University of British Columbia
Electrical and Computer Engineering
Digital Systems and Microcomputers
CPEN312

Lecture 16: Timers/Counters in the 8051 Microcontroller

Dr. Jesús Calviño-Fraga P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

March 28, 2017

Objectives

- Understand the advantages of using timers/counters.
- Understand the different operation modes of the 8051's timers/counters.
- Configure and use the timers/counters in the 8051.
- Measure frequency using the counters in the 8051.
- Measure period using a timer in the 8051.
- Measure pulse width using the gate control.
- Use a timer/counter to measure a physical quantity

Timing & machine cycles

- For example a 40µs delay wit a 24MHz clock and 12 clock periods per cycle:

```
mydelay:
    mov R0, #37
L1:
    djnz R0, L1
    ret
```

lcall: two machine cycles.

The ***djnz*** instruction takes two cycles or 24 clocks: 24/24MHz=1µs, for a total of 37 µs.

Two machine cycles each.

- For many applications this is ok, but it has disadvantages:
  1. It keeps the MCU busy just wasting time.
  2. Correct time is tedious to achieve, especially if interrupts are used.

Timing & machine cycles

- For the original 8051 one machine cycle takes 12 oscillator periods. For newer parts the machine cycle could be six, four, or even one oscillator period.
- For the CV-8052, one machine cycle takes 1 oscillator period. Since the clock is set to 33.33 MHz: One cycle takes 30 ns.
- A better solution is to use dedicated hardware for timing and counting: Timers and Counters!

Timers/Counters

- Timers/Counters have some advantages over timing loops:
  - The processor is not tied while counting.
  - Combined with interrupts, produces very efficient (small and fast) code.
  - They are usually independent on how many clocks per cycle the MCU takes.
  - Many timers/counters can be set to work concurrently.
  - Timers/counters work similarly in many different microprocessor architectures. Some times they are identical!

8051's Timers/Counters

- The original 8051 has only two timers/counters: 0 and 1.
- Newer 8051 microcontrollers usually have:
  1. The 8051 timers/counters: timers 0 and 1
  2. The 8052 timer/counter: timer 2
  3. Additional timers (3, 4, 5, etc.) Not available in the CV-8052.
  4. The Programmable Counter Array (PCA). Not available in the CV-8052.
- Let us begin with timers 0 and 1:

Timer 0 and Timer 1 Operation Modes

- Timer 0 and 1 have four modes of operation:
  - Mode 0: 13-bit timer/counter (compatible with the 8048 microcontroller, the predecessor of the 8051). <u>Do not use this mode; use mode 1 instead!</u>
  - Mode 1: 16-bit timer/counter.
  - Mode 2: 8-bit auto reload timer counter.
  - Mode 3: Special mode 8-bit timer/counter (timer 0 only). (I have never used it!)
- Timer 1 can be used as baud rate generator for the serial port. Some 8051/8052 microcontrollers have a dedicated baud rate generator.

TMOD timer/counter mode control register (Address 89H)

Is this SFR bit addressable?

Timer 1				Timer 0			
GATE	**C/T***	**M1**	**M0**	**GATE**	**C/T***	**M1**	**M0**

Bit	**Name**		**Description**
7 & 3	GATE		1: uses either INT0 or INT1 pins to enable/disable the timer/counter
6 & 2	C/T*		0: timer; 1: counter (pins T0 and T1)
All the other pins!	M1	M0	
	0	0	13-bit timer/counter
	0	1	16-bit timer/counter
	1	0	8-bit auto-reload timer/counter
	1	1	Special mode

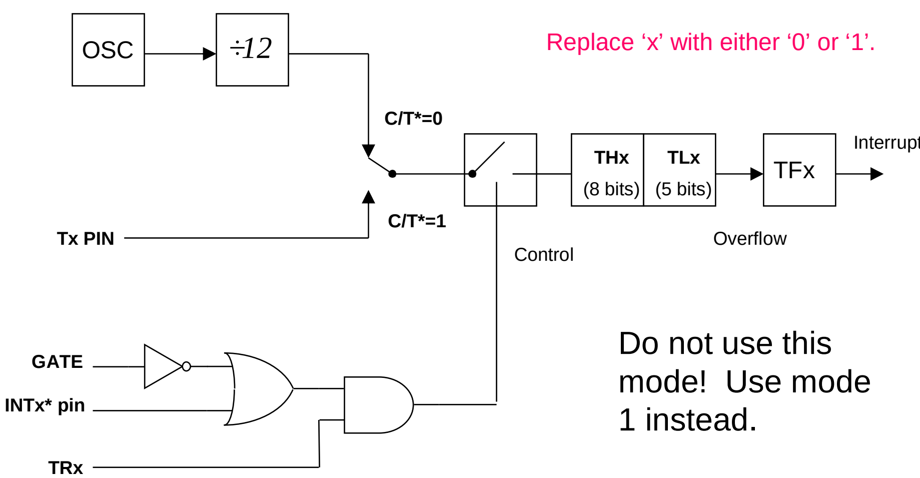
TCON: timer/counter control register. (Address 88H)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit	Name	Description
7	TF1	Timer 1 overflow flag.
6	TR1	Timer 1 run control.
5	TF0	Timer 0 overflow flag.
4	TR0	Timer 0 run control.
3	IE1	Interrupt 1 flag.
2	IT1	Interrupt 1 type control bit.
1	IE0	Interrupt 0 flag.
0	IT0	Interrupt 0 type control bit.

Timer/Counter 0 or 1 in Mode 0

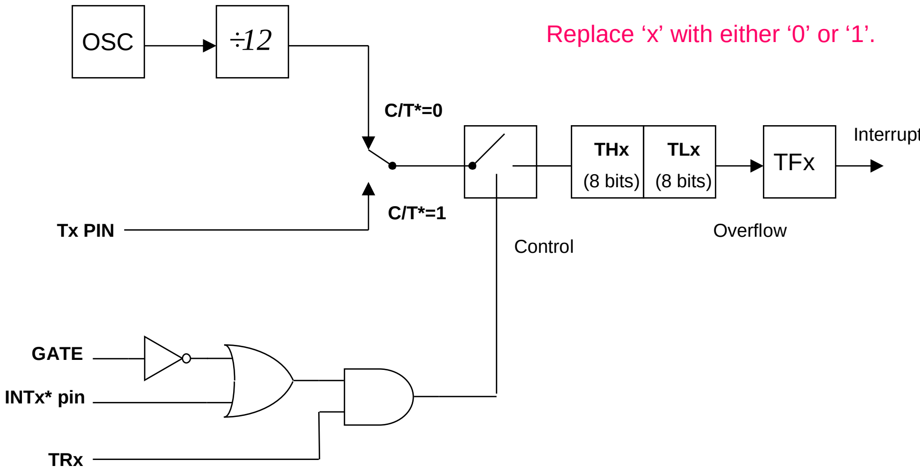
Replace ‘x’ with either ‘0’ or ‘1’.



Do not use this mode! Use mode 1 instead.

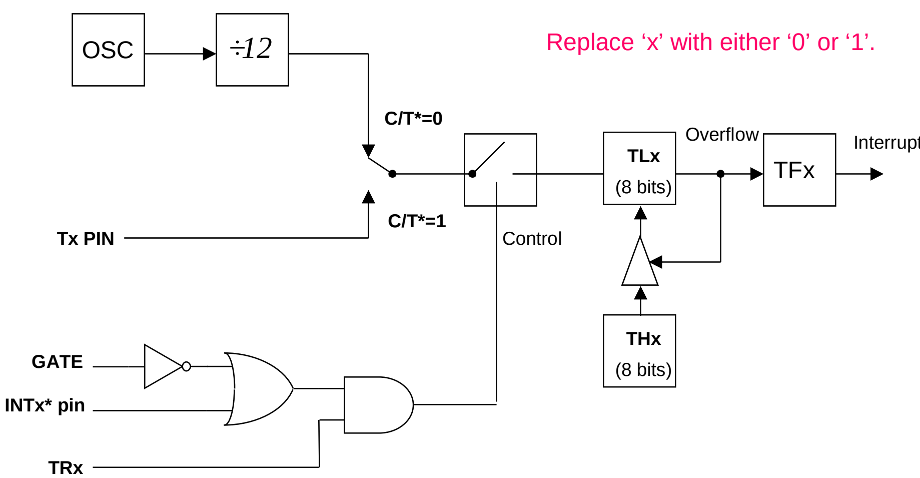
Timer/Counter 0 or 1 in Mode 1

Replace ‘x’ with either ‘0’ or ‘1’.



Timer/Counter 0 or 1 in Mode 2

Replace ‘x’ with either ‘0’ or ‘1’.



Timer/Counter 0 in Mode 2

```
myprogram:
    ; After reset, the stack pointer register is set to 07h
    ; We may need space for variables, so move the SP
    mov SP, #7fH
    ; Enable timer 0
    mov a, TMOD
    anl a, #0f0H
    orl a, #00000010B ; GATE=0, C/T*=0, M1=1, M0=0: 8-bit auto reload timer
    mov TMOD, a
    mov TH0, #080H ; Set the interrupt rate
    setb TR0 ; Enable timer 0
    setb ET0 ; Enable timer 0 interrupt (future lecture!)
    setb EA
Blink:
    cpl P1.0
    mov R0, #200
L0: djnz R0, L1
    jmp Blink
L1: mov R1, #200
L2: djnz R1, L2
    jmp L0
```

$$Rate = \frac{12}{OSC} \times (100H - TH0)$$

$$Rate = \frac{12}{11.1111MHz} \times (100H - 80H) = 138.2\mu s$$

Timer/Counter 2

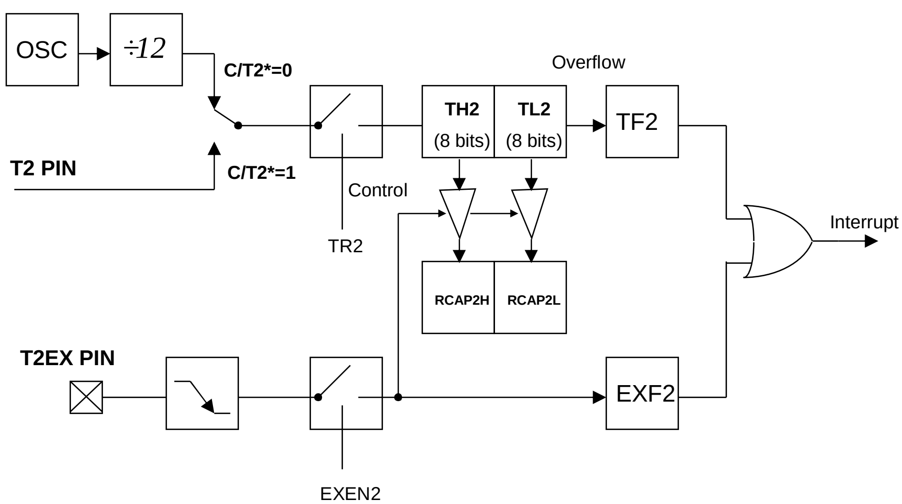
- It is a 16-bit timer/counter.
- It has four modes of operation:
  - Capture
  - Auto-reload
  - Baud rate generation
  - Programmable clock out (not implemented in CV-8052)

T2CON: timer/counter 2 control register. (Address C8H)

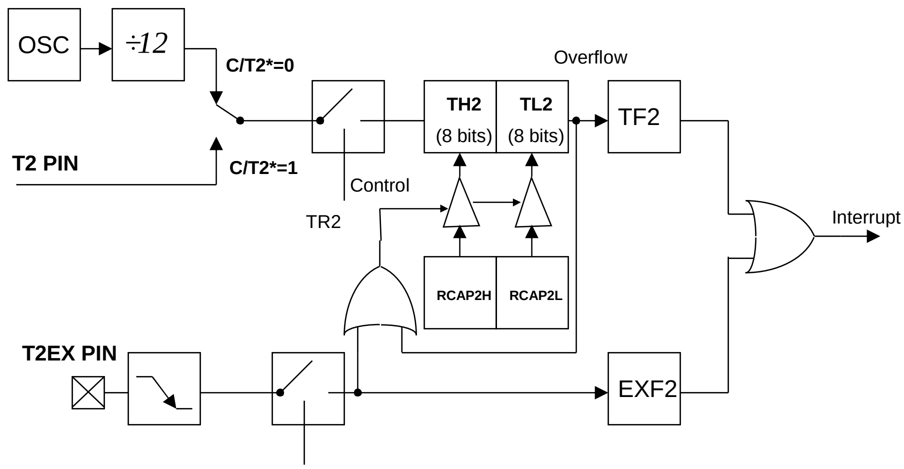
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2*	CP/RL2*

Bit	Name	Description
7	TF2	Timer/counter 2 overflow flag.
6	EXF2	Timer/counter 2 external flag.
5	RCLK	Receive clock flag.
4	TCLK	Transmit clock flag.
3	EXEN2	Timer/Counter 2 external enable.
2	TR2	Start/stop for timer/counter 2.
1	C/T2*	Timer or Counter select.
0	CP/RL2*	Capture/Reload Flag.

Timer/Counter 2 in capture mode



Timer/Counter 2 in auto-reload mode



Example: Time Delay Using a Timer

- To use a timer to implement a delay we need to:
  - Initialize the timer: use TMOD SFR.
  - Load the timer: use THx and TLx.
  - Clear the timer overflow flag: TFx=0;
  - Start the timer: Use TRx.
  - Check the timer overflow flag: Use TFx.

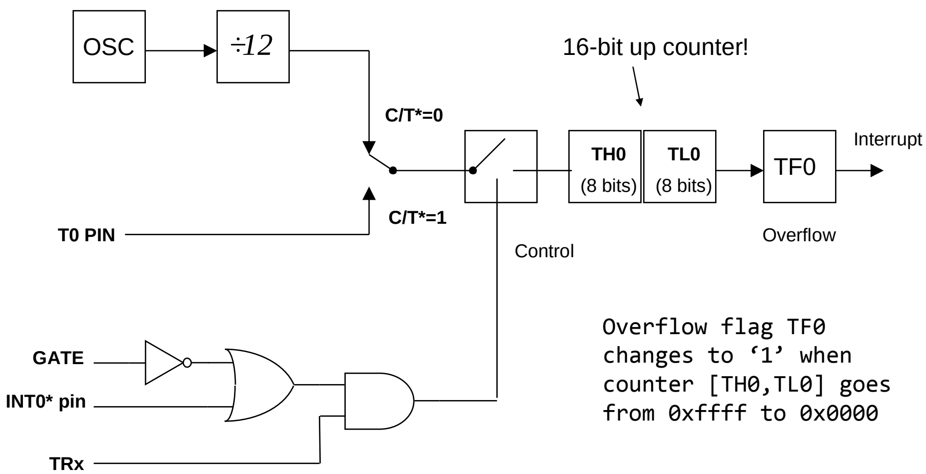
For the registers above ‘x’ is either ‘0’ for timer 0, or ‘1’ for timer 1.

Time Delay Using a Timer

- Implement a 10 ms delay subroutine using timer 0. Assume the routine will be running in a CV-8052 soft processor.

First, we have to find the divider (TH0, TL0) needed for a 10 ms delay…

Timer 0 in Mode 1



Overflow flag TF0 changes to '1' when counter [TH0,TL0] goes from 0xffff to 0x0000

Calculating TH0 and TL0

$$\text{Rate}=\frac{\text{CLK}/12}{2^{16}-[\text{THn,TLn}]}=\frac{33.3333\text{MHz}/12}{65536-[\text{THn,TLn}]}$$

$$[\text{THn,TLn}]=65536-\frac{2.77777\text{MHz}}{\text{Rate}}=65536-\frac{2.77777\text{MHz}}{(1/10\text{ms})}=27778$$

Maximum delay achievable?

$$\text{Rate}=\frac{\text{CLK}/12}{2^{16}-[\text{THn,TLn}]}=\frac{2.777777\text{MHz}}{65536-[\text{THn,TLn}]}$$

$$[\text{THn,TLn}]=0$$

$$\text{Rate}=\frac{2.777777\text{MHz}}{65536}=42.39Hz\rightarrow 23.59ms$$

Time Delay Using Timer 0

```
Wait10ms:
    ; Initialize the timer
    mov a, TMOD
    anl a, #11110000B ; Clear bits for timer 0, keep bits for timer 1
    orl a, #00000001B ; GATE=0, C/T*=0, M1=0, M0=1: 16-bit timer
    mov TMOD, a
    clr TR0 ; Disable timer 0
    ; Load the timer [TH0, TL0]=65536-(2777777/(1/10E-3))
    mov TH0, #high(27778)
    mov TL0, #low(27778)
    clr TF0 ;Clear the timer flag
    setb TR0 ; Enable timer 0
Wait10ms_L0:
    jnb TF0, Wait10ms_L0 ; Wait for overflow
    ret
```

Time Delay Using Timer 0

```
; Let the Assembler do the calculation for us!
XTAL equ 33333333
FREQ equ 100 ; 1/100Hz=10ms
RELOAD_TIMER0_10ms equ 65536-(XTAL/(12*FREQ))

Wait10ms:
    ; Initialize the timer
    mov a, TMOD
    anl a, #11110000B ; Clear bits for timer 0, keep bits for timer 1
    orl a, #00000001B ; GATE=0, C/T*=0, M1=0, M0=1: 16-bit timer
    mov TMOD, a
    clr TR0 ; Disable timer 0
    mov TH0, #high(RELOAD_TIMER0_10ms )
    mov TL0, #low(RELOAD_TIMER0_10ms )
    clr TF0 ;Clear the timer flag
    setb TR0 ; Enable timer 0
Wait10ms_L0:
    jnb TF0, Wait10ms_L0 ; Wait for overflow
    ret
```

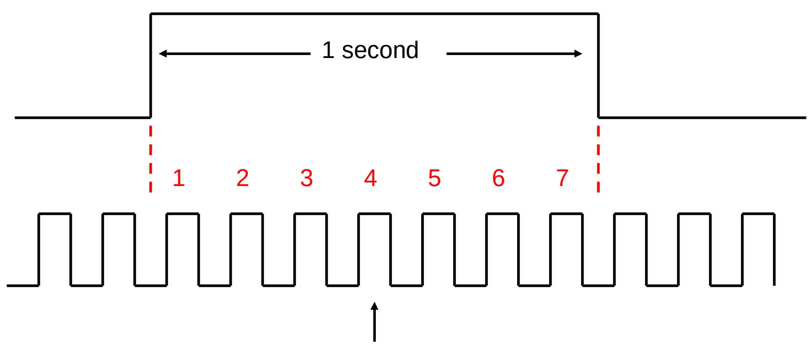

Timer/Counter Applications

- Measure Frequency (need for lab 5!)
- Measure Period
- Measure Pulse Width
- Measure a physical quantity. Time permitting!

Using a Counter to Measure Frequency

- By definition “frequency” in Hz is the number of pulses in one second, so:
  1) Set up the counter to count pulses in one of the pins in the microcontroller.
  2) Reset the counter to zero.
  3) Enable the counter.
  4) Wait one second. Use delay loops or another timer.
  5) Disable the counter. The counter (THx, TLx) has the frequency in Hz!

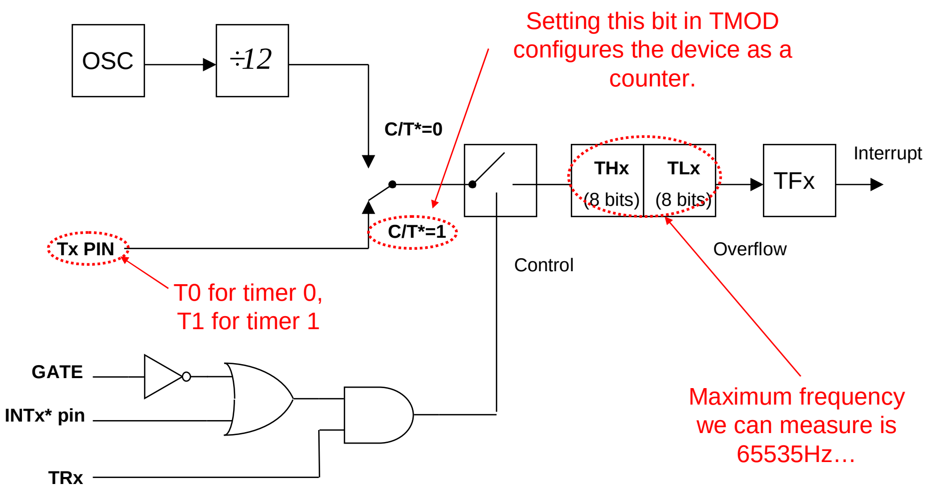
Using a Counter to Measure Frequency



Quick: what is the frequency of this signal?

About 7 Hz!

Using a Counter to Measure Frequency



CV-8052 Pin Assignments

	JP1		
LCD_DATA[0]	1	2	*LCD_DATA[1]*
LCD_DATA[2]	3	4	*LCD_DATA[3]*
LCD_DATA[4]	5	6	*LCD_DATA[5]*
LCD_DATA[6]	7	8	*LCD_DATA[7]*
LCD_EN	9	10	*LCD_RS*
5V	11	12	*GND*
LCD_RW	13	14	*TXD*
LCD_ON	15	16	*RXD*
FL_DQ[0]	17	18	*FL_DQ[1]*
FL_DQ[2]	19	20	*FL_DQ[3]*
FL_DQ[4]	21	22	*FL_DQ[5]*
FL_DQ[6]	23	24	*FL_DQ[7]*
FL_RST_N	25	26	*FL_WE_N*
FL_OE_N	27	28	*FL_CE_N*
3.3V	29	30	*GND*
TDO	31	32	*TDI*
TCS	33	34	*TCK*
Not used	35	36	*Not used*
T0	37	38	***T1***
T2	39	40	*T2EX*

	JP2		
P0.0	1	2	*P0.1*
P0.2	3	4	*P0.3*
P0.4	5	6	*P0.5*
P0.6	7	8	*P0.7*
P1.0	9	10	*P1.1*
5V	11	12	*GND*
P1.2	13	14	*P1.3*
P1.4	15	16	*P1.5*
P1.6	17	18	*P1.7*
P2.0	19	20	*P2.1*
P2.2	21	22	*P2.3*
P2.4	23	24	*P2.5*
P2.6	25	26	*P2.7*
P3.0	27	28	*P3.1*
3.3V	29	30	*GND*
P3.2	31	32	*P3.3*
P3.4	33	34	*P3.5*
P3.6	35	36	*P3.7*
INT0	37	38	*INT1*
Not Used	39	40	*Not used*

Using a Counter to Measure Frequency

```
; On the CV-8052, with a 33.33MHz clock, one cycle takes 30ns
Wait1s:
    mov R2, #180
L3: mov R1, #250
L2: mov R0, #250
L1: djnz R0, L1 ; 3 machine cycles-> 3*30ns*250=22.5us
    djnz R1, L2 ; 22.5us*250=5.625ms
    djnz R2, L3 ; 5.625ms*180=1s (approximately)
    ret

;Initializes timer/counter 0 as a 16-bit counter
InitTimer0:
    clr TR0 ; Stop timer 0
    mov a, #11110000B ; Clear the bits of timer 0
    anl a,TMOD
    orl a, #00000101B ; Set timer 0 as 16-bit counter
    mov TMOD, a
    ret
```


Using a Counter to Measure Frequency

```
; Configure T0 as an input (original 8051 only).
; Not needed but harmless in the CV-8052
setb T0
; 1) Set up the counter to count pulses from T0
lcall InitTimer0
; Stop counter 0
clr TR0
; 2) Reset the counter
mov TL0, #0
mov TH0, #0
; 3) Start counting
setb TR0
; 4) Wait one second
lcall Wait1s
; 5) Stop counter 0, TH0-TL0 has the frequency!
clr TR0
; Do something useful with the frequency!
```

Using a Counter to Measure Frequency

- There is a 65535 Hz limit, but it can be easily solved:
  - Check the counter overflow flag (TF0) and increment a register as it changes.
  - Use a shorter time window, for example 100 ms. You loose significant digits by doing this!
- Carefully calibrating the delay subroutine and synchronizing the input frequency with the beginning of the delay will result in more accurate measurements.
- Warning: Microcontrollers with built in oscillators are a lot less accurate than microcontrollers with external crystals. Typically 1% versus 0.01%!

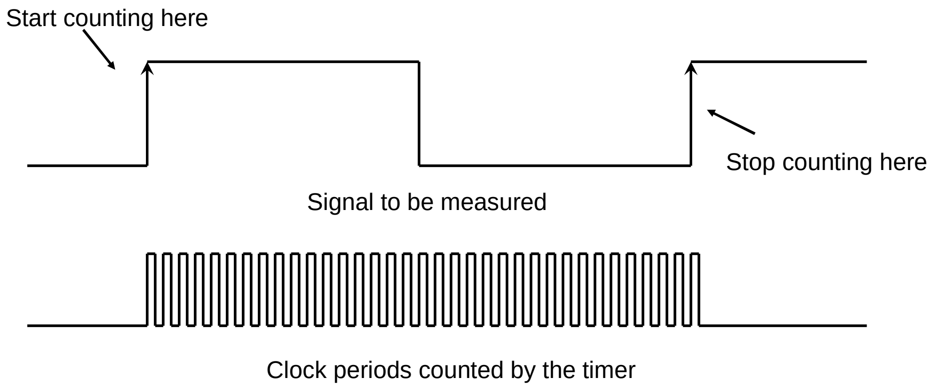
Measure Period Using a Timer in the 8051

- We can measure the period of a wave in integer numbers of the minimum timer clock period. Some math may be required!
- Works quite well for slow frequencies.
- We can also measure the period of a signal very easily using a plain software counter! Of course, some calibration and care for execution time is required.
- Measuring period could be way faster than measuring frequency.

Measure Period Using a Timer in the 8051

- To measure period we have to:
  1) Set up the timer.
  2) Connect the signal to be measured to <u>any</u> available pin. Also, set the pin as input.
  3) Reset the timer to zero.
  4) Wait for the input signal to transition from zero to one.
  5) Start the timer.
  6) Wait for the input signal to transition from zero to one.
  7) Stop the timer! The timer SFRs (THx, TLx) have the period in timer-input-period units!

Measure Period Using a Timer in the 8051



Signal period is about 35 timer periods. If using a CV-8052, the period of the signal would be approximately T=(35*12/33.33MHz)=12.6µs

Using a timer to Measure Period

```
InitTimer0:
   clr TR0 ; Stop timer 0
   mov TMOD, #11110000B
   ;Set timer 0 as 16-bit timer
   orl TMOD, #00000001B
   mov TMOD, a
   ret
```

```
    lcall InitTimer0
    setb P1.7 ; Used as input
; Stop timer 0 (just in case!)
    clr TR0
; Reset the timer
    mov TL0, #0
    mov TH0, #0
; Wait for the signal to be zero
W1: jb P1.7, W1
; Wait for the signal to be one
W2: jnb P1.7, W2
    setb TR0 ; Start timing
; Wait for the signal to be zero
W3: jb P1.7, W3
; Wait for the signal to be one
W4: jnb P1.7, W4
    clr TR0 ; Stop timer 0
;TH0-TL0 has the period!
```

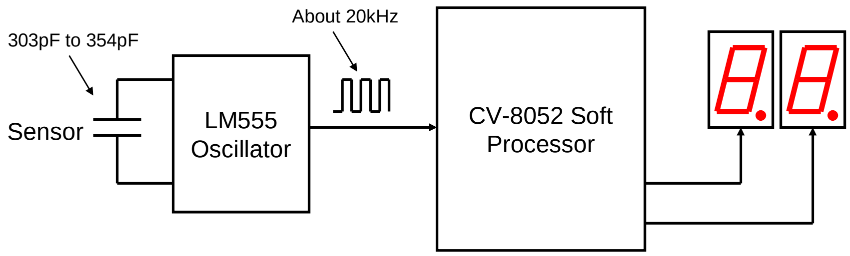

Example: Measuring and Displaying Relative Humidity

- For this example I'll be using the **HCH-1000** Capacitive Humidity Sensor from Honeywell. (DigiKey part number 480-2903-ND)

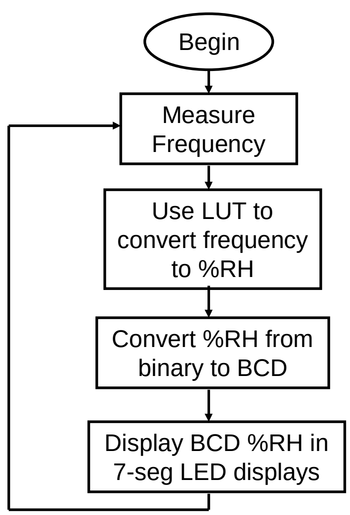
Characteristic @ 20kHz	Value	Unit	Note
Typical Normal Capacitance	330	pF	at 55% RH
Typical Sensitivity	0.6	pF/%RH	10% RH to 95% RH

A LM555 timer will convert the capacitance to frequency. See block diagram in next slide:

Example: Hardware Block Diagram



Example: Software Block Diagram



Initialize stack, ports, timer, etc.

Use the method described in this lecture.

The most tedious part is to create the LUT! We need to carefully design the LM555 oscillator and calculate the LUT.

This was covered in previous lecture. BCD numbers are easy to display.

This was covered in previous lecture also!

Example: LM555 design.

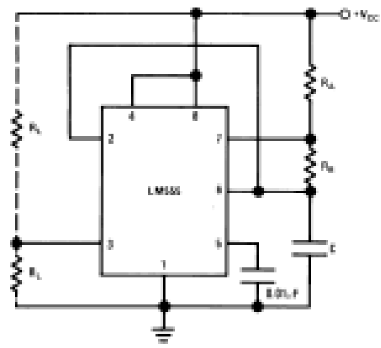


Figure taken from LM555 datasheet from National Semiconductors.

Astable Oscillator

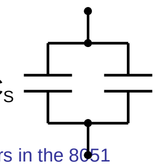
The frequency is given by:

$$f = \frac{1.44}{(R_A + 2R_B) \times C}$$

Also, from the datasheet:

$$C \geq 1000\,pF$$

Therefore I will connect the humidity sensor in parallel with a fixed capacitor:



$$C = C_S + C_F$$

Example: LM555 design.

$$C = C_S + C_F = 330\,pF + 680\,pF = 1010\,pF$$

$$f = \frac{1.44}{(R_A + 2R_B) \times C} = \frac{1.44}{(R_A + 2R_B) \times 1010\,pF} = 20kHz$$

$$(R_A + 2R_B) = \frac{1.44}{20kHz \times 1010\,pF} = 71287\Omega$$

For an almost square wave: $R_B >> R_A$

$R_B = 33k\Omega + 2k\Omega$

$$\boxed{R_A = 1k\Omega,\ R_B = 35k\Omega}$$

Example: LUT generation

RH%	Cs(pF)	Frequency
0	297.00	20759
1	297.60	20746
2	298.20	20734
3	298.80	20721
	.	
	.	
54	329.40	20093
55	330.00	20081
56	330.60	20069
	.	
	.	
96	354.60	19603
97	355.20	19592
98	355.80	19581
99	356.40	19569

Created with MS Excel

$$f = \frac{1.44}{(R_A + 2R_B)\times C}$$

$$C = C_S + 680\,pF$$

$$R_A = 1k\Omega$$

$$R_B = 35k\Omega$$

Example: Code

```
$MODCV
org 0000H
   ljmp MyProgram

;-----------------------------
; Look-up tables
;-----------------------------

Dec7Seg:
    DB 0C0H, 0F9H, 0A4H, 0B0H, 099H
    DB 092H, 082H, 0F8H, 080H, 090H
    DB 10111111B ; '-' to indicate an error
```

Example: Code

```
Freq2RH:
    DW 20759, 20746, 20734, 20721, 20708
    DW 20696, 20683, 20670, 20658, 20645
    DW 20632, 20620, 20607, 20595, 20582
    DW 20570, 20557, 20545, 20532, 20520
    DW 20507, 20495, 20482, 20470, 20458
    DW 20445, 20433, 20421, 20408, 20396
    DW 20384, 20371, 20359, 20347, 20335
    DW 20322, 20310, 20298, 20286, 20274
    DW 20261, 20249, 20237, 20225, 20213
    DW 20201, 20189, 20177, 20165, 20153
    DW 20141, 20129, 20117, 20105, 20093
    DW 20081, 20069, 20057, 20045, 20033
    DW 20021, 20010, 19998, 19986, 19974
    DW 19962, 19951, 19939, 19927, 19915
    DW 19904, 19892, 19880, 19868, 19857
    DW 19845, 19833, 19822, 19810, 19799
    DW 19787, 19775, 19764, 19752, 19741
    DW 19729, 19718, 19706, 19695, 19683
    DW 19672, 19660, 19649, 19638, 19626
    DW 19615, 19603, 19592, 19581, 19569
```


Example: Code

```
;Converts the hex number in R0 to BCD in R1
;Changes R0 to R2 and the accumulator.
hex2bcd:
    mov R1, #0   ; Set BCD result to 00
    mov R2, #8   ; Loop counter.
hex2bcd_L0:
    mov a, R0    ; Shift R0 left through carry
    rlc a
    mov R0, a
    mov a, R1    ; Do (2*BCD)+carry=BCD+BCD+carry
    addc a, R1
    da a
    mov R1, a
    djnz R2, hex2bcd_L0
    ret
```

Example: Code

```
; On the CV-8052, with a 11.11MHz clock, one cycle takes 90ns
Wait1s:
    mov R2, #60
L3: mov R1, #250
L2: mov R0, #250
L1: djnz R0, L1 ; 3 machine cycles-> 3*90ns*250=67.5us
    djnz R1, L2 ; 67.5us*250=0.0169s
    djnz R2, L3 ; 0.0169s*60=1s (approximately)
    ret

;Initializes timer/counter 0 as a 16-bit counter
InitTimer0:
    clr TR0 ; Stop timer 0
    mov a, #0F0H
    anl a, TMOD
    orl a, #05H ; 16-bit counter
    mov TMOD, a
    ret
```


Example: Code

```
; Display the BCD number stored in R1
DisplayBCD:
    mov dptr, #Dec7Seg
    mov A, R1 ; Display MSB
    swap A
    anl A, #0FH
    movc A, @A+dptr
    mov HEX1, A
    mov A, R1 ; Display LSB
    anl A, #0FH
    movc A, @A+dptr
    mov HEX0, A
    ret
```

Example: Code

```
MyProgram:
    ; Initialize the hardware:
    mov SP, #7FH
    lcall InitTimer0

forever:
    ; Measure the frequency applied to pin T0
    clr TR0 ; Stop counter 0
    mov TL0, #0
    mov TH0, #0
    setb TR0 ; Start counter 0
    lcall Wait1s ; Wait one second
    clr TR0 ; Stop counter 0, TH0-TL0 has the frequency
```


Example: Code

```
    ; Scan the LUT to find the best RH% match.  This is
    ; not using a binary search.  That is left for you to
    ; do as an exercise.
    mov dptr, #Freq2RH
    mov R0, #0 ; Current entry in the LUT...
X0:
    ; Get the MSB
    clr a
    movc a, @a+dptr
    mov R6, a
    ; Get the LSB
    inc dptr
    clr a
    movc a, @a+dptr
    mov R7, a
    inc dptr
```

Example: Code

```
; Compare the frequency in TH0-TL0 with the table value
    clr c
    mov a, R7
    subb a, TL0
    mov a, R6
    subb a, TH0
    jc Done
    inc R0
    cjne R0, #100, X0 ; Have we checked the whole table?
    mov R1, #0AAH ; Not found, display "--"
    sjmp Display
Done:
    lcall hex2bcd
Display:
    lcall DisplayBCD
    ; Repeat!
    sjmp forever
END
```


Exercises

- Write a one second delay function using timer 1. This function will run in a standard 8051 microcontroller with a 11.11MHz clock (12 clocks per cycle). What changes need to be made to run the function in a CV-8052 with a 33.33MHz clock (1 clock per cycle)?
- Program profiling is used to find the usage of resources by a piece of code (a subroutine, for example). A profile value often needed is execution time. Show how to use timer 0 to find out the execution time of a subroutine.

Exercises

- From the examples given in this lecture, explain how to use the timer overflow flag to measure frequencies higher than 65535 Hz while using a 1-second time interval. (Note: you'll need this for the lab assignment).
- Modify the code for the relative measurement example above to use a binary search to find the best RH% match to a frequency from the LUT. Do you need to make any changes to the LUT?