

```

;*****
;* Titolo:                AVR_PSS.ASM
;* Applicazione:         Alimentatore stabilizzato con controllo  Vout ed Iout
;* Versione:             0.2
;* Last Updated:        19.09.2006
;* Target:              ATmega8535 clock interno a 8MHz
;**** Registers definitions ****
.def VA1                =r1   ; prime due cifre x voltmetro A
.def VA2                =r2   ; ultime due cifre x voltmetro A
.def VB1                =r3   ; prime due cifre x voltmetro B
.def VB2                =r4   ; ultime due cifre x voltmetro B
.def adc_low            =r9
.def adc_high           =r10
.def temp              =r16   ; "general scratch"
.def AVAL              =r17   ; registro della porta A
.def BVAL              =r18   ; registro della porta B
.def CVAL              =r19   ; registro della porta C
.def DVAL              =r20   ; registro della porta D
.def flagvari          =r21   ; flag
.def count             =r22   ; scansione bit della tastiera
.def conta             =r23   ; conteggi
.def conta1            =r24   ;
.def conta2            =r25   ;
.def conta3            =r26   ;
.def icount            =r27   ; contatore inizio pos.
.def fcount            =r28   ; contatore fine pos.
.equ FAN_BIT           =5     ; bit di controllo ventola
.equ FAN               = (1<<FAN_BIT)
;**** include files ****
.nolist
.include "m8535def.inc"
.list
;**** Source Code ****
.cseg                ; segmento di CODICE
.org 0x0000
    rjmp reset        ; partenza da reset
.org 0x000E
    rjmp adc_read     ; lettura ADC
.org 0x0020
    ; ***** ID Tag *****
.db "VAPRSSv .0 2"   ; AVR_PSS v0.2
reset:
    ; ***** Inizializzazione *****
.equ RAMINI = 0x0060
stack_init:
    ; posizionamento dello stack pointer
    ldi    r16,low(RAMEND)
    out   SPL,r16
    ldi    r16,high(RAMEND)
    out   SPL+1,r16
; **** Definizioni delle porte utilizzate ****
; Porta A  segnali ADC
; Porta C  Out  dati/comandi a LCD (8 bit)
; Porta D  controllo LCD  e Ventola
; **** SET-UP INIZIALE ****
set_port:
    ldi    temp,0x0F        ; Port A usata come output
    out   DDRA,temp        ; pred. la porta
    ser   temp              ; Port C usata come Output (1=output)
    out   DDRC,temp        ; pred. la porta
    clr   CVAL              ; azzero tutti i bit
    out   PORTC,CVAL       ; ed il registro della porta
    ldi    temp,0xFE        ; Port D usata come Input/Output 1=output 0=input (Rx)
    out   DDRD,temp        ; pred. la porta
    clr   DVAL              ; set-up del registro (tutto a 0)
    out   PORTD,DVAL
    clr   temp
    out   EEARH,temp
    rcall init_lcd         ; inizializzazione modulo LCD
inizio:ldi    icount,banner1 ; punto al primo mess. della EEPROM
    ldi    fcount,banner2  ; punto al messaggio successivo
    rcall leggoeprom       ; scrivo il primo messaggio
    rcall attesa
    ldi    conta3,80       ; tempo di visualizzazione iniziale
girol:rcall long_delay
    dec   conta3
    brne girol
    ldi    CVAL,HOME_2     ; primo carattere della seconda riga
    rcall wc_lcd
    rcall attesa
    ldi    icount,banner2  ; punto la secondo mess. della EEPROM
    ldi    fcount,prima_r
    rcall leggoeprom       ; scrivo il secondo messaggio
    rcall attesa
    ldi    conta3,60       ; perdo tempo

```

Listato AVR\_PSS.ASM

```

giroll:rcall    long_delay
dec            conta3
brne          giroll
ldi           CVAL,CL_LCD      ; pulitura display
rcall        wc_lcd
rcall        attesa
ldi           icount,prima_r   ; punto al primo mess. della EEPROM
ldi           fcount,seconda_r ; punto al messaggio successivo
rcall        leggoeeprom      ; scrivo il primo messaggio
rcall        attesa
ldi           CVAL,HOME_2     ; primo carattere della seconda riga
rcall        wc_lcd
rcall        attesa
ldi           icount,seconda_r ; punto al primo mess. della EEPROM
ldi           fcount,lowi     ; punto al messaggio successivo
rcall        leggoeeprom      ; scrivo il primo messaggio
rcall        attesa
clr           adc_low         ;set_adc
clr           adc_hig
ldi           temp,$86        ; abilit. ADC + prescaler a 64
out           ADCSR,temp
sei
main: ldi     temp,$67         ; sel CH#7, Vref=VCC, left adj (lettura voltmetro A)
out           ADMUX,temp
ldi           temp,$CE        ; con interrupt
out           ADCSR,temp      ; start conversione!
rcall        long_delay       ; aspetto che sia completata
rcall        long_delay
ldi           CVAL,$80        ; mi posiziono per scrivere
rcall        wc_lcd
rcall        attesa
rcall        calc_dec1        ; calcolo i valori del 1o voltmetro
rcall        visual_a         ; li visualizzo
rcall        long_delay
rcall        long_delay
ldi           temp,$66        ; sel CH#6, Vref=VCC, left adj (lettura voltmetro B)
out           ADMUX,temp
ldi           temp,$CE        ; con interrupt
out           ADCSR,temp      ; start conversione!
rcall        long_delay
rcall        long_delay
ldi           CVAL,$8A        ; mi posiziono per scrivere
rcall        wc_lcd
rcall        attesa
rcall        calc_dec2        ; calcolo i valori del 2o voltmetro
rcall        visual_b         ; li visualizzo
rcall        long_delay
rcall        long_delay
ldi           temp,$45        ; sel CH#5, Vref=VCC, righth adj (lettura amperometro A)
out           ADMUX,temp
ldi           temp,$CE        ; con interrupt
out           ADCSR,temp      ; start conversione!
rcall        long_delay
rcall        long_delay
ldi           CVAL,$C0        ; mi posiziono per scrivere
rcall        wc_lcd
rcall        attesa
rcall        verifica_ia      ; leggo il valore
rcall        long_delay
rcall        long_delay
ldi           temp,$44        ; sel CH#4, Vref=VCC, righth adj (lettura amperometro B)
out           ADMUX,temp
ldi           temp,$CE        ; con interrupt
out           ADCSR,temp      ; start conversione!
rcall        long_delay
rcall        long_delay
ldi           CVAL,$CA        ; mi posiziono per scrivere
rcall        wc_lcd
rcall        attesa
rcall        verifica_ib      ; verifica presenza corrente sul canale A
rcall        long_delay
rcall        long_delay
rjmp         main
; riprendo dall'inizio
; ***** funzioni del programma *****
verifica_ia: ; verifica presenza corrente sul canale A
ldi           count,$FC       ; predispongo un valore di soglia
cp            adc_low,count
brlo         low_a            ; verifica se c'è consumo
ldi           icount,vuoto     ; altrimenti cancello scritta
ldi           fcount,fine
rcall        leggoeeprom
cbr          DVAL,FAN         ; spengo la ventola

```

```

        out        PORTD,DVAL
        rjmp       fineia
low_a:ldi        count,$EF        ; predispongo un valore di soglia
        cp        adc_low,count
        brlo      mid_a          ; verifica consumo
        ldi       icount,lowi    ; altrimenti scrivo
        ldi       fcount,midi
        rcall     leggoeeprom
        cbr       DVAL,FAN       ; spengo la ventola
        out       PORTD,DVAL
        rjmp       fineia        ; finito lettura corrente canale A
mid_a:ldi        count,$E2        ; predispongo un valore di soglia
        cp        adc_low,count
        brlo      hig_a          ; verifica consumo
        ldi       icount,midi    ; altrimenti scrivo
        ldi       fcount,higa
        rcall     leggoeeprom
        sbr       DVAL,FAN       ; accendo la ventola
        out       PORTD,DVAL
        rjmp       fineia        ; finito lettura corrente canale A
hig_a:ldi        count,$80
        cp        adc_low,count
        brlo      fineia
        ldi       icount,higa    ; altrimenti scrivo
        ldi       fcount,venton
        rcall     leggoeeprom
        sbr       DVAL,FAN       ; accendo la ventola
        out       PORTD,DVAL
fineia:ret
verifica_ib:    ; verifica presenza corrente sul canale A
        ldi       count,$FA      ; predispongo un valore di soglia
        cp        adc_low,count
        brlo      low_b          ; verifica se c'è consumo
        ldi       icount,vuoto   ; altrimenti cancello scritta
        ldi       fcount,fine
        rcall     leggoeeprom
        cbr       DVAL,FAN       ; spengo la ventola
        out       PORTD,DVAL
        rjmp       fineib        ; finito lettura corrente canale B
low_b:ldi        count,$ED        ; predispongo un valore di soglia
        cp        adc_low,count
        brlo      mid_b          ; verifica consumo
        ldi       icount,lowi    ; altrimenti scrivo
        ldi       fcount,midi
        rcall     leggoeeprom
        cbr       DVAL,FAN       ; spengo la ventola
        out       PORTD,DVAL
        rjmp       fineib        ; finito lettura corrente canale B
mid_b:ldi        count,$E2        ; predispongo un valore di soglia
        cp        adc_low,count
        brlo      hig_b          ; verifica consumo
        ldi       icount,midi    ; altrimenti scrivo
        ldi       fcount,higa
        rcall     leggoeeprom
        sbr       DVAL,FAN       ; accendo la ventola
        out       PORTD,DVAL
        rjmp       fineib        ; finito lettura corrente canale B
hig_b:ldi        count,$80
        cp        adc_low,count
        brlo      fineib        ; finito lettura corrente canale B
        ldi       icount,higa    ; altrimenti scrivo
        ldi       fcount,venton
        rcall     leggoeeprom
        sbr       DVAL,FAN       ; accendo la ventola
        out       PORTD,DVAL
fineib:ret
calc_dec1:    ; calcolo dei decimali voltmetro#1
        clr       VAL            ; parto con i registri azzerati
        ldi       count,$50      ; valore base (5V)
        clr       contal
p_contr:cp     count,adc_hig      ; confronto il valore da ADC
        breq      offs_ok        ; se sono uguali, offset ok, esco
        inc       count          ; altrimenti incremento conteggio
        rjmp     p_contr         ; e ricontrollo
offs_ok:ldi    icount,tab_val    ; punto tabella conv. unità
        mov       contal,count
        andi      contal,0xF0    ; isolo le unità
        swap      contal        ; scambio nibble
        add       icount,contal
        rcall     sleepro        ;leggo i decimali
        mov       VAL,temp      ; salvo
        ldi       icount,tab_dec ; punto tabella conv. decimali

```

```

mov     contal,count
andi   contal,0x0F      ; isolo i decimali
add    icalount,contal ; aggiorno pointer tabella dec.
rcall  sleeptro        ;leggo i decimali
mov    VA2,temp        ; salvo
ret                                         ; posso uscire avendo i valori da visualizzare
calc_dec2:
        ; calcolo dei decimali voltmetro#2
clr    VB1             ; parto con i registri azzerati
ldi    count,$50      ; valore base (5V)
clr    contal
p_contrl:cp
count,adc_hig         ; confronto il valore da ADC
breq   offs_o         ; se sono uguali, offset ok, esco
inc    count          ; altrimenti incremento conteggio
rjmp   p_contrl       ; e ricontrollo
offs_o:ldi
icalount,tab_val     ; punto tabella conv. unita
mov    contal,count
andi   contal,0xF0    ; isolo le unita
swap   contal         ; scambio nibble
add    icalount,contal
rcall  sleeptro        ;leggo i decimali
mov    VB1,temp       ; salvo
ldi    icalount,tab_dec ; punto tabella conv. decimali
mov    contal,count
andi   contal,0x0F    ; isolo i decimali
add    icalount,contal ; aggiorno pointer tabella dec.
rcall  sleeptro        ;leggo i decimali
mov    VB2,temp       ; salvo
ret                                         ; posso uscire avendo i valori da visualizzare
visual_a:
        ; visualizzazione dei valori (Voltmetro A)
mov    temp,VA1       ; carico digit MSD (1a cifra)
andi   temp,$F0       ; isolo il digit
swap   temp           ; lo sposto
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
mov    temp,VA1       ; carico digit LSD (2a cifra)
andi   temp,$0F       ; isolo il digit
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
ldi    CVAL,'.'       ; punto
rcall  wd_lcd         ; lo visualizzo
rcall  attesa
mov    temp,VA2       ; carico digit MSD (3a cifra)
andi   temp,$F0       ; isolo il digit
swap   temp           ; scambio posizione
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
mov    temp,VA2       ; carico digit LSD (4a cifra)
andi   temp,$0F       ; isolo il digit
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
ret
visual_b:
        ; visualizzazione dei valori (Voltmetro B)
mov    temp,VB1       ; carico digit MSD (1a cifra)
andi   temp,$F0       ; isolo il digit
swap   temp           ; lo sposto
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
mov    temp,VB1       ; carico digit LSD (2a cifra)
andi   temp,$0F       ; isolo il digit
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd
rcall  attesa
ldi    CVAL,'.'       ; punto
rcall  wd_lcd         ; lo visualizzo
rcall  attesa
mov    temp,VB2       ; carico digit MSD (3a cifra)
andi   temp,$F0       ; isolo il digit
swap   temp           ; scambio posizione
ldi    CVAL,$30       ; preparo il valore ascii
add    CVAL,temp      ; lo posso visualizzare
rcall  wd_lcd

```

```

    rcall    attesa
    mov     temp,VB2          ; carico digit LSD (4a cifra)
    andi   temp,$0F         ; isolo il digit
    ldi    CVAL,$30         ; preparo il valore ascii
    add    CVAL,temp        ; lo posso visualizzare
    rcall  wd_lcd
    rcall  attesa
    ret

adc_read:    ; lettura dell'ADC su interrupt
    in     temp,ADCL        ; leggo il byte inferiore
    mov    adc_low,temp     ; salvo il valore
    in     temp,ADCH        ; leggo il byte superiore
    mov    adc_hig,temp     ; salvo il valore
    reti

sleepro:out EEARL,icount    ; lettura singolo byte della eeprom
    sbi    EECR,EERE
    in     temp,EEDR        ; leggo
    ret

leggoeeprom:clr temp
    out    EEARH,temp
punto:out    EEARL,icount    ; posizione inizio messaggio
    sbi    EECR,EERE        ; strobe alla eeprom
    in     temp,EEDR        ; leggo il dato
datoaLCD:mov CVAL,temp
    rcall  wd_lcd
    rcall  attesa
    inc    icount           ; incremento la posizione
    cp     icount,fcount    ; fino alla fine del messaggio
    brne   punto
    ret

long_delay:
    ldi    conta1,90
g_1: ldi    conta2,$FF
g_2: dec    conta2
    brne   g_2
    dec    conta1
    brne   g_1
    ret

.include "geslcd.asm"

.eseg      ;***** Segmento dati in EEPROM *****
.org 0x0000
banner1:  .db "PSS-IW3QBN v1.0 "      ; messaggio all'accensione (16 crt)
banner2:  .db " ATmega32 "
prima_r:  .db "XX.xxV   XX.xxV"
seconda_r: .db "loadA   loadB "
lowi:     .db "low I"
midi:     .db "mid I"
higa:     .db "hig I"
venton:   .db "**fan*"
vuoto:    .db " "
fine:     .db " "

tab_dec:  ; ad ogni valore (4bit) corrisponde una coppia di valori per i decimali
.db 0x00,0x06,0x12,0x18,0x25,0x31,0x37,0x43,0x50,0x56,0x62,0x68,0x75,0x81,0x87,0x93
tab_val:  ; ad ogni valore (4 bit) corrisponde una coppia di valori
.db 0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,0x11,0x12,0x13,0x14,0x15

```