Embedded System Design Laboratory

A Few Words From Dilbert

I Didn't know how to design a power supply, so I put a nail in a piece of wood.

I'm on vacation tomorrow, so I'll give you my files in case you need to make changes.

Once I had the idea, it all came together pretty quickly.

October 2, 2002

Stanford University - EE281 Lecture #2
Lecture #2 Outline

• Status Check
• AVR Processor Resources
  – Interrupts
  – Timers
• Extra Lab Time?
Status Check

• How is Lab #1 going?
• Got access to the EE281 lab yet?
• More STK500 kits are coming…
AVR Processor Resources

- Interrupts
- Timers
- UART (Universal Asynchronous Receiver/Transmitter)
- A/D Converters (Analog to Digital)
- SPI (Serial Peripheral Interface)
- Analog Comparator
General Purpose Ports
- PORTA
- PORTB
- PORTC
- PORTD
- (Special Functions)

Special Purpose Pins
- Crystal (XTAL1/XTAL2)
- RESET
- ICP, OLE, OC1B

Power (VCC/GND)
• 32 Registers (R0-R31)
• 4K Prog ROM
• 512 bytes RAM
• 512 bytes EEPROM
• 32 I/O lines
• 13 Interrupts
• Lots of fun built-in peripherals
Interrupts

- Interrupts halt normal code execution in order to go do something more important or time sensitive
- Interrupt “Handlers”
  - Using the Interrupt Vectors
- Interrupts are used for:
  - RESET
  - Timers and Time-Critical Code
  - Hardware signaling
    - “I’m done”
    - “Something’s happened that you want to know about”
    - “I have something for you”
## Interrupt Vectors

### Table 2. Reset and Interrupt Vectors

<table>
<thead>
<tr>
<th>Vector No.</th>
<th>Program Address</th>
<th>Source</th>
<th>Interrupt Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$000</td>
<td>RESET</td>
<td>External Reset, Power-on Reset and Watchdog Reset</td>
</tr>
<tr>
<td>2</td>
<td>$001</td>
<td>INT0</td>
<td>External Interrupt Request 0</td>
</tr>
<tr>
<td>3</td>
<td>$002</td>
<td>INT1</td>
<td>External Interrupt Request 1</td>
</tr>
<tr>
<td>4</td>
<td>$003</td>
<td>TIMER1 CAPT</td>
<td>Timer/Counter1 Capture Event</td>
</tr>
<tr>
<td>5</td>
<td>$004</td>
<td>TIMER1 COMPA</td>
<td>Timer/Counter1 Compare Match A</td>
</tr>
<tr>
<td>6</td>
<td>$005</td>
<td>TIMER1 COMPB</td>
<td>Timer/Counter1 Compare Match B</td>
</tr>
<tr>
<td>7</td>
<td>$006</td>
<td>TIMER1 OVF</td>
<td>Timer/Counter1 Overflow</td>
</tr>
<tr>
<td>8</td>
<td>$007</td>
<td>TIMER0, OVF</td>
<td>Timer/Counter0 Overflow</td>
</tr>
<tr>
<td>9</td>
<td>$008</td>
<td>SPI, STC</td>
<td>Serial Transfer Complete</td>
</tr>
<tr>
<td>10</td>
<td>$009</td>
<td>UART, RX</td>
<td>UART, Rx Complete</td>
</tr>
<tr>
<td>11</td>
<td>$00A</td>
<td>UART, UDRE</td>
<td>UART Data Register Empty</td>
</tr>
<tr>
<td>12</td>
<td>$00B</td>
<td>UART, TX</td>
<td>UART, Tx Complete</td>
</tr>
<tr>
<td>13</td>
<td>$00C</td>
<td>ANA_COMP</td>
<td>Analog Comparator</td>
</tr>
</tbody>
</table>
Interrupts: Code Example

; setup reset/interrupt vectors
.cseg
.org 0x000
rjmp reset ; $000 HW Reset or Watchdog Handler
rjmp reset ; $001 External IRQ 0 Handler
rjmp reset ; $002 External IRQ 1 Handler
rjmp reset ; $003 Timer/Counter1 Capture Event Handler
rjmp reset ; $004 Timer/Counter1 Compare Match A Handler
rjmp reset ; $005 Timer/Counter1 Compare Match B Handler
rjmp reset ; $006 Timer/Counter1 Overflow Handler
rjmp Timer0Isr ; $007 Timer/Counter0 Overflow Handler
rjmp reset ; $008 SPI Serial Transfer Complete Handler
rjmp reset ; $009 UART Rx Complete Handler
rjmp reset ; $00A UART Data Register Empty Handler
rjmp reset ; $00B UART Tx Complete Handler
rjmp reset ; $00C Analog Comparator Handler

; begin code

reset: ; your main code goes here

Timer0Isr: ; Timer0 overflow interrupt code here
   RETI ; don’t forget to return from your interrupt!
Timers: Why we need them

- Provide accurately timed delays or actions independent of code execution time

- How are Timers used?
  - Accurate delay
    - Read the timer, store value as K. Loop until timer reaches K+100.
  - Schedule important events
    - Setup an Output Compare to trigger an interrupt at a precise time
  - Measure time between events
    - When event#1 happens, store timer value as K
    - When event#2 happens, read timer value and subtract K
    - The difference is the time elapsed between the two events
AVR Timer/Counter 0

• 8 Bit Up Counter
  – counts from 0 to 255 (0xFF), then loops to 0
  – Internal or External Clock source
    • Prescaler

• Interrupt on Overflow
  – Transition from 255 to 0 can trigger interrupt if desired
AVR Timer/Counter 0 (cont’d)
AVR Timer/Counter 1

- 16 Bit Up Counter
  - Counts from 0 to 65535 (0xFFFF), then loops
  - Internal clock source with prescaler or External Clock
- Dual Comparators
- Interrupts possible on:
  - Overflow
  - Compare A/B
  - Input Capture of external event on ICP pin
- Can also act as an 8, 9 or 10 bit PWM Up-Down Counter
AVR Timer/Counter 1 (cont’d)
The 8515 has two output compares (OCR1A/B)

- OCR1A/B are 16-bit registers
- When the value of OCR1x matches that of Timer1:
  - A user-defined action can take place on the OC1x pin (set/clear/inv)
  - An interrupt can be triggered
  - Timer1 can be cleared to zero
- Once set up, output compares operate continuously without software intervention
- Great for:
  - Precise recurring timing
  - Frequency/Tone generation (maybe sound effects)
  - All kinds of digital signal generation
    - Infrared communications
    - Software-driven serial ports
Timer 1 and PWM

- Pulse-Width Modulation
  - Useful for using digital circuits to achieve analog-like control of motors, LEDs, etc
  - Timer 1 has two channels of PWM output on OCR1A and OCR1B
Timer Control: I/O space

• Timer 0
  – Timer/Counter0 (TCNT0)
  – Control Register (TCCR0)

• Timer 1
  – Timer/Counter1 (TCNT1)
  – Control Register A & B (TCCR1A/B)
  – Input Capture Register (ICR1)
  – Timer/Counter1 Output Compare Register A and B (OCR1A/B)

• Timer Interrupt Registers
  – Timer Interrupt Mask Register (TIMSK)
  – Timer Interrupt Flag Register (TIFR)
  – Common to Both Timers
Timer/Counter Clock Sources

- Prescaler
  - Shut Off
  - Divided System Clock
  - External Input (rising or falling)

<table>
<thead>
<tr>
<th>CS02</th>
<th>CS01</th>
<th>CS00</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Stop, the Timer/Counter0 is stopped.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CK</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CK/8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>CK/64</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>CK/256</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>CK/1024</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>External Pin T0, falling edge</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>External Pin T0, rising edge</td>
</tr>
</tbody>
</table>
Timer: Example Code

- Timer0.asm
  - Gives a complete example of one way to use timer 0 with a timer interrupt handler
  - Heavily commented
  - Highlights helpful coding practices for all programs
    - Use .equ to define constants
    - Use .def to define register “nicknames”
  - Available on the course website