Today: Timing and Interrupts

- Some timing examples
- Critical sections
- PWM generation and capturing

Administrivia

- Next week:
  - Lab exercise 3: make sure to attend one of the official lab sessions
  - Milestone 3 due next Friday

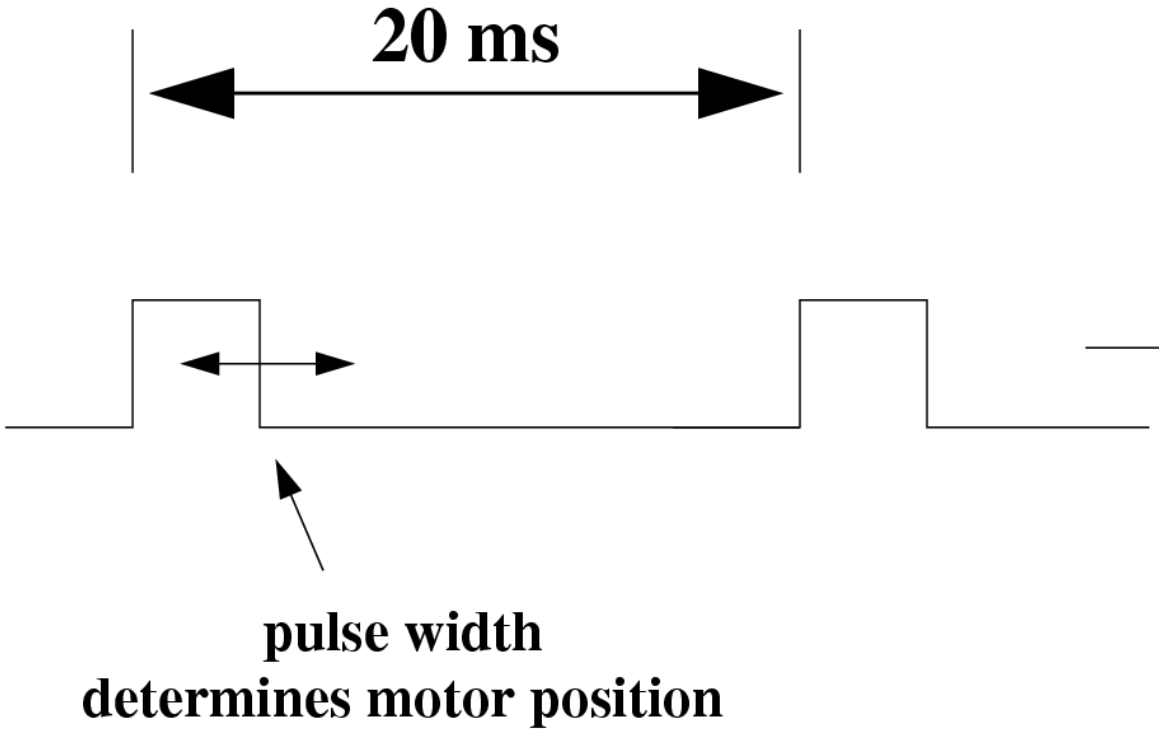
Actuator of the Day...

Actuator of the Day...

RC Servo Motors

- Connector fits on a standard 3-pin header (so easy to connect to your circuit board)
- 3 pins: power (red), ground (black), and PWM command signal (white)

RC Servo Motors



Internal circuit translates pulse width into a goal position

RC Servo Motors

- Internal potentiometer measures the current orientation of the shaft
- Uses a **Position Servo Controller**: the difference between current and commanded shaft position determines shaft velocity. More on this later...
- Mechanical stops limit the range of motion
  - These stops can be removed for unlimited rotation

RC Servos: Power Issues

These servos require significant peak currents and can cause fluctuations in the power supply

- Isolate the PIC power supply from the servo supply
- The servos can handle power supplies of 4.8V – 6.0V
- 3-4 AA batteries are a good solution
- Don't forget to connect the grounds! Why?

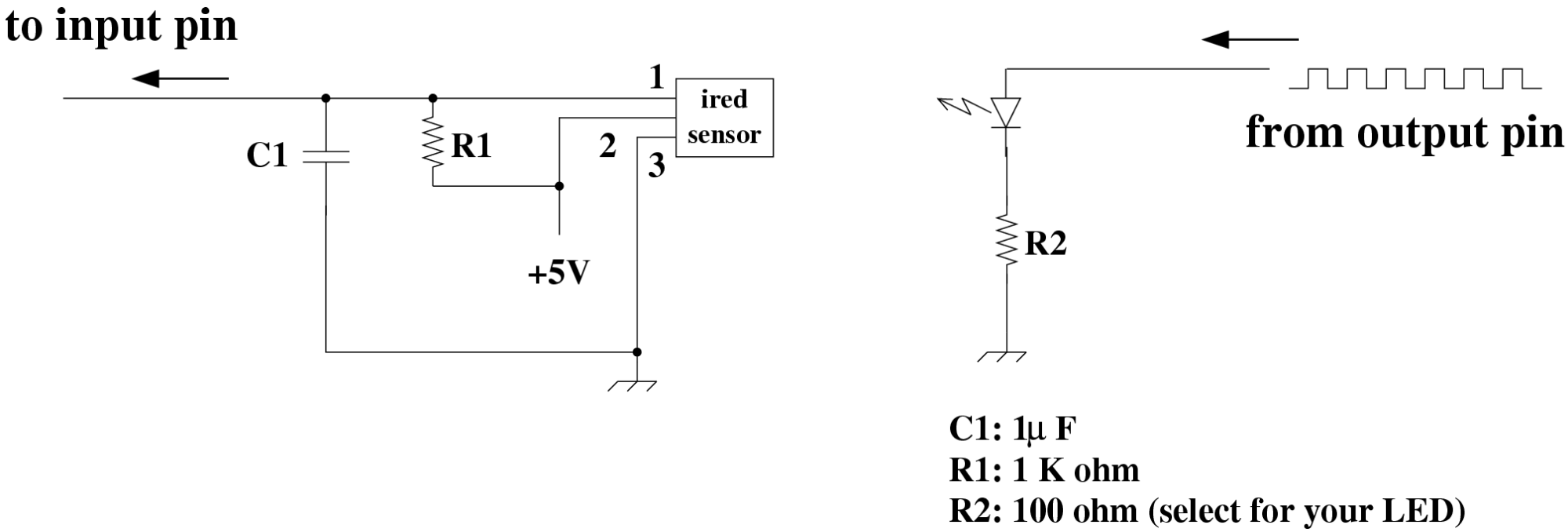
RC Servo Motor Uses

- Small-scale robotics
- Autonomous vehicles
- Directing sensors
- Animatronics

IRED LED Flashing Example

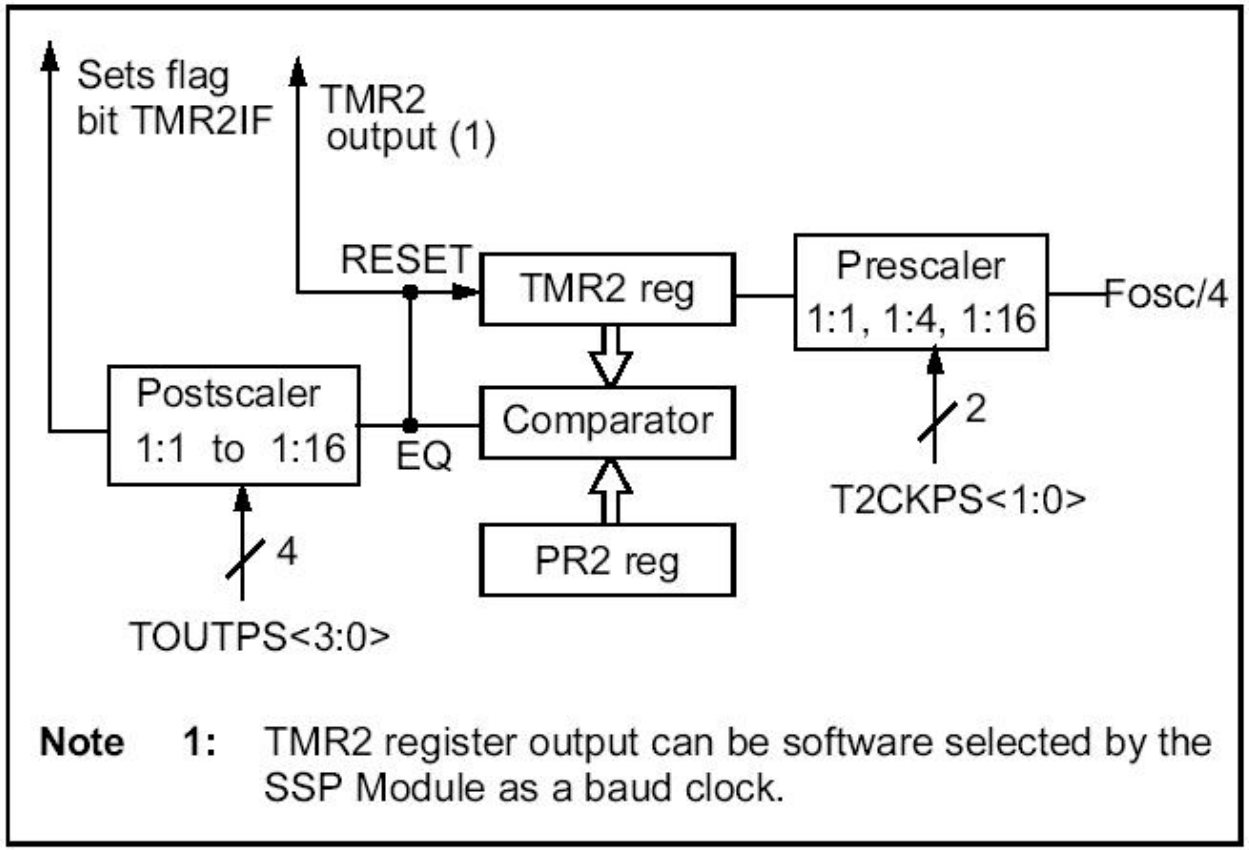
Suppose we want to flash an LED at 40 KHz

- How would we configure the timer?
- What would the ISR look like?



Use Timer 2?

FIGURE 8-1: TIMER2 BLOCK DIAGRAM



What parameters do we need?

Timer 2 Configuration

We need an interrupt frequency of 80 KHz

- Prescaler = 1
- PR2 = 62
- Postscaler = 1

- 20000000/(4 * 1 * 62 * 1) = 80645 Hz
  - Close enough (less than 1% off)

An ISR Implementation

```
#int_TIMER2
void flash_isr() {
    static short state;
    state ^= 1;
    output_bit(pin_A0, state);
}
```

What does **static** do?

Critical Sections

What are they about?

Critical Sections

- Some datastructures or hardware elements must be accessed atomically relative to other threads
  - In particular, we must be concerned with the datastructures that are shared between the main program and the interrupt service routines
- A critical section is a piece of code that assumes that access is atomic

Critical Sections

How do we implement a critical section

- Inside of an interrupt service routine?
- Outside?

Critical Sections

Inside of an interrupt service routine?

- The global interrupt bit is cleared as we enter the interrupt service routine
- So – no other interrupts can occur

Critical Sections

Outside of an interrupt service routine?

- Can explicitly disable the global interrupt bit or the bit for a specific interrupt
- In CCSC, this looks like:

```
DISABLE_INTERRUPTS(INT_TIMER0);
```

Disabling Interrupts

- Don't forget to re-enable the interrupts
- Keep the time period as short as possible
  - Otherwise, we can miss the event for which an interrupt was raised
  - Often, we will precompute a set of quantities outside of the critical section

  ... and then copy these quantities to some shared datastructure within the critical section

Critical Sections

What happens if the interrupt service routine needs to assume that a datastructure does not change over several interrupts?

Critical Sections

What happens if the interrupt service routine needs to assume that a datastructure does not change over several interrupts?

- Then we must create a 3rd copy of the datastructure that is private to the ISR
  - The ISR then copies the data from the shared structure at appropriate times

Use of the Timer Systems

- Counting external events
- Capture/Compare/PWM (CCP) module:
  - The 628 has 1 module; the 876 has 2
  - Capture mode: measure time between events
  - Compare mode: when a specific time arrives, indicate with a change in RB3 output state
  - Pulse width modulation (PWM) generation

Pulse-Width Modulation

FIGURE 11-3: PWM OUTPUT

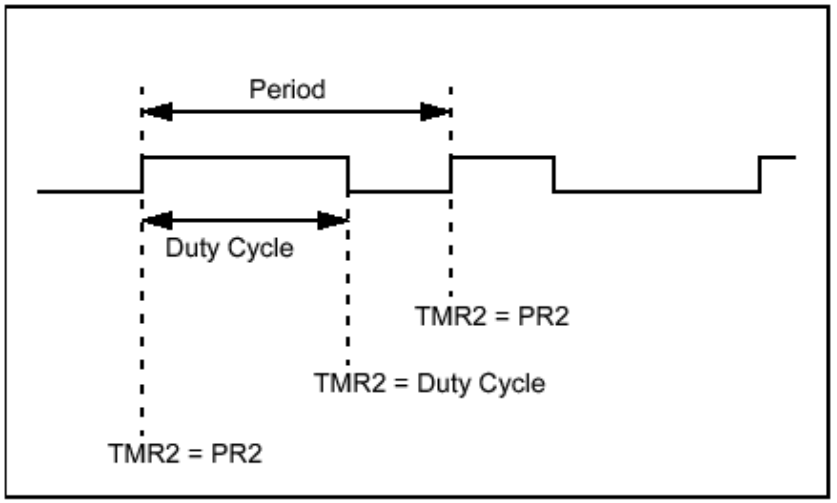
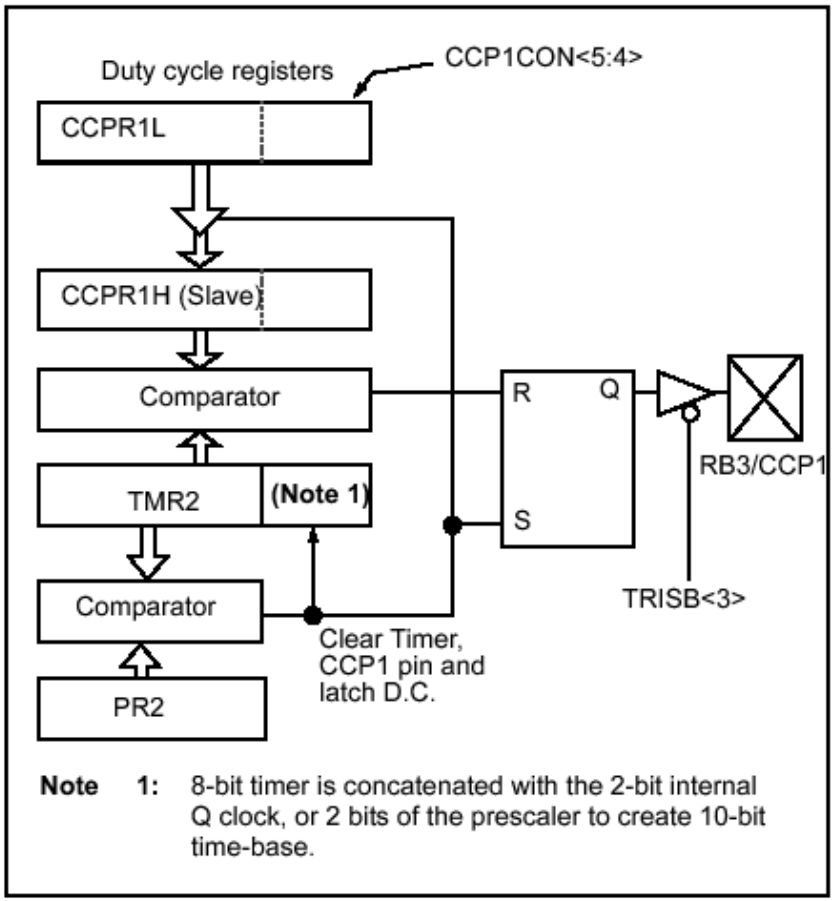


FIGURE 11-2: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler to create 10-bit time-base.

- Timer 2 is used
- Two critical registers:
  - CCPR1: duty cycle
  - PR2: period
- Output to RB3

PWM in Code: The Flashing IRED LED Again

```
setup_ccp1(CCP_PWM);
set_pwm1_duty(63);
                    // Pre, period, post
setup_timer_2(T2_DIV_BY_1, 125, 1);
```

Period is an 8-bit number

Why is the period one half of the interrupt interval in the previous LED example?

PWM in Code: The Flashing IRED LED Again

Why is the period one half of the interrupt interval in the previous LED example?

- During the period, we turn the LED on **and** off...

Other Notes

- By using the PWM hardware, no CPU time is used to generate the LED flash
  - In our previous example, we used a large amount of the available CPU time
- The duty parameter is actually a 10-bit number
  - 1 duty count is equal to ¼ of a period count
  - set_pwm1_duty() can accept an 8-bit or a 10-bit number. When an 8-bit number is specified, it is multiplied by 4 before being placed in the register

How Might We Vary the Brightness of an LED?

How Might We Vary the Brightness of an LED?

- Configure the PWM module with a period of about 1 ms
- Vary the duty cycle:
  - The longer the duty cycle, the brighter the LED

Servo Motor Control

For our servo motors, we want a period of no more than 20ms

- How do we configure the PWM hardware to generate this?

Servo Motor Control

How do we configure the PWM hardware to generate this?

- Unfortunately, with a 20 MHz clock, we can only achieve periods as low as 0.8192 ms
- We must find some other way...

Compare Mode

Next Time

- Lab:
  - Lab exercise 3: next week
- Lectures next week:
  - Finish up on timing issues
  - Synchronous serial communication
  - DEH Chapter 9
  - DwPICM Chapters 7, 9