 0 and 1 control the timer mode so that there are four modes as shown in the Figure 3. Timer 1 has a similar set of mode bits but for Timer 1 the mode bits are bits 4 and 5 of TMOD.

T0M1	T0M0	Mode description
0	0	13-bit counter timer for 8048 compatibility
0	1	16- bit counter timer
1	0	8-bit counter timer with auto-reload. T0H is loaded automatically into T0L when T0L overflows.
1	1	Dual 8-bit counter timers. (see user's manual)

Figure 4

Mode select bits for Timer 0

Bit 2 for Timer 0 and bit 6 for Timer 1 in TMOD are called the timer source bits and determine whether each timer is used as a counter or a timer. Setting this bit to 0 make a timer and setting it to 1 makes it a counter. In the counter mode the Timers count external inputs on one of the input pins. In the timer mode, the timers count FTX clock.

Bit 3 for Timer 0 and bit 7 for Timer 1 are the gate control bits. These bits turns the timer on or off to allow it to be used to count external events. This bit is cleared to 0 for normal timer operation. Figure 5 summarizes the timer modes.

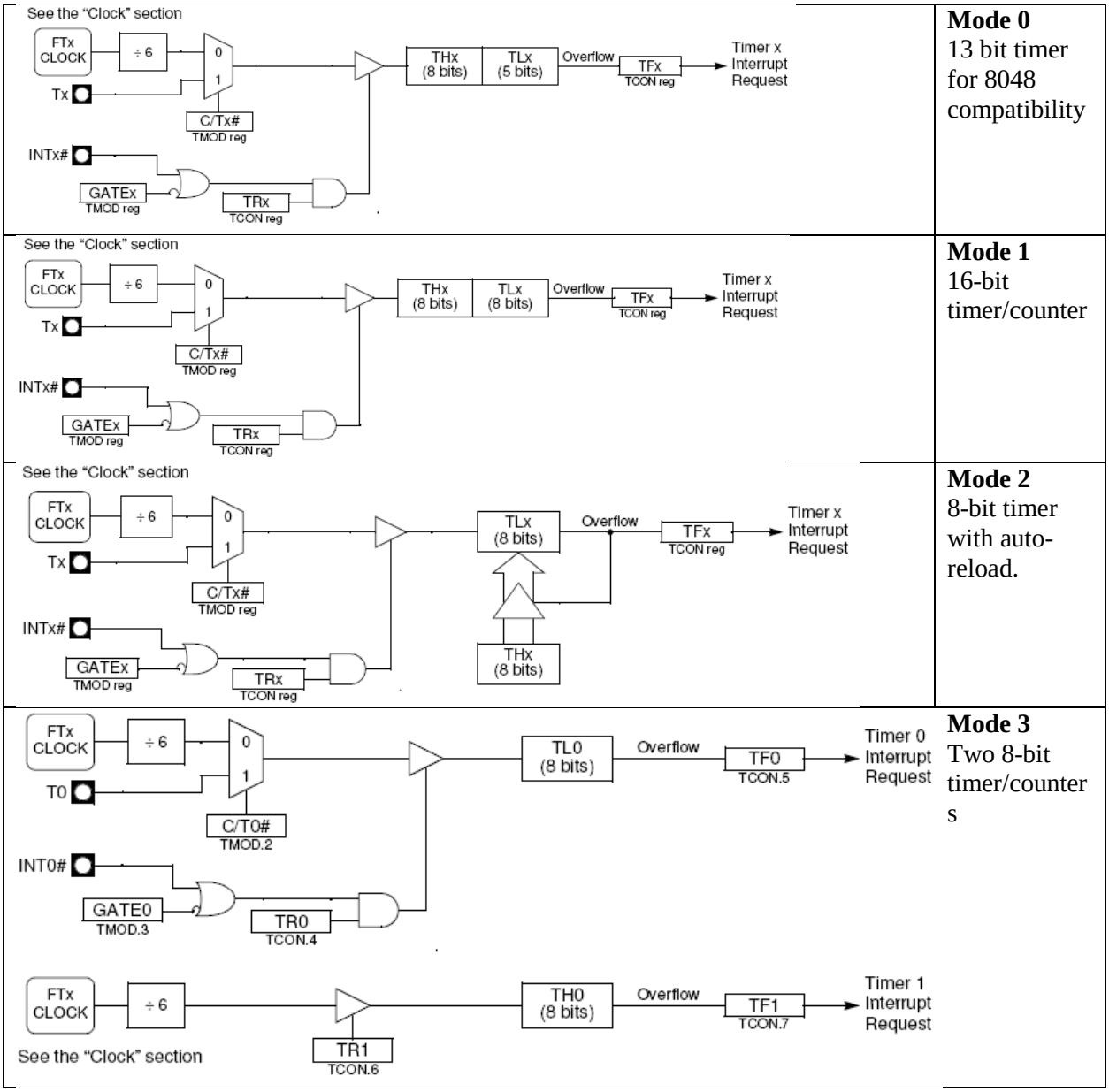

Figure 5

Timer mode summaries.

Setting up Timers in C

Example 1: Set timer 0 to roll over every millisecond. Toggle bit P3.2 each time the timer rolls over. Use polling.

TMOD Register. We can use a 16-bit timer so we want mode 1. We are not using the gated input and we want the timer to count FT0 Clock so all of the other bits are 0. Set TMOD to 0000 0001B = 01H.

TCON Register. In TCON we will need to set TR0 to 1 to get timer 0 to run and we can check TF0 to see when Timer 0 overflows.

Timer Register values for overflow. Setting CCKON to 01h will make FTClock = 28.2076MHz/6 = 4.701267 MHz and the processor core will be double clocked. This gives a period of 0.2127086 µsecond period. To get 1 millisecond = 1000 µseconds we need

$$\frac{1\,\text{count}}{0.2127086\,\text{microseconds}} x \frac{1000\,\text{microseconds}}{\text{millisecond}} = 4701\,\text{counts}$$

Since this is a 16-bit timer it overflows when it runs from 65,535 to 65,536. So we want to initialize the timer to 65,536 – 4701 = 60835 which corresponds to 0EDA3H. So we want to load TH0 with 0EDH and TL0 with A3H.

```
//TimerPolled1Khz.c
// This program produces a square wave 2 msec period on
//  P3.2 using timer 0 in a polled mode.
#include<REG51ac2.h>

void main(void)
   {CKCON = 0x01;   // x2 mode
    TMOD = 0x01;    //Timer 0 mode = not gated, internal clock, 16-bit mode
    //For fosc = 28.2076MHz in x2 mode timer is clocked at 28.2076Mhz/6 =
4.701MHz
    // so period is 1/4.701Mhz = .2127086 usec.  To get 1 msec we need
    // 1000/.2127086 = 4701 counts.  65536 - 4701 = 60835 = 0xEDA3.
    TH0 = 0xED;     //Timer 0 set to ECDC = 60835
    TL0 = 0xA3;
    while(1)
      {TR0 = 1;         //Timer 0 run control bit in TCON
       P3 = P3 ^ 4;     //Toggle bit 3.2
       TF0 = 0;         //Reset timer overflow flag

       while(TF0 == 0); //Wait for time 0 to time out
       TR0 = 0;         //Turn off timer 0
       TH0 = 0xED;      //Reload timer 0
       TL0 = 0xA3;
      }
   }
```

Figure 6

The c code for running Timer 0 for 1 millisecond and toggling P3.2.

Example 2: Use timer 0 and timer 1with interrupts to produce a square wave on P3.2. The square wave should have a 2 msec high time and a 1 msec low time.

TMOD Register. We can use a 16-bit timer so we want mode 1. We are not using the gated input and we want the timer to count FT0 and FT1 clocks so all of the other bits are 0. Set TMOD to 0001 0001B = 11H.

TCON Register. In TCON we will need to set TR0 and TR1 to 1 to get timer 0 and time 1 to run.

Timer Register values for overflow. FT0 is running at 28.2076MHz/6 = 4.701267 MHz which corresponds to a 0.2127086 µsecond period. To get a 1 msec low time we need

$$\frac{1\,\text{count}}{0.2127086\,\text{microseconds}} x \frac{1000\,\text{microseconds}}{\text{millisecond}} = 4701\,\text{counts}$$

To get a 2 msec high time we will need a count of 2 x 4701 = 9402 counts.

Since this is a 16-bit timer it overflows when it runs from 65,535 to 65,536. So we want to initialize the timer to 65,536 – 4701 = 60835 which corresponds to 0EDA3H. So we want to load TH1 with 0EDH and TL1 with A3H. (Timer 1 will count the 1 msec time) Timer 0 will count 65,536 – 9402 = 56134 which corresponds to 0DB46H. We will make TH0 = 0DBh and TL0 = 046h.

Enabling the interrupt. Timer 0 uses interrupt 1so we need to enable the timer 0 interrupt by setting ET0 = 1. We must also set the global interrupt enable bit to 1 using EA = 1. These two enables are in the interrupt enable register IE at location 0A8H. (This is a SFR.) Interrupt 1 has a vector at location 0BH so we must place a jump to the interrupt service routine (ISR) at this location. Finally we must write the ISR. For this application the ISR is going to toggle a bit on P3.2 so it's trivial.

```
//TimerInts.c
// This program produces a square wave with a 2 msec high time and 1 msec low
//  time on P3.2 using timer 0 and timer 1 in an interrupt mode.
#include<REG51ac2.h>

void main(void)
   {CKCON = 0x01;   // x2 mode
    TMOD = 0x11;    //Timer 0 mode = not gated, internal clock, 16-bit mode
                    //Timer 1 mode = not gated, internal clock, 16-bit mode
    //For fosc = 28.2076MHz in x2 mode timer is clocked at 28.2076Mhz/6 =
4.701MHz
    // so period is 1/4.701Mhz = .2127086 usec.  To get 1 msec we need
    // 1000/.2127086 = 4701 counts.  65536 - 4701 = 60835 = 0xEDA3.
    // For 2 msec we need 9402 counts.  65536 - 9402 = 55134 = 0xDB46.

    TH0 = 0xDB;     //Timer 0 set to DB46 -> 55134
    TL0 = 0x46;

    TH1 = 0xED;     //Timer 1 set to EDA3 -> 60835
    TL1 = 0xA3;
    TR0 = 1;
    ET0 = 1;        //Timer 0 interrupt enable
    EA = 1;         //Global interrupt enable
    while(1);
   }
//
void T0Int() interrupt 1 using 1
  {P3 = P3 | 4;  //bit P3.2 to 1
   TR0 = 0;      //Turn timer 0 off
   TH1 = 0xDB;   //Timer 1 set to DB46 = 55134
   TL1 = 0x46;
   TR1 = 1;      //Turn timer 1 on
   ET0 = 0;      //Timer 0 interrupt off
   ET1 = 1;      //Timer 1 interrupt on
  }
//
void T1Int(void) interrupt 3 using 1
  {P3 = P3 & 0xB1;   //bit P3.1 to 0
   TR1 = 0;          //Turn timer 1 off
   TH0 = 0xED;       //Timer 0 set to EDA3 = 60835
   TL0 = 0xA3;
   TR0 = 1;          //Turn timer 0 on
   ET1 = 0;          //Timer 1 interrupt off
   ET0 = 1;          //Timer 0 interrupt on
  }
```

Figure 7

The c code for running Timer 1 for 1 msec and Timer 0 for 2 millisecond and toggling P3.2. This version uses an interrupt with Timer 0 and Timer 1.

Example 3: Use timer 0 with an interrupt to produce a 50 KHz square wave on P3.2. Timer 0 is used in auto reload mode.

Timer Register values for overflow. FT0Clock is running at 28.2076MHz/6 = 4.701267 MHz which corresponds to a 0.2127086 μsecond period. A 50 KHz square wave has a 0.02 millisecond period but there are two transitions per cycle so we need a 0.01 millisecond period for the timer.

To get 0.01 millisecond = 10 μseconds we need

$$\frac{1\,\text{count}}{0.2127086\,\text{microseconds}} x \frac{10\,\text{microseconds}}{\text{millisecond}} = 47\,\text{counts}$$

An 8-bit counter can count to 255 so we can use Timer 0 in the 8-bit auto-reload mode with TH0 loaded initially with 256 – 47 = 209 counts which corresponds to 0D1h.

TMOD Register.
For 8-bit auto-reload we want Timer 0 in mode 2. We are not using the gated input and we want the timer to count FT0 so all of the other bits are 0. Set TMOD to 0000 0010B = 02H.

TCON Register. In TCON we will need to set TR0 to 1 to get timer 0 to run.

Enabling the interrupt. Timer 0 uses interrupt 1so we need to enable the timer 0 interrupt by setting ET0 = 1. We must also set the global interrupt enable bit to 1 using EA = 1. These two enables are in the interrupt enable register IE at location 0A8H. (This is a SFR.) Interrupt 1 has a vector at location 0BH so we must place a jump to the interrupt service routine (ISR) at this location. Finally we must write the ISR. For this application the ISR is going to toggle a bit on P0.0 so it's trivial.

```
//TimerReload.c
// This program produces a 50KHz square wave on P3.2 using
//  Timer 0 in the 8-bit autoreload mode.
//  50KHz corresponds to a 20usec period but interrupt complements
//  P3.2 at twice that rate so we need a 10 usec period for T0.
#include<REG51ac2.h>
void SqWave();
void main(void)
   {CKCON = 0x01;   // x2 mode
    TMOD = 0x02;    //Timer 0 mode = not gated, internal clock, 8-bit, auto
reload
    //For fosc = 28.2076MHz in x2 mode timer is clocked at 28.2076Mhz/6 =
4.701MHz
    // so period is 1/4.701Mhz = .2127086 usec.  To get 10 usec we need
    // 10/.2127608 = 47 counts.  256 - 47 = 209 = 0xD1
    TH0 = 0xD1;     //Timer 0 high set to 209 for 10 micro-seconds
    TR0 = 1;        //Timer 0 run control bit in TCON
    ET0 = 1;        //Timer 0 interrupt enable
    EA = 1;         //Global interrupt enable
    while(1);
   }
//
//Timer 0 comes in on Interrupt 1.
void SqWave() interrupt 1 using 1
  {P3 = P3 ^ 4;  //Exclusive or with 00000100 for bit 3.2
  }
```

Figure 8
The c code for running Timer 0 in an 8-bit auto reload mode and toggling P3.2 every 10 microseconds.