More Fun with Timer Interrupts

Chords

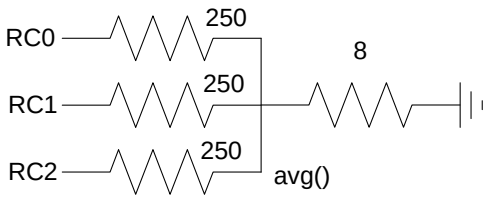
Objective: Play a musical chord each time you press a button:

Button	RC0	RC1	RC2
Timer	Timer0	Timer1	Timer3
RB0	A3	C4	E4
RB1	B3	D4	F4
RB2	C4	E4	G4

Calculations: Assume Timer 0 / 1 / 3 with a pre-scalar of 1

	A3	B3	C4	D4	E4	F4	G4	A4
Hz	220	246.94	261.63	293.66	329.63	349.23	392	440
N	22,727.27	20,247.83	19,110.96	17,026.49	15,168.52	14,317.21	12,755.1	11,363.64

Hardware: Connect RC0 / RC1 / RC2 to an 8 Ohm speaker. Limit the current from each pin on the PIC to 20mA (250 Ohms @ 5V).



Software: Global Variables:

```
// Global Variables

const unsigned char MSG0[21] = "Chord.C             ";
const unsigned char MSG1[21] = "Timer 0/1/2/3       ";

const unsigned int A3 = 22727;
const unsigned int B3 = 20247;
const unsigned int C4 = 19110;
const unsigned int D4 = 17026;
const unsigned int E4 = 15168;
const unsigned int F4 = 14317;
const unsigned int G4 = 12755;
const unsigned int A4 = 11363;

unsigned int N0, N1, N3;
```

Interrupt Service Routine:

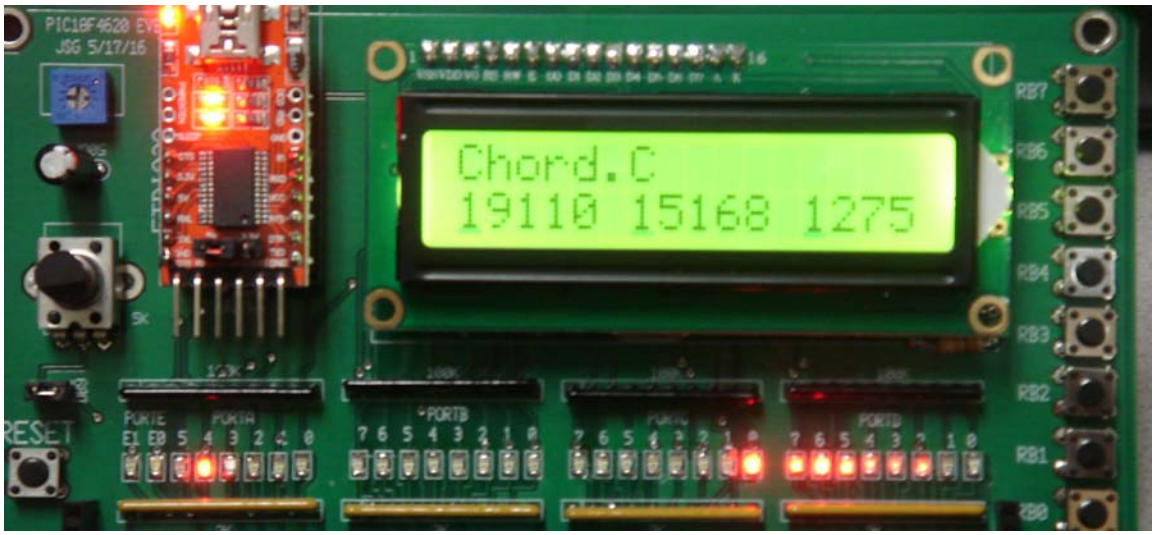
```
// Interrupt Service Routine
void interrupt IntServe(void)
{
   if (TMR0IF) {
      TMR0 = -N0;
      if (PORTB) RC0 = !RC0;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      TMR1 = -N1;
      if (PORTB) RC1 = !RC1;
      TMR1IF = 0;
      }
   if (TMR2IF) {
      if (RB0) { N0 = A3;  N1 = C4;  N3 = E4; }
      if (RB1) { N0 = B3;  N1 = D4;  N3 = F4; }
      if (RB2) { N0 = C4;  N1 = E4;  N3 = G4; }
      if (RB3) { N0 = D4;  N1 = F4;  N3 = A4; }
      TMR2IF = 0;
      }
   if (TMR3IF) {
      TMR3 = -N3;
      if (PORTB) RC2 = !RC2;
      TMR3IF = 0;
      }
   }
```

Interrupt Initialization:

```
// set up Timer0 for PS = 1
   T0CS = 0;
   T0CON = 0x88;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;
// set up Timer1 for PS = 1
   TMR1CS = 0;
   T1CON = 0x81;
   TMR1ON = 1;
   TMR1IE = 1;
   TMR1IP = 1;
   PEIE = 1;
// set up Timer2 for 1ms
   T2CON = 0x4D;
   PR2   = 249;
   TMR2ON = 1;
   TMR2IE = 1;
   TMR2IP = 1;
   PEIE = 1;
// set up Timer3 for PS = 1
   TMR3CS = 0;
   T3CON = 0x81;
   TMR3ON = 1;
   TMR3IE = 1;
   TMR3IP = 1;
   PEIE = 1;
// turn on all interrupts
   GIE = 1;
```

Main Loop: Interrupts do all the work - just display the timing for each interrupt (N)

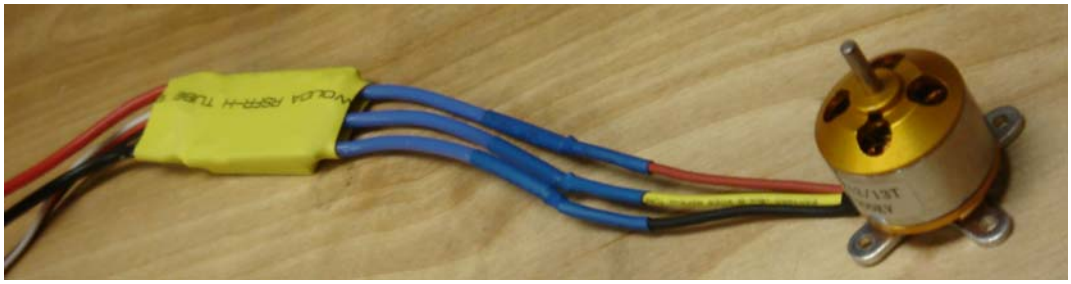
```
while(1) {
   LCD_Move(0,0);  LCD_Out(N0, 4);
   LCD_Move(0,8);  LCD_Out(N1, 4);
   LCD_Move(1,0);  LCD_Out(N3, 4);
   }
```



Display resulting from pressing button RB2: the period of the three notes played are 19,110 / 15,168 / 12,750 clocks
The resistors are connected to RC0 / RC1 and RC2

Quad Copter Motor Controller (Quad.C)

Objective: Control the speed of a BLDC motor (quad-copter motor) using the push buttons:



Hardware:

- Blue Wires from the controller: phase A / B / C to the BLDC motor
- Power (black / red wires):
  - Red = +6 to +12V DC, capable of 1A
  - Black = ground
- Signal: (3-wire black / red / white)
  - Black: ground
  - Red: +5V
  - White: Signal: 0.9ms to 2.0ms pulse @ 50Hz

Software: The controller for the quad copter motor uses format which is fairly standard with R/C controllers:

- The white signal wire is TTL logic levels (0 / 5V)
- The frequency of the pulse should be 50Hz
- The pulse width determines the motor speed
  - 0.9ms Idle
  - 1.2ms Slow
  - 2.0ms Fast

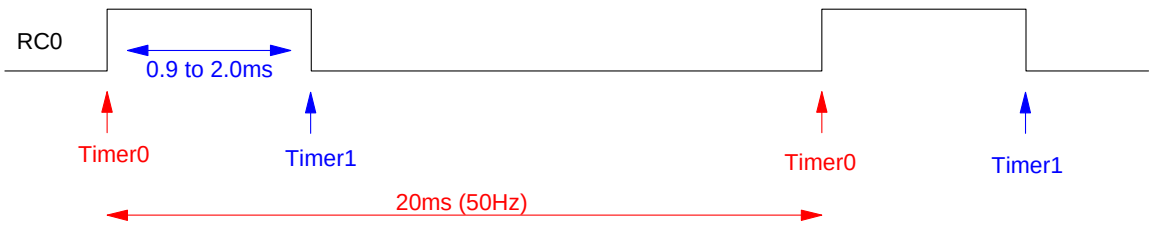
To generate this signal, use two interrupts:

Timer0 Interrupt

- Called every 20ms (50Hz).
- RC0 is set every interrupt
- In Timer0, the next Timer1 interrupt is set up for 0.9 to 2.0ms later

Timer1

- Clear RC0



Software:

Timer0 Interrupt: 20ms is 200,000 clocks. To trigger a Timer0 interrupt every 200,000 clocks, let

- PS = 4
- Y = 50,000

Timer1 Interrupt: 0.9 to 2.0ms is 9,000 to 20,000 clocks. Let PS = 1.

Code: (Quad.C)

Interrupt Service Routine:

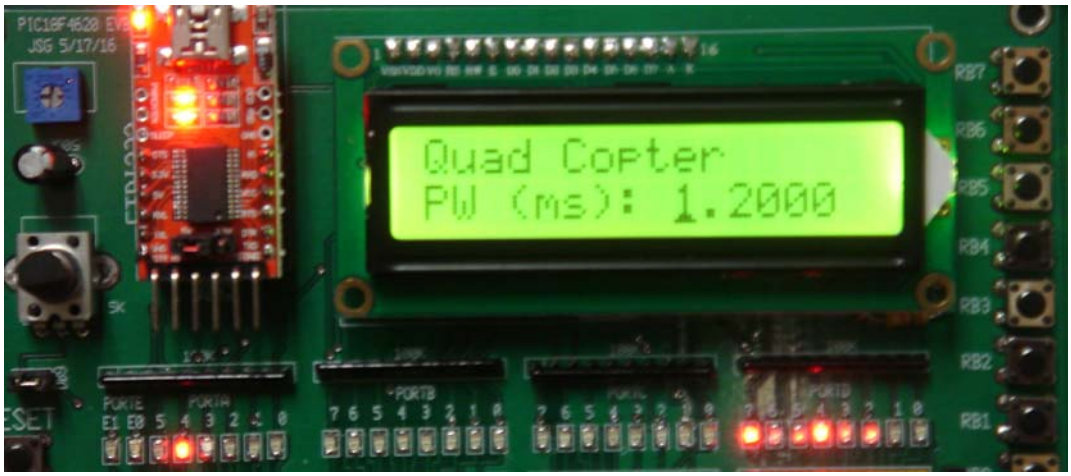
```
void interrupt IntServe(void)
{
   if (TMR0IF) {        // PS = 4
      TMR0 = -50000;
      TMR1 = -N;
      T0 += 1;
      RC0 = 1;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      RC0 = 0;
      T1 += 1;
      TMR1IF = 0;
      }
   }
```

Interrupt Set-Up:

```
// set up Timer0 for PS = 4
   T0CS = 0;
   T0CON = 0x81;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;
// set up Timer1 for PS = 1
   TMR1CS = 0;
   T1CON = 0x81;
   TMR1ON = 1;
   TMR1IE = 1;
   TMR1IP = 1;
   PEIE = 1;
// turn on all interrupts
   GIE = 1;
```

Main Loop:

```
        while(1) {
           if (RB0) N =  9000;
           if (RB1) N = 12000;
           if (RB2) N = 13000;
           if (RB3) N = 14000;
           if (RB4) N = 15000;
           if (RB5) N = 16000;
           if (RB6) N = 17000;
           if (RB7) N = 18000;

           LCD_Move(1,9);  LCD_Out(N, 4);

           }
        }
```

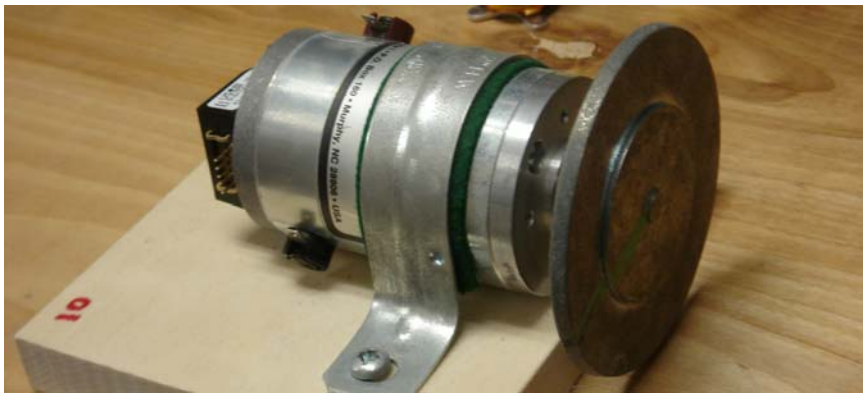


Resulting display when pressing RB1: 1.2ms period pulse.
A 50Hz pulse is output on RC0 which is sent to the white -wire of the quad copter controller.

DC Servo Motor: Frequency Counter (Freq.C)

Objective:

- Measure how fast a DC Servo Motor is spinning
- Measure the frequency of a 0V / 5V square wave in cycles / second



The sensor for this motor is an optical encoder with the following pin-outs:

- Ground
- Index
- Ch A
- +5V
- Ch B

As you rotate the motor, the optical encoder outputs 200 pulses per rotation. By counting the number of pulses per second, you know the motor's speed. To do this, use two interrupts:

Timer0: Interrupt every 1 second (10,000,000 clocks)

- PS = 256
- N = 39,062

Timer1:

- Set the input to Timer1 to RC0 (instead of the clock)
- Set the pre-scalar to 1

Timer1 then counts each pulse from the optical encoder. (Note: The encoder isn't able to drive the LEDs on your PIC board - which draw about 2mA. For the encoder to work, you need to remove the LED jumper on PORTC).

Software:

Interrupt Service Routine:

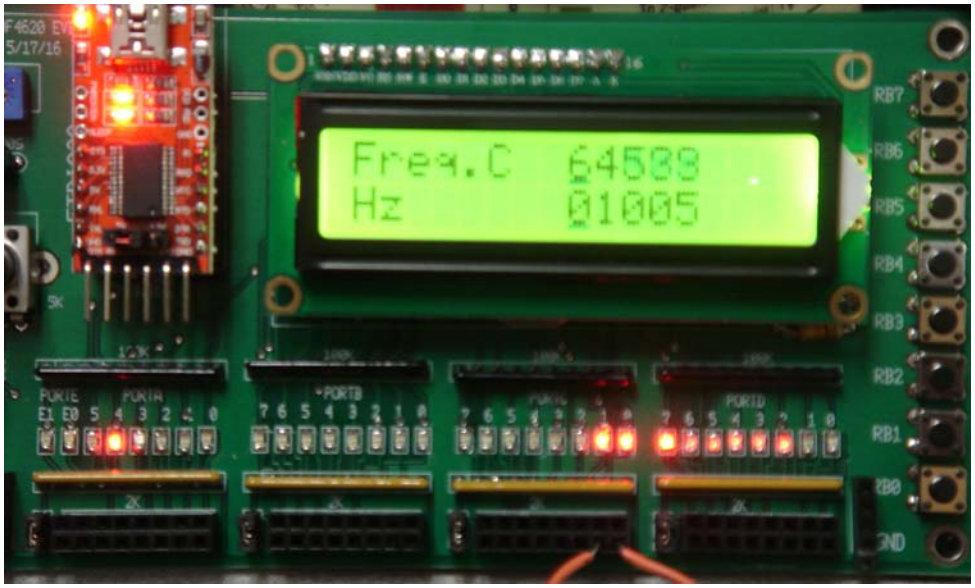
```
// Global Variables
unsigned int T0, T1, T2;

// Interrupt Service Routine
void interrupt IntServe(void)
{
   if (TMR0IF) {
      TMR0 = -39250;
      T0 += 1;
      T2 = T1;
      T1 = TMR1;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      T1 += 1;
      TMR1IF = 0;
      }
   if (TMR2IF) {
      RC1 = !RC1;
      TMR2IF = 0;
      }
   }
```

Interrupt Initialization

```
// set up Timer0 for PS = 256, input = clock
   T0CS = 0;
   T0CON = 0x87;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;
// set up Timer1 for PS = 1, input = RC0
   T1CON = 0x81;
   TMR1ON = 1;
   TMR1IE = 0;
   TMR1IP = 0;
   PEIE = 0;
   TMR1CS = 1;
   TRISC0 = 1;
// set up Timer2 for 0.5ms (1kHz reference signal on RC0)
   T2CON = 0x4D;
   PR2   = 124;
   TMR2ON = 1;
   TMR2IE = 1;
   TMR2IP = 1;
   PEIE = 1;
// turn on all interrupts
   GIE = 1;
```

Main Loop:

```
  while(1) {
      Hz = T1 - T2;
      LCD_Move(0,8);  LCD_Out(TMR1, 0);
      LCD_Move(1,8);  LCD_Out(Hz, 0);
      }
```



Display when measuring a 1kHz square wave (output on RC1).
The running time of Timer1 is displayed on the top row, the cycles per second (Hz) is displayed on the second row.

DC Motor / Fan Tachometer (Tach.C)

Problem: Measure the speed of the 12V motor

I/O Pins:

- Black: Gound
- Red: Power (0V to +12V DC, capable of 10mA)
- Blue: Tachometer output (add a 1k pull-up resistor to +5V to read this signal)

Software:

Interrupt Service Routine:

```
unsigned long int TIME;
unsigned long int T2, T1;
unsigned long int PERIOD;


// Subroutine Declarations
#include <pic18.h>

#include        "lcd_portd.h"


// Subroutines
#include        "lcd_portd.c"

// Interrupt Service Routinevoid interrupt IntServe(void)
{
   if (TMR0IF) {
      TIME = TIME + 0x10000;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      TMR1 = -1;
      T2 = T1;
      T1 = TIME + TMR0;
```

```
            PERIOD = T1 - T2;
            TMR1IF = 0;
            }
         if (TMR2IF) {
            RC1 = !RC1;
            TMR2IF = 0;
            }
         }
```

Interrupt Initialization:

```
      // set up Timer0 for PS = 1
         T0CS = 0;
         T0CON = 0x88;
         TMR0ON = 1;
         TMR0IE = 1;
         TMR0IP = 1;
         PEIE = 1;
      // set up Timer1 for PS = 1, input = RC0
         T1CON = 0x81;
         TMR1ON = 1;
         TMR1IE = 1;
         TMR1IP = 1;
         PEIE = 1;
         TMR1CS = 1;
         TRISC0 = 1;
      // set up Timer2 for 0.5ms
         T2CON = 0x4D;
         PR2   = 124;
         TMR2ON = 1;
         TMR2IE = 1;
         TMR2IP = 1;
         PEIE = 1;

      // turn on all interrupts
         GIE = 1;
```

Main Loop:

```
         while(1) {

            LCD_Move(0,5);  LCD_Out(TIME + TMR0, 7);
            LCD_Move(1,5);  LCD_Out(PERIOD, 7);

            }
```

Display of Tach.C when reading a 1kHz square wave output on RC0.
Row #1 displays the running time since reset in seconds.
Row #2 displays the period of the waveform seen on RC0.

Pulse Width Modulation (PWM.C)

Objective: Turn on and off a motor / light / heater from 0% on to 100% on

- With 10,000 levels of grey, and
- At 1kHz

Solution: Use Timer interrupts:

Timer0 sets RC0

- Timer0 interrupts every 1ms for 1kHz
- When called, it sets up a Timer1 interrupt from 100 to 9900 clocks in the future

When Timer0 kicks in

- Timer1 clears RC0

Software:

Interrupt Service Routine: Pulse Width is passed in PWM

- 0 = 0%
- 10000 = 100%

```
void interrupt IntServe(void)
{
   if (TMR0IF) {
      TMR0 = -10000;
      TMR1 = -PWM;
      TIME += 1;
      RC0 = 1;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      RC0 = 0;
      TMR1IF = 0;
      }

   }
```

Interrupt Initialization:

```
// set up Timer0 for PS = 1
   T0CS = 0;
   T0CON = 0x88;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;
// set up Timer1 for PS = 1
   T1CON = 0x81;
   TMR1ON = 1;
   TMR1IE = 1;
   TMR1IP = 1;
   PEIE = 1;
   TMR1CS = 0;
// turn on all interrupts
   GIE = 1;
```

Main Loop:

```
while(1) {
   if (RB0) PWM = 100;
   if (RB1) PWM = 1000;
   if (RB2) PWM = 2000;
   if (RB3) PWM = 3000;
   if (RB4) PWM = 4000;
   if (RB5) PWM = 5000;
   if (RB6) PWM = 6000;
   if (RB7) PWM = 9900;

   LCD_Move(0,7);  LCD_Out(TIME, 3);
   LCD_Move(1,7);  LCD_Out(PWM, 2);

   }
```