2η εργαστηριακή άσκηση

Φωτεινή Κυριακοπούλου 03120182
Ιωάννης Μπουφίδης 03120162

Ζήτημα 2.1

```
.include "m328PBdef.inc"

.equ FOSC_MHZ=16      ; MHz
.equ DEL_ms=500       ; ms
.equ DEL_NU=FOSC_MHZ*DEL_ms

.org 0x0
rjmp init
.org 0x4
rjmp ISR1

init:
    ; init Stack Pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPL, r24

    ; enable interrupt INT1 on falling edge
    ldi r24, (1 << ISC11) | (0 << ISC10)
    sts EICRA, r24
    ldi r24, (1 << INT1)
    out EIMSK, r24

    ser r26
    out DDRB, r26             ; PORTB as output
    out DDRC, r26             ; PORTC as output
    clr r16                   ; interruption counter to 0
    out PORTC, r16

    sei                       ; enable global interrupts

; main program
main:
    clr r26
loop:
    out PORTB, r26

    ldi r24, LOW(DEL_NU)
    ldi r25, HIGH(DEL_NU)     ; set delay
    rcall delay_mS

    inc r26
```

```
    cpi r26, 16
    breq main
    rjmp loop

; delay routine
delay_mS:
    ldi r23, 249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24, 1
    brne delay_mS

    ret

; interrupt routine
ISR1:
    ldi r24, (1 << INTF1)
    out EIFR, r24             ; clear INTF1
    ldi r24, LOW(100*FOSC_MHZ)
    ldi r25, HIGH(100*FOSC_MHZ)
    rcall delay_mS            ; delay 100ms
    in r24, EIFR
    cpi r24, 0                ; INTF1 = 1 -> there's been a debounce
    brne ISR1                 ; wait until there's no debounce

    in r17, PIND
    sbrs r17, 6               ; if PD6 is pressed
    rjmp isr1_exit            ; then exit the interrupt routine

    inc r16                   ; increment counter
    cpi r16, 32               ; check if the counter reached 32
    brne update_leds          ; if not, update the LEDs
    clr r16                   ; reset the counter to zero
update_leds:
    out PORTC, r16
isr1_exit:
    reti
```

2η εργαστηριακή άσκηση

Ζήτημα 2.2

```
.include "m328PBdef.inc"

.equ FOSC_MHZ=16      ; MHz
.equ DEL_ms=1000      ; ms
.equ DEL_NU=FOSC_MHZ*DEL_ms

.def result = r16
.def mask = r18
.def bits = r19

.org 0x0
rjmp init
.org 0x2
rjmp ISR0

init:
    ; init Stack Pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPL, r24

    ; enable INT0 on falling edge
    ldi r24, (1 << ISC01) | (0 << ISC00)
    sts EICRA, r24
    ldi r24, (1 << INT0)
    out EIMSK, r24

    ser r26
    out DDRC, r26             ; PORTC as output


    sei                       ; enable global interrupts

; main program
main:
    clr r26
loop:
    out PORTC, r26

    ldi r24, LOW(DEL_NU)
    ldi r25, HIGH(DEL_NU)
    rcall delay_mS

    inc r26
```

```
    cpi r26, 32
    breq main
    rjmp loop

; interrupt routine
ISR0:
    cli                         ; disable interrupts

    ldi result, 0               ; initialize the result register
    in r17, PINB                ; read PORTB
    com r17                     ; inverse logic input
    ldi mask, 0b011111          ; initialize mask register
    ldi bits, 5                 ; bits to check

    and r17, mask               ; isolate PB4-PB0 bits
int_loop:
    lsr r17
    brcc check                  ; if LSB is 1
    inc result                  ; increment the result register
    lsl result                  ; and shift left to check next bit
check:
    dec bits                    ; else if there are more bits
    brne int_loop               ; repeat
isr0_exit:
    lsr result                  ; else shift the last "1" to LSB
    out PORTC, result           ; output the result
    sei                         ; enable interrupts
    reti

; delay routine
delay_mS:
    ldi r23, 249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24, 1
    brne delay_mS

    ret
```

Ζήτημα 2.3

Ο κώδικας assembly της άσκησης είναι ο ακόλουθος:

```
.include "m328pbdef.inc"

.org 0x0
rjmp reset
.org 0x4
rjmp ISR1

reset:
    ; init Stack Pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPL, r24

    ; enable INT1 on falling edge
    ldi r24, (1 << ISC01) | (0 << ISC00)
    sts EICRA, r24
    ldi r24, (1 << INT1)
    out EIMSK, r24

    ser r21
    out DDRB, r21             ; PORTB as output
    sei                       ; enable interrupts
main:
    rjmp main

; interrupt routine
ISR1:
    ser r26
    out PORTB, r26            ; turn on all LEDs
    call delay_half           ; delay 0.5 seconds
    ldi r26, 1
    out PORTB, r26            ; turn off the LEDs except PB0
    call delay_three          ; delay 3 seconds
    clr r26
    out PORTB, r26            ; turn off PB0 too
    reti

delay_half:
    ldi r24, LOW(16*500)
    ldi r25, HIGH(16*500)
    call delay_ms
    ret
```

```
delay_three:
    ldi r24, LOW(16*3000)
    ldi r25, HIGH(16*3000)
    call delay_ms
    ret

; delay routine
delay_mS:
    ldi r23, 249
loop_inn:
    sei                 ; allow delay routine to be interrupted
                        ; start the interrupt routine all over again
    dec r23
    nop
    brne loop_inn

    sbiw r24, 1
    brne delay_mS

    ret
```

Ο κώδικας C της άσκησης είναι ο ακόλουθος:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

// flag to check when INT1 is activated
volatile uint8_t buttonPressed = 0;

int main() {
    // PD3 as input
    DDRD &= ~(1 << 3);
    PORTD |= (1 << 3);

    // INT1 on rising edge
    EICRA = (1 << ISC10) | (1 << ISC11);

    // enable INT1
    EIMSK |= (1 << INT1);

    // enable global interrupts
    sei();

    // PORTB as output
    DDRB = 0xFF;

    while (1) {
        // check if PD3 is pressed
        if (buttonPressed) {
           buttonPressed = 0;
```

```
            // Turn on LEDs for 0.5 sec
            PORTB = 0xFF;
            for (int i=0; i<500; i++) {
                _delay_ms(16);
                // if PD3 is pressed
                if(buttonPressed) break;
            }
            // start all over again
            if(buttonPressed) continue;

            // turn off the LEDs except PB0 for 3 sec
            PORTB = 0x01;
            for (int i=0; i<3000; i++) {
                _delay_ms(16);
                if(buttonPressed) break;
            }
            // turn off PB0 too
            PORTB = 0x00;
        }
    }
}

// INT1 interrupt routine
ISR(INT1_vect) {
    // set flag
    buttonPressed = 1;
}
```