CSE 466: Software for Embedded Systems

Autumn, 2003

- **Instructor: Bruce Hemingway**
  Office: CSE 464
  Office Hours: MW 11:30 A.M.-12:30 P.M.
  e-mail: bruceh@cs.washington.edu
  Phone: (206) 543-6274
- **Teaching Associate: Waylon Brunette**
- **Teaching Assistants: William Welbourne and Tom Anderl**
- **Class Meeting Times and Location:**
- **Lectures: MWF 9:30-10:20 A.M. LOWE 102**
  Lab: Tues. Section 1, 2:30- 5:20 P.M. CSE 003
  Thurs. Section 2, 2:30- 5:20 P.M. CSE 003

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

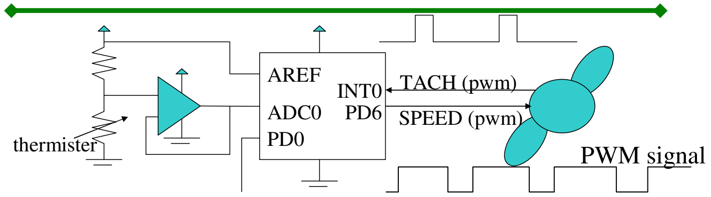
What is an Embedded System

- 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) PD0 = 1;
```

Task: FanPWM (periodic, hard constraint)

```
count++;
if (count == 0) PD6 = 1;
if (count > Thi) PD6 = 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
- 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 MAYBE– Final Group Project Participation Required
- Reading and Source Material assigned as needed

Business Matters

- Lecture slides will be handed out and 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…**
- Sign up for CSE466 mailing list

Grading

- Lab reports: Demo required, sometimes Report, sometimes hand-in code
- Ratios:
  - Lab: 50%
    Exams total: 40%
    Class and Lab 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, LEDs, audio transducers, A/D converters, pulse-width modulators, wireless communications, Berkeley Motes and TinyOS.
- **Required Readings:** We'll be using the Atmel ATmega16 microprocessor extensively, and will refer to the datasheet, which is found at: http://www.atmel.com/atmel/acrobat/doc2466.pdf It is 323 pages.
- Atmel has made available CD-ROMs of their data and application notes, which will be available for each student at the first lecture. The CD-ROM includes copies of the assembly tools as well.
- Hardbound copies of the datasheet as a coursepak are available at the Communication Copy Center in the Communications bldg, Rm B-042, cost $21.45.

CSE466 Syllabus-2

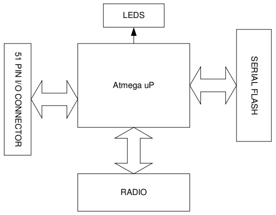
- Introduction: What is an Embedded System.
- AVR Development Tools
- Reading the AVR datasheet
- The Rule of (Ohm's) Law
- Timers, Interrupts,A/D converters
- Interrupt-driven Task Structures
- Pulse Width Modulation & DACs
- Table-driven Wave Synthesis
- Event-driven OS programming- TinyOS
- Wireless networking- using Motes
- Debugging tools: Logic analyzer
- Safety, Ethics, and Societal Impact
- Design Trade-offs Memory, Speed, Power, Cost
- Serial Interfaces: SPI, I2C, USB

Family of Motes

Mote Type	WeC	Renee	Mica	Mica2	Mica2Dot
Microcontroller					
Type	AT90LS8535	Atmega163	Atmega128	Atmega128	Atmega128
CPU Clock (Mhz)	4	4	4	7.3827	4
Program Memory (KB)	8	16	128	128	128
Ram (KB)	0.5	1	4	4	4
UARTs	1	1	2 (only 1 used)	2	2
SPI	1	1	1	1	1
I2C	Software	Software	Software	Hardware	Hardware
Nonvolatile storage					
Chip	24LC256		AT45DB041B		
Size (KB)	32		512		
Radio Communication					
Radio	RFM TR1000			Chipcon CC1000	
Frequency	916 (single freq)			916/433 (multiple channels)	
Radio speed (kbps)	OOK		ASK	FSK	
Transmit Power Control	Programmable resistor potentiometer			Programmable via CC1000 registers	
Encoding	SecDed (software)			Manchester (hardware)	

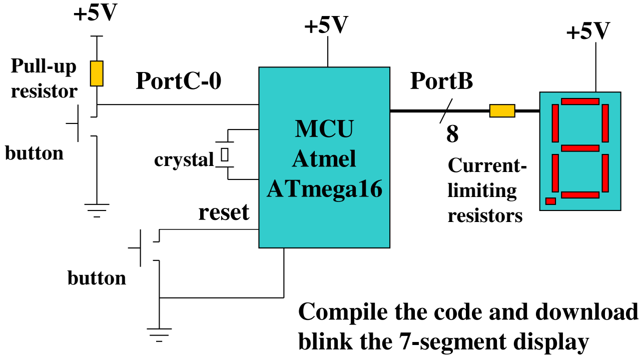
Common Platform Architecture

- Atmega uP
  - 32Khz crystal and 4Mhz (7.3728Mhz Mica2) crystal.
  - 10 bit ADC
  - UART (Mica2/Mica2Dot have 2)
  - SPI bus
  - I2C bus (hardware for mica2/mica2dot)
- Radio (Chipcon 1000)
- External serial flash memory (512K byte)
- Connectors for interfacing to sensor and programming boards
- 3 programmable leds (1 for Mica2Dot)
- JTAG port



Lab1

GND, VCC, XTAL, Reset



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