Microprocessor Fundamentals

Topic 1

Microprocessor Fundamentals
Objectives

• Understand the difference between microprocessors, microcontrollers, and microcomputers
• Review of number systems and conversions
• Understand the development process
• Understand the numbering system and selection of various AVR ICs
• Examine some Internet resources
Microprocessor

• A Microprocessor is a single integrated circuit (IC) that accepts and executes coded instructions (machine code or machine language) for the purpose of manipulating data and controlling the associated circuitry (RAM, ROM, and I/O ICs) in a digital system.
A Microcomputer is an integrated system of computer components typically containing data storage ICs, mass storage elements, and standard I/O devices, capable of a wide variety of applications. Microcomputers contain a microprocessor at the heart of the system, controlling data flow, manipulating data, and executing instructions.

- Data storage ICs: RAM, ROM, and I/O
- Mass storage elements: External Drives, Hard Drives, CD-ROM Drives
- Standard I/O devices: High resolution monitor, keyboard, mouse, printer, etc.
Microcontroller

- **A Microcontroller** is also a single integrated circuit that accepts and executes coded instructions for the purpose of manipulating data and controlling a digital system similar to a microprocessor.

- The difference between a microcontroller and a microprocessor is that the microcontroller also contains RAM, ROM, and I/O circuitry in that single IC package.

- This allows miniaturization of single application, microprocessor controlled, digital systems because the required associated circuitry is contained within the integrated circuit of a microcontroller.
Microcontroller

- Its construction and integration of common computer circuits make the microcontroller ideal for single function, programmable, control systems such as those found in:
  - Microwave Ovens, Dishwashers, Washing Machines
  - Cell Phones, iPODs, other MP3 devices
  - PDAs, watches
  - Automobiles, GPS Systems
  - House Alarm Systems,
  - and many other household and industrial applications.
A “short list” of embedded systems

- Anti-lock brakes
- Auto-focus cameras
- Automatic teller machines
- Automatic toll systems
- Automatic transmission
- Avionic systems
- Battery chargers
- Camcorders
- Cell phones
- Cell-phone base stations
- Cordless phones
- Cruise control
- Curbside check-in systems
- Digital cameras
- Disk drives
- Electronic card readers
- Electronic instruments
- Electronic toys/games
- Factory control
- Fax machines
- Fingerprint identifiers
- Home security systems
- Life-support systems
- Medical testing systems
- Modems
- MPEG decoders
- Network cards
- Network switches/routers
- On-board navigation
- Pagers
- Photocopiers
- Point-of-sale systems
- Portable video games
- Printers
- Satellite phones
- Scanners
- Smart ovens/dishwashers
- Speech recognizers
- Stereo systems
- Teleconferencing systems
- Televisions
- Temperature controllers
- Theft tracking systems
- TV set-top boxes
- VCR’s, DVD players
- Video game consoles
- Video phones
- Washers and dryers

And the list goes on and on
Embedded systems are everywhere

- In 2000, more than a quarter billion, 8-bit embedded processors were being sold each month (http://www.extremetech.com/article2/0,3973,18917,00.asp)
Embedded systems are everywhere

• The average middle-class American household has about 40 to 50 microprocessors in it - plus another 10 processors for every PC
  (http://www.extremetech.com/article2/0,3973,18917,00.asp)

• There's a microprocessor in your:
  • The microwave oven
  • The washer, dryer, and dishwasher
  • Color TV and another one in the remote control
  • The VCR (and its remote)
  • Your stereo receiver, CD player, and DVD player An automatic garage door opener (and each remote control for it) also contains a microprocessor
Embedded systems are everywhere

- There's a microprocessor in your:
  - An automatic garage door opener (and each remote control) also contains a microprocessor
  - The average new car has a dozen
    - BMW 7-series has 63
    - Mercedes S-class has 65
  - Every modern car has electronic ignition
    - Ford, Jaguars and Volvos, use a PowerPC to control the engine
  - Automatic transmissions and Antilock brakes are microprocessor controlled as well
Embedded systems are everywhere

• Got a Volvo?
  – The processor in its automatic transmission communicates with the processors behind each side-view mirror.
  – Allows the outside mirrors to automatically tilt down and inward whenever you put car into reverse gear, the better to see the back end of the car
Embedded systems are everywhere

- The Additional Processors in Your PC
  - There's an 8-bit processor in your keyboard
  - Another processor in your mouse
  - There’s one in each hard disk drive and floppy drive (if you still have one)
  - One in your CD-ROM
  - A big one in your graphics accelerator
  - A CPU buried in your USB interface
  - A processor handling your NIC
Common characteristics of embedded systems

- **Single-functioned**
  - Executes a single program, repeatedly
- **Tightly-constrained**
  - Low cost, low power, small, fast, etc.
- **Reactive and real-time**
  - Continually reacts to changes in the system’s environment
  - Must compute certain results in real-time without delay
- **Programmed in C or Java**
NUMBER SYSTEMS
Number Systems

• So, why do we need to understand the different numbering systems and how to convert between them?
  – Humans (including programmers and engineers) think in decimal (at least early on in their education they do)
  – When working with I/O to control devices, we are looking at individual bits or bit streams
  – Large binary numbers are difficult to read, its easier to use some form of short-hand: hexadecimal (and sometimes octal)
# Number Systems

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Binary</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00000000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>00000001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>00000010</td>
<td>002</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>00000011</td>
<td>003</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>00000100</td>
<td>004</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>00000101</td>
<td>005</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>00000110</td>
<td>006</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>00000111</td>
<td>007</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>00001000</td>
<td>010</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>00001001</td>
<td>011</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>00001010</td>
<td>012</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>00001011</td>
<td>013</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>00001100</td>
<td>014</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>00001101</td>
<td>015</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>00001110</td>
<td>016</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>00001111</td>
<td>017</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>00010000</td>
<td>020</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>00010001</td>
<td>021</td>
</tr>
</tbody>
</table>
Conversions

- Decimal to Hexadecimal:
  Ex: convert $40_{10}$ to hex:
  We only need a 2 digit number to express 40 in hex:
  $40 \div 16 = 2.5$ or $2 \text{ r} 8$
  $8$ is too small to be divided by 16: $28$
Conversions

• Decimal to Hexadecimal:
  Ex: convert $210_{10}$ to hex:
Conversions

- Decimal to Hexadecimal:
  Ex: convert $210_{10}$ to hex:

  We only need a 2 digit number to express 210 in hex:

  $$210 \div 16 = 13.125 \text{ or } 13 \text{ r}2$$

  (13 in decimal is D in hex) \hspace{1cm} D-

  2 is too small to be divided by 16: \hspace{1cm} D2
Conversions

- Hexadecimal to Decimal:
  Ex: convert E7 to decimal:
  \[ E \times 16^1 + 7 \times 16^0 = \]
  \[ 14 \times 16^1 + 7 \times 16^0 = \]
  \[ 224 + 7 = 231 \]
Conversions

• Hexadecimal to Decimal:
  Ex: convert 2C to decimal:
Conversions

• Hexadecimal to Decimal:
  Ex: convert 2C to decimal:
  $2 \times 16^1 + C \times 16^0 =$
  $2 \times 16^1 + 12 \times 16^0 =$
  $32 + 12 = 44$
Conversions

• Decimal to Binary:

Ex: convert $40_{10}$ to binary:

\[
\begin{align*}
40 \div 2 &= 20, \text{ r } 0 & 0 \\
20 \div 2 &= 10, \text{ r } 0 & 00 \\
10 \div 2 &= 5, \text{ r } 0 & 000 \\
5 \div 2 &= 2, \text{ r } 1 & 1000 \\
2 \div 2 &= 1, \text{ r } 0 & 01000 \\
1 \div 2 &= 0, \text{ r } 1 & 101000
\end{align*}
\]
Conversions

- Decimal to binary:
  Ex: convert $155_{10}$ to binary:
Conversions

- Decimal to binary:

  Ex: convert 155₁₀ to binary:

  155 ÷ 2 = 77, r 1
  77 ÷ 2 = 38, r 1
  38 ÷ 2 = 19, r 0
  19 ÷ 2 = 9, r 1
  9 ÷ 2 = 4, r 1
  4 ÷ 2 = 2, r 0
  2 ÷ 2 = 1, r 0
  1 ÷ 2 = 0, r 1

  155₁₀ = 10011011₂
Conversions

• Binary to Decimal:

Ex: convert 10011101 to decimal:

$1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$

$128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 = 157$
Conversions

• Binary to Decimal:
  Ex: convert 11101101 to decimal:
Conversions

• Binary to Decimal:

Ex: convert 11101101 to decimal:

\[1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =
\]

\[128 + 64 + 32 + 0 + 8 + 4 + 0 + 1 = \]

237
Conversions

• Hex to binary:
  Ex: convert $3F_{16}$ to binary:
  $3 = 0011$,  $F = 1111$
  $3F = 00111111$
Conversions

- Hex to binary:
  Ex: convert $F7_{16}$ to binary:
Conversions

- Hex to binary:
  Ex: convert $F7_{16}$ to binary:
  $F = 1111$, $7 = 0111$
  $F7 = 11110111$
Conversions

• Binary to Hex:

Ex: convert 10011101 to hex:
1001 = 9, 1101 = D
10011101 = 9D
Conversions

- Binary to Hex:
  Ex: convert 11101101 to hex:
Conversions

• Binary to Hex:

Ex: convert 11101101 to hex:
1110 = E, 1101 = D
11101101 = ED
Conversions

- Conversions:
  - You need to be able to do this with and without a calculator
  - Many conversions have to be done quickly, on-the-fly
  - Used to verify data or I/O
DEVELOPMENT PROCESS
Development Process

1. Gather and verify all project requirements and specifications
2. Layout a plan for your program
   1. Pseudo-code, algorithm, or flowchart (maybe all 3)
3. Choose a target processor or controller
4. Write the program using a text editor or an Integrated Development System (IDS)
5. Assemble the program
   1. Remove syntax errors and re-assemble until all errors are removed

Continued……
Development Process

6. Simulate the program in the IDS
   1. Remove all logic errors
   2. Go to step 4 and repeat 4 & 5 until all logic errors are removed

7. Program the AVR
   1. Remove errors:
      1. Example: Target range of addresses could be incorrect
      2. Example: Expected support ICs may be missing
      3. Go to step 4 and repeat 4, 5, & 6 until all errors are removed

8. Test the program on the hardware
   1. Remove errors:
      1. Example: I/O port addresses may be incorrect
      2. Example: Logic for interfaced hardware may be incorrect
      3. Example: Timing may be incorrect
      4. Go to step 4 and repeat 4, 5, 6, & 7 until all errors are removed

* The various books for this class have slightly different steps
Choosing an AVR

- The IC number tells us some things (but not all):

  AT90S1200

  - SRAM memory size
  - CPU Model #
  - EEPROM data memory size
  - 1 kB of Flash Program memory

Memory Sizes (SRAM and EEPROM)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1k</td>
<td>2k</td>
<td>4k</td>
<td>8k</td>
<td>16k</td>
<td>32k</td>
</tr>
</tbody>
</table>

May also need to know the number of total pins, the pinout of the IC, the number of I/O pins (and registers), other features
Choosing an AVR

• May also need to know:
  – The number of total pins,
  – The pinout of the IC,
  – The number of I/O pins (and registers),
  – And other features

• Have to look it up
  – Good place to start:
    – Appendix A in Morton book
    – http://www.avrfreaks.net/
    – http://www.atmel.com
In this topic we:

- Discussed the difference between microprocessors, microcontrollers, and microcomputers
- Reviewed number systems and conversions
- Discussed the development process
- Discussed the numbering system and selection of various AVR ICs
- Examined some Internet resources