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, $90^\circ$ 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.

![Rotating clockwise and counter-clockwise](image)

Rotary Encoders

- If $B = 0$ when $A \uparrow$ ⇒ Clockwise
- If $B = 0$ when $A \downarrow$ ⇒ Counter clockwise
- If $A = 1$ when $B \uparrow$ ⇒ Clockwise
- If $A = 1$ when $B \downarrow$ ⇒ Counter clockwise

![Rotating clockwise and counter-clockwise state machine](image)
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 0 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 Lab 8 have 64 states per revolution
- Must have pull-up resistors on switch outputs

Lab 8 – Part A

- 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.
- Use the LCD routines from the previous labs.
- How you work with the encoder inputs is up to you.
  - It can be done with multiple “if” statements.
  - It can be done with a state machine.
- Test the program by rotating the encoder and seeing if the count value changes correctly.

Lab 8 – Part B

- Problem: When the encoder is rotated rapidly the count doesn’t keep up (try it).
  - Transitions can be lost while the program is in delays for the LCD and other time-consuming tasks.
- Solution: Modify the program to use interrupts to handle the encoder inputs.
  - Program can respond to input transitions regardless of what it is doing.
  - This should allow the count value to not miss counts when the encoder is rotated rapidly.
  - Use “Pin Change Interrupts” to generate interrupts whenever an input from the encoder changes.
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 numbers:

<table>
<thead>
<tr>
<th>PORTB</th>
<th>PORTC</th>
<th>PORTD</th>
<th>PCINT23</th>
<th>PCINT22</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 (D13)</td>
<td>5 (A5)</td>
<td>7 (D7)</td>
<td>PCINT5</td>
<td>PCINT13</td>
</tr>
<tr>
<td>4 (D12)</td>
<td>4 (A4)</td>
<td>6 (D6)</td>
<td>PCINT4</td>
<td>PCINT12</td>
</tr>
<tr>
<td>3 (D11)</td>
<td>3 (A3)</td>
<td>5 (D5)</td>
<td>PCINT3</td>
<td>PCINT11</td>
</tr>
<tr>
<td>2 (D10)</td>
<td>2 (A2)</td>
<td>4 (D4)</td>
<td>PCINT2</td>
<td>PCINT10</td>
</tr>
<tr>
<td>1 (D9)</td>
<td>1 (A1)</td>
<td>3 (D3)</td>
<td>PCINT1</td>
<td>PCINT9</td>
</tr>
<tr>
<td>0 (D8)</td>
<td>0 (A0)</td>
<td>2 (D2)</td>
<td>PCINT0</td>
<td>PCINT8</td>
</tr>
</tbody>
</table>

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

```c
PCMSK0 |= ((1 << PCINT5) | (1 << PCINT1));
```

Pin Change Interrupts

• Pin change interrupt registers

<table>
<thead>
<tr>
<th>PCIE2</th>
<th>PCIE1</th>
<th>PCIE0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCIF2</td>
<td>PCIF1</td>
<td>PCIF0</td>
</tr>
<tr>
<td>PCINT23</td>
<td>PCINT22</td>
<td>PCINT21</td>
</tr>
</tbody>
</table>

| PCINT14 | PCINT13 | PCINT12 | PCINT11 | PCINT10 | PCINT9 | PCINT8 |
| PCINT7  | PCINT6  | PCINT5  | PCINT4  | PCINT3  | PCINT2  | PCINT1  | PCINT0  |

• To enable a pin change interrupt:
  – Set the PCIEEx 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

Lab 8 – Part B

• Start with your code from Part 8A 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.
• How does the program know when to update the number on the LCD?
• Test the program by spinning the knob and see if it can now keep up and show 64 counts per revolution.