CSE 466: Software for Embedded Systems

Autumn, 2002

- **Instructor: Bruce Hemingway**
  Office: Sieg 327A
  Office Hours: MWF 11 A.M.-12 P.M. , TTh 1 P.M.- 2 P.M
  e-mail: bruceh@cs.washington.edu
  Phone: (206) 543-6274

- **Assisted by: Doug Beale, Waylon Brunette, and Kevin Chan**

- **Class Meeting Times and Location:**
- **Lectures: MWF 9:30-10:20 A.M. EE 045**
  Lab: Tues. Section 1, 2:30- 5:20 P.M. Sieg 327
  Thurs. Section 2, 2:30- 5:20 P.M. Sieg 327

What is an Embedded System?

- It's not a desktop system
  - Fixed or semi-fixed functionality (not user programmable)
  - Lacks some or all traditional human interfaces: screen, keyboard, pointing device, audio
  - May have stringent real-time requirements (Hard and Soft)
  - Usually has sensors and actuators for interface to physical world
- It may:
  - replace discrete logic circuits
  - provide feature implementation path
  - Make maintenance easier
  - Protect intellectual property
  - Improve mechanical performance
  - Replace analog circuits

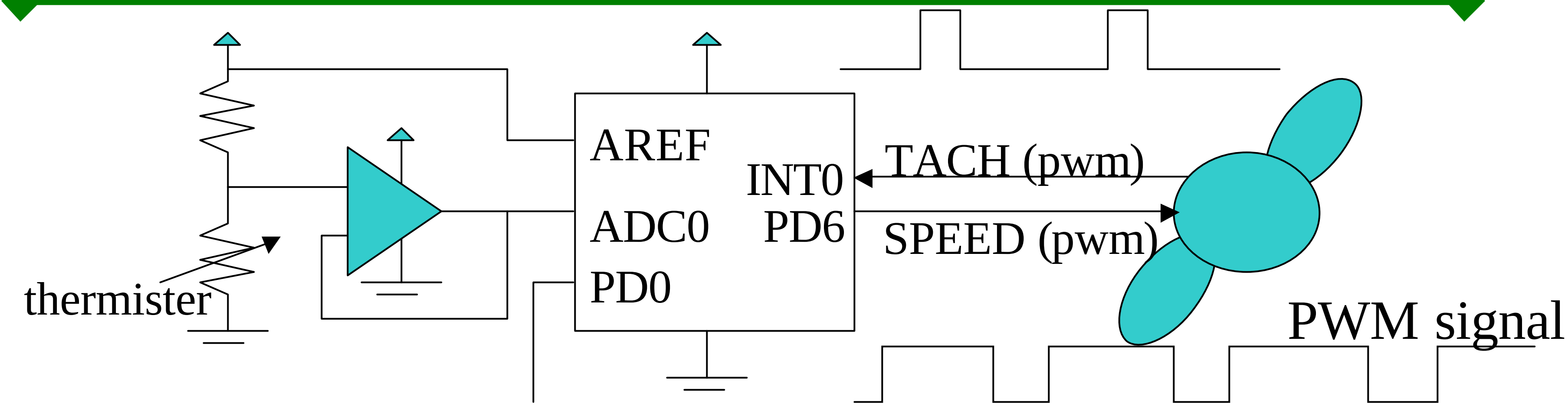
What is an Embedded System

- Figures of Merit for embedded systems
  - Reliability – it can never crash
  - Safety – Involves things that move and can harm/kill a person
  - Power Consumption – may run on limited power supply. Want slowest possible clock, least amount of memory. **You will always be resource constrained!**
  - Cost – Engineering Cost, Mfg Cost, Schedule tradeoffs
  - Product life cycle issues: maintainability, upgradeability, serviceability
  - Performance

“To Have and Have Not” …

- **We don’t have**
  - User Interface
  - Dynamic Linking and Loading
  - Virtual Memory, Protection Modes
  - Disk
  - Processes
- **Instead we have**
  - Real Time Kernel (very small OS) (If we’re lucky)
  - Tasks (threads)
  - Task communication primitives
  - ADC
  - Timers
  - Event Capture
  - PWM

An Example: Temp Controller w/ AVR MCU



Task: Tachometer (external interrupt)

now = getTime();

period = then - now; //overflow?

then = now;

return;

Task: TempControl (periodic, soft constraint)

if (Temp > setpoint) Thi++;

if (Temp < setpoint) Thi--;

if (period<min || period>max) GP4 = 1;

Task: FanPWM (periodic, hard constraint)

count++;

if (count == 0) GP0 = 1;

if (count > Thi) GP0 = 0;

return;

Task: Main

Thi = 0;

setup timer for 1ms interrupt;

setup timer for 100ms interrupt;

while (1) ;

Capacity

- Assume:
  - 8 MHz processor @ one instruction/cycle
  - Assume fan runs between 30Hz and 60Hz
  - Assume 256ms period on speed control PWM, with 1ms resolution.
- What percent of the the available cycles are used for the temperature controller?
  - [total instructions in one second] / (8m I/sec)
- How much RAM do you need?
- How much ROM?

Resource Analysis of Temp Controller

Task: Tachometer (external interrupt)

```
now = getTime();
period = then - now; //overflow?
then = now;
return;
```

Task: TempControl (periodic, soft constraint)

```
if  (Temp > setpoint) Thi++;
if  (Temp < setpoint) Thi--;
if  (period<min || period>max) GP4 = 1;
```

Task: FanPWM (periodic, hard constraint)

```
count++;
if (count == 0) GP0 = 1;
if (count > Thi) GP0 = 0;
return;
```

Task: Main

```
Thi = 0;
setup timer for 1ms interrupt;
setup timer for 100ms interrupt;
while (1) ;
```

Task	ROM	RAM	Instructions/Sec
Tach	~4	2 (period, then)	4 * 60 = 240
FanPWM	~8	1 (count)	8 * 1000 = 8000
TempControl	~10	1 (THI)	10 * 2 = 20

Total Instructions/Sec = 8260, at 8MIPS, that's .1% utilization

Other resources? local variables, stack

Class and Lab Policies

- Lecture
  - See Syllabus and Schedule. Generally coordinated with design problems
  - Mondays– this week's lab assignment
  - Wednesdays– background and some theory
  - Fridays– discuss lab and more background for next lab
- Homework assignments will be short but will precede lecture. Probably 1/week. Graded on an "effort" basis (1, 2, or 3 points). Must be turned in prior to start of class when due.
- Lab
  - Implementation of the design, as specified in class
  - Lab reports due prior to start of next lab section (2:30pm)
- Exams
  - Two, based on lecture, lab, and reading
- No Final
- Reading and Source Material assigned as needed

Business Matters

- Lecture slides will be on line after class
- Go to the 466/schedule link for links to lecture slides, labs, etc.
- If you have a home PC, get and use the tools!
- The Documents:
  - Atmel CD-Rom Data Books
  - ATmega16 Datasheet– on CD, on web, in course pak
  - Prototyping with the Design Kit on web
  - HWLab web page docs
- **“Lab equipment required for the duration of a course or project must be first checked out from the Lab Manager and secured with a deposit check of $200 made payable to "University of Washington" (note that this check will not be cashed but will be returned to the student upon the return of all checked-out equipment in good condition).” from lab policy…**
- When it’s ready, sign-up for CSE466 mailing list (majordomo)

Grading

- Lab reports: 10pts each. Demo required
- Homeworks: 3 points each (will be scaled by difficulty)
- Ratios:
  - Lab: 25%
    Homework: 25%
    Exams total: 40%
    Class Participation: 10%

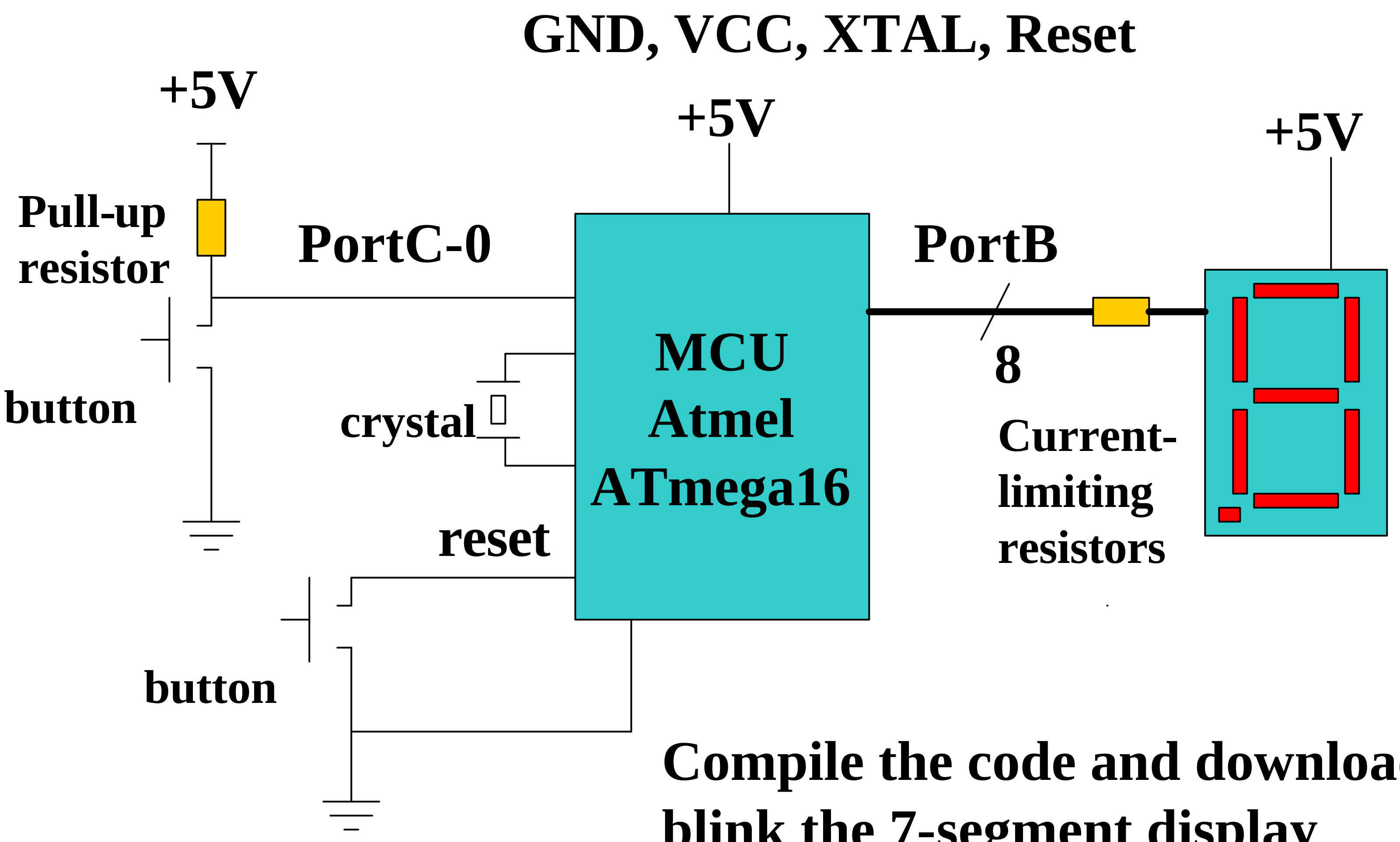
CSE466 Syllabus-1

- The course will focus on software issues in embedded systems including use of an advanced 8-bit microcontroller and its development environment, interrupt programming and management, and peripheral interfacing and drivers.
- Laboratory assignments will use prototyping boards, Personal Digital Assistants, LEDs, stepper motors, A/D converters, IrDA communications, and accelerometers.

CSE466 Syllabus-2

- Introduction: What is an Embedded System.
- AVR Development Tools
- Reading the AVR datasheet
- The Rule of (Ohm's) Law
- Memory spaces
- Timers, Interrupts,A/D converters
- Interrupts; Stepper motors
- Interrupt-driven Task Structures
- Accelerometers; Semaphores
- Control, Hysteresis & Feedback
- Pulse-width measurement
- Closing the loop
- Palm and IrDA
- Event-driven OS programming
- Noise & bypassing; Testability
- Debugging tools: Logic analyzer
- Pulse Width Modulation & DACs
- Safety, Ethics, and Societal Impact
- Design Trade-offs Memory, Speed, Power, Cost
- Serial Interfaces: SPI, I2C, USB

Lab1



Compile the code and download; blink the 7-segment display