X M AVR

E M

K K , *el*21045 M Δ , *el*21170

22 O 2024

0.1 Z 2.1

T 2.1. Π INT1 INT1 , int1counter , reset 63. Π PB5 , INT1 . E isrloop , 2.4.

```
;
; Ex2_1.asm
;
; Created: 10/16/2024 1:10:26 PM
; Author : User
;

.include "m328PBdef.inc"          ; ATmega328pB microcontroller definitions

.equ FOSC_MHZ=16                            ; Microcontroller operating frequency in MHz

.equ DEL_mS=500                                 ; Delay in mS (valid number from 1 to 4095)

.equ DEL_NU=FOSC_MHZ*DEL_mS         ; delay_mS routine: (1000*DEL_NU+6) cycles

.def int1counter=r16
.def pin=r17

.org 0x0
        rjmp reset

.org 0x4
        rjmp isr1

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

; Init PORTB as output
```

```
        ser r26
        out DDRB, r26

; Init PORTC as output
        ldi r26, 0b00111111
        out DDRC, r26

; Init PORTD as input
        clr r26
        out DDRD, r26

; Interrupt on rising edge of INT1 pin
        ldi r24, (1<<ISC11) | (1<<ISC10)
        sts EICRA, r24

; Enable the INT1 interrupt
        ldi r24, (1<<INT1)
        out EIMSK, r24

        clr int1counter
        sei ; Enable general flag of interrupts

loop1:
        clr r26
loop2:
        out PORTB, r26

        ldi r24, LOW(DEL_NU)        ;
        ldi r25, HIGH(DEL_NU)       ; Set delay (number of cycles)
        rcall delay_mS              ;

        inc r26

        cpi r26, 16                 ; compare r26 with 16
        breq loop1
        rjmp loop2

; External interrupt 1 service routine
isr1:
        push r24
        push r25
        in r24, SREG
        push r24
isrloop:
        ldi r24, (1<<INTF1)
        out EIFR, r24
        ldi r24, LOW(5*16)          ;
        ldi r25, HIGH(5*16)         ; Set delay (number of cycles)
        rcall delay_mS              ;
        in r24, EIFR
        sbrc r24, INTF1             ; skip next instruction if EIFR & (1<<INTF1) == 0
        rjmp isrloop                    ; don't continue until EIFR & (1<<INTF1) == 0 to avoid debouncing
        in pin, PIND
        sbrs pin, 5                     ; skip next instruction if PB5=1 (not pressed)
        rjmp end                        ; skip interrupt if PB5=0 (pressed)
        inc int1counter
        sbrc int1counter, 6             ; skip next instruction if int1counter & (1<<6) == 0
        clr int1counter                 ; reached 2^6 = 64, return to 0
        out PORTC, int1counter
end:
        pop r24
        out SREG, r24
        pop r25
        pop r24
        reti                            ; Return from interrupt

; delay of 1000*F1+6 cycles (almost equal to 1000*F1 cycles)
delay_mS:
; total delay of next 4 instruction group = 1+(249*4-1) = 996 cycles
        ldi r23, 249                    ; (1 cycle)
loop_inn:
        dec r23                         ; 1 cycle
        nop                             ; 1 cycle
        brne loop_inn                   ; 1 or 2 cycles

        sbiw r24, 1                     ; 2 cycles
        brne delay_mS                   ; 1 or 2 cycles

        ret                             ; 4 cycles
```

0.2 Z 2.2

Γ Z 2.2, . A 16 32, . Γ interrupts, INT1 INT0 . H interrupt , bits PORTB (5 LSBs bitwise AND) .

```
;
; Ex2_2.asm
;
; Created: 10/16/2024 12:58:39 PM
; Author : User
;


;
; Ex2_2.asm
;
; Created: 10/16/2024 12:53:50 PM
; Author : User
;

```

```
.include "m328PBdef.inc"

.equ FOSC_MHZ=16
.equ DEL_mS=2000
.equ DEL_NU=FOSC_MHZ * DEL_mS
.equ INT_mS=500
.equ INT_NU=FOSC_MHZ * INT_mS

.def DELL=r24
.def DELH=r25
.def TMP=r26
.def COUNTER=r27

; ---- INTERRUPTS ----
.org 0x0
        rjmp reset
.org 0x2
        rjmp ISR0

ISR0:
        push TMP                ;
        push DELL               ; (this will be used for input)
        push DELH               ; Save registers to stack
        in TMP, SREG            ;
        push TMP                ;
        push COUNTER            ;

        ; TODO: output
        in DELL, PINB           ; Read input
        andi DELL, 15           ; Keep 4 LSBs
        clr TMP                 ; Initialize result
        clr COUNTER             ; Initialize counter
check_bit:
        lsr DELL
        brcs skip_inc           ; If popped a 0, skip increasing TMP
        inc TMP
skip_inc:
        inc COUNTER
        cpi COUNTER, 4          ; If haven't checked 4 bits, repeat
        brne check_bit

        out PORTC, TMP          ; Output

        ldi DELL, low(INT_NU)   ;
        ldi DELH, high(INT_NU)  ; Call delay
        call delay_mS           ;

        pop COUNTER             ;
        pop TMP                 ;
        out SREG, TMP           ;
        pop DELH                ; restore registers from stack
        pop DELL                ;
        pop TMP                         ;

        reti                        ; return

reset: ; configure eternal Interrupts
        ldi TMP, (1 << ISC01) | (1 << ISC00)        ; set rising edge
        sts EICRA, TMP                                  ;
        ldi TMP, (1 << INT0)                            ; enable INT0
        out EIMSK, TMP                                   ;
        sei                                         ; Enable Interrupts

; ---- MAIN CODE ----

; init SP
ldi TMP, LOW(RAMEND)
out SPL, TMP
ldi TMP, HIGH(RAMEND)
out SPH, TMP

; set PORTB, PORTD as INPUT
clr TMP
out DDRB, TMP
out DDRD, TMP
; set PORTC as OUTPUT
ser TMP
out DDRC, TMP

init:
        clr COUNTER                 ; reset to 0
loop_:
        out PORTC, COUNTER                  ; output
        ldi DELL, low(DEL_NU)         ;
        ldi DELH, high(DEL_NU)         ; call delay
        rcall delay_mS                  ;

        inc COUNTER                 ; increase

        cpi COUNTER, 32                  ; If COUNTER exceedes range
        breq init                   ; then go to beginning
        rjmp loop_                  ; else loop

delay_mS:        ; Given procedure
        ldi TMP, 249
loop_inn:
        dec TMP
        nop
        brne loop_inn
        sbiw DELL, 1
        brne delay_mS
        ret
```

0.3 Z 2.3

Γ Z 2.3 `delay_mS` 1ms. K ms `FLAG` (0: LEDs, 1: `PB0` , 2: LEDs `PORTB`) 0. T `FLAG` 2 `INT1` . T 1 2, 500ms, 0 1, 4500ms. Θ `COUNT = 4500` `FLAG` 0, `COUNT = 500` `ISR1` , `COUNT` , ( ) `FLAG` . T ms , `CNT` . T `C` `_delay_ms(1)` , `flg` `count`

```
;
; Ex2_3_asm.asm
;
; Created: 10/16/2024 2:28:35 PM
; Author : User
;

.include "m328PBdef.inc"

.equ FOSC_MHZ=16
.equ FLASH_MS=500
.equ DEL_MS=4500

.def CNT=r24 ; count number of delays that have happened
.def CNT_HI=r25
.def TMP=r26
.def TMP_HI=r27
.def FLG=r28 ; 0 --> Closed, 1 --> PB0, 2 --> All

; ---- INTERRUPTS ----
.org 0x0
        rjmp reset
.org 0x4
        rjmp ISR1

ISR1:
        push TMP
        in TMP, SREG
        push TMP

        ldi FLG, 2
        ldi CNT   , low (FLASH_MS)
        ldi CNT_HI, high(FLASH_MS)

        pop TMP
        out SREG, TMP
        pop TMP

        reti

; ---- MAIN CODE ----
```

```
delay_mS:         ; Given procedure, altered so it runs for 1ms (no args)
        ldi TMP, low(3999)
        ldi TMP_HI, high(3999)
loop_inn:
        sbiw TMP, 1
        brne loop_inn
        ret

reset: ; configure eternal Interrupts
        ldi TMP, (1 << ISC11) | (1 << ISC10)        ; set rising edge
        sts EICRA, TMP                              ;
        ldi TMP, (1 << INT1)                        ; enable INT1
        out EIMSK, TMP                              ;

; init SP
ldi TMP, LOW(RAMEND)
out SPL, TMP
ldi TMP, HIGH(RAMEND)
out SPH, TMP

; set PORTB as OUTPUT
ser TMP
out DDRB, TMP
; out DDRC, TMP
; set PORTD as INPUT
clr TMP
out DDRD, TMP

sei                                          ; enable Interrupts

init:
        clr FLG                     ;
        ldi CNT    , low(DEL_MS)        ; load 4.5s (no matter if FLG = 0 or 1)
        ldi CNT_HI, high(DEL_MS);

loop_:
        sei
        cpi FLG, 1                  ; Check FLG:
        breq one_bit                  ;   if FLG == 1, open 1 bit
        brlo no_bits                  ;   else if FLG < 1, open 0 bits
all_bits:                       ;   else open all bits
        ser TMP
        out PORTB, TMP
        jmp cont
no_bits:
        clr TMP
        out PORTB, TMP
        jmp cont
one_bit:
        ldi TMP, 1
        out PORTB, TMP
        jmp cont
cont:
        rcall delay_mS                  ; run delay
        sbiw CNT, 1                             ; decrease counter
        brne loop_                  ; if counter != 0, skip FLG updates
        dec FLG                              ; decrease flag
        brpl foo                  ; if N == 1 (FLG < 0)
        ldi FLG, 0                  ;    then FLG = 0
foo:
        cli
        cpi FLG, 2
        breq loop_
        ldi CNT    , low(DEL_MS)          ; load 4.5s (no matter if FLG = 0 or 1)
        ldi CNT_HI, high(DEL_MS);
        sei
        jmp loop_
```

```
/*
 * main.c
 *
 * Created: 10/16/2024 4:46:07 PM
 *  Author: User
 */

#include <xc.h>

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define FLG_OFF 0
#define FLG_ONE 1
#define FLG_ALL 2

#define CNT_ALL 500
#define CNT_ONE 4500

int flg = FLG_OFF, count = 0;

ISR (INT1_vect)
{
        flg = FLG_ALL, count = CNT_ALL;
}

#define max(a, b) ((a) > (b) ? (a) : (b))

int main ()
{
        EICRA = (1 << ISC11) | (1 << ISC10);
        EIMSK = 1 << INT1;
        sei();

        DDRB = 0xff; // input
        DDRD = 0x00; // output

        while (1) {
                while (count--) {
                        PORTB = (flg == FLG_OFF) ? 0x00 :
                        (flg == FLG_ONE) ? 0x01 : 0x0ff;
                        _delay_ms(1);
                }
                flg = max(flg - 1, 0);
                count = CNT_ONE;
        }
}
```