University of British Columbia
ELEC291/ELEC292

Lab 3: 555 timer and Capacitance Meter

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

January 28, 2022

Objectives

- Lab 3 requirements.
- The 555 timer.
- Read capacitors.
- Measure Frequency
- Measure Period
- 32-bit unsigned integer arithmetic.

Lab Requirements

- 555 Timer:
  a) Test the 555 timer as an A-stable oscillator.
  b) Use a 555 timer to measure capacitance:
     – Range: 1nF to 1µF. No decimal places required.
     – Display value using LCD.
     – Program AT89LP51RC2 using assembly language.

555 Timer astable (oscillator)

From National Semiconductors LM555 Timer datasheet

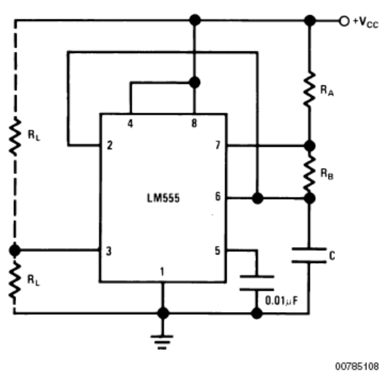


FIGURE 4. Astable

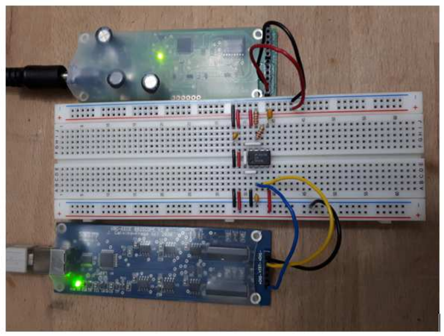
$$f = \frac{1}{T} = \frac{1.44}{(R_A + 2R_B)C}$$

Information in BB2Scope Manual

555 Timer A-stable Oscillator

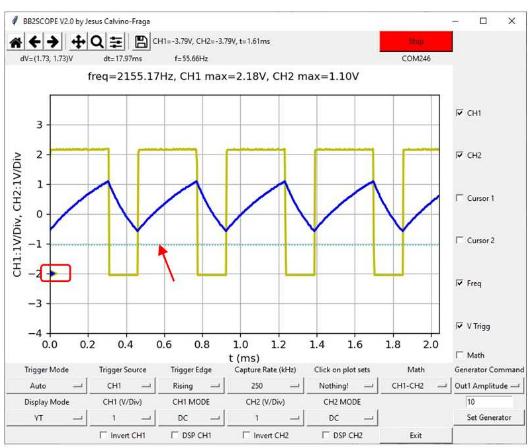
The picture of the bread-boarded circuit below shows a 555 timer configured as an A-stable oscillator (R1=R2=2.2kΩ; C=0.1uF). The power for the 555 timer IC (approximately 5V) is obtained from the Gen2BB board +5V output. The schematic diagram of this circuit can be found here:

https://ohmslawcalculator.com/555-astable-calculator

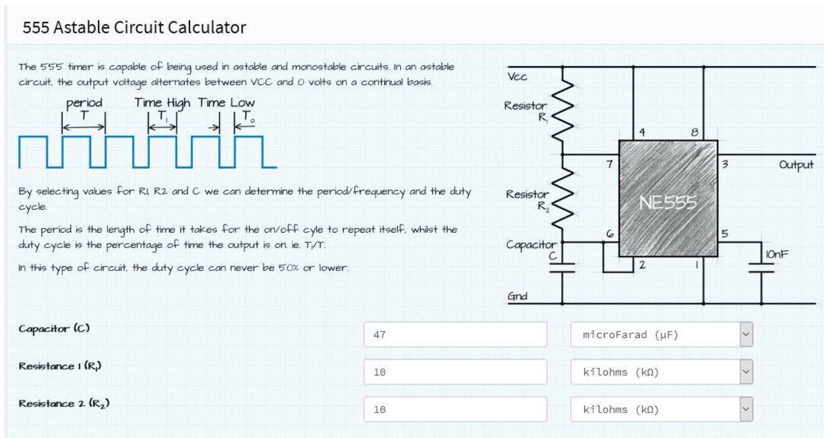


Information in BB2Scope Manual

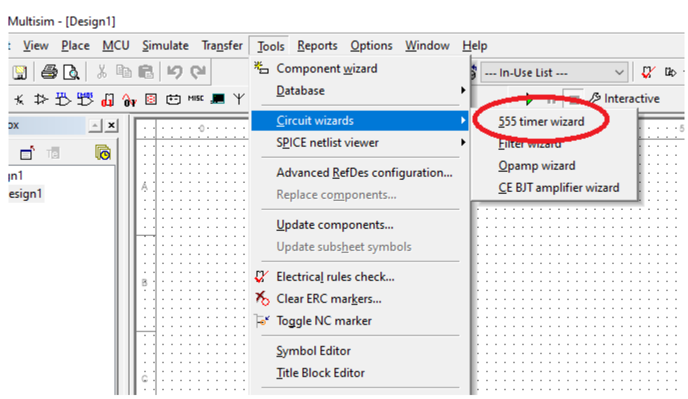
The figure below was obtained while the 555 timer A-stable oscillator was connected to BB2Scope board. CH1 shows the output of the timer (IC pin 3), while CH2 shows the voltage at the timing capacitor (IC pin 2).



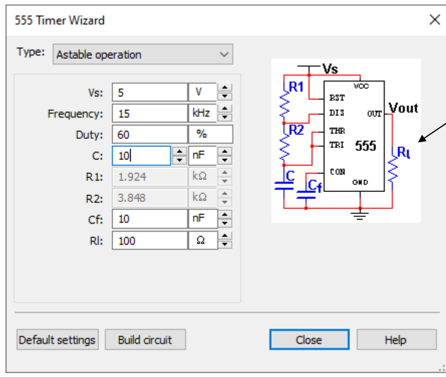
https://ohmslawcalculator.com/555-astable-calculator



Multisim 555 Timer Wizard



Multisim 555 Timer Wizard



Don't put R_L!

Capacitor Types

- Ceramic. Small values, small size, good price, good tolerance (lowest around ±1%)
- Electrolytic. Large value, high tolerance (±10% minimum), don't age well, big size, very temperature sensitive.
- Tantalum. Large value, low voltage, small size, expensive, lowest tolerance around ±5%.
- Mica. Best capacitors ever! Lowest tolerance around ±0.5%. Very small values. VERY expensive, around 4$ each!
- Polyester Film. Wide range values, inexpensive, good tolerance, price depends on tolerance and voltage rating
- Glass.

How to read Capacitor Codes

- Large capacitors have their values printed on them, for example 10µF, 50V, 85C.
- Most small capacitors use a three number code system:

1	0	4

10×10^4 pF

How to read Capacitor Codes

- Tolerance is indicated by a letter after the value:

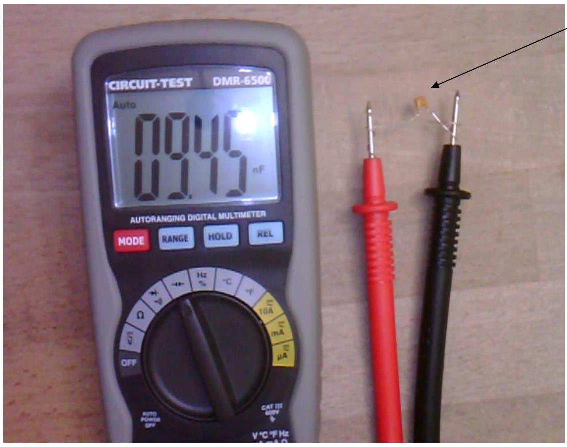
E	± 0.5%
F	± 1%
G	± 2%
H	± 3%
J	± 5%
K	± 10%
M	± 20%
N	± 30%
P	+100% ,-0%
Z	+80%, -20%

If tolerance is not indicated assume it is 'Z': +80%, -20%.

How to read Capacitor Codes

- Examples:
  - 103J
    - 10 x 10^3 ± 5% = 0.01μF ± 5%
  - 681
    - 68 x 10^1 +80%, -20% = 680pF +80%, -20%
  - 104Z
    - 10 x 10^4 +80%, -20% = 0.1μF +80%, -20%
  - 224M
    - 22 x 10^4 ± 20% = 0.22μF ± 20%
  - 473K
    - 47 x 10^3 ± 10% = 47nF ± 10%

Using the Kit Multimeter to Measure Capacitance

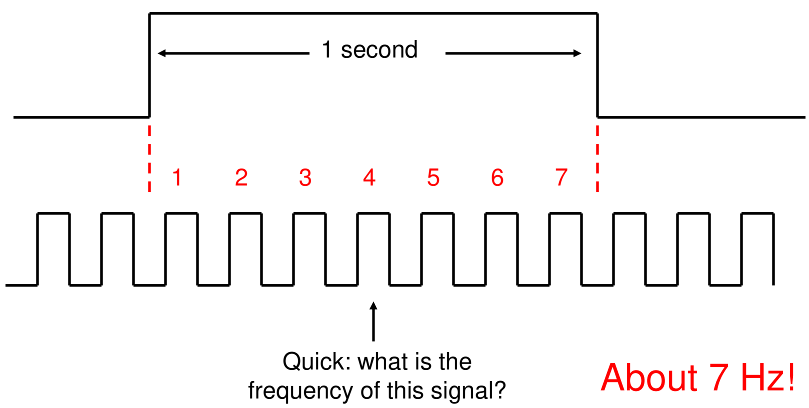


Marking says: 103

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.
  5) Disable the counter. The counter register (THx, TLx) has the frequency in Hz!

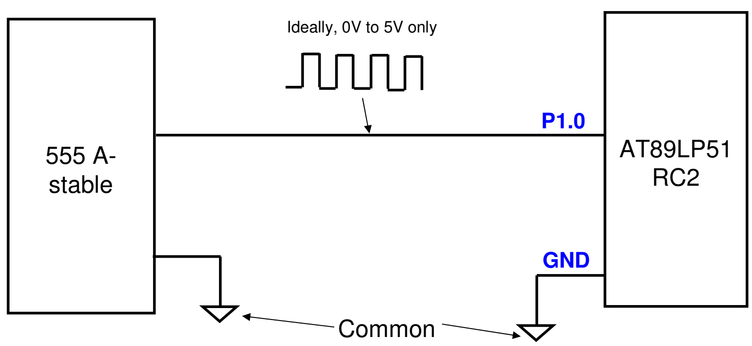
Using a Counter to Measure Frequency



Freq_RC2.asm

- Available on Canvas

Testing Freq_RC2.asm with the 555 timer



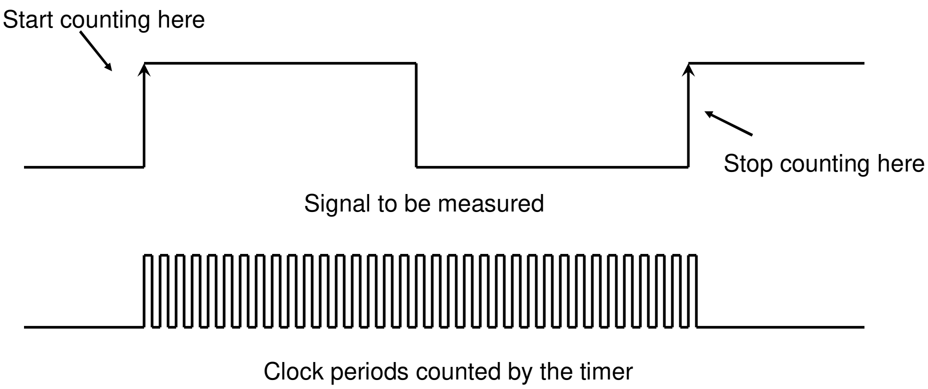
Measure Period Using a Timer in the 8051

- We can measure the period of a wave in integer numbers of the timer clock period. Some math may be required!
- Works quite well for slow signals.
- Measuring period is 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. For a AT89LP51RC2 (@22.1184MHz), the period of the signal would be approximately T=(35/22.1184E6)=1.582393µs

32-bit Arithmetic Library Math32

- ***Math32.asm*** has the following functions:
  - **Hex2bcd**: Converts the 32-bit binary number in 'x' to a 10-digit packed BCD in 'bcd' using the double-dabble algorithm. Largest BCD possible is 4,294,967,295.
  - **Bcd2hex**: Converts the 10-digit packed BCD in 'bcd' to a 32-bit binary number in 'x'. Largest hex number is `0xffffffff`.
  - **add32**: x = x + y
  - **sub32**: x = x − y
  - **mul32**: x = x * y
  - **div32**: x = x / y
  - **x_lt_y**: mf=1 if x < y (mf is a bit)
  - **x_gt_y**: mf=1 if x > y
  - **x_eq_y**: mf=1 if x = y
  - **x_gteq_y**: mf=1 if x >= y
  - **x_lteq_y**: mf=1 if x <= y
  - **xchg_xy**: exchange x and y
  - **copy_xy**: copy x to y
- ***Math32.asm*** has the following macros:
  - **Load_X or Load_Y**: load x or y with a 32-bit constant

Math32 Test Program

- ***Mathtest.asm*** shows how to:
  - Define the x, y, bcd, and mf variables.
  - Include the math32 library in your program.
  - Use the Load_X and Load_X macros.
  - Convert a binary to BCD using bin2bcd and display it using the LCD.
  - Use the add32, sub32, mul32, and div32 functions.
  - Evaluate a formula using only integers.

Warning: You are using integer arithmetic!

Formula we want to calculate

$$Vout = \frac{ADC}{\left(2^{10}-1\right)} \times V_{REF} = \frac{ADC}{1023} \times 410$$

Suppose ADC=612, compute Vout

$$Vout = \frac{612}{1023} \times 410$$

Wrong!

Vout=(612/1023) x 410 = (0) x 410 =0

$\boxed{\text{Vout=(612 x 410) / 1023 = (250920) /1023 = 245}}$ Right!

Final Remarks

- Lab 3 is an excellent starting point for project 1!
- Copy 16-bit timer count to 32-bit x:

```
mov x+0, TL2
mov x+1, TH2
mov x+2, #0 ; load high bits with zero
mov x+3, #0 ; load high bits with zero
```