Timer1 Compare Interrupts

NDSU ECE 376

Lecture #24

Inst: Jake Glower

Please visit Bison Academy for corresponding lecture notes, homework sets, and solutions

Timer1 Compare Mode:

The PIC we use is able to measure time to 100ns. If you want to drive an output pin high or low at a precise time (accurate to 100ns), Timer1 compare interrupts are used.

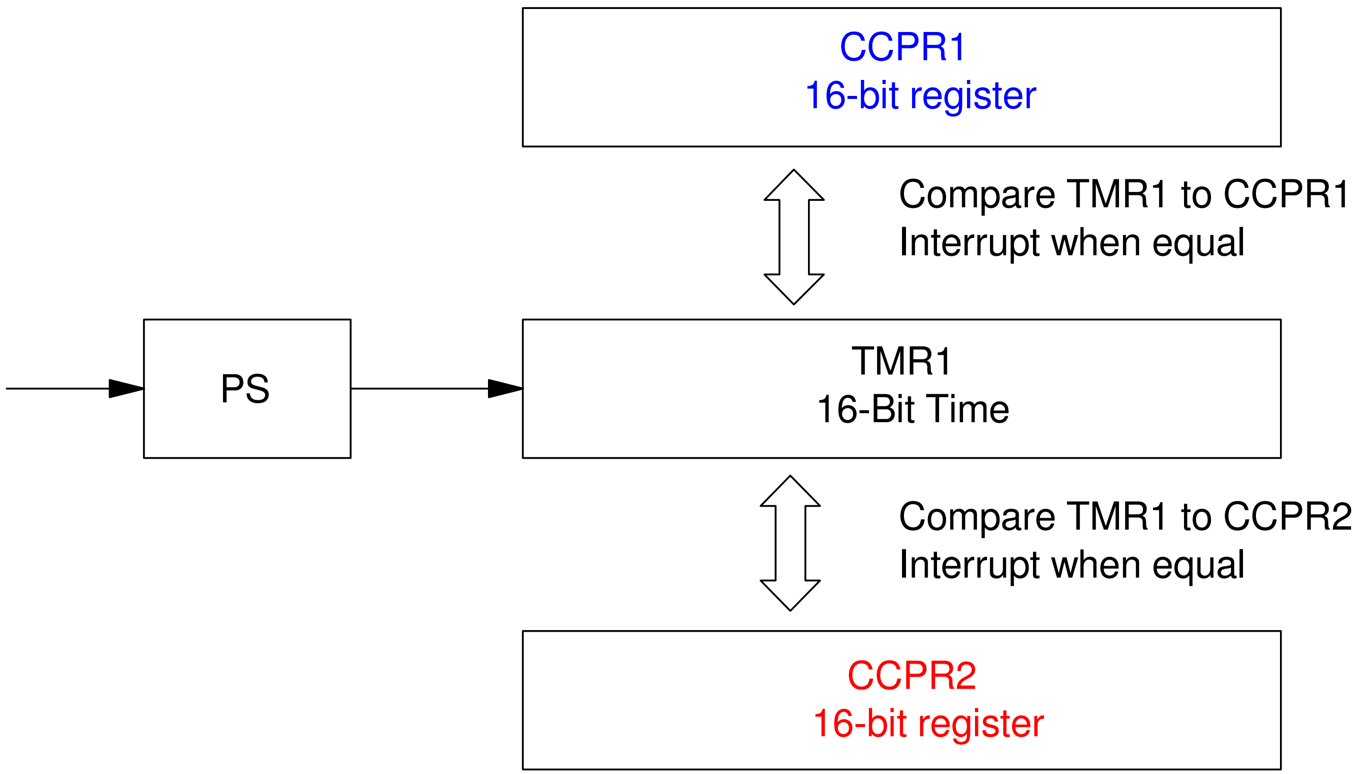
There are several reasons you might want to do this:

- To output a precise frequency
- To generate a pulse with a precise duration
- To output 0V and 5V using PWM

How Timer1 Compare Works

Timer1 runs in the background

- When TMR1 = CCPR1, a Capture1 interrupt is triggered
- When TMR1 = CCPR2 a Capture2 interrupt is triggered



Timer1 Compare

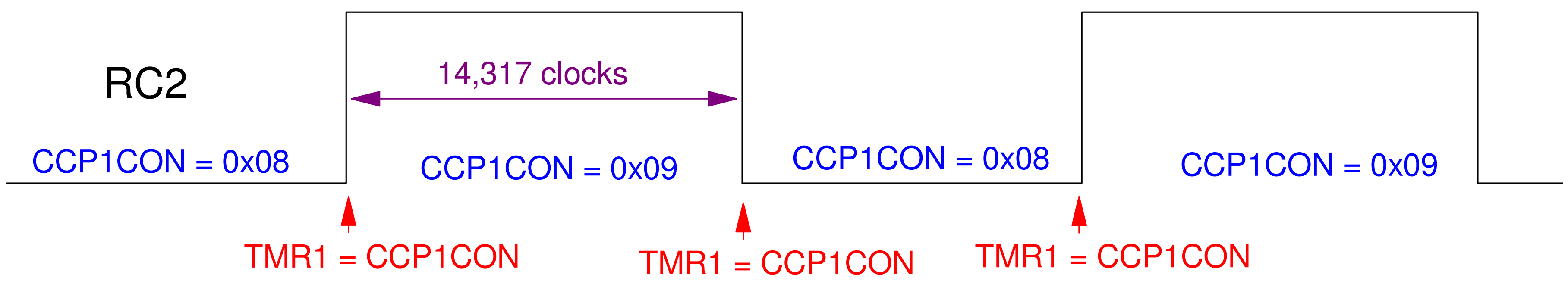
Interrupt	Description	Input	Output	Conditions	Enable	Flag
Timer 1	Trigger after N events N = 1 .. 2^{19} 100ns to 0.52 sec	RC0 TMR1CS = 1 OSC/4 TMR1CS = 0	none	N = (PS)(Y) T1CON = 0x81: PS = 1 T1CON = 0x91: PS = 2 T1CON = 0xA1: PS = 4 T1CON = 0xB1: PS = 8 TMR1 = -Y	TMR1ON = 1 TMR1IE = 1 TMR1IP = 1 PEIE = 1	TMR1IF
Timer 1 Compare Mode 1	Drive a pin high or low at a precise time Interrupt when TMR1 = CCPR1	OSC/4	RC2	Interrupt when CCPR1 = TMR1 CCP1CON = 0x08: Set RC2 CCP1CON = 0x09: Clear RC2 CCP1CON = 0x0A: no change	CCP1IE = 1 TMR1ON = 1 PEIE = 1	CCP1IF
Timer 1 Compare Mode 2	Drive a pin high or low at a precise time Interrupt when TMR1 = CCPR2	OSC/4	RC1	Interrupt when CCPR2 = TMR1 CCP2CON = 0x08: Set RC1 CCP2CON = 0x09: Clear RC1 CCP2CON = 0x0A: no change	CCP12E = 1 TMR1ON = 1 PEIE = 1	CCP2IF

Output a Precise Frequency: Capture1.C

Problem: Output the note F4 (349.228Hz) on pin RC2

Solution: Toggle RC2 every 14317 clocks (rounded down)

$$N = \left(\frac{10{,}000{,}000}{2 \cdot 349.228 Hz}\right) = 14317.29$$



Assume

- PS = 1 (Timer1 counts every 100ns)

Increment CCPR1 by 14317 every interrupt

CCPR1 += 14317

This sets up the next interrupt 14,317 clocks after the last interrupt

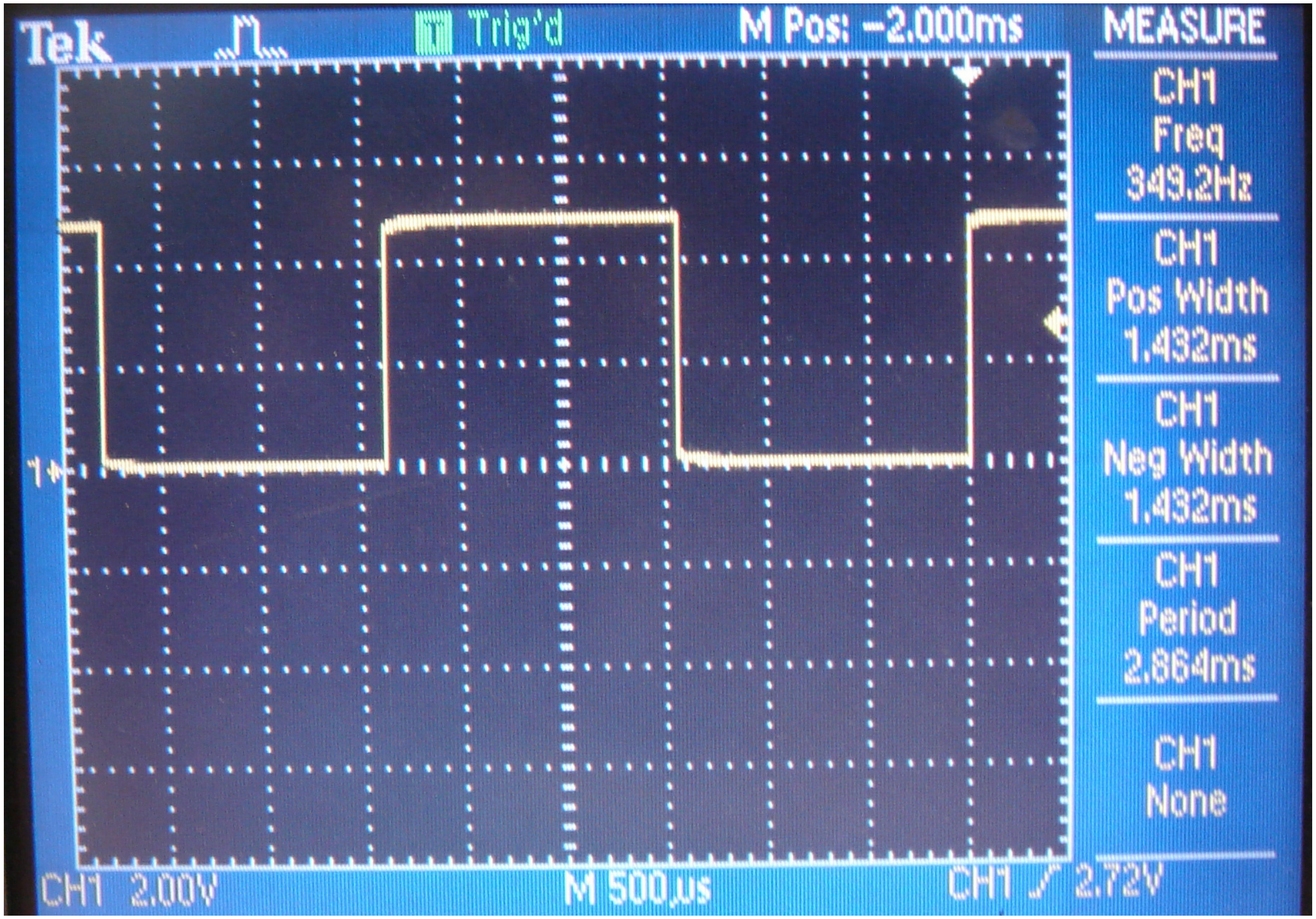
```
void interrupt IntServe(void)
{
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      CCP1CON = CCP1CON ^ 0x01;    // toggle between 0x08 & 0x09
      CCPR1 += 14317;
      CCP1IF = 0;
      }
   }
```

Another Option

- RC0 has a 50 clock delay in it's output
- Frequency is still correct

```
void interrupt IntServe(void)
{
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      RC0 = !RC0;
      CCPR1 += 14317;
      CCP1IF = 0;
      }
   }
```

Result



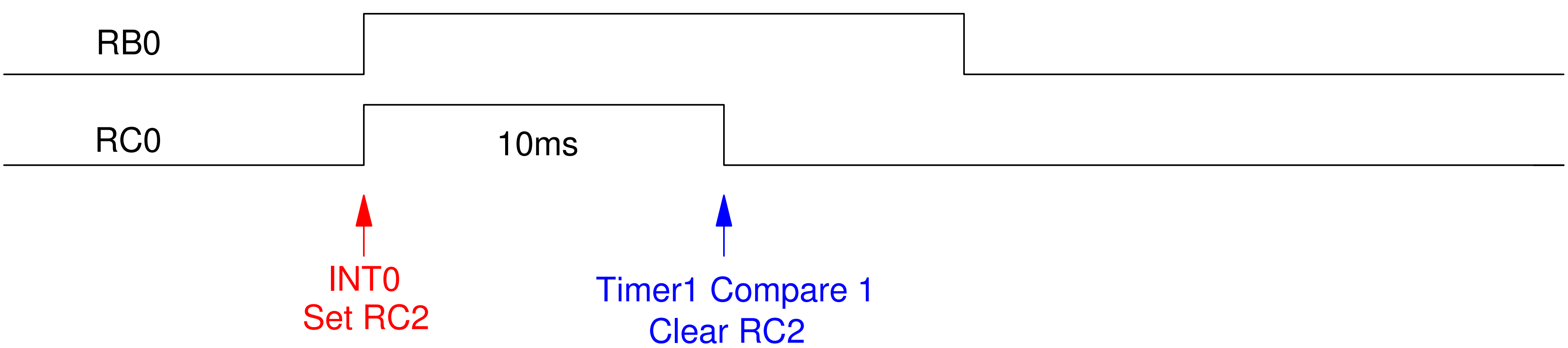
Compare1.C: RC2 outputs a 349.228Hz using Timer1 Compare interrupts

Precise Pulse Width

Problem: Output a pulse that is precisely 10ms long when RB0 is pressed.

Solution: Use two different interrupts:

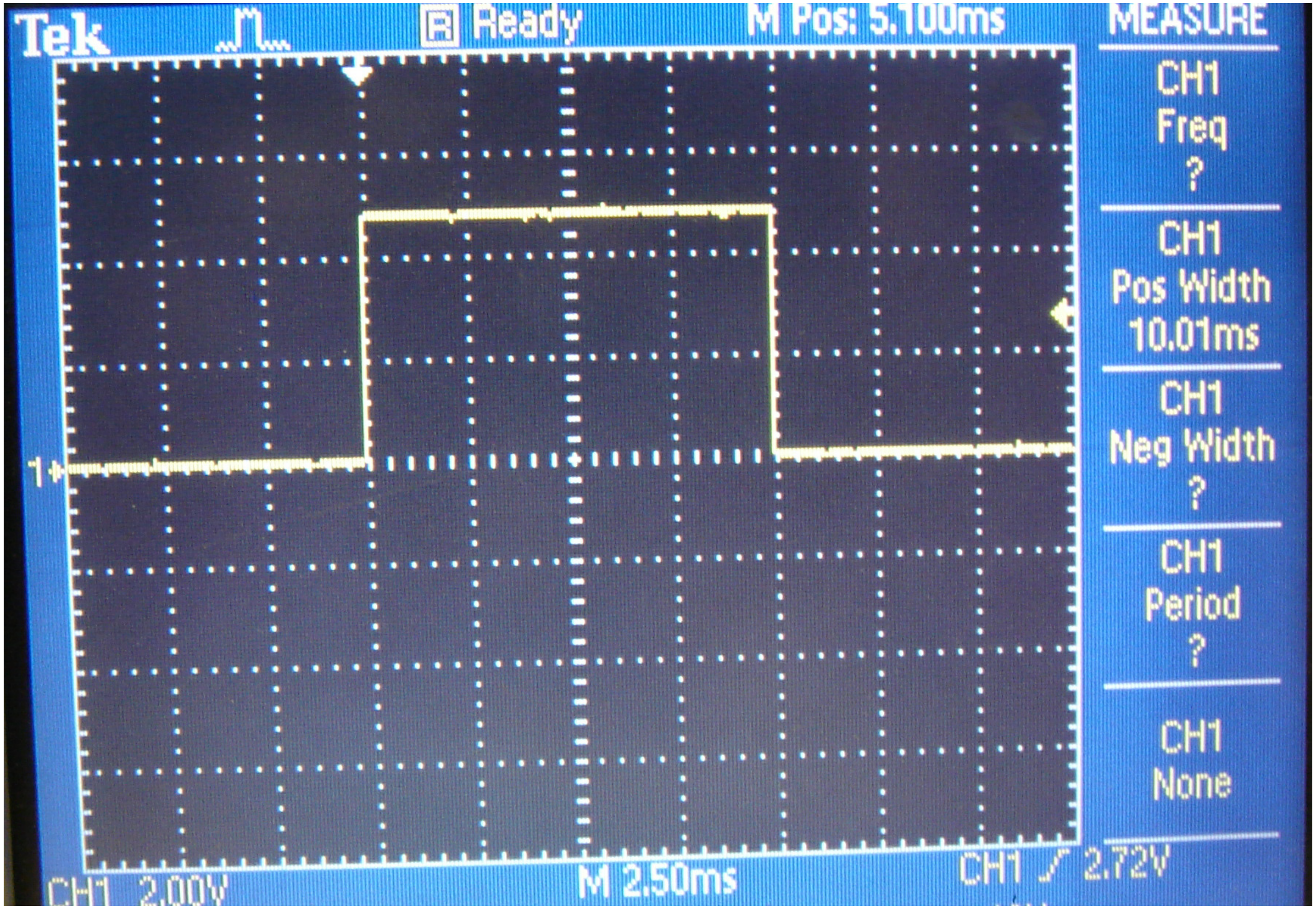
- INT0 records the time that the button was pressed (time of rising edge on RB0). RC0 is set at that time.
- Compare1 kicks in 10ms later. At that time, RC0 is cleared.



Code: Interrupt Service Routine:

```
// Interrupt Service Routine

void interrupt IntServe(void)
{
   if (INT0IF) {
      RC2 = 1;
      CCPR1 = TMR1 + 12500;    // 10ms with PS = 8
      CCP1CON = 0x09;          // clear RC2 when TMR1 == CCPR1
      INT0IF = 0;
      }
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      RC2 = 0;                 // not needed - just to be sure
      CCP1IF = 0;
      }
   }
```

Resulting Signal on RC2:



Compare2.c: A 10ms pulse is generated every time you press RB0

Pulse Width Modulation

Output an analog-like signal

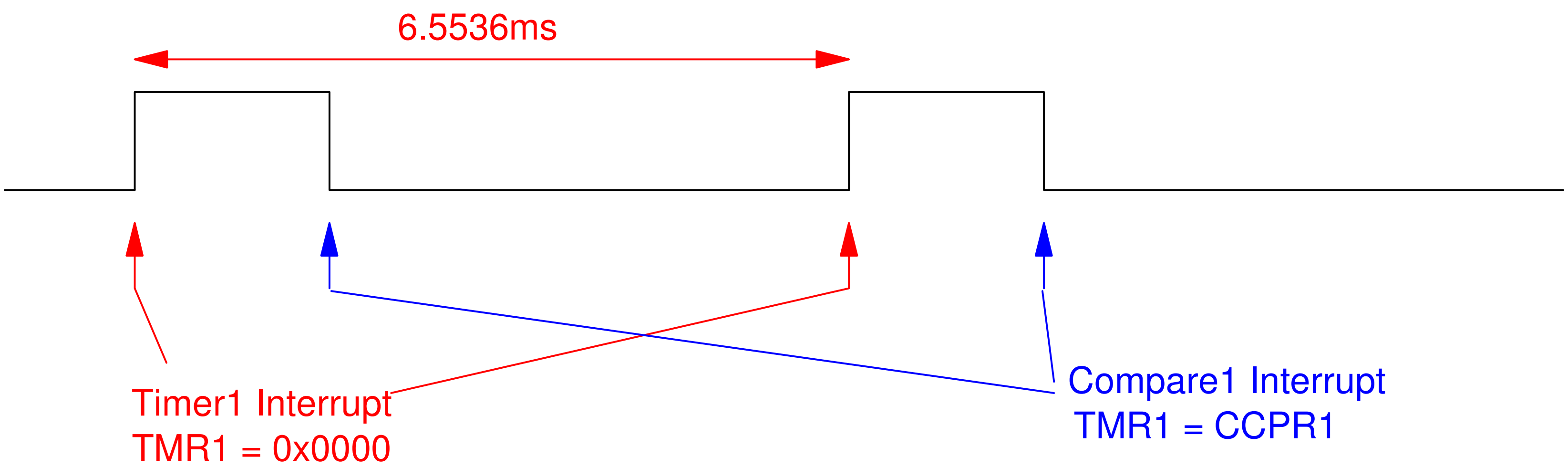
- Average = 0.00V to 5.00V with 65,536 steps

Timer1:

- Set RC2 when TMR1 = 0 (every 65,536 clocks)

Capture1:

- Clear RC2 when TMR1 = CCPR1



Code: PWM.C

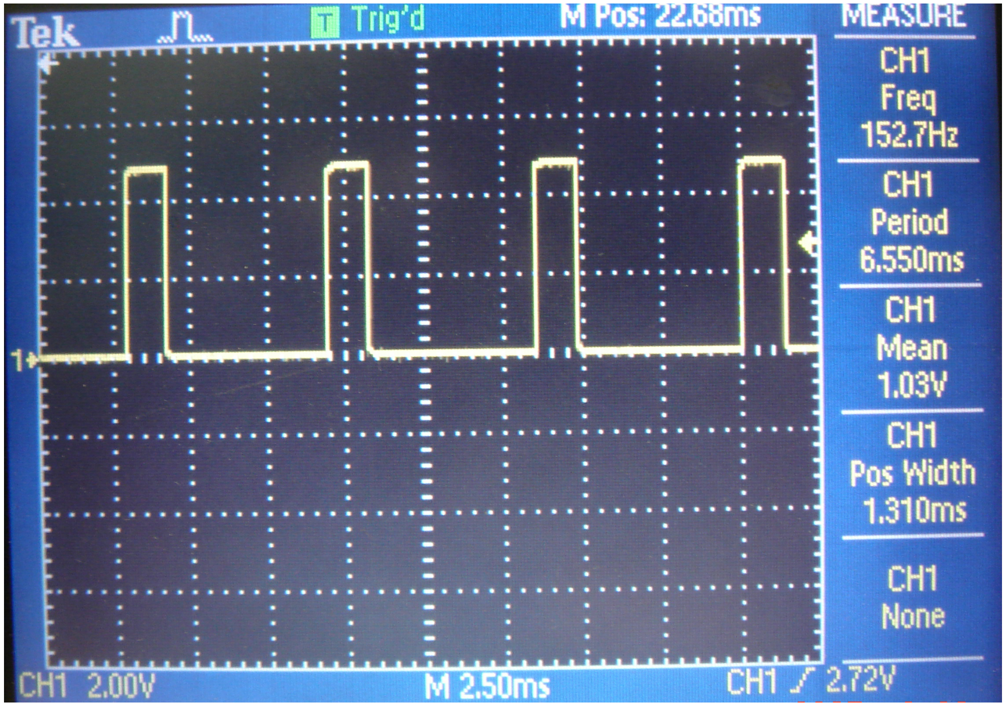
```
// Global Variables
unsigned long int TIME;
unsigned int PWM;

void interrupt IntServe(void)
{
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      if(RC2) {
         CCP1CON = 0x09;      // clear RC2 when TMR1 == PWM
         CCPR1 = PWM;
         }
      else {
         CCP1CON = 0x08;      // set RC2 when TMR1 == 0
         CCPR1 = 0;
         }
      CCP1IF = 0;
      }
   }
```

Example: Output 1.000V (average)

$$PWM = \left(\frac{1V}{5V}\right) = 20\%$$

CCPR1 should then be 20% of its maximum value:

$$CCPR1 = 0.2 \cdot 65,536 = 13,107$$



PWM Limitations

65,536 steps from 0.00V to 5.00V

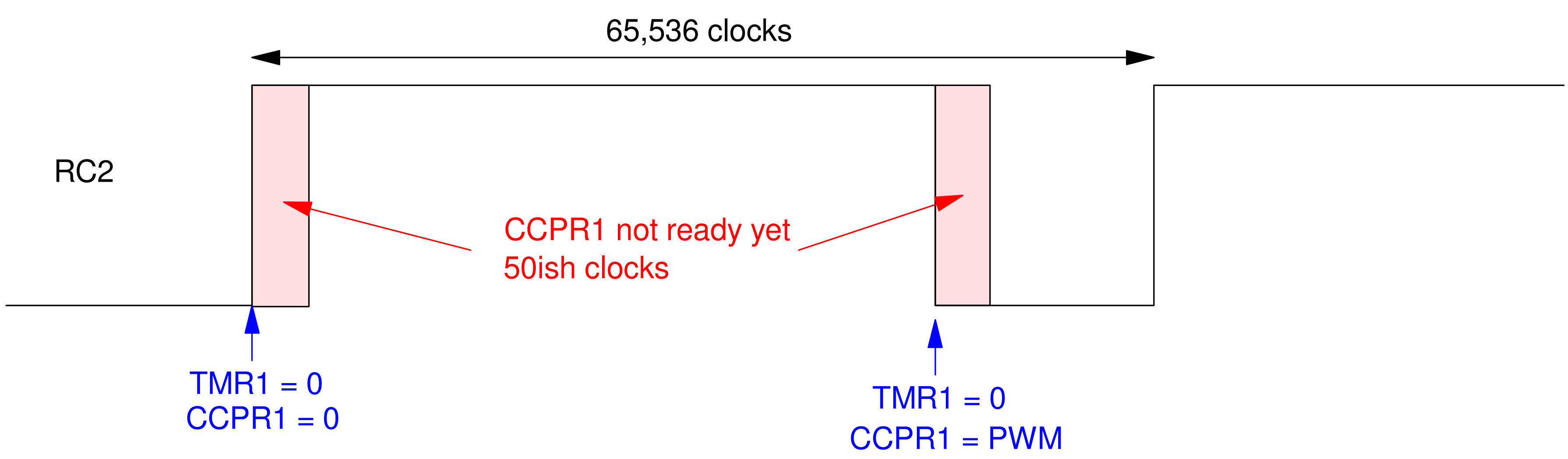
Minimum pulse width = 50 clocks

- 0.076% on

Maximum pulse width = 65,536 - 50

- 99.924% on

It takes about 50 clocks to trigger an interupt



Fun with Timer1 Compare Interrupts:

- Can you hear a 1% difference in frequency at 349.228Hz?

Procedure

Each Trial:

- Play 349.228Hz for 500ms
- Pause 100ms
- Play 352.72Hz for 500ms (1% more)
- Pause 1000ms

Problem: You know that the 2nd frequency is 1% higher

- Biases the result
- (not a blind experiment)

Code

```
if (CCP1IF) {
   if(PLAY) RC0 = !RC0;
   else RC0 = 0;
   CCPR1 += N;
   CCP1IF = 0;
   }
--------------
while(1) {
   N = 14317;
   PLAY = 1;
   Wait_ms(500);
   PLAY = 0;
   Wait_ms(100);
   N = 14175;
   PLAY = 1;
   Wait_ms(500);
   PLAY = 0;
   Wait_ms(1000);
   }
```

Take 2: Blind Experiment

- Not double blind

Procedure

Operator presses RB0 or RB1

- Play 349.228Hz for 500ms
- Pause 100ms
- if RB0: Play same note
- if RB1: Play different note
- Pause 1000ms

Problem: The person running the test knows the answer

- Not a double-blind experiment

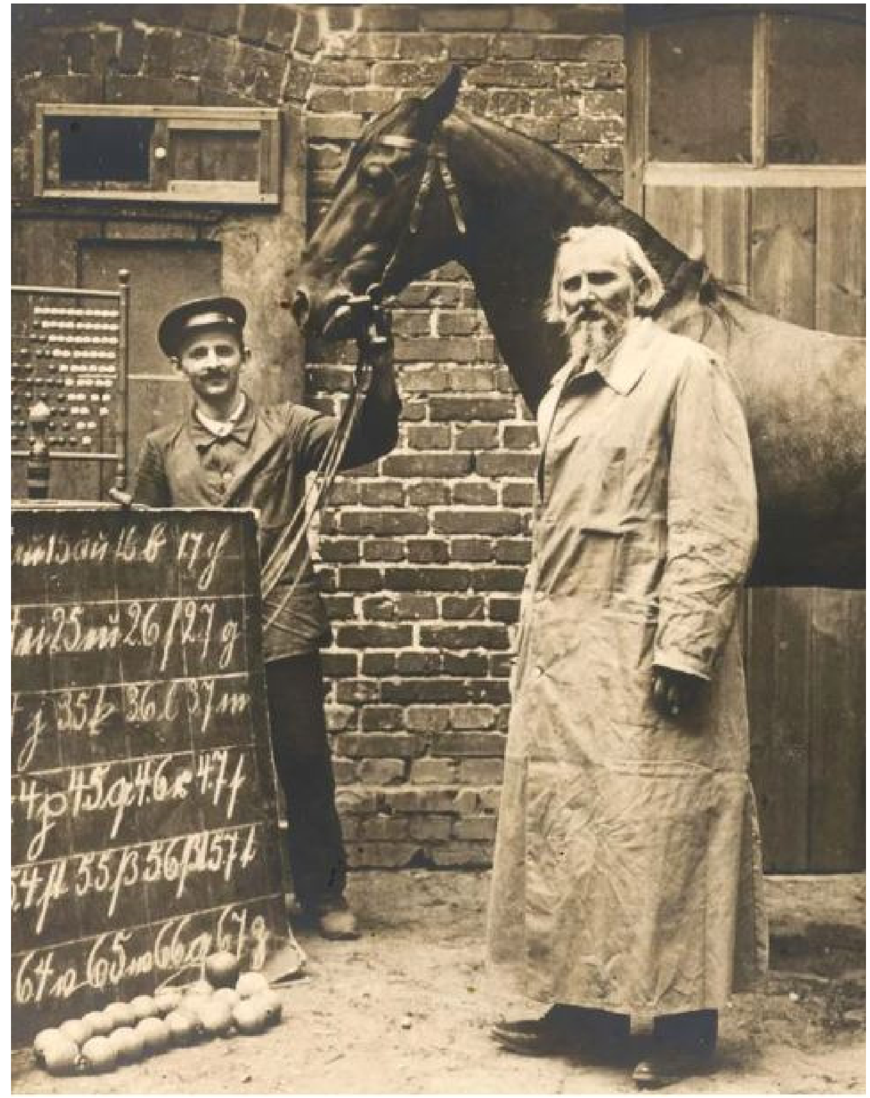
Code

```
while(1) {
    N = 14317;
    PLAY = 1;
    Wait_ms(500);
    PLAY = 0;
    Wait_ms(100);
    if(RB0) N = 14317;
    if(RB1) N = 14175;
    PLAY = 1;
    Wait_ms(500);
    PLAY = 0;
    Wait_ms(1000);
    }
```

Importance of double-blind experiments

www.britannica.com/topic/Clever-Hans

- Clever Hans was a sensation in 1891 - 1907
- This horse could add, subtract, read, and spell



Actually, Clever Hans was reading body language

- The observers knew the answers
- They were giving clues
- Similar to a "tell" in poker

Double-Blind Experiment

Procedure

Flip a coin

- If heads, play two different notes
- If tails, play the same note twice

Operator 'guesses' if the notes were same or different

- Press RB0 (same) or RB1 (different)
- Computer tallies correct / incorrect guesses

Null Hypothesis

- You cannot tell the difference
- p = 0.5 (50/50 odds)
- Chi-squared test

Code

```
while(1) {
    COIN = TRM1 % 2;
    N = 14317;
    PLAY = 1;
    Wait_ms(500);
    PLAY = 0;
    Wait_ms(100);
    if(COIN) N = 14175;
    else N = 14317;
    PLAY = 1;
    Wait_ms(500);
    PLAY = 0;
    Wait_ms(1000);
    }
```

Importance of your question / hypothesis...

What is the smallest frequency difference you can hear?

- Too broad
  - *100Hz? 1kHz? 10kHz?*
  - *1%? 0..1%? 0.01%?*
- May take years and hundreds of experiments to answer

Better:

Can you hear a 1% difference in frequency at 359.228Hz?

- Needs to be specific enough to be doable
- Needs to be general enough to be interesting