.include "m169def.inc" **; CPU definitions**

.org 0x1000

.include "print.inc" **; used for show\_char to display on the LCD**

.dseg **; data memory**

s: .byte 1 **; counter used for seconds**

ms: .byte 1 **; counter used for 1/10th of a second**

.cseg **; program memory**

.org 0 **; start of the program**

jmp setup **; SETUP OF DISPLAY / JOYSTICK / OSCILLATOR & INTERRUPT**

.org 0x000A **; memory address of the interrupt handler**

jmp inthandlr **; " INTERRUPT HANDLER "**

.org 0x100 **; start of program**

**; ===INTERRUPT HANDLER===**

inthandlr:

push r16 **; store r16 & SREG into the stack**

in r16, SREG

push r16

lds r16,s **; increment seconds counter**

inc r16

sts s,r16

lds r16,ms **; increment milliseconds counter**

inc r16

sts ms,r16

pop r16 **; restore r16 & SREG**

out SREG,r16

pop r16

reti **; return from the interrupt**

**; === SETUP ===**

setup:

**;stack initialisation , used for calling subprograms**

ldi r16,0xFF

out SPL,r16

ldi r16,0x04

out SPH,r16

call init\_disp **; display initialisation**

**; joystick initialisation**

in r17,DDRE

andi r17,0b11110011

in r16,PORTE

ori r16,0b00001100

out DDRE,r17

out PORTE,r16

ldi r16,0b00000000

sts DIDR1,r16

in r17,DDRB

andi r17,0b00101111

in r16,PORTB

ori r16,0b11010000

out DDRB,r17

out PORTB,r16

cli **; disables interrupts**

ldi r16,8

sts ASSR,r16 **; sets the crystal oscillator as the source of frequency 32768Hz**

ldi r16,1

sts TIMSK2,r16 **; enables the interrupts from the counter**

ldi r16, 1 **; 32768/(1)\*256 = 128 ~ 256 used because thats the overflow "rate" for 1Byte incrementing**

sts TCCR2A , r16 **; division ratio is 128 which is approximately 1 second**

clr r16

out EIMSK, r16 **; disables the interrupt from the joystick**

sei **; enables interrupts**

**; === RESET ===**

reset:

ldi r20,0 **; used throught program to reset registers to 0**

ldi r21,0 **; 1/10th of a second (0-9)**

ldi r22,0 **; 2nd digit of seconds (0-9)**

ldi r23,0 **; 1st digit of seconds (0-6)**

ldi r24,0 **; 2nd digit of minutes (0-9)**

ldi r25,0 **; 1st digit of minutes (0-6)**

sts s,r25 **; setting initial value of counters to 0**

sts ms,r25

jmp display **; calling "display" function to display timer after reset**

**; === MAIN ===**

main:

jmp display **; displays values stored in registers r21 - r25**

jmp millisec **; function for checking the 1/10th of a second**

**; === MILLI-SECs ===**

millisec:

lds r16,ms **; loads value of the 1/10th of a second counter into r16**

cpi r16,13 **; compares the value to 13 which is approximately 1/10th of a second**

brlo secsmins **; branches to "secsmins" if 1/10th a second has not been counted yet**

sts ms,r20 **; else resets the counter for 1/10th of a second to 0**

inc r21 **; increments the register for the 1/10th of a second**

cpi r21,10 **; checks if the register for 1/10th of a second (0-9) has reached 10**

brlt secsmins **; if the value is between 0 and 9 branch to "secsmins"**

ldi r21,0 **; else reset the register for 1/10th of a second to 0 then continue to "secsmins"**

**; === SECs/MINs ===**

secsmins:

lds r16,s **; loads value of the seconds counter into r16**

cpi r16,128 **; compares the value to 128 which is approximately 1 second**

brlo main **; branches back to "main" if a second hasn't been counted yet**

ldi r17,0

sts s, r17 **; else resets the 1/10th of a second and seconds counter to 0**

sts ms,r17

inc r22 **; increments the register for the 2nd digit of seconds**

cpi r22,10 **; check if 2nd digit of seconds (0-9) has reached 10**

brlt main **; if the value is between 0 and 9 branch back to the "main"**

subi r22,10 **; else reset the register for the 2nd digit of a second to 0**

inc r23 **; increment the 1st digit of seconds**

cpi r23,6 **; check if 1st digit of seconds (0-6) has reached 6**

brlt main **; if the value is between 0 and 6 branch back to the "main"**

subi r23,6 **; else reset the register for the 1st digit of seconds to 0**

inc r24 **; increment 2nd digit of minutes**

cpi r24,10 **; check if 2nd digit of a minutes (0-9) has reached 10**

brlt main **; if the value is between 0 and 9 branch back to the "main"**

subi r24,10 **; else reset the register for the 2nd digit of minutes to 0**

inc r25 **; increment the 1st digit of minutes**

cpi r25,6 **; check if 1st digit of minutes (0-6) has reached 6**

brlt main **; if the value is between 0 and 6 branch back to the "main"**

subi r25,6 **; else reset the value of the 1st digit of minutes to 0**

jmp main **; return to the "main" ; could start counting hours**

**; === DISPLAY ===**

display:

ldi r17,6 **; starts printing from position 6 on the display**

mov r16,r21 **; displays 1/10th of a second**

call print

mov r16,r22 **; displays the 2nd digit of seconds**

call print

mov r16,r23 **; displays the 1st digit of seconds**

call print

mov r16,r24 **; displays the 2nd digit of minutes**

call print

mov r16,r25 **; displays the 1st digit of minutes**

call print

jmp millisec **; calls function for the initial increment of time "millisec"**

**; === PRINT ===**

print:

subi r16, -'0' **; adds ASCII value of '0' to current digit**

call show\_char **; displays the digit's ASCII value (0-9)**

dec r17 **; decrement the printing position on the display**

call joystick **; checking for input from the joystick**

ret **; return to the "display" function**

**; === JOYSTICK ===**

joystick:

in r18,PINB **;read inputs from joystick registers**

in r19,PINE

andi r18,0xD0 **; mask registers accordingly**

andi r19,0x0C

or r18,r19 **; combine masked values into register r18**

ldi r19,255 **; loop to create a delay**

loop:

dec r19

brne loop

in r16,PINB **; read inputs from joystick again**

in r19,PINE

andi r16,0xD0 **; mask registers accordingly**

andi r19,0x0C

or r16,r19 **; combine masked values into register r16**

cpse r18,r16 **; compare registers r18 & r16 ( inputs before & after delay )**

jmp joystick **; if they are NOT equal "INVALID INPUT" jump back to "joystick"**

sbrs r18,2 **; else if the 2nd LMSB is set to 0 then the joystick LEFT was triggered**

jmp reset **; call "reset" function for the timer**

sbrs r18,3 **; else if the 3rd LMSB is set to 0 then the joystick RIGHT was triggered**

ret **; return to print function and continue**