Unit B - Rotary Encoders
Rotary Encoders

- Electromechanical devices used to measure the angular position or rotation of a shaft.
- Two types:
  - Absolute: Output a binary number for the current angular position of the shaft.
    - $0000 = 0^\circ$, $0001 = 22.5^\circ$, $0010 = 45^\circ$, etc.
  - Incremental: Outputs signals that indicate a change in angular position and the direction of rotation.
- Many uses in controlling mechanical devices
  - Scanners, printers, mice, robots, manufacturing equipment, etc.
Rotary Encoders

- Incremental encoders produce **quadrature** outputs
- Output is two square waves, 90° out of phase, as the device is rotated
- By examining the state of the two outputs at the transitions, we can tell which way it’s being rotated.

![Diagram of A and B signals for clockwise and counter-clockwise rotation]
Rotary Encoders

- If $B = 0$ when $A \uparrow \Rightarrow$ Clockwise
- If $B = 0$ when $A \downarrow \Rightarrow$ Counter clockwise
- If $A = 1$ when $B \uparrow \Rightarrow$ Clockwise
- If $A = 1$ when $B \downarrow \Rightarrow$ Counter clockwise
Rotary Encoders

- Can implement this as a state machine

Rotating clockwise

Rotating counter-clockwise
Gray Codes

• The two bit output sequence is a “Gray Code”.
  – Each adjacent element differs by only one bit.

• In normal binary codes, multiple bits change from one code to the next (011→100)

• Impossible for hardware to make sure all the bits change at the same time.

• Gray codes are used with many electromechanical devices.

<table>
<thead>
<tr>
<th>3-Bit Binary</th>
<th>3-Bit Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 0 0</td>
</tr>
</tbody>
</table>
Rotary Encoders

- Encoder has three terminals
  - A, B and common
- As it rotates the two switches open and close
- Ones used in our lab have 64 states per revolution
- Must have pull-up resistors on switch outputs
Rotary Encoder Lab

• Write a program that monitors the two inputs from the encoder and increments or decrements a count value each time the encoder changes state.

• Display the count value on the LCD, update only when it changes.

• When the count is a multiple of eight, play one of eight musical tones for one second.

• Implement a state machine with four states:
  – “A” and “B” inputs from encoder cause state transitions.
  – State transitions cause count to go up or down.
Rotary Encoder Lab

• Test the program by rotating the encoder and seeing if the count value changes correctly.

• Problem: When a tone is being played, the program ignores the encoder inputs (try it).
  – Transitions can be lost while the program is in delays and other time-consuming tasks.

• Solution: Modify the program to use interrupts to handle the encoder inputs.
  – Use “Pin Change Interrupts” to generate interrupts whenever an input from the encoder changes.
  – Program responds to input transitions regardless of what it is doing, allowing the count value to change properly when tones are being played.
Pin Change Interrupts

• All the input pins in Ports B, C and D can trigger a pin change interrupt.
• When enabled, a 0→1 or 1→0 transition on the pin will cause an interrupt.
• Separate ISRs for each of the three ports:
  – Port B: PCINT0_vect
  – Port C: PCINT1_vect
  – Port D: PCINT2_vect
• All the pins in one port must use the same interrupt service routine. Up to the ISR to figure out what to do.
Pin Change Interrupts

- Pin change interrupt registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>PCIE2</th>
<th>PCIE1</th>
<th>PCIE0</th>
<th>PCIF2</th>
<th>PCIF1</th>
<th>PCIF0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Change Int. Control Register (PCICR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pin Change Int. Flag Register (PCIFR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pin Change Mask Register 0 (PCMSK0) for Port B</td>
<td>PCINT7</td>
<td>PCINT6</td>
<td>PCINT5</td>
<td>PCINT4</td>
<td>PCINT3</td>
<td>PCINT2</td>
</tr>
<tr>
<td>Pin Change Mask Register 1 (PCMSK1) for Port C</td>
<td>PCINT14</td>
<td>PCINT13</td>
<td>PCINT12</td>
<td>PCINT11</td>
<td>PCINT10</td>
<td>PCINT9</td>
</tr>
<tr>
<td>Pin Change Mask Register 2 (PCMSK2) for Port D</td>
<td>PCINT23</td>
<td>PCINT22</td>
<td>PCINT21</td>
<td>PCINT20</td>
<td>PCINT19</td>
<td>PCINT18</td>
</tr>
</tbody>
</table>

- To enable a pin change interrupt:
  - Set the PCIEx bit to a one for the port
  - Set the PCINTxx bit in the mask register for the I/O pin
  - Call sei() to enable global interrupts
Pin Change Interrupts

- Pin Change Interrupt numbers:

<table>
<thead>
<tr>
<th>(PCMSK0)</th>
<th>(PCMSK1)</th>
<th>(PCMSK2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTB</td>
<td>PORTC</td>
<td>PORTD</td>
</tr>
</tbody>
</table>

| 5 (D13)  | PCINT5   | 5 (A5)   | PCINT13  | 5 (D5)   | PCINT21  |
| 4 (D12)  | PCINT4   | 4 (A4)   | PCINT12  | 4 (D4)   | PCINT20  |
| 3 (D11)  | PCINT3   | 3 (A3)   | PCINT11  | 3 (D3)   | PCINT19  |
| 2 (D10)  | PCINT2   | 2 (A2)   | PCINT10  | 2 (D2)   | PCINT18  |
| 1 (D9)   | PCINT1   | 1 (A1)   | PCINT9   | 1 (D1)   | PCINT17  |
| 0 (D8)   | PCINT0   | 0 (A0)   | PCINT8   | 0 (D0)   | PCINT16  |

- Use the names above to enable interrupts for various pins:

```
PCMSK0 |= ((1 << PCINT5)| (1 << PCINT1));
```
Interrupt-Based Rotary Encoder Lab

- Start with your polling-based rotary encoder lab and modify it to use interrupts to handle the encoder inputs.
- Decide what tasks should be done in the ISR and what stays in the main loop.
  - Hint: Don’t do anything that requires delays in the ISR.
- Test the program by continuing to rotate the knob while a tone is being played. Once the tone is finished the new count value should be displayed.