RTOS

# Blink led:

## Code:

#include <avr/io.h>

#include <avr/interrupt.h>

#define LED\_PIN PB5

void setup() {

// Set LED\_PIN as output

DDRB |= (1 << LED\_PIN); // or DDRB |= (1<< DDB5)

// Set INT0 to trigger on rising edge

EICRA |= (1 << ISC01) | (1 << ISC00);

// Enable INT0

EIMSK |= (1 << INT0);

// Enable global interrupts

sei();

}

ISR(INT0\_vect) {

// Toggle LED state

PORTB ^= (1 << LED\_PIN); // or PINB ^= (1 << LED\_PIN)

}

int main(void) {

setup();

while(1) {

// Main loop

}

return 0;

}

## DDRB:

DDRB is a register in the AVR microcontroller, and it stands for **Data Direction Register for port B**. This register is used to configure the direction (input or output) of each of the pins on port B.

In the ATmega328P, port B corresponds to digital pins 8 to 13 on an Arduino Uno board. Here’s how it works:

* If a bit in the DDRB register is set to 1, the corresponding port B pin is configured as an output.
* If a bit in the DDRB register is set to 0, the corresponding port B pin is configured as an input.

For example, in the line DDRB |= (1 << LED\_PIN);, the LED\_PIN is being set as an output. The |= operator is a bitwise OR assignment which sets the LED\_PIN bit in the DDRB register to 1, without affecting the other bits. This means that the LED\_PIN will be an output pin, and you can control the voltage on this pin using the PORTB register.

## EICRA:

EICRA stands for External Interrupt Control Register A. It’s a special function

register in the AVR microcontroller used to set the conditions for the external interrupt

to be triggered.

In the ATmega328P, there are two external interrupts: INT0 and INT1. The EICRA

register is used to set whether these interrupts are triggered by a low level, a logical

change, a falling edge, or a rising edge on the interrupt pin.

Here’s how it works:• Bits ISC11 and ISC10 are used for INT1.

* Bits ISC01 and ISC00 are used for INT0.

Each pair of bits can be set to one of the following configurations:

* 00: The low level of INTx generates an interrupt request.
* 01: Any logical change on INTx generates an interrupt request.
* 10: The falling edge of INTx generates an interrupt request.
* 11: The rising edge of INTx generates an interrupt request.
* In the code you asked about, EICRA |= (1 << ISC01) | (1 << ISC00); sets the interrupt to be triggered on the rising edge of the INT0 pin. This is because ISC01 and ISC00 are both set to 1.

Please note that after setting up EICRA, you need to enable the interrupt by setting the

appropriate bit in the EIMSK register, and also enable global interrupts using the sei() function. This allows the microcontroller to respond to the interrupt when it occurs.

## EIMSK:

EIMSK stands for **External Interrupt Mask Register**. It’s a special function register in the AVR microcontroller used to enable or disable external interrupts.

In the ATmega328P, there are two external interrupts: INT0 and INT1. The EIMSK register is used to enable or disable these interrupts.

Here’s how it works:

* Bit INT1 is used to interrupt INT1.
* Bit INT0 is used to interrupt INT0.

If a bit in the EIMSK register is set to 1, the corresponding interrupt is enabled. If a bit is set to 0, the corresponding interrupt is disabled.

For example, in the line EIMSK |= (1 << INT0);, the INT0 interrupt is being enabled. The |= operator is a bitwise OR assignment which sets the INT0 bit in the EIMSK register to 1, without affecting the other bits. This means that the INT0 interrupt will be enabled, and the microcontroller will jump to the ISR(INT0\_vect) function whenever a rising edge is detected on the INT0 pin (if the ISC01 and ISC00 bits in the EICRA register are both set to 1).

Please note that after setting up EIMSK, you also need to enable global interrupts using the sei() function. This allows the microcontroller to respond to the interrupt when it occurs.

## sei():

The sei() function is a part of the AVR Libc library, which provides a high-level interface to the features of the AVR microcontroller. sei() stands for **Set Interrupts**.

When you call sei(), it enables global interrupts for the AVR microcontroller. This means that the microcontroller can respond to interrupt signals. Interrupts are events that require immediate attention from the microcontroller. When an interrupt occurs, the microcontroller pauses its current task, jumps to a special function known as an Interrupt Service Routine (ISR) to handle the event, and then returns to its previous task.

sei() is called after setting up the EICRA and EIMSK registers. This allows the microcontroller to respond to the INT0 interrupt when it occurs.

Please note that there is also a corresponding cli() function, which stands for **Clear Interrupts**. This function disables global interrupts, meaning the microcontroller will not respond to interrupt signals until sei() is called again.

## ISR(INT0\_vect):

This is an Interrupt Service Routine (ISR) associated with the INT0 interrupt vector in AVR microcontrollers.

* ISR(INT0\_vect): This is the declaration of the ISR. The ISR() macro is a special function used in AVR programming to handle interrupt events. The argument INT0\_vect specifies which interrupt vector the ISR is associated with. In this case, it’s INT0, which is one of the external interrupts in AVR microcontrollers.
* {}: These are the opening and closing braces of the function. Any code that you want to execute when the interrupt occurs would go between these braces. In this case, the braces are empty, meaning no action is taken when the INT0 interrupt occurs.

Interrupts are a mechanism by which a microcontroller can be instructed to leave its normal program execution flow to execute a special function (the ISR) when a certain event occurs. This event could be a change in a digital input pin (like a button press), a timer reaching a certain value, a communication event (like receiving a byte via SPI or UART), etc.

After the ISR has finished executing, the microcontroller returns to wherever it left off in its normal program execution. This is why interrupts are useful for responding quickly to external events, even if the microcontroller is in the middle of doing something else.

In the context of INT0, this is typically an external interrupt that is triggered by a voltage level change (either from low to high, or high to low, or both) on a specific microcontroller pin.

PINB |= (1 << PB5);: This line of code toggles the state of the 5th bit of the PORTB register (which corresponds to pin PB5 on the microcontroller). The |= operator performs a bitwise OR operation with the current value of PINB and the result of (1 << PB5). The expression (1 << PB5) creates a value where only the 5th bit is set to 1 and all other bits are 0. The effect of the bitwise OR operation is to set the 5th bit of PINB to 1, regardless of its previous state. If the 5th bit of PINB was already 1, it remains 1. If it was 0, it becomes 1. This effectively turns the LED on and off.