EMBEDDED SYSTEMS LAB 2

Fall Semester 2023
Lab Experiment Lab 2– External Interrupt (Assembler)
Instructor: Dr. Prof. Fangning Hu
Author of the report: Faraz Ahmed and Sohaib Salman

Experiment conducted by: Faraz Ahmed, Sohaib Salman
Place of execution: Research1
Date of Execution: 12th September 2023

Introduction

The objective of this laboratory exercise revolves around the practical application of interrupts in microcontroller programming. The primary goal is to establish a circuit wherein the pressing of a button triggers an external interrupt. This is achieved by integrating a pull-up resistor in the circuit configuration, ensuring that the input voltage to the pin maintains a high level before button activation and transitions to a low level upon pressing.

Additionally, a pivotal aspect of this exercise entails determining the behavior of an LED in response to the button press. In this particular case, the LED is programmed to execute a distinctive pattern: two rapid blinks upon button press, while under normal circumstances, it engages in a slower blinking pattern. This dynamic behavior is orchestrated through the implementation of an Interrupt Service Routine (ISR), a critical component in interrupt-driven programming.

The concept of interrupts plays a crucial role in the operation of microcontrollers, as they necessitate an immediate response from the device. When an interrupt event occurs, the microcontroller temporarily halts its ongoing task to execute an Interrupt Service Routine (ISR). Once the ISR is completed, the microcontroller resumes its paused task and continues with normal operations. The Atmega328 microcontroller offers various internal and external interrupt sources, with external interrupts being triggered by changes in voltage on pins INT0 or INT1. When an interrupt is activated, the corresponding Interrupt Flag is set to 1. An Interrupt Request is generated only if both the interrupt Flag and an Interrupt Enable bits are set to one.

Pre-Lab Tasks

1. In order to configure external interrupts on the ATmega328 microcontroller, we need to understand the relevant registers and their associated pins. Let's go through the necessary steps:

   EMISK Register (External Interrupt Mask Register):

   - This register controls which external interrupts are enabled or disabled.
   - The bit corresponding to a specific external interrupt, when set to '1', enables it.
   - EMISK I/O Address: 0x5B

   EICRA Register (External Interrupt Control Register A):

   - This register determines the triggering condition for the external interrupt pins INT0 (PD2) and INT1 (PD3).

- Each pair of bits (ISC01, ISC00 for INT0 and ISC11, ISC10 for INT1) control the triggering conditions for the respective interrupts.
- EICRA I/O Address: 0x69
- Bit Descriptions:
    - ISC01 and ISC00 (Bits 1 and 0, respectively, for INT0):
        - 00: The low-level of INT0 generates an interrupt request.
        - 01: Any logical change on INT0 generates an interrupt request.
        - 10: The falling edge of INT0 generates an interrupt request.
        - 11: The rising edge of INT0 generates an interrupt request.
    - ISC11 and ISC10 (Bits 3 and 2, respectively, for INT1):
        - 00: The low-level of INT1 generates an interrupt request.
        - 01: Any logical change on INT1 generates an interrupt request.
        - 10: The falling edge of INT1 generates an interrupt request.
        - 11: The rising edge of INT1 generates an interrupt request.

Assembly Instructions for Configuration:

- To enable an external interrupt (e.g., INT0), set the corresponding bit in the EMISK register:

```
LDI R16, (1<<INT0)      ; Load R16 with bit mask for INT0
STS 0x5B, R16           ; Store R16 in EMISK register (Enable INT0)
```

- To set the triggering condition (e.g., rising edge) for INT0, modify the corresponding bits in the EICRA register:

```
LDI R16, (1<<ISC01)     ; Load R16 with bit mask for ISC01
STS 0x69, R16           ; Store R16 in EICRA register (Set ISC01)
 LDI R16, (1<<ISC00)    ; Load R16 with bit mask for ISC00
 STS 0x69, R16          ; Store R16 in EICRA register (Set ISC00)
```

By utilizing these assembly instructions, we can effectively enable and configure external interrupts on the ATmega328 microcontroller, allowing us to respond to specific events with the desired interrupt handling routine.

Lab Assignments

1. Setting up circuit
2. When the button is clicked, the LED turns off, and on. If the button is pressed and held, the LED remains off, and turns off when the button is released.

Code:

```
.org 0x0000
        rjmp main
        .org INT0addr
        rjmp INT0_handler
        .org INT_VECTORS_SIZE
        .def dreg = r18

main:
        cli
        ldi r17, LOW(RAMEND)
        out SPL, r17
        ldi r17, HIGH(RAMEND)
        out SPH, r17            ; set Stack Pointer
        sbi DDRB, 0x00          ; set PB0 as output (Pin 8)
        sbi PORTB, 0x00         ; set PB0
        cbi DDRD, 0x02          ; set PD2 as input
        cbi PORTD, 0x02
        ldi r16, 0x01           ; enable external interrupts
        out EIMSK, r16
        ldi r16, 0x05           ; change triggers
        sts EICRA, r16
        ldi r16, 0x20
        sei

loop:
        rjmp loop

INT0_handler:
        push r17
        in r17, SREG            ; push status register to SP
        com r16
        out PORTB, r16          ; negate PORTB (PINB5)
        out SREG, r17           ; restore status register
        pop r17

        reti                    ; return from handler (INT0)
```

Code Explanation:

1. Initialization:
    - The program starts at address 0x0000 and jumps to the main function.
    - The address for the INT0 ISR is set at INT0addr.
    - INT_VECTORS_SIZE serves as a placeholder for the size of interrupt vectors.
2. Main Function:
    - Global interrupts are disabled (cli) to ensure a clean setup.
    - Stack Pointer (SP) is initialized with the value of RAMEND.
    - DDRB is set to make Pin 8 (PB0) an output (LED pin).
    - PORTB is set high to turn on the LED.
    - DDRD is cleared for Pin 2 (PD2), making it an input for the button. The internal pull-up resistor is enabled.
    - External Interrupt INT0 is enabled, and triggering conditions are set for rising edges.
    - Global interrupts are enabled (sei).
3. Infinite Loop:
    - The program enters an infinite loop to keep it running continuously.
4. INT0 Interrupt Service Routine (ISR):
    - When INT0 is triggered, the program jumps to this routine.
    - The value of register r17 is pushed onto the stack.
    - The status register is saved.
    - The value in r16 is complemented (bitwise negation).
    - The result is written to PORTB (toggling Pin 8, PB0).
    - The original status register value is restored, and r17 is popped from the stack.
    - reti is used to return from the interrupt, re-enabling global interrupts.