GOAL
The goal of this lab is to understand the concept of taking input to AVR, interrupt and analog comparator and finally implement it in AVR.

BACKGROUND
PUSHBUTTON
It is an input device providing 0/1 value signal and is interfaced as shown below:

Fig. 1. Pushbutton interfacing
**INTERFACE LOGIC:**

When a port (B,C or D) in a microcontroller is set as an input then the status of those pins in a port eg, PB3, PD2, etc is stored in PIN register namely PINA for port A, PINB for port B. We use the concept of masking learned in lab2 to identify if the switch is ON or OFF.

Example:

\[(\text{PINC} & 0b00000001) == 0b00000001\]

If, PC0 is connected to a switch and if a switch sends logic ‘1’ when ON then, above logic is “TRUE” when switch is ON else its OFF.

**INTERNAL INTERRUPT**

Timer Overflow:

Timer1 uses 16-bit counter which can count up to 65535 before overflow occur. The counting is done using TCNT1 register as we discussed in Timer lab. After 65535, timer overflow occurs and atmega can identify this event. We can use this event to trigger a specific function called Interrupt Service Routine (ISR), where we take care of any operation which needs immediate attention.

Thus, say for a microcontroller operating at 1Mhz Clock, with prescaler 1024, I can get \(Y = 976\) for 1sec delay using the following formula

\[Y = \left(\frac{\text{clock_speed}}{\text{prescaler_value}} \times \text{desired\_delay\_in\_seconds}\right) - 1\]

Now, for an overflow to occur after 1 second, all we need to do is set a TCNT value as,

\[TCNT_x = \text{MAX} - Y\]

Where MAX is maximum timer can count and for TIMER1 it is 65535.

ISR (TIMER1_OVF_vect)

\[
\{ \\
\text{PORTD} ^= 0xFF; \\
\text{TCNT1} = 64559;
\}
\]

int main(void)

\[
\{ \\
\text{DDRD} = 0xFF; \\
\text{TCCR1B} = 5; \\
\text{TCNT1} = 0; \\
\text{sei();} \\
\text{while (1)} \\
\text{\{ \\
\text{PORTD} ^= 0xFF; \\
\text{TCNT1} = 0; \\
\text{\} \\
\}}
\}
\]

int main(void)

\[
\{ \\
\text{DDRD} = 0xFF; \\
\text{TIMSK1} = (1<<TOIE1); \\
\text{TCCR1B} = 5; \\
\text{TCNT1} = 64559; //65535 - 976 \\
\text{sei();} \\
\text{while (1)} \\
\text{\{ \\
\text{PORTD} ^= 0xFF; \\
\text{TCNT1} = 0; \\
\text{\} \\
\}}
\}
\]
EXTERNAL INTERRUPT

Code inside main() is executed sequentially. When we interface our microcontroller to interact with real world, things are not sequential and external world may act in random fashion, interrupt is there to take care of those unpredictable situations. Those external response needs immediate service and interrupt will hold microcontroller’s current operation and respond to interrupt service and resume previous ongoing task.

Interruptions are off by default and can be enabled using sei() and disabled by cli(). To use interrupt, we first enable interrupt by using sei() function, that sets I-bit in SREG.

<table>
<thead>
<tr>
<th>Name</th>
<th>SREG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>0x5F</td>
</tr>
<tr>
<td>Reset</td>
<td>0x00</td>
</tr>
<tr>
<td>Property</td>
<td>When addressing as I/O Register: address offset is 0x3F</td>
</tr>
</tbody>
</table>

Fig. 2. SREG register

When interrupt occurs, we service the interrupt and disable I-bit so that no other interrupt occurs during the service. External interrupt is in pin PD2 and PD3.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F:

<table>
<thead>
<tr>
<th>Name</th>
<th>EIMSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>0x3D</td>
</tr>
<tr>
<td>Reset</td>
<td>0x00</td>
</tr>
<tr>
<td>Property</td>
<td>When addressing as I/O Register: address offset is 0x1D</td>
</tr>
</tbody>
</table>

Fig. 3. External Interrupt Mask Register
When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

**Name:** EIFR  
**Offset:** 0x3C  
**Reset:** 0x00  
**Property:** When addressing as I/O Register: address offset is 0x1C

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTF1</td>
<td>INTF0</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>R/W</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. External Interrupt Flag Register

The External Interrupt Control Register A contains control bits for interrupt sense control.

**Name:** EICRA  
**Offset:** 0x69  
**Reset:** 0x00  
**Property:** -

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ISC11</td>
<td>ISC10</td>
<td>ISC01</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. External Interrupt Control Register A

**ANALOG COMPARATOR**

Analog comparator is in pin 12 and 13. If the value at AIN0 > AIN1 then ACO in ACSR register is set 1.
PUSHBUTTON INTERFACE

Coding Steps:
- define library
- main function: check if PC0 is ‘1’, use masking operation described above, if (PC0 is 1 then turn ON LED) else (turn OFF led)

![Schematic to connect the pushbutton](image)

Fig. 6. Schematic to connect the pushbutton

INTERRUPT INTERFACE

Steps:
- define F_CPU (for delay), delay, interrupt and IO library
- Initialization: LED Port, Switch input, enable external interrupt 0, enable external interrupt flag 0, set rising edge trigger on interrupt 0 and enable global interrupt
- Set ISR: “ISR(INT0_vect)” to perform fast led blink code
- Infinite loop function: slow led blink code

ANALOG COMPARATOR INTERFACE

Coding Steps:
- define library
- main function: check AC0 bit in ACSR register, if it is 1 then AIN0>AIN1 (Turn ON LED) else (Turn OFF LED)
Prelab

1. Answer the questions
   a) Explain in detail each bit of following register: EIMSK, EIFR, EICRA, ACSR and TIMSK1.
   b) What is Interrupt Service Routine?
   c) Mention 4 different internal interrupts.
   d) What is Switch debouncing? How do you avoid them using hardware and software?
EXPERIMENTS

1. Using the pushbutton
   a) Write a code to interface pushbutton. Connect LED and Switch to any pin you want.

2. Programming the interrupt
   a) Write a code to blink LED using external interrupt, your code should blink LED faster when interrupt occurs and slowly during normal operation. You can use _delay_ms() function to control led blinking.

3. Timer1 Overflow Interrupt
   a) Write a code for timer overflow interrupt. You can blink led using the interrupt to verify the operation.

4. Programming the comparator
   a) Write a code to verify the operation of the analog comparator.

POSTLAB

Include the following elements in the report document:

<table>
<thead>
<tr>
<th>Section</th>
<th>Element</th>
</tr>
</thead>
</table>
| 1 | Theory of operation
   Include a brief description of every element and phenomenon that appears during the experiments. |
| 2 | Prelab report |
| 3 | Results of the experiments |
| | Experiment | Experiment Results |
| 1 | a. Code with comments | b. Picture of the circuit wired on the breadboard |
| 2 | a. Code with comments | b. Picture of the circuit wired on the breadboard |
| 3 | a. Code with comments | b. Picture of the circuit wired on the breadboard |
| 4 | c. Code with comments | d. Picture of the circuit wired on the breadboard |
| 4 | Answer the questions |
| | Question no. | Question |
| 1 | | What difference did you find in experiment 2 and 3? |
| 2 | | What are the other interrupt sources in timer apart from the overflow? |
| 5 | Conclusions |
| | Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc. |