Timer2 Interrupts

Examples

ECE 376 Embedded Systems

Jake Glower - Lecture #19

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

Examples of Timer2 Interrupts:

Once you can keep track of time, there's lots of things you can do. A short list is:

- Build a better wait routine
- Build a stopwatch that's accurate to 0.0001 second (N=1,000)
- Generate musical notes
- Drive a stepper motor (step every 20ms)
- Vary the light output (pulse width modulation)

Better Wait Routine:

- Wait X milliseconds (precisely)

Set up Timer2 for 1ms (a nice round number)

- N = 10,000 (1ms)
  - A = 10, B = 250, C = 4

T2CON 0x4D

7	6	5	4	3	2	1	0
-	A3	A2	A1	A0	TMR2ON	C1	C0
0	1	0	0	1	1	0	1
	A = 10 (9 + 1)					C = 4	

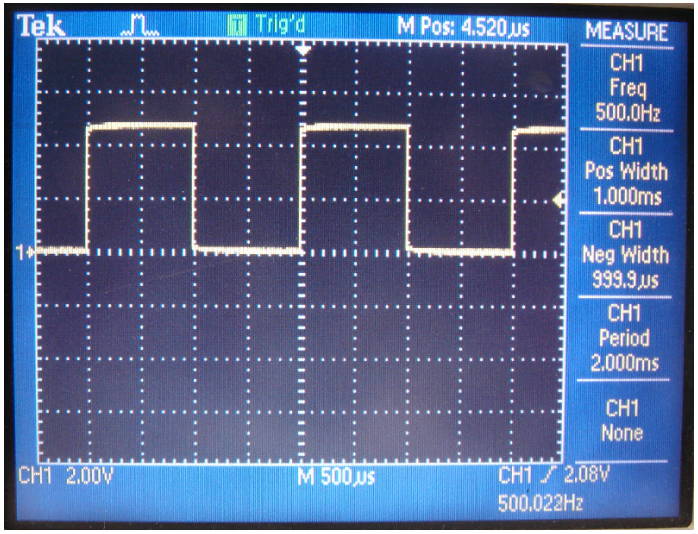
Binary Clock

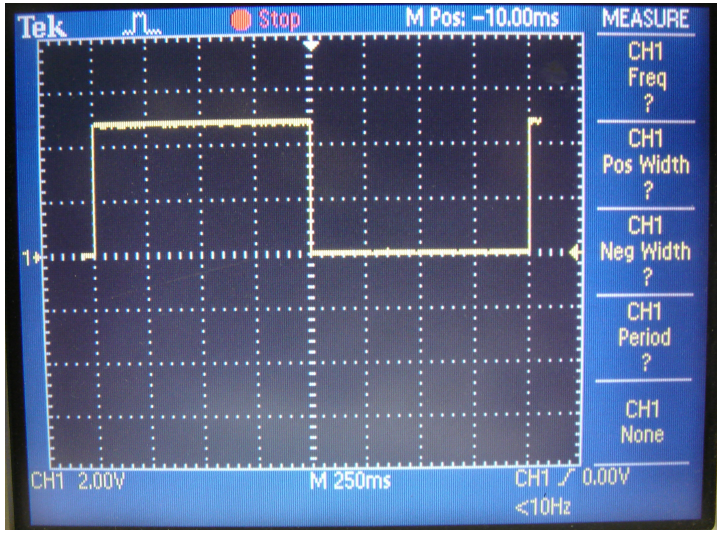
- N = 10,000 (1ms)

Interrupt Service Routine	Main Routine

```
void interrupt IS(void)
{
   RA1= !RA1;
   if(DELAY) DELAY -= 1;
   }
```

```
while(1) {
   RA2 = !RA2;
   while(DELAY);
   DELAY = 1000;
   TIME = TIME + 1;
   LCD_Move(1,0);
   LCD_Out(TIME, 3, 0);
   }
```





Build a Stopwatch

- Accurate to 0.0001 second
- 100x better resolution than the Olympics

Functions:

- RB0: Start
- RB1: Stop
- RB2: Clear

N = 1,000

- A = 1, B = 250, C = 4

T2CON 0x05

7	6	5	4	3	2	1	0
-	A3	A2	A1	A0	TMR2ON	C1	C0
0	0	0	0	0	1	0	1
	A = 1 (0 + 1)					C = 4	

- PR2 = 249

Stopwatch: Interrupt Service Routine

- RA0: Interrupts every 100us
  - RA0 = 5000Hz
  - RA1 = 2500Hz
  - RA2 = 1250Hz

```
nsigned int TIME;
unsigned char RUN;

// Subroutines
#include "LCD_PortD.C"

void interrupt IntServe(void)
{
   if (TMR2IF) {
      PORTA += 1;
      if (RB1) RUN = 1;
      if (RB0) RUN = 0;
      if (RB2) TIME = 0;
      if (RUN) TIME += 1;
      TMR2IF = 0;
      }
   }
```

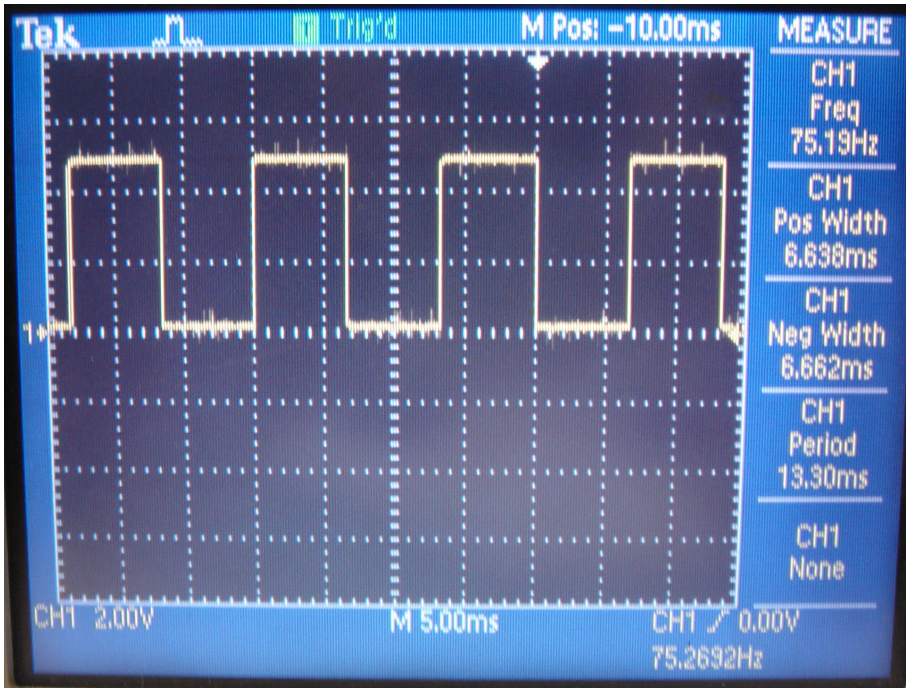
Stopwatch: Main Routine

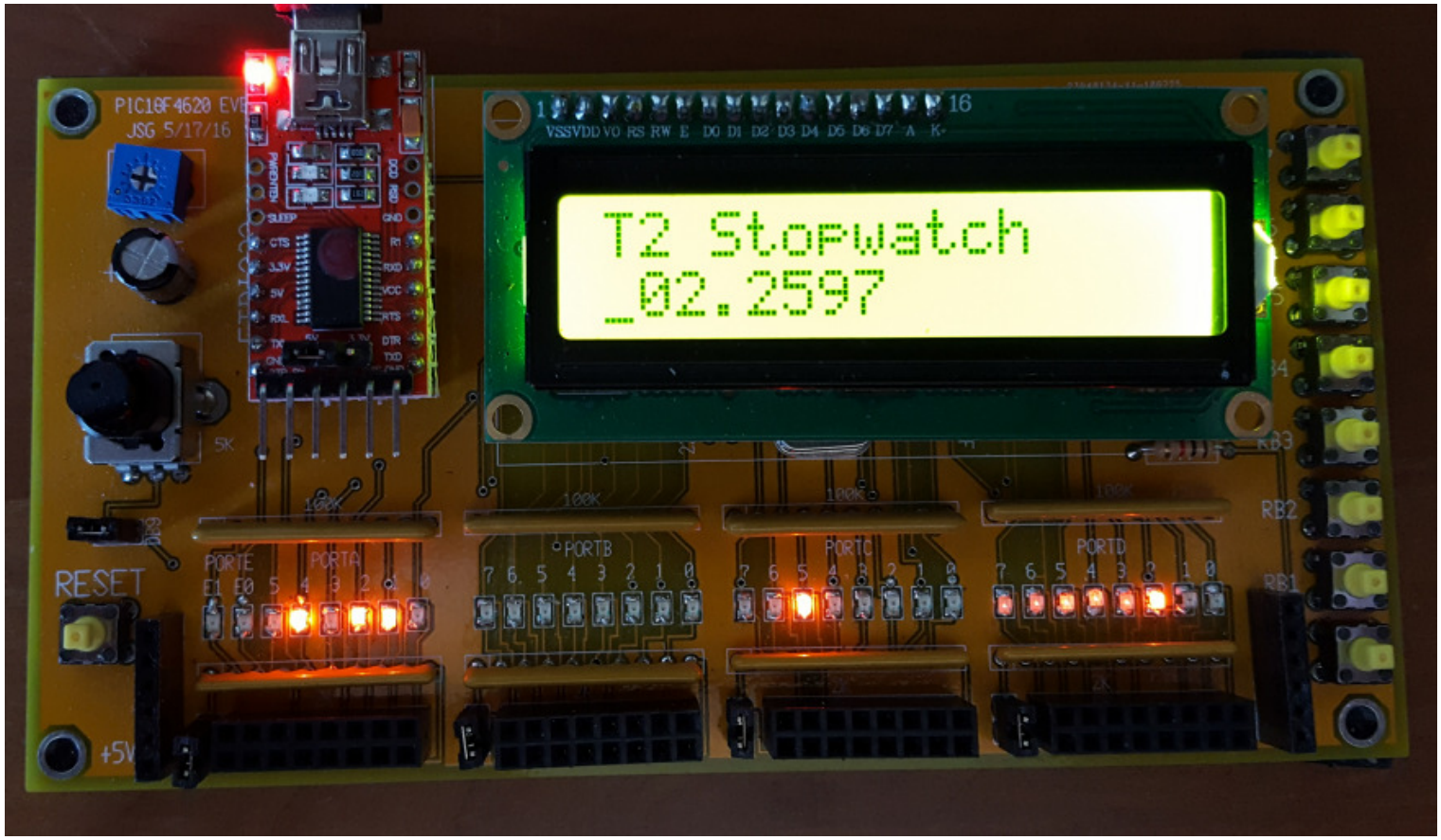
- Thee main routine only displays whatever the current time is.
  - RC0: Loop time = 6.638ms
  - Doesn't tell you much...
- The interrupt does all the work.

```
// initialize Timer2 for 1ms
  PR2 = 249;
  T2CON = 0x4D;
  TMR2IE = 1;
  PEIE = 1;
  TMR2IP = 1;

  TIME = 0;
  RUN = 0;
  GIE = 1;

  while(1) {
     RC0 = !RC0;
     LCD_Move(1,0);
     LCD_Out(TIME, 5, 3);
     }
  }
```





LCD Display of Time, Accurate to 0.0001s (one interrupt)

Build a 3-Key piano:

- You can change the condition of the interrupt (N)
- Example: 3-key piano

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

Note	A4	B4	C5
Hz	440	493.88	523.25
N	11,363.64	10,123.92	9,555.66
A	12	12	12
B	236.74	210.91	199.08
C	4	4	4

Interrupt:

```
void interrupt IS(void)
{
 if (TMR2IF) {
    if(PORTB) RA1 = !RA1;
    else RA1 = 0;
    TMR2IF = 0;
    }
 }
```

Main Routine

```
 while(1) {
    if (RB0) PR2 = 236;
    if (RB1) PR2 = 210;
    if (RB2) PR2 = 198;
    };
```

3-Key Piano: Result

440.00 Hz

493.88 Hz

523.25 Hz

Stepper Motor:

Step every 20ms (200,000 clocks)

- Step every 100th interrupt
- N = 2,000
- A = 10, B = 200, C = 1

T2CON 0x4C	7	6	5	4	3	2	1	0
	-	A3	A2	A1	A0	TMR2ON	C1	C0
	0	1	0	0	1	1	0	0
		A = 10 (9 + 1)					C = 1	

Stepper Motor

Let interrupts do the work

Interrupt Service Routine

```
void interrupt IS(void)
{
   N = (N + 1) % 100;
   if(N == 0) {
      STEP += 1;
      PORTC = TABLE[STEP % 4];
      RA1 = !RA1;
      }
   TMR2IF = 0;
   }
```

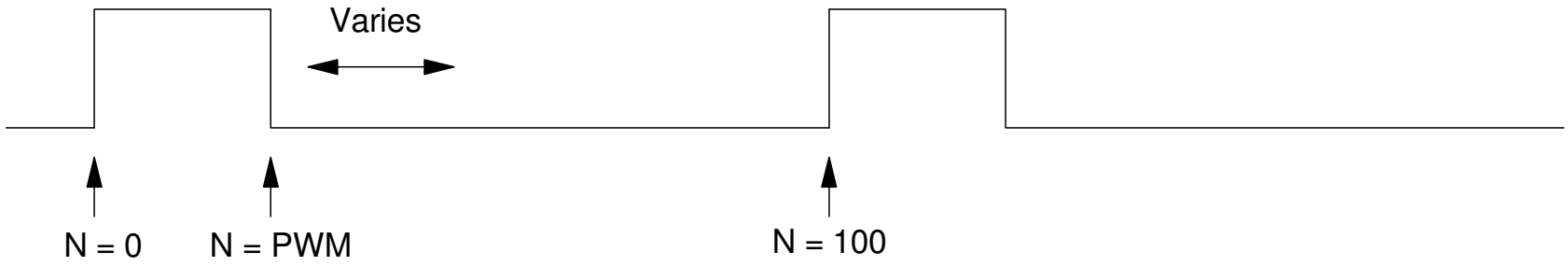
Main Routine

```
while(1) {
   LCD_Move(0,8);
   LCD_Write(STEP, 5, 0);
   }
```

Pulse Width Modulation

- A way to make a binary output look like an analog output
- Allow you to output any color on a Piranah RGB LED

Vary the on-time from 0% (PWM = 0) to 100% (PWM = 64)



Pulse Width Modulation

- Set N = 200
  - A = 1, B = 200, C =1

Interrupt Service Routine

```
void interrupt IS(void)
{
   N = (N + 1) % 64;
   if(N < PWM) PORTC = 0x3F;
   else PORTC = 0;

   if(N == 0) RA1 = !RA1;

   TMR2IF = 0;

   }
```

Main Routine

```
while(1) {

   if(RB0) PWM = 0;
   if(RB1) PWM = 9;
   if(RB2) PWM = 18;
   if(RB3) PWM = 27;
   if(RB4) PWM = 36;
   if(RB5) PWM = 45;
   if(RB6) PWM = 54;
   if(RB7) PWM = 64;

   LCD_Move(1,0);
   LCD_Write(PWM, 3, 0);
   }
```

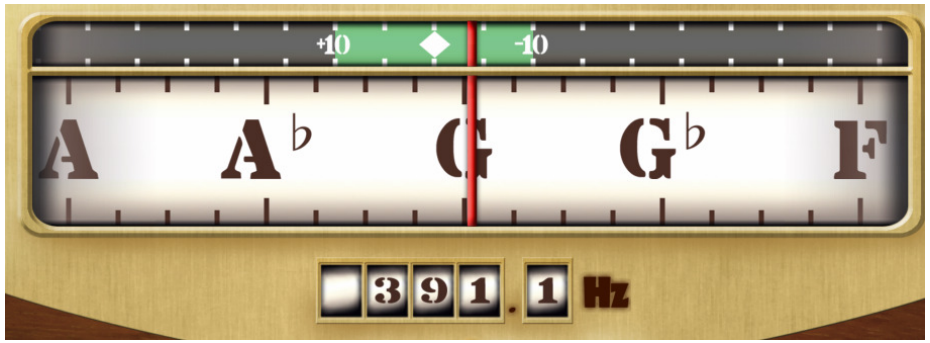
PWM Validation:

Frequency on RA1

N = 200 * 64 = 12,800

f = (10,000,000) / 2N

f = 390.625Hz



Voltage on PORTC

RB7: 4.55V

RB6: 3.84V

RB5: 3.20V

RB4: 2.56V

RB3: 1.92V

RB2: 1.28V

RB1: 0.64V

RB0: 0.00V

Summary

With Timer2 Interrupts, the PIC processor can do two things at the same time

Interrupt:

- Measure time
  - Resolution = 1ms (N = 10,000),
  - Resolution = 0.1ms (N = 1,000), or
- Output a frequency
  - $N = \left(\frac{10{,}000{,}000}{2 \cdot Hz} \right)$

Main Routine:

- Drive the LCD display, read the buttons, etc. in the main routine

Communication with the main routine is through global variables