Debugging Interrupts

- Debugging systems with many interrupts can be difficult.
- The primary need is for visibility into what is happening.
- By using a primitive 3-bit DAC and some debug code we can determine how often interrupts occur and how long it takes for them to run.
- By commenting out some code you can also tell if a interrupt is hung in an endless loop.
The DAC circuit is shown below. This example is for use on the ATMega128 board supplied by OSU.

The output voltage of this DAC is:

\[ V_{out} = V_{ref} \times \frac{bit\_value}{2^{number\_of\_bits}} \]
Since we have 3 bits and our reference voltage is 5 volts, the output voltages we can obtain with different bit values is:

\[ V_{out} = 5 \times \frac{\text{bit value}}{8} \]

<table>
<thead>
<tr>
<th>Bit value</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0V</td>
</tr>
<tr>
<td>001</td>
<td>0.625V</td>
</tr>
<tr>
<td>010</td>
<td>1.25</td>
</tr>
<tr>
<td>011</td>
<td>1.875</td>
</tr>
<tr>
<td>100</td>
<td>2.5</td>
</tr>
<tr>
<td>101</td>
<td>3.125</td>
</tr>
<tr>
<td>110</td>
<td>3.75</td>
</tr>
<tr>
<td>111</td>
<td>4.375</td>
</tr>
</tbody>
</table>
Debugging Interrupts

- In our code, we associate a bit value and thus a voltage for each interrupt.
- When we are executing in `main()` the output voltage is zero.

```c
//3-bit DAC binary values
#define TCNT0_ISR 0x01
#define TCNT1_ISR 0x02
#define TCNT3_ISR 0x03
#define ADC_ISR 0x04
#define TWI_ISR 0x05
#define USART0_ISR 0x06
#define NOT_IN_ISR 0xF8 //reset lower three bits to zero
```
# Debugging Interrupts

- We get distinct voltages for each interrupt.

<table>
<thead>
<tr>
<th>Bit value</th>
<th>Output Voltage</th>
<th>Running Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0V</td>
<td>main()</td>
</tr>
<tr>
<td>001</td>
<td>0.625V</td>
<td>TCNT0 ISR</td>
</tr>
<tr>
<td>010</td>
<td>1.25</td>
<td>TCNT1 ISR</td>
</tr>
<tr>
<td>011</td>
<td>1.875</td>
<td>TCNT3 ISR</td>
</tr>
<tr>
<td>100</td>
<td>2.5</td>
<td>ADC ISR</td>
</tr>
<tr>
<td>101</td>
<td>3.125</td>
<td>TWI ISR</td>
</tr>
<tr>
<td>110</td>
<td>3.75</td>
<td>USART0 ISR</td>
</tr>
<tr>
<td>111</td>
<td>4.375</td>
<td>unused</td>
</tr>
</tbody>
</table>
Debugging Interrupts

- We use one variable to determine if the interrupt debug code is used.

```c
// define to show interrupts via 3-bit DAC
#define SHOW_INTERRUPTS
```

- The beginning and end of each ISR is bracketed with debug statements.

```c
#if define SHOW_INTERRUPTS
    PORTG |= (USART0_ISR); // set interrupt value to this ISR
#endif

// ISR code goes here

#if define SHOW_INTERRUPTS
    PORTG &= NOT_IN_ISR; // set interrupt value back to zero
#endif
```
For example, the ADC ISR looks like this.

```c
ISR(ADC_vect){
  #ifdef SHOW_INTERRUPTS
    PORTG |= (USART0_ISR); // set interrupt value to this ISR
  #endif

  OCR2 = ADCH;            // update pwm dimmer
  adc_flag = ADC_DONE;    // set done flag

  #ifdef SHOW_INTERRUPTS
    PORTG &= NOT_IN_ISR;  // set interrupt value back to zero
  #endif
}
// ADC_vect_ISR
```
Debugging Interrupts

- Five different interrupts, their periods and duration seen here. Plenty of time for `main()` and other interrupts to run. This is a good picture.
Debugging Interrupts

- You can determine relative or absolute ISR run times if you zoom in.
Debugging Interrupts

- ISR for TCNT0 (0.625V) ran 34.6µS, USART0 ISR (3.77V) ran 2µS.