

Panel 1

Prior to Le13

Timers

ME430 Mechatronics

1

Panel 2

What are Timers?

A timer is a Special Function Register that automatically increments its value after a certain number of instruction cycles

Example:

Timer = 0b 0000 0000 0000 0000

(a certain amount time passes for one timer tick)

Timer = 0b 0000 0000 0000 0001

(lots of time passes)

(65534 timer ticks worth of time pass)

Timer = 0b 1111 1111 1111 1111

(timer overflows and restarts on next timer tick)

Timer = 0b 0000 0000 0000 0000

2

Panel 3

Timers

There are 4 Timers on the 18F4520

- Timer0 => 16 bit (primary timer)
- Timer1 => 16 bit
- Timer2 => 8 bit (used with PWM module)
- Timer3 => 16 bit

Why use a timer when you can use a delay function call?

3

Panel 4

The three hardware peripherals we'll study:

Timers

Analog-to-Digital

Pulse Width Modulation

4

Panel 5

How fast does the timer tick?

The timer runs at or slower than the instruction cycle:

Set instruction cycle ratio

$$\text{Clock Frequency} : \text{Instruction Cycle Frequency}$$

$$4 : 1$$

but Instruction Cycle Frequency : Timer Frequency

Can be set to a variety of values

5

Panel 6

Timer prescaler options:

Table 1: Timer Prescaler Options

Library config statement	Divides the instruction cycle frequency by how much?
TO_PS_1_1	1:1 prescale
TO_PS_1_2	1:2 prescale
TO_PS_1_4	1:4 prescale
TO_PS_1_8	1:8 prescale
TO_PS_1_16	1:16 prescale
TO_PS_1_32	1:32 prescale
TO_PS_1_64	1:64 prescale
TO_PS_1_128	1:128 prescale
TO_PS_1_256	1:256 prescale

6

Panel 7

Let's do some math with timer ticks:

	Frequency	Period
Clock Use 1 MHz		
Instruction Cycle		
Timer Use Prescaler of 16		
Timer overflow: How long does it take for 65536 ticks?		-----

7

Panel 8

Your Turn

	Frequency	Period
Clock Use 500 kHz	-----	
Instruction Cycle	-----	-----
Timer Use Prescaler of 32	-----	-----
Timer overflow: How long does it take for 65536 ticks?		

8

Panel 9

Using 4 MHz make the math pretty:

	Frequency	Period
Clock Use 4 MHz	
Instruction Cycle		
Timer Use Prescaler of 128		

9

Panel 10

What are the two factors in determining the timer tick timing?

The Clock Frequency

OSCCON REGISTER

IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

bit 6-4 **IRCF2:IRCF0**: Internal Oscillator Frequency Select bits

- 111 = 8 MHz (INTOSC drives clock directly)
- 110 = 4 MHz
- 101 = 2 MHz
- 100 = 1 MHz⁽³⁾
- 011 = 500 kHz
- 010 = 250 kHz
- 001 = 125 kHz
- 000 = 31 kHz (from either INTOSC/256 or INTRC directly)⁽²⁾

The Timer prescaler value

10

Panel 11

Let's go look at the code!

There are some handy Timer library functions

2.9 TIMER FUNCTIONS

The timer peripherals are supported with the following functions:

TABLE 2-12: TIMER FUNCTIONS

Function	Description
CloseTimerx	Disable timer <i>x</i> .
OpenTimerx	Configure and enable timer <i>x</i> .
ReadTimerx	Read the value of timer <i>x</i> .
WriteTimerx	Write a value into timer <i>x</i> .

11

Panel 12

OpenTimer0

Function: Configure and enable timer0.

Include: `timers.h`

Prototype: `void OpenTimer0(unsigned char config);`

Arguments: `config`
 A bitmask that is created by performing a bitwise AND operation ('&') with a value from each of the categories listed below. These values are defined in the file `timers.h`.

Enable Timer0 Interrupt:

<code>TIMER_INT_ON</code>	Interrupt enabled
<code>TIMER_INT_OFF</code>	Interrupt disabled

Timer Width:

<code>TO_8BIT</code>	8-bit mode
<code>TO_16BIT</code>	16-bit mode

Clock Source:

<code>TO_SOURCE_EXT</code>	External clock source (I/O pin)
<code>TO_SOURCE_INT</code>	Internal clock source (Tosc)

External Clock Trigger (for `TO_SOURCE_EXT`):

<code>TO_EDGE_FALL</code>	External clock on falling edge
<code>TO_EDGE_RISE</code>	External clock on rising edge

Prescale Value:

<code>TO_PS_1_1</code>	1:1 prescale
<code>TO_PS_1_2</code>	1:2 prescale
<code>TO_PS_1_4</code>	1:4 prescale
<code>TO_PS_1_8</code>	1:8 prescale
<code>TO_PS_1_16</code>	1:16 prescale
<code>TO_PS_1_32</code>	1:32 prescale
<code>TO_PS_1_64</code>	1:64 prescale
<code>TO_PS_1_128</code>	1:128 prescale
<code>TO_PS_1_256</code>	1:256 prescale

Remarks: This function configures timer0 according to the options specified and then enables it.

File Name: `t0open.c`

Code Example:

```
OpenTimer0( TIMER_INT_OFF &
            TO_8BIT &
            TO_SOURCE_INT &
            TO_PS_1_32 );
```

12

Panel 13

Let's go play with MPLAB

```

void main (void)
{
    // Run the clock at 500 kHz
    OSCCONbits.IRCF2 = 0;
    OSCCONbits.IRCF1 = 1;
    OSCCONbits.IRCF0 = 1;

    // Setup the time with a 1:32 prescaler
    OpenTimer0( TIMER_INT_OFF & TO_16BIT & TO_SOURCE_INT & TO_PS_1_32 );

    // Therefore the timer0 freq is 500 kHz / 4 / 32 = 3906 Hz = 256 uSeconds

    ADCON1 = 0x0F;
    TRISB = 0x00;
    PORTB = 0x00;

    while (1)
    {
        WriteTimer0(0x0000);
        PORTB = 0x01;
        // Let's say I want a 0.1 second delay
        while(ReadTimer0() < 1)
        {
            // Could do stuff while waiting for timer
        }

        WriteTimer0(0x0000);
        PORTB = 0x02;
        // Let's say I want a 0.2 second delay
        while(ReadTimer0() < 1);

        WriteTimer0(0x0000);
        PORTB = 0x04;
        // Let's say I want a 1 second delay
        while(ReadTimer0() < 1);

        WriteTimer0(0x0000);
        PORTB = 0x04;
        // Let's say I want a 20 second delay
        while(ReadTimer0() < 1);
        WriteTimer0(0x0000);
        while(ReadTimer0() < 1);
    }
}

```

13

Panel 14

Running the timers demo in MPLAB

- Download the .c file off the website called timers using blocking code.c (in the preclass area)
- Put it in a new folder, build a project, add a linker, etc.
- Use the Simulator as the debugger
- Setup the Simulator frequency to 500 kHz
- View the stopwatch
- Add breakpoints after the while statements
- Change the delay values and test the code

14

Panel 15

Timer ticks we need to wait for 0.1 second

Timer ticks we need to wait for 0.2 seconds

15

Panel 16

Timer ticks we need to wait for 1 second

Timer ticks we need to wait for 20 seconds

16

Panel 17

In class what are we gonna do?

Let's play with an example program using the PICDEM 2 boards

Download the Timer 0 Demo Code

Go get your MPLAB ICD 2 and PICDEM 2 board

Create a new folder put in the:

timer example.c

PICDEM 2 LCD module.c

PICDEM 2 LCD module.h

Start a new project with these files

17

Panel 18

Things to do with the example program:

Watch it run

Preset the Timer 0 within the interrupt

Make the interrupts go every 1 second by presetting timer value

Make the interrupt not happen

Play with two timers!

- Make Timer 1 have an interrupt that toggles RB3

18

Panel 19

OpenTimer1	
Function:	Configure and enable timer1.
Include:	timers.h
Prototype:	void OpenTimer1(unsigned char config);
Arguments:	config A bitmask that is created by performing a bitwise AND operation (&) with a value from each of the categories listed below. These values are defined in the file timers.h.
Enable Timer1 Interrupt:	TIMER_INT_ON Interrupt enabled TIMER_INT_OFF Interrupt disabled
Timer Width:	T1_8BIT_RW 8-bit mode T1_16BIT_RW 16-bit mode
Clock Source:	T1_SOURCE_EXT External clock source (I/O pin) T1_SOURCE_INT Internal clock source (T0SC)
Prescaler:	T1_PS_1_1 1:1 prescale T1_PS_1_2 1:2 prescale T1_PS_1_4 1:4 prescale T1_PS_1_8 1:8 prescale
Oscillator Use:	T1_OSC1EN_ON Enable Timer1 oscillator T1_OSC1EN_OFF Disable Timer1 oscillator
Synchronize Clock Input:	T1_SYNC_EXT_ON Sync external clock input T1_SYNC_EXT_OFF Don't sync external clock input
Use With CCP:	
For devices with 1 or 2 CCPs	T3_SOURCE_CCP Timer3 source for both CCPs T1_CCP1_T3_CCP2 Timer1 source for CCP1 and Timer3 source for CCP2 T1_SOURCE_CCP Timer1 source for both CCPs
For devices with more than 2 CCPs	T34_SOURCE_CCP Timer3 and Timer4 are sources for all CCPs T12_CCP12_T34_CCP345 Timer1 and Timer2 are sources for CCP1 and CCP2 and Timer3 and Timer4 are sources for CCP3 through CCP5 T12_CCP1_T34_CCP2345 Timer1 and Timer2 are sources for CCP1 and Timer3 and Timer4 are sources for CCP2 through CCP5 T12_SOURCE_CCP Timer1 and Timer2 are sources for all CCPs
Remarks:	This function configures timer1 according to the options specified and then enables it.
File Name:	t1open.c
Code Example:	OpenTimer1(TIMER_INT_ON & T1_8BIT_RW & T1_SOURCE_EXT & T1_PS_1_1 & T1_OSC1EN_OFF &