Assignment 1 ML **J L Gouws**
March 31, 2022 **19G4436**

main.asm

```
1  .INCLUDEPATH "/usr/share/avra" ;set the inlude path to the correct
       place
2  ;.DEVICE ATmega16
3  .NOLIST
4  .INCLUDE "m16def.inc"
5  .LIST
6
7  .def MUL_LOW=R0             ;
8  .def MUL_HIGH=R1                 ;
9  .def ZERO=R2                ;
10 .def step=R3
11 .def dstep=R4
12 .def slow=R5
13 .def retReg=R6
14 .def tmp1=R16
15 .def tmp2=R17
16 .def tmp3=R18
17 .def arg1=R19               ;
18 .def arg2=R20               ;
19 .def tasknum=R21
20 .def stepCount=R22
21
22 .CSEG                              ; start the code segment
23 .org 0x000                        ; locate code at address $000
24   RJMP  START                     ; Jump to the START Label
25 .org INT0addr
26   JMP    INT0_ISR
27 .org INT1addr
28   JMP    INT1_ISR
29 .org URXCaddr
30   JMP    urxc_isr
31 .org UTXCaddr
32   JMP    utxc_isr
33 .org OC0addr
34   JMP    t0_OC_ISR
35 .org OVF2addr
36   JMP    t2_OV_ISR
37 .org ADCCaddr
38   JMP    ADC_ISR
39 .org OC1Aaddr
40   JMP    t1_OCA_ISR
41 .org OVF1addr
42   JMP    t1_OV_ISR
43
```

```
44 .org $02A                                    ; locate code past the
       interupt vectors
45
46 START:
47    LDI    tmp1, LOW(RAMEND)
48    OUT    SPL, tmp1
49    LDI    tmp1, HIGH(RAMEND)          ; initialise the stack pointer
50    OUT    SPH, tmp1
51
52    EOR    zero, zero                  ; make zero register, well zero
53
54    ; set task num to zero
55    LDI    tasknum, 0x00
56    LDI    XH, HIGH(TASK_NUM_RAM)
57    LDI    XL, LOW(TASK_NUM_RAM)
58    ST     X, tasknum                  ; make the task number buffer zero
59
60    ; initialize the microcontroller
61    CALL   init_LCD
62    CALL   init_EEP
63    CALL   init_UART
64    CALL   init_stepper
65    CALL   init_IO
66    CALL   send_menu
67    CALL   init_watchdog
68    SEI
69
70 MAIN_LOOP:
71    NOP
72    NOP
73    NOP
74    WDR
75    RJMP MAIN_LOOP
76
77 .include "IO.asm"
78 .include "LCD.asm"
79 .include "UART.asm"
80 .include "EEP.asm"
81 .include "stepperMotor.asm"
82 .include "taskHandler.asm"
83 .include "ADC.asm"
84 .include "watchdog.asm"
```

taskHandler.asm

```
1 ; found this at
2 ; https://www.avrfreaks.net/forum/how-do-you-make-jump-table-avr-
      assembly
3 .DSEG
```

```
4  TASK_NUM_RAM: .BYTE 2 ;
5
6  .MACRO jumpto
7     LDI   XH, HIGH(@0)   ; NB Can't use Z register for IJMP
8     LDI   XL, LOW(@0)
9     MOV   tmp1, @1
10    LSL   tmp1
11    ADD   XL, tmp1
12    ADC   XH, ZERO
13    PUSH  XL
14    PUSH  XH
15    RET
16 .ENDMACRO
17
18 .CSEG
19 do_task:
20    CALL  clear_line
21    jumpto taskTable, tasknum
22 taskTable:
23    RCALL task0
24    RET
25    RCALL task1
26    RET
27    RCALL task2
28    RET
29    RCALL task3
30    RET
31    RCALL task4
32    RET
33    RCALL task5
34    RET
35    RCALL  task6
36    RET
37    RCALL  task7
38    RET
39    RCALL  task8
40    RET
41    RCALL  task9
42          RET
43    RCALL  task10
44          RET
45    RCALL  task11
46    RET
47
48 task0:
49    CALL  send_menu
50    RET
51 task1:
52    LDI   tmp1, 1
53    MOV   dstep, tmp1
```

```
54    LDI    arg1, 10                    ; set number of steps
55    LDI    arg2, 100
56    RJMP   do_motor_task
57  task2:
58    LDI    tmp1, -1
59    MOV    dstep, tmp1
60    LDI    arg1, 10                    ; set number of steps
61    LDI    arg2, 100
62    RJMP   do_motor_task
63  task3:
64    LDI    tmp1, 2
65    MOV    dstep, tmp1
66    LDI    arg1, 80                    ; set number of steps
67    LDI    arg2, 25
68    RJMP   do_motor_task
69  task4:
70    LDI    tmp1, -2
71    MOV    dstep, tmp1
72    LDI    arg1, 80                    ; set number of steps
73    LDI    arg2, 25
74    RJMP   do_motor_task
75  task5:
76    LDI    tasknum, 0x00
77    CALL   stepper_disable
78    RET
79  task6:
80    LDI    tasknum, 0x00
81    CALL   stepper_enable
82    RET
83  task7:
84    LDI    tasknum, 0x00
85    CALL   convert_voltage
86    RET
87  task8:
88    LDI    tasknum, 0x00
89    CALL   adc_disable
90    CALL   clear_lights
91    RET
92  task9:
93    LDI    tasknum, 0x00
94    LDI    XH, HIGH(RAMMESSAGE3)
95    LDI    XL, LOW(RAMMESSAGE3)
96    CALL   message_to_LCD
97    RET
98  task10:
99    LDI    tasknum, 0x00
100   CALL   clear_LCD
101   RET
102 task11:
103   NOP
```

```
104   NOP
105   RJMP  task11
106   RET
107
108 do_motor_task:
109   CALL  stepper_is_enabled
110   SBRS  retReg, 0
111   RET
112   CBI   UCSRB, RXEN
113   CALL  clear_terminal
114   CALL  enable_buttons
115   CALL  step_motor
116   RET
117
118
119 adc_done:
120   MOV   arg1, retReg
121   CALL  output_lights
122   RET
123
124 stepper_done:
125   LDI   tasknum, 0x00
126   SBI   UCSRB, RXEN               ; Re-enable receiving
127   RCALL do_task
128   CALL  disable_buttons
129   RET
130
131 stepper_off_handle:
132   LDI   tasknum, 0x00
133   SBI   UCSRB, RXEN               ; Re-enable receiving
134   CALL  clear_line
135   RET
136
137 menu_text:          .db 0x0C,"Project Tasks: ",0x0d,0x0a
138 menu_text1:         .db "--------------",0x0d,0x0a
139 menu_text2:         .db "1) Rotate clockwise for 5 seconds ",0x0d,0x0a
140 menu_text3:         .db "2) Rotate anti-clockwise for 5 seconds",0x0d,0
      x0a
141 menu_text4:         .db "3) Rotate clockwise for 10 seconds",0x0d,0x0a
142 menu_text5:         .db "4) Rotate anti-clockwise for 10 seconds ",0x0d
      ,0x0a
143 menu_text6:         .db "5) Disable stepper",0x0d,0x0a
144 menu_text7:         .db "6) Enable stepper ",0x0d,0x0a
145 menu_text8:         .db "7) Start ADC PWM task ",0x0d,0x0a
146 menu_text9:         .db "8) Stop ADC PWM task",0x0d,0x0a
147 menu_text10:        .db "9) Print 3rd stored message to LCD",0x0d,0x0a
148 menu_text11:        .db "10) Clear LCD ",0x0d,0x0a
149 menu_text12:        .db "11) Reset Microcontroller ",0x0d,0x0a,0x00,0
      x00
150 blankterminal:      .db 0x0C,0x00
```

```
151 blankline:          .db 0x0D,"                ", 0x0D,0x00,0x00
```

IO.asm

```
1  .DSEG
2  lightValue: .BYTE 1
3
4  .CSEG
5  init_IO:
6    IN    tmp1, DDRD
7    ANDI  tmp1, 0xF0
8    OUT   DDRD, tmp1              ; set up the d pins data directions
9    LDI   tmp1, 0x7E
10   OUT   DDRA, tmp1              ; set up the d pins data directions
11   IN    tmp1, PORTD
12   ANDI  tmp1, 0xF0
13   ORI   tmp1, 0x0C
14   OUT   PORTD, tmp1             ; enable pull ups for buttons
15   LDI   tmp1, 0x0A
16   OUT   MCUCR, tmp1             ; falling edge triggered.
17   IN    tmp1, TIMSK             ; timer 1 for PWM
18   ANDI  tmp1, 0xC3
19   ORI   tmp1, 0x14              ; enable overflow and OCA interupts
        for timer 1
20   OUT   TIMSK, tmp1
21   RET
22
23 enable_buttons:
24   LDI   tmp1, 0xC0              ; 0b11000000
25   OUT   GICR, tmp1              ; int0 and int1 enabled
26   OUT   GIFR, tmp1              ; clear int0 flag and in1 flags
27   RET
28
29 disable_buttons:
30   OUT   GICR, zero              ; int0 and int1 disabled
31   RET
32
33 output_lights:
34   OUT   TCCR1A, zero
35   LDI   tmp1, 0x01              ; prescalar of 1, 8 makes the timer
        too slow
36   OUT   TCCR1B, tmp1
37   LDI   tmp2, 0x3F
38   MUL   tmp2, arg1
39   ADD   MUL_LOW, tmp2
40   ADC   MUL_HIGH, zero
41   OUT   OCR1AH, MUL_HIGH
42   OUT   OCR1AL, MUL_LOW
43   LDI   XH, HIGH(lightValue)
```

```
44    LDI    XL, LOW(lightValue)
45    LSR    arg1
46    ANDI   arg1, 0x7E
47    ST     X, arg1
48    RET
49
50  clear_lights:
51    OUT    TCCR1A, zero
52    OUT    TCCR1B, zero
53    IN     tmp1, PORTA
54    ANDI   tmp1, 0x81
55    OUT    PORTA, tmp1
56    RET
57
58  int0_ISR:
59    CALL   pause_stepper
60    OUT    GICR, ZERO                ; disable external interrupts
61    IN     tmp1, TIMSK
62    ANDI   tmp1, 0x2F                  ; mask off TIMSK
63    ORI    tmp1, 0x40                  ; enable overflow interupts
64    OUT    TIMSK, tmp1
65    LDI    tmp1, 0x05
66    OUT    TCCR2, tmp1               ; start timer2 with prescaler set to
          /1024
67    RETI
68
69  int1_ISR:
70    CALL   start_stepper
71    OUT    GICR, ZERO                  ; disable external interrupts
72    IN     tmp1, TIMSK
73    ANDI   tmp1, 0x2F                  ; mask off TIMSK
74    ORI    tmp1, 0x40                  ; enable overflow interupts
75    OUT    TIMSK, tmp1
76    LDI    tmp1, 0x05
77    OUT    TCCR2, tmp1               ; start timer2 with prescaler set to
          /1024
78    RETI
79
80  t2_OV_ISR:
81    OUT    TCCR2, zero               ; stop counter
82    OUT    TCNT2, zero               ; zero counter
83    RCALL  enable_buttons
84    RETI
85
86  t1_OCA_ISR:
87    IN     tmp1, PORTA
88    ANDI   tmp1, 0x81
89    OUT    PORTA, tmp1
90    RETI
91
```

```
92  t1_OV_ISR:
93    LDI   XH, HIGH(lightValue)
94    LDI   XL, LOW(lightValue)
95    LD    tmp1, X
96    IN    tmp2, PORTA
97    ANDI  tmp2, 0x81
98    OR    tmp1, tmp2
99    OUT   PORTA, tmp1
100   RETI
```

stepperMotor.asm

```
1   ;           This file is responsible for handling the stepper motor
2   ; J L Gouws 19G4436
3   ;
4
5   .DSEG
6   RAM_STEPS: .BYTE  9      ;
7
8   .CSEG
9   init_stepper:
10    IN    tmp1, DDRD
11    ANDI  tmp1, 0x0F                ; lower bits of the DDRD
12    LDI   tmp2, 0xF0
13    OR    tmp1, tmp2                ; set D4-D7 to output
14    OUT   DDRD, tmp1
15    IN    tmp1, PORTD
16    ANDI  tmp1, 0x0F                ; lower bits of the PORTD
17    ORI   tmp1, 0x10                ; lock motor on first driver
18    OUT   PORTD, tmp1
19    EOR   step, step                ; set step to 0
20    LDI   ZH, HIGH(2 * STEP_TABLE)
21    LDI   ZL, LOW(2 * STEP_TABLE)
22    CALL  read_steps_to_RAM
23    RCALL init_timer0
24    RET
25
26  init_timer0:
27    IN    tmp1, TIMSK
28    ANDI  tmp1, 0xFC                ; mask off TIMSK
29    ORI   tmp1, 0x02
30    OUT   TIMSK, tmp1
31    LDI   tmp1, 156                 ; output compare 0.125 x 10^{-6} x
          256 x 156 = 4.992ms
32    OUT   OCR0, tmp1
33    LDI   tmp1, 0x00                ; reset timer0s counter
34    OUT   TCNT0, tmp1
35    RET
36
```

```
37 step_motor:
38   MOV   slow, ZERO
39   MOV   stepCount, ZERO
40   RCALL start_stepper
41   RET
42
43 read_steps_to_RAM:
44   LDI   XH, HIGH(RAM_STEPS)
45   LDI   XL, LOW(RAM_STEPS)
46 readPGM:
47   LPM   tmp1, Z+
48   ST    X+, tmp1
49   CPI   tmp1, 0x00
50   BRNE  readPGM
51   RET
52
53 ; makes the motor make one step
54 NEXTSTEP:
55   ADD   step, dstep                ; get to the next step
56   LDI   tmp1, 0x07
57   AND   step, tmp1
58   INC   stepCount                  ; get to the next step
59   LDI   XH, HIGH(RAM_STEPS)
60   LDI   XL, LOW(RAM_STEPS)
61   ADD   XL, step
62   ADC   XH, zero
63   LD    tmp2, X
64   IN    tmp1, PORTD
65   ANDI  tmp1, 0x0F                 ; tmp1 now contains the masked off
         values
66                                    ; of portD
67   OR    tmp1, tmp2
68   OUT   PORTD, tmp1
69   RET
70
71 t0_OC_ISR:
72   OUT   TCCR0, zero                ; stop timer 0
73   INC   slow
74   CP    slow, arg2
75   BRNE  contSteps
76   MOV   slow, zero
77   RCALL nextstep
78   CP    stepCount, arg1
79   BRNE  contSteps
80   LDI   stepCount, 0x00;
81   MOV   slow, zero
82   CALL  stepper_done
83   RETI                             ; this is done
84 contSteps:
85   RCALL start_stepper
```

```
86    RETI
87
88  stepper_disable:
89    IN      tmp1, TIMSK
90    ANDI    tmp1, 0xFC                  ; mask off TIMSK
91    OUT     TIMSK, tmp1
92    IN      tmp1, PORTD
93    ANDI    tmp1, 0x0F                  ; tmp1 now contains the masked off
          values
94                                        ; of portD
95    OUT     PORTD, tmp1                 ; the lower bits of PORTD are now
          off
96    RET
97
98  stepper_is_enabled:
99    IN      tmp1, TIMSK
100   ANDI    tmp1, 0x03                ; mask off TIMSK
101   MOV     retReg, tmp1
102   ANDI    tmp1, 0x01                ; mask off TIMSK
103   LSR     retReg
104   OR      retReg, tmp1
105   RET
106
107 stepper_enable:
108   RCALL   init_timer0
109   LDI     XH, HIGH(RAM_STEPS)
110   LDI     XL, LOW(RAM_STEPS)
111   ADD     XL, step
112   ADC     XH, zero
113   LD      tmp2, X
114   IN      tmp1, PORTD
115   ANDI    tmp1, 0x0F
116   OR      tmp1, tmp2
117   OUT     PORTD, tmp1
118   RET
119
120 pause_stepper:
121   OUT     TCCR0, zero                 ; stop timer 0
122   RET
123
124 start_stepper:
125   OUT     TCNT0, zero
126   LDI     tmp1, 0x04                  ; 0b00000100 | clock prescalar 256
127   OUT     TCCR0, tmp1                 ; stop timer 0
128   RET
129
130 STEP_TABLE:        .db 0x10, 0x30, 0x20, 0x60, 0x40, 0xC0, 0x80, 0x90, 0
      x00, 0x00
```

UART.asm

```
1  ;                File for handling UART and related things
2
3  init_UART:
4    ;set baud rate (9600,8,n,2)
5          LDI    tmp1, 51
6          LDI    tmp2, 0x00
7          OUT    UBRRH, tmp2
8          OUT        UBRRL, tmp1
9    ; set rx and tx enable
10         SBI    UCSRB, RXEN
11         SBI    UCSRB, TXEN
12   ; enable uart interrupts, both transmit and receive
13         SBI    UCSRB, RXCIE
14         SBI    UCSRB, TXCIE
15         RET
16
17 send_menu:
18   LDI    ZH, HIGH(2 * menu_text)
19   LDI    ZL, LOW(2 * menu_text)
20   LPM    tmp1, Z+
21   OUT    UDR, tmp1
22   RET
23
24 clear_terminal:
25   LDI    ZH, HIGH(2 * blankterminal)
26   LDI    ZL, LOW(2 * blankterminal)
27   LPM    tmp1, Z+
28   OUT    UDR, tmp1
29   RET
30
31 clear_line:
32   LDI    ZH, HIGH(2 * blankline)
33   LDI    ZL, LOW(2 * blankline)
34   LPM    tmp1, Z+
35   OUT    UDR, tmp1
36   RET
37
38 wait_transmit:
39   SBIS   UCSRA, TXC                ; wait for bit data to be sent
40   RJMP   wait_transmit
41   SBI    UCSRA, TXC
42   RET
43
44 URXC_ISR: ; on receive
45   LDI    XH, HIGH(TASK_NUM_RAM)
46   LDI    XL, LOW(TASK_NUM_RAM)
47   IN     tmp1, UDR
48   CPI    tmp1, '.'
49   BREQ   set_task
50   OUT    UDR, tmp1                 ; echo to terminal
```

```
51    RCALL wait_transmit
52    LD    tmp3, X                    ; get current stored number
53    LDI   tmp2, 10
54    MUL   tmp3, tmp2                 ; multiply tmp3 by tmp2
55    CP    MUL_HIGH, zero
56    BRNE  reset_task
57    MOV   tmp3, MUL_LOW
58    SUBI  tmp1, '0'
59    BRMI  reset_task
60    ADD   tmp3, tmp1
61    ST    X, tmp3
62    RETI
63  set_task:
64    LD    tasknum, X
65    CPI   tasknum, 0x00
66    BRLT  reset_task
67    CPI   tasknum, 0x0C
68    BRGE  reset_task
69    RJMP  do_set_task
70  reset_task:
71    LDI   tasknum, 0x0
72  do_set_task:
73    ST    X, zero                    ; clear the input register.
74    CALL  do_task
75    RETI
76
77
78  UTXC_ISR: ; continue transmitting
79    LPM   tmp1, Z+
80    CPI   tmp1, 0x00
81    BREQ  donetx
82    OUT   udr, tmp1
83  donetx:
84    RETI
85
86  sendfromram:
87    LD tmp1, x+
88    CPI tmp1, 0x00
89    BREQ txcexit
90    OUT UDR, tmp1
91  txcexit:
92    RETI
```

EEP.asm

```
1  .ESEG
2  EEMSG: .DB "Why am I here",0x00,"Not this message",0x00,"The message is
       long",0x00
3
```

```
4  .DSEG
5  RAMMESSAGE1: .BYTE 20
6  RAMMESSAGE2: .BYTE 20
7  RAMMESSAGE3: .BYTE 20
8
9  .CSEG
10 ; Reads the EEPROM messages into RAM
11 init_EEP:
12   LDI   XH, HIGH(RAMMESSAGE1)
13   LDI   XL, LOW(RAMMESSAGE1)
14   LDI   YH, HIGH(EEMSG)
15   LDI   YL, LOW(EEMSG)
16   CALL  read_EEP
17   LDI   XH, HIGH(RAMMESSAGE2)
18   LDI   XL, LOW(RAMMESSAGE2)
19   CALL  read_EEP
20   LDI   XH, HIGH(RAMMESSAGE3)
21   LDI   XL, LOW(RAMMESSAGE3)
22   CALL  read_EEP
23   RET
24
25 ; Reads an individual message into RAM, stops on null byte
26 ; Y -- Location of first byte in EEPROM to read
27 ; X -- Location to store byte in RAM
28 read_EEP:
29   OUT   EEARH, YH
30   OUT   EEARL, YL
31   SBI   EECR, EERE   ; read from EEPROM
32   IN    tmp1, EEDR
33   ST    X+, tmp1
34   ADIW  YL, 1
35   CPI   tmp1, 0x00
36   BRNE  read_EEP
37   RET
```

ADC.asm

```
1  convert_voltage:
2    LDI   tmp1, 0b01100000          ; AVCC selected as reference
3    OUT   ADMUX, tmp1               ; ADLAR set so most significant 8
         bits are in ADCH
4    LDI   tmp1, 0b11001111          ; ADC enabled conversion started no
         auto trigger
5                                    ; 128 prescaler interrupt enabled
6    OUT   ADCSRA, tmp1
7    RET
8
9  adc_disable:
10   LDI   tmp1, 0x00                ; AVCC selected as reference
```

```
11    OUT    ADMUX, tmp1                ; ADLAR set so most significant 8
          bits are in ADCH
12    LDI    tmp1, 0x00                 ; ADC enabled conversion started no
          auto trigger
13                                       ; 128 prescaler interrupt enabled
14    OUT    ADCSRA, tmp1
15
16  ADC_ISR:
17    IN     retReg, ADCH
18    CALL   adc_done
19    RETI
```

LCD.asm

```
1  ;                          LCD file
2  .MACRO LCD_WRITE
3    CBI PORTB, 1
4  .ENDMACRO
5  .MACRO LCD_READ
6    SBI PORTB, 1
7  .ENDMACRO
8  .MACRO LCD_E_HI
9    SBI PORTB, 0
10 .ENDMACRO
11 .MACRO LCD_E_LO
12   CBI PORTB, 0
13 .ENDMACRO
14 .MACRO LCD_RS_HI
15   SBI PORTB, 2
16 .ENDMACRO
17 .MACRO LCD_RS_LO
18   CBI PORTB, 2
19 .ENDMACRO
20
21 ;This is a one millisecond delay
22 Delay:
23    PUSH   r16
24    LDI    r16, 11
25 Delayloop1:
26    PUSH   r16
27    LDI    r16, 239 ; for an 8MHz xtal
28 Delayloop2:
29    DEC    r16
30    BRNE   Delayloop2
31    POP    r16
32    DEC    r16
33    BRNE   Delayloop1
34    POP    r16
35    RET
```

```
36 ; waits 800 clock cycles (0.1ms on 8MHz clock)
37 Waittenth:
38   PUSH  r16
39   LDI   r16, 255
40 decloop:
41   DEC   r16
42   NOP
43   NOP
44   BRNE  decloop
45   POP   r16
46   RET
47
48 ; return when the lcd is not busy
49 Check_busy:
50   PUSH  r16
51   LDI   r16, 0b00000000
52   OUT   DDRC, r16        ; portc lines input
53   LCD_RS_LO      ;RS lo
54   LCD_READ       ;read
55 Loop_Busy:
56   RCALL Delay    ; wait 1ms
57   LCD_E_HI       ; E hi
58   RCALL Delay
59   IN    r16, PINC        ; read portc
60   LCD_E_LO       ; make e low
61   SBRC  r16, 7   ; check the busy flag in bit 7
62   RJMP  Loop_busy
63   LCD_WRITE      ;
64   LCD_RS_LO      ; rs lo
65   POP   r16
66   RET
67
68 ; write char in r16 to LCD
69 Write_char:      ;rcall Check_busy
70   PUSH  r17
71   RCALL Check_busy
72   LCD_WRITE
73   LCD_RS_HI
74   SER   r17
75   OUT   DDRC, r17        ; c output
76   OUT   PORTC, R16
77   LCD_E_HI
78   LCD_E_LO
79   CLR   r17
80   OUT   DDRC, r17
81   ;rcall delay
82   POP   r17
83   RET
84 ;write instruction in r16 to LCD
85 Write_instruc:
```

```
86    PUSH  r17
87    RCALL Check_busy
88    LCD_WRITE
89    LCD_RS_LO
90    SER   r17
91    OUT   DDRC, r17        ; c output
92    OUT   PORTC, R16
93    ;rcall delay
94    LCD_E_HI
95    LCD_E_LO
96    CLR   r17
97    OUT   DDRC, r17
98        ;rcall delay
99    POP   r17
100   RET
101
102
103 Init_LCD:
104   PUSH  r16
105   CLR   r16
106   OUT   DDRC, r16
107   OUT   PORTC, r16
108   SBI   DDRB, 2 ;reg sel output
109   SBI   DDRB, 0 ; enable output
110   SBI   PORTB, 2
111   SBI   PORTB, 0
112   SBI   DDRB, 1 ; rw output
113   LDI   r16, 0x38
114   RCALL Write_instruc
115   LDI   r16, 0x0E ; turn lcd on with cursor
116   RCALL Write_instruc
117   LDI   r16, 0x06
118   RCALL Write_instruc
119   LDI   r16, 0x01
120   RCALL Write_instruc
121   POP   r16
122   RET
123
124  message_to_LCD:
125    LDI  tmp2, 0x00
126  one_char:
127    CPI  tmp2, 16
128    BRNE write_one_char
129    LDI  tmp1, 0b11000000
130    CALL Write_instruc
131  write_one_char:
132    LD   tmp1, X+
133    CPI  tmp1, 0x00
134    BREQ m2ldone
135    CALL write_char
```

```
136     INC   tmp2
137     RJMP  one_char
138   m2ldone:
139     RET
140
141 clear_LCD:
142     CBI     PORTB, 0
143     CBI     PORTB, 1
144     LDI     R16, 0x01
145     RCALL   write_instruc
146     RET
```

watchdog.asm

```
1  init_watchdog:
2     LDI     tmp1, 0x0A                ; Enable watchdog 65ms time out
3     OUT     WDTCR, tmp1
4     RET
```