ECE 376 - Homework #10

ITimer1 Capture - Timer1 Compare.

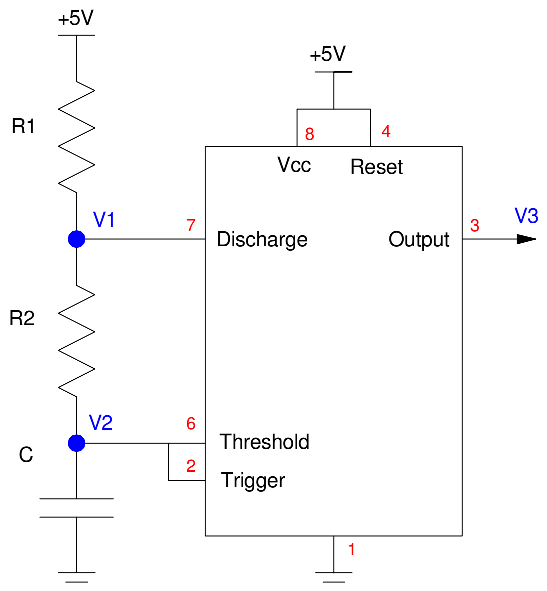
Timer1 Capture: Capacitor Meter

Problem 1-5) Use Timer1 Capture to measure time to 1 clock (100ns).

1) Requirements:

- Measure the period of a 555 timer with a resolution of 100ns (Timer1 Capture).
- From this, compute the value of C

Hardware: Output a square wave using a 555 timer



- $period = (R_1 + 2R_2) \cdot C \cdot \ln(2)$
- R1 = 1k
- R2 = 3.3k
- C = 1uF (varies)

2) C code and flow chart:

Computations

$$C = \left(\frac{T}{(R_1+2R_2)\ln(2)} \right) = 0.0001898T$$

With T measured to 100ns

$$N = 10^7 T$$

$$C = 18.98 \cdot 10^{-12} N \qquad \text{Farads}$$

$$C = 18.98N \qquad \text{pF}$$

If you capture every 256th rising edge

$$C = \left(\frac{18.98}{256} \right) N = 0.07379N \quad \text{pF}$$

C-Code and flow chart.

```
< insert code >

< one flow chart for the main routine, one for each interrupt >
```

3) Test: Collect data in lab to verify that your interrupts are working properly.

Toggle RA1 every Timer1 interrupt (2^16 clocks).

- Expected period = 2 * 65,536 = 131,072 clocks
- Measured period = 13.1063808ms = 131,063 clocks

Measure a 2ms square wave (555 timer with 0.36uF)

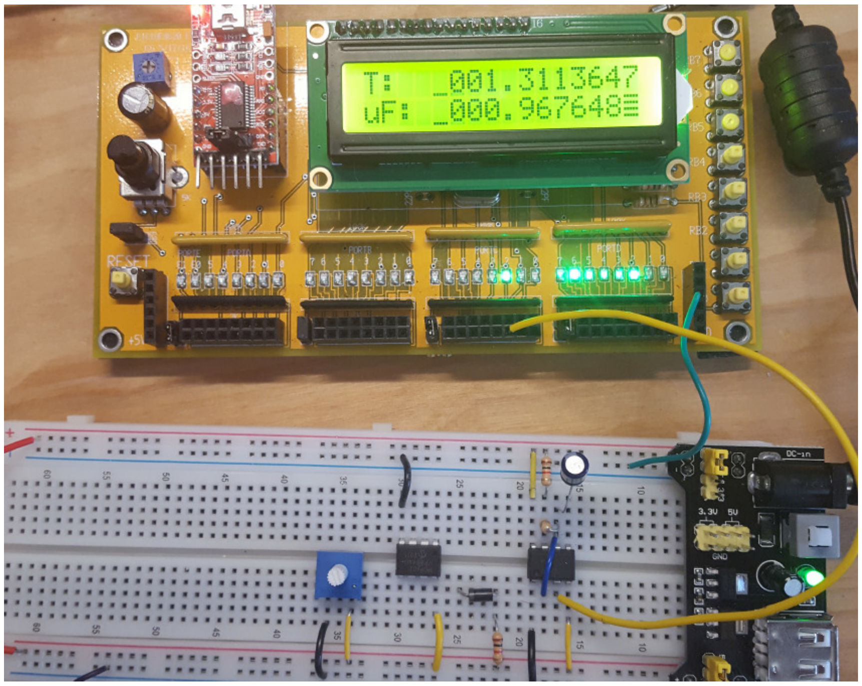
- Measured period = 1.7807872ms
- Calculated period = 1.8960*ms*

4) Validation: Collect data to validate your design works.

C	T (ms)	uF (meas)	C Lovum multimeter	Error
10uF	42.6246ms	8.036608 uF	10.20uF	-21.21 %
1uF	5.124096 ms	0.968528 uF	1.059 uF	-8.54%
0.18uF	0.8742656 ms	0.165136 uF	0.1785 uF	-7.49%
0.1uF	0.5922362 ms	0.112130 uF	0.1038 uF	+8.02%
0.015uF	0.0775216 ms	0.014638 uF	0.01530 uF	-5.33%

note: both readings might be correct. C is specified at 1kHz. Our meter uses 23Hz - 13kHz.

5) Demo



```
T:   _001.3113647
uF:  _000.967648
```

Timer1 Compare:

Can you detect a 1% change in frequency at 440Hz?

6) Requirements: Press RB0 to start.

- The PIC flips a coin (head or tails)
- The PIC will then play 440Hz for 500ms
- Then pause 100ms
- Then play either 440Hz or 444.44Hz for 500ms, depending upon the coin toss (random).

The operator then must press a button

- RB0 if the notes sound like they're the same
- RB1 if the notes sound like they're different

The PIC then records whether you were correct or not, displays the running total on the LCD, the repeats.

7) C-Code and flow chart.

```
< insert code here >

< one flow chart for the main routine and each interrupt >
```

Interrupt Service Routine

```
void interrupt IntServe(void)
{
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      if(PLAY) RC0 = !RC0;
      else RC0 = 0;
      CCPR1 += N;
      CCP1IF = 0;
      }
   }
```

8) Test: Collect data in lab to verify that your interrupts are working properly.

Test Code: Play 440.0Hz

```
while(1) {
   N = 11354; // 440Hz
   PLAY = 1;
}
```

Resulting frequency = 441.0Hz

Test Code: Play 444.44Hz

```
while(1) {
   N = 11251; // 444.44Hz
   PLAY = 1;
}
```

Resluting frequency = 445.0Hz

Test Code: Random number generator

```
while(1) {
   while(RB0);
   while(!RB0);
   DIE = TMR1 & 1;
   LCD_Move(0,0);  LCD_Write(DIE + 48);
}
```

Result

```
00100000111010110001010100010101111110100111110011101
```

- 25 0's
- 28 1's

9) Validation: Collect data to validate your design works.

- 18 tests
- Correct 15 times
- Incorrect 3 times

Guess	p	np	N	chi-squared
correct	0.5	9	15	4.00
incorrect	0.5	9	3	4.00
			Total	8.00

From StatTrek, a chi-scored critical value of 8.00 with 1 degree of freedom corresponds to a probability of 0.995

I can be 99.5% certain that I can hear a 1% difference in frequency at 440Hz (i.e. I'm not guessing)

10) Demo