Supported Hardware Platforms [Recitation]

18-549: Embedded Systems Design

- Prof. Priya Narasimhan

Electrical & Computer Engineering Department
Carnegie Mellon University

Electrical&Computer ENGINEERING

Lecture material and readings are posted at
http://www.ece.cmu.edu/~ece549

Introduction

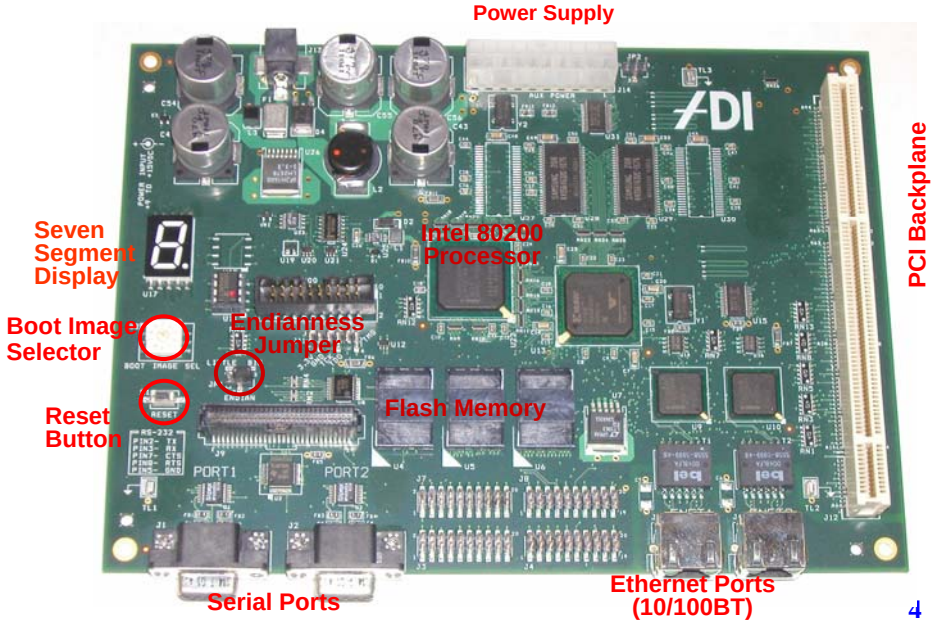
- TAs for Spring 2007
  - Adrian Ng (ECE IMB student)
  - Patrick Lanigan (ECE Grad. Student)
  - Jun Han (ECE IMB student)
  - Donnie Kim (ECE Grad. Student)
- Supported Platforms
  - Part I - XBoard
  - Part II - Telos Motes
- This recitation is being videotaped and will be posted at
  mms://wms.andrew.cmu.edu/001/recitation_1_22_07.wmv
- References
  - 18-349 Embedded Systems – Lecture #7 on X-Board
  - http://www.tinyos.net/scoop/special/support

Part I

Supported Platform I – The XBoard

3

Supported Platform I – The XBoard (1 of 2)



4

Supported Platform I – The XBoard (2 of 2)

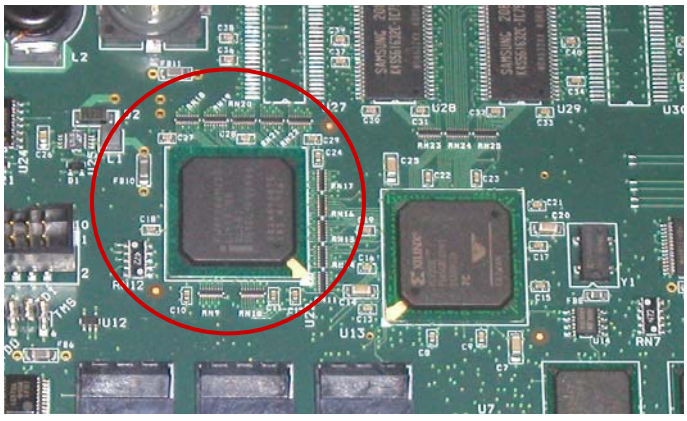
Hardware Specification	Value/Limit
Processor	XScale-based Intel 80200, 733 MHz
SDRAM Memory	128Mbyte, 100MHz
Flash Memory	4Mbyte, 100-150ns
Ethernet – 2 Ports	10/100 BaseT (untested)
Serial – 2 Ports	RS232, 115.2K
PCI	Single-slot, 64 bit, 33 MHz, 264 MBps
Debug Support	PROM-ICE Connector 80200 JTAG Connector

Conforms to the ARM architecture

5

XBoard – Intel 80200 Processor (1 of 3)

- Intel 80200 Processor



6

XBoard – Intel 80200 Processor (2 of 3)

- ARM ISA
  - RISC Architecture
- 32-bit Instructions (All instructions are of equal length)
- 8-bit / 32-bit data type
- Endianess
  - Optional – can be Big or Little Endian (By setting/unsetting a jumper)

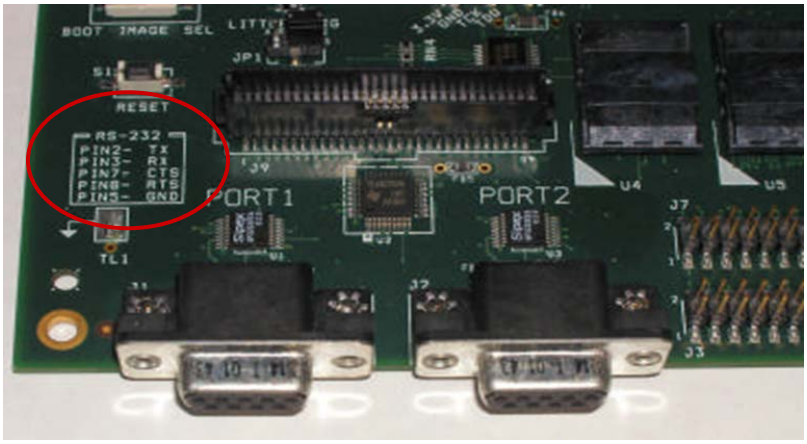
7

XBoard – Intel 80200 Processor (3 of 3)

- Registers
  - 13 General Purpose Registers (R0-R12)
  - Stack Pointer (R13)
  - Link Register (R14)
  - Program Counter (R15)
  - CPSR (Current Program Status Register)
    - Contains condition codes from the last ALU operation
    - Set processor execution mode
    - Enable/disable interrupts
  - SPSR (Saved Program Status Register)
    - Used by exception handler
- Exceptions
    - Reset, Undefined Instructions, SWI, IRQ
- Device Address Space Accesses
  - Memory-mapped I/O
    - In other words, standard Load and Store instructions can manipulate devices
      - E.g keypad, display

8

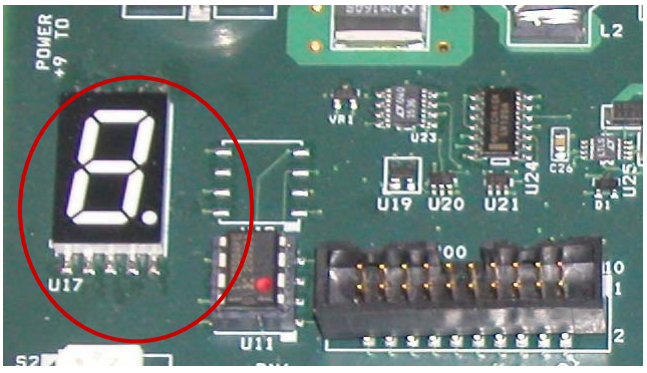
XBoard - Serial Port



- **Uses RS-232 standard**
- **Maximum Bit-Rate is 115.2kbps**

9

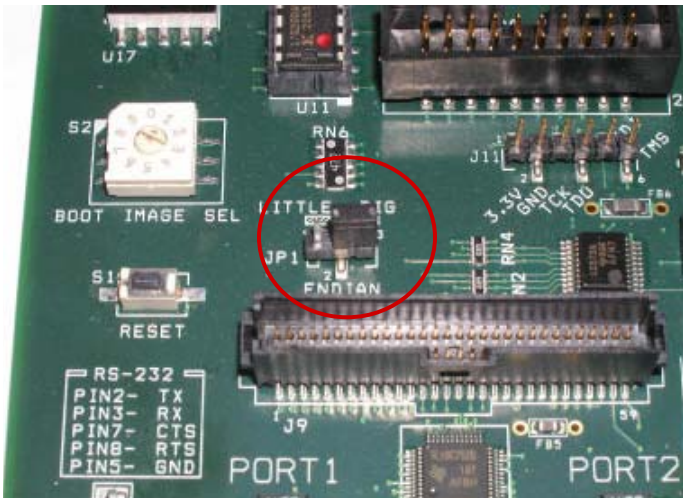
XBoard – Seven-Segment Display



- **Seven-Segment Display controlled by firmware**
- **Decimal Point Display not controlled by firmware**
- **Decimal Point Display is a power indicator connected to 3.3V rail**

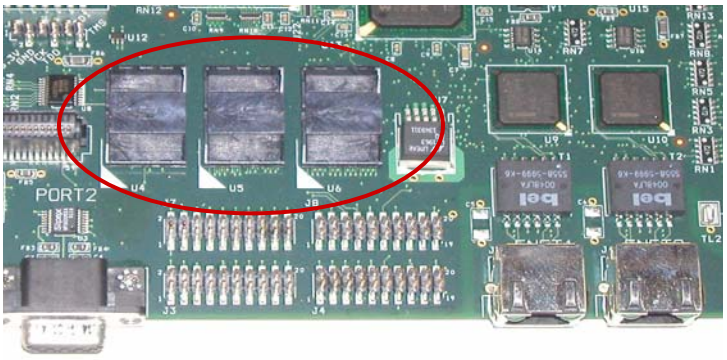
10

XBoard – Jumper for selecting Big/Little Endian



11

XBoard – Flash Memory



- 4MB of Flash Memory
- Managed by the Flash Library (to be discussed in the next slide)

12

Accessing Flash Memory (Flash Library)

- To save data persistently
- Flash Library Provided
  - *flash_lib.a*, *ActivateFlash.a*
- The Flash Library APIs allows you to:
  - Initialize/finalize the flash device
  - Identifies bad blocks on the device
  - Read from or write to flash memory location
    - E.g File processing

13

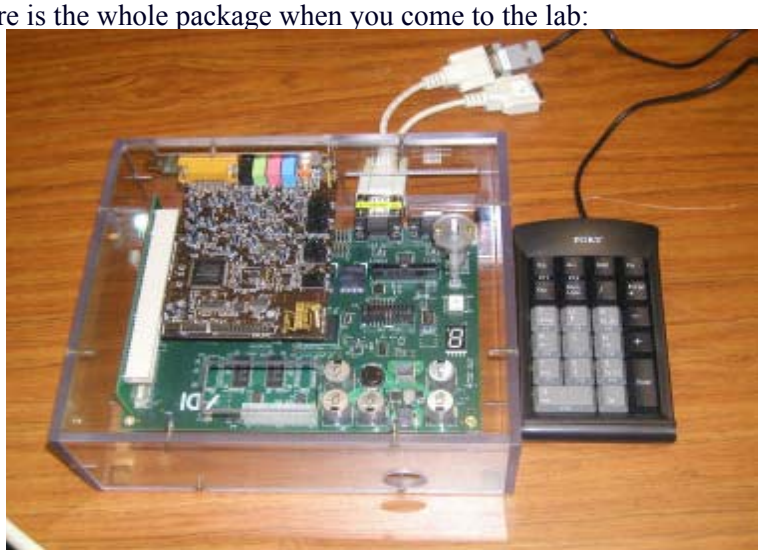
XBoard - Accessories

- Keypad
- Protecting Case
- Sound Card
- JTAG cable (for debugging purposes)

14

X-Board + Sound Card + KeyPad + JTAG

- Here is the whole package when you come to the lab:



15

Creating a New CodeWarrior Project

- Invoke the CodeWarrior IDE:
  - Start → Programs → ARM Developer Suite → CodeWarrior
- Create a new project:
  - File → New.
  - Choose the Project Tab.
  - Location – Choose new project's location.
  - Name – Choose appropriate project name.
- Add files to your project:
  - Choose Project → Add Files

16

Compiler Settings

- Choose Edit → DebugRel Settings → Language Settings
  - For ARM Assembler, C Compiler, C++ Compiler, Thumb C Compiler, and Thumb C++ Compiler select:
    - Architecture or Processor: ARM7TDMI
    - Floating Point: Pure-endian softfp
    - Byte Order: Big Endian

17

AXD (Debugger) Configuration

- Select ADP Target (ARM Processor).
  - Click Configure – check the following settings:
    - Select: ARM serial driver
    - Configure: Port – COM1, Baud Rate - 115200
    - Heartbeat: should be disabled
    - Channel Viewers: should be disabled
    - Endianness: Big
  - If the connections are all correct, AXD should now be able to identify your X-Board.

18

AXD (Debugger) Usage

- Go to File → Load Image
  - Find the directory containing your AXF (ARM image) file
  - Usually the DebugRel folder of the current project
- Go to: Execute → Run (or F5)
  - Click twice to run your program
- You can also set breakpoints, step through the execution, etc.

19

HyperTerminal Configuration

- Open the HyperTerminal windows application
  - Programs → Accessories → Communications → HyperTerminal
  - Try COM3 in the labs
  - COM port properties:
    - Bitrate: 1200 bps (you can experiment with higher bitrates)
    - Data bits: 8
    - Parity: none
    - Stop bits: 1
    - Flow control: None

20

Part II – Telos Motes

Supported Platform II – Telos Motes

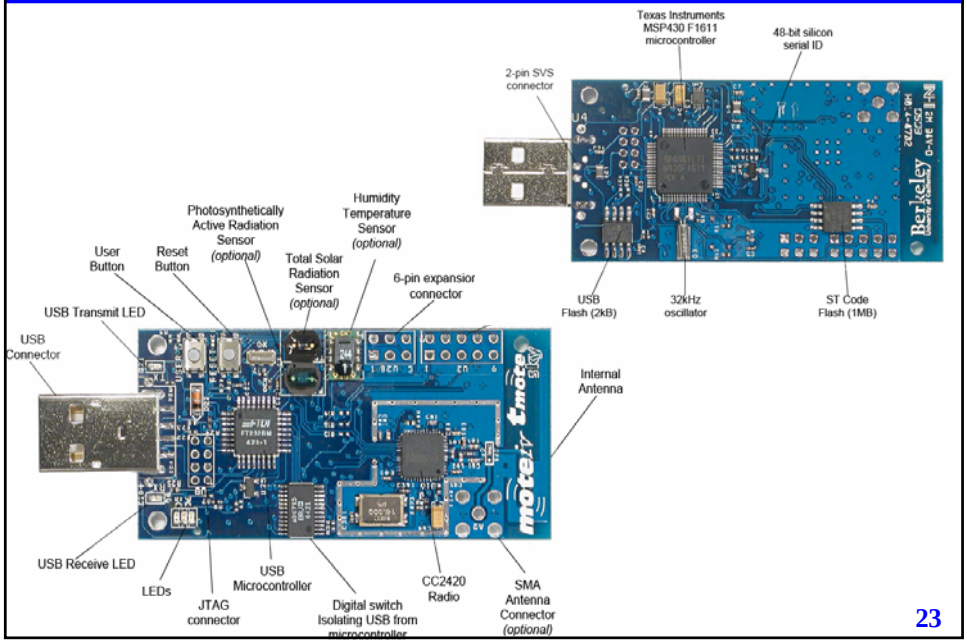
21

Part II – Telos Motes

Hardware Specifications

22

Supported Platform II – Telos Motes



23

Supported Platform II – Telos Motes

Hardware Specification	**Value/Limit**
Microcontroller	TI MSP430 F1611, up to 8 Mhz
Memory	10 Kbyte SRAM, 48 Kbyte flash
Storage	1 MB serial flash
Radio	Chipcon CC2420, 2.4 Ghz, 250 Kbps (IEEE 802.15.4)
On-board sensors	Humidity, temperature, light
Serial interface	USB (FTDI drivers)
Debug support	Optional JTAG connector (untested)

24

Supported Platform II – Telos Motes

- Microprocessor - TI MSP430F1611
  - Memory
    - 48kB+256B Flash Memory (ROM)
    - 10kB RAM
  - 16bit RISC architecture, 125 nsec Instruction Cycle Time
  - Low Supply-Voltage Range, 1.8V ~ 3.6V
  - Power (Ultralow-Power Consumption)
    - Active Mode : 330uA @ 1MHz, 2.2V
    - Standby Mode: 1.1 uA
    - Off Mode : 0.2uA
  - 12-Bit A/D Converters / 12-Bit D/A Converters
  - Two 16-Bit Timers (Timer_A, Timer_B)
  - Two Serial Communication Interfaces (USART0, USART1) for either Asynchronous UART or Synchronous SPI
  - Supply Voltage Supervisor/Monitor
  - Three-Channel Internal DMA

25

Supported Platform II – Telos Motes

memory organization (MSP430F161x)

		MSP430F1610	MSP430F1611	MSP430F1612
Memory Main: interrupt vector Main: code memory	Size Flash Flash	32KB 0FFFFh – 0FFE0h 0FFFFh – 08000h	48KB 0FFFFh – 0FFE0h 0FFFFh – 04000h	55KB 0FFFFh – 0FFE0h 0FFFFh – 02500h
RAM (Total) Extended Mirrored	Size Size Size	5KB 024FFh – 01100h 3KB 024FFh – 01900h 2KB 018FFh – 01100h	10KB 038FFh – 01100h 8KB 038FFh – 01900h 2KB 018FFh – 01100h	5KB 024FFh – 01100h 3KB 024FFh – 01900h 2KB 018FFh – 01100h
Information memory Boot memory	Size Flash Size ROM	256 Byte 010FFh – 01000h 1KB 0FFFh – 0C00h	256 Byte 010FFh – 01000h 1KB 0FFFh – 0C00h	256 Byte 010FFh – 01000h 1KB 0FFFh – 0C00h
RAM (mirrored at 018FFh - 01100h)	Size	2KB 09FFh – 0200h	2KB 09FFh – 0200h	2KB 09FFh – 0200h
Peripherals	16-bit 8-bit 8-bit SFR	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h

26

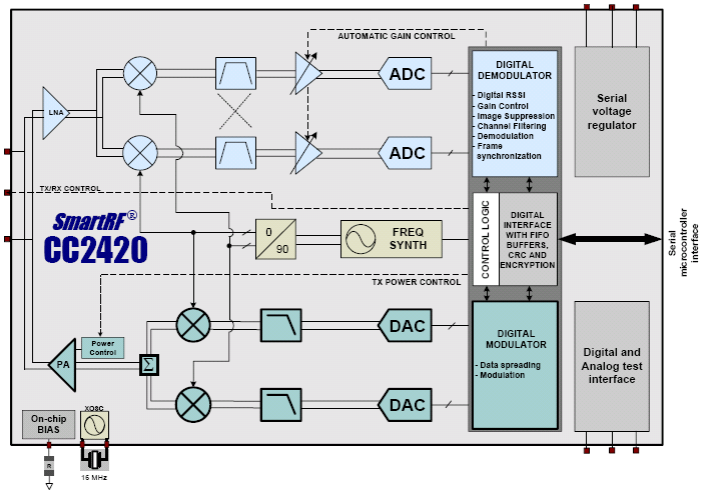
Supported Platform II – Telos Motes

- Chipcon CC2420
  - Radio Frequency Chip
  - 2.4 GHz IEEE 802.15.4 System
  - Low Supply Voltage
    - 2.1 – 3.6 V (with integrated voltage regulator)
    - 1.6 – 2.0 V (with external voltage regulator)
  - Low Current Consumption
    - RX Max: 19.7 mA
    - TX Max: 17.4 mA
  - Adjustable Power Level
  - Data Rate: 250kbps
  - 128B for RX and TX data buffering
  - Digital RSSI/LQI support

27

Supported Platform II – Telos Motes

- Chipcon CC2420



28

Supported Platform II – Telos Motes

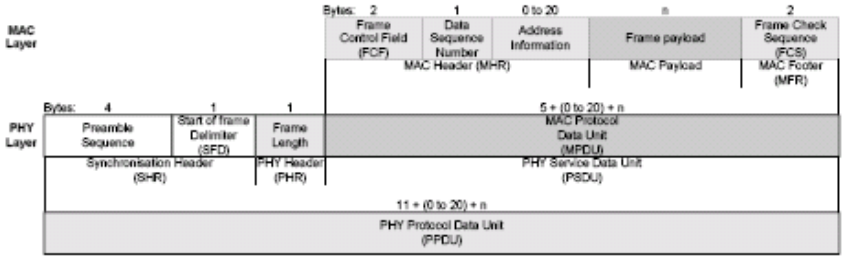


Figure 17. Schematic view of the IEEE 802.15.4 Frame Format [1]

29

Part II – Telos Motes

Programming Telosb (Tmote) Motes

30

Installing Telos Motes and TinyOS

- Moteiv's Boomerang TinyOS installer
  - Download the Boomerang Software from *http://www.moteiv.com/software/*
  - Automatically Installs:
    - Cygwin
    - Java
    - Moteiv's distribution of the TinyOS that includes extended API support, example applications, and additional development tools
    - USB Serial COM Driver
- Manually installing components in Windows
  - http://www.moteiv.com/community/Tmote_Windows_install
- Manually installing components in Linux
  - http://www.moteiv.com/community/Tmote_Linux_install

31

Learning TinyOS and NesC Syntax

- To start learning TinyOS and NesC:
  - Getting Started
    - http://www.tinyos.net/scoop/special/support
    - Follow the tutorials linked at the above website
  - NesC language
    - A C-like language for programming structured component-based applications
    - The TinyOS system, libraries, and applications are written in nesC
    - http://nescc.sourceforge.net/
  - Search for help
    - http://www.mail-archive.com/tinyos-help@millennium.berkeley.edu/
    - Also, Google is very useful
  - Library documents
    - Boomerang provides a nice library documentation
    - Ex. C:\cygwin\opt\moteiv\doc\nesdoc\index.html

32

Getting Started

- Checking mote connectivity
  - Boomerang provides an executable that shows list of connected Tmote devices
  - type `motelist`
    - Which is located at `/usr/local/bin/motelist`
- Directory for example applications
  - */opt/moteiv/apps/*
    - Moteiv's Tmote specific example applications
    - Use `make tmote`
  - /opt/tinyos-1.x/apps/
    - Example applications provided by TinyOS community
    - Use `make telosb`

33

TinyOS Directory Structure

- Within </opt/tinyos-1.x>

```
/apps
    /Blink
    /CntToLedsAndRfm
    …
/tools
    /java
    …
/tos
    /interface
    /lib
    /platform
        /mica2
        /telos
        /pc
        …
    /sensorboards
    /systems
    /types
```

34

The make System

- In the application's directory (i.e. /apps/Blink)
  - make (re)install, <node id> <platform>
    - <node id> is an integer between 0 and 255 (16-bit value)
    - <platform> may be mica2, tmote, or pc
    - make <platform>
      - Compiles the application for <platform>
    - make install <platform>
      - Compiles and downloads the application to the mote
    - make reinstall <platform>
      - Programs the module with the already compiled binary image
  - make clean
  - make pc
    - Generates an executable that can be run on a PC for simulation
    - Handy for debugging

35

Compiling and Downloading Applications

- Demo:

36

Debugging Tips

- Develop applications under a private directory
  - Ex. /opt/tinyos-1.x/myProject
- Debug with TOSSIM
  - http://www.cs.berkeley.edu/~pal/pubs/nido.pdf
  - Use dbg statements as printf statements
    - `dbg(DBG_USR1, “testing\n”);`
  - `export DBG=usr1`
- Debug with LEDs
- Subscribe and/or search TinyOS mailing lists
  - Many questions have already been answered by others.

37

Debugging Tips

- Demo: Running TOSSIM

38