```
.include "M8515def.inc" ;@18,432MHz

.def _NOP               =r0
.def _Read              =r1
.def _Write             =r2
.def _Precharge =r3
.def _Activate  =r4
.def _NC3               =r5
.def _NC4               =r6
.def Reload             =r7

.def Color              =r8
.def Byte               =r9
.def Byte0              =r10

.def temp               =r16            ;Allgemeines Hilfsregister
.def loop               =r17            ;Allgemeiner Schleifenzähler

.def temp2              =r18            ;Allgemeines Hilfsregister

.def ZeileL             =r19
.def ZeileH             =r20

.def AdressLL   =r26            ;Spaltenadresse Low
.def AdressLH   =r27            ;Spaltenadresse High
.def AdressHL   =r28            ;Zeilenadresse Low
.def AdressHH   =r29            ;Zeilenadresse High

.equ VStart     = 0                     ;Beginng von VSync in Zeile : 500 + x
.equ VLang      = 5                     ;VSync Länge in Zeilen
.equ Offs       = 15            ;Beginn des neuen Bildes, x Zeilen nach Ende von VSync

;IO Pins:
;PortA: Pixeldaten 0-7
;PortC: Adresse 0-7, RAMDAC Daten 0-7
.equ AdL                =PortC
;PortB: Adresse 8-11, BS0-1, RAMDAC WR, RAMDAC RD
.equ AdH                =PortB
.equ RS0                =1
.equ RS1                =0
.equ BS0                =4
.equ BS1                =5
.equ WR                 =6
.equ RD                 =7

;PortD
.equ Comm               =PortD
.equ RXD                =0
.equ TXD                =1
.equ DQM                =2
.equ CKE                =3
.equ CS                 =4
.equ RAS                =5
.equ CAS                =6
.equ WE                 =7

;Port E
.equ HSync = 0                  ;HSync direkt an Monitor
.equ VSync = 1                  ;VSync direkt an Monitor
.equ Blank = 2                  ;Blanking an den RAMDAC

        ;Command                        CKE                     CS              RAS
CAS                     WE              DQM
```

```
.equ Activate           =(1<<CKE)                                               |(1<<CAS)       |
(1<<WE)
.equ Read                       =(1<<CKE)                       |(1<<RAS)
|(1<<WE)
.equ Write                      =(1<<CKE)                       |(1<<RAS)
.equ Precharge          =(1<<CKE)                                               |(1<<CAS)
.equ ModeRegister       =(1<<CKE)
.equ NoOP                       =(1<<CKE)                       |(1<<RAS)       |(1<<CAS)       |
(1<<WE)
.equ AutoRefresh        =(1<<CKE)
|(1<<WE)
.equ SelfRefresh        =
(1<<WE) |(1<<DQM)
.equ SelfRefreshExit=(1<<CKE)                   |(1<<RAS)       |(1<<CAS)       |(1<<WE)        |
(1<<DQM)
.equ Deselect           =(1<<CKE)       |(1<<CS)
.equ BurstTerminate     =(1<<CKE)                       |(1<<RAS)       |(1<<CAS)
|(1<<DQM)

.org 0
        rjmp    RESET

.org OVF0addr
        out TCNT0, Reload
        cbi PortE, HSync                        ;H-Sync: Low
        rjmp Timer

RESET:
        ser     temp
        out PORTA, temp
        out     DDRB,temp
        out PORTB, temp
        out     DDRC,temp
        out     DDRD,temp
        out     DDRE,temp
        out PORTE, temp

        ldi temp, NoOP|(1<<DQM)
        out PORTD, temp

        clr temp
        out     DDRA,temp
        out PORTC, temp

        ldi temp, LOW(RAMEND)
        out SPL, temp
        ldi temp, HIGH(RAMEND)
        out SPH, temp

        ldi temp, NoOP
        mov _NOP, temp

        ldi temp, Read
        mov _Read, temp

        ldi temp, Write
        mov _Write, temp

        ldi temp, Precharge
        mov _Precharge, temp

        ldi temp, Activate
        mov _Activate, temp

        clr AdressLL
```

```
        ldi AdressLH, 192
        clr AdressHL
        ldi AdressHH, 192

        clr ZeileL
        ldi ZeileH, 192

        ldi temp, 64                                    ; 59 für 19,2kBaud bei 18,432MHz
        out UBRRL, temp                                 ; Baudrate einstellen
        ldi temp, (1<<TXEN)|(1<<RXEN)
        out UCsRb, temp


        ldi temp, (1<<TOIE0)                        ;Timer0 Interrupt aktivieren
        out TIMSK, temp

        ldi temp, 2                                             ;Teiler auf 8
        out TCCR0, temp

        ldi temp, 256-72                                ;32kHz Horizontalfrequenz
        mov Reload, temp
        out TCNT0, Reload

        rcall Init
        rcall Init

;       clr AdressLL
;writesd1:
;       mov temp, AdressLL
;       rcall WriteByte
;       ldi temp, 1
;       add AdressLL, temp
;       clr temp
;       adc AdressLH, temp
;       cpi AdressLH, 192+2
;       brne writesd1
;       ldi AdressLH, 192
;
;       ldi temp, 1
;       add AdressHL, temp
;       clr temp
;       adc AdressHH, temp
;       cpi AdressHH, 192+2
;       brne writesd1
;       ldi AdressHH, 192


        ldi AdressLL, 0
        ldi AdressLH, 192
        ldi AdressHL, 0
        ldi AdressHH, 192

writesdc:                               ;SDRAM löschen
        clr Byte

        rcall WriteLine

        cpi AdressLL, 0
        brne writesdc
        cpi AdressLH, 192
        brne writesdc

        ldi AdressLH, 192

        ldi temp, 1
```

```
        add AdressHL, temp
        clr temp
        adc AdressHH, temp
        cpi AdressHH, 192+2
        brne writesdc


        clr AdressLL
        ldi AdressLH, 192
        ldi AdressHL, 16
        ldi AdressHH, 192

        ldi temp, 20                            ;Textfarbe laden
        mov Color, temp

        clr ZeileL
        ldi ZeileH, 192

        clr AdressLL
        ldi AdressLH, 192
        ldi AdressHL, 0
        ldi AdressHH, 192

        ldi ZL, low(Text1*2)            ;Pointer auf Text laden
        ldi ZH, high(Text1*2)
        rcall Disptext                          ;Text ausgeben

        ldi ZL, low(Text2*2)
        ldi ZH, high(Text2*2)
        rcall Disptext

        ldi ZL, low(Text3*2)
        ldi ZH, high(Text3*2)
        rcall Disptext

        ldi ZL, low(Text4*2)
        ldi ZH, high(Text4*2)
        rcall Disptext

        ldi ZL, low(Text5*2)
        ldi ZH, high(Text5*2)
        rcall Disptext

        ldi ZL, low(Text6*2)
        ldi ZH, high(Text6*2)
        rcall Disptext

        ldi ZL, low(Text7*2)
        ldi ZH, high(Text7*2)
        rcall Disptext

        ldi ZL, low(Text8*2)
        ldi ZH, high(Text8*2)
        rcall Disptext

        ldi ZL, low(Text9*2)
        ldi ZH, high(Text9*2)
        rcall Disptext

        ldi ZL, low(Text10*2)
        ldi ZH, high(Text10*2)
        rcall Disptext

        ldi ZL, low(Text11*2)
        ldi ZH, high(Text11*2)
```

```
        rcall Disptext

        ldi ZL, low(Text12*2)
        ldi ZH, high(Text12*2)
        rcall Disptext

        ldi ZL, low(Text13*2)
        ldi ZH, high(Text13*2)
        rcall Disptext

        ldi ZL, low(Text14*2)
        ldi ZH, high(Text14*2)
        rcall Disptext

        ldi ZL, low(Text15*2)
        ldi ZH, high(Text15*2)
        rcall Disptext

        ldi ZL, low(Text16*2)
        ldi ZH, high(Text16*2)
        rcall Disptext

        ldi ZL, low(Text17*2)
        ldi ZH, high(Text17*2)
        rcall Disptext

        ldi ZL, low(Text18*2)
        ldi ZH, high(Text18*2)
        rcall Disptext

        ldi ZL, low(Text19*2)
        ldi ZH, high(Text19*2)
        rcall Disptext

        ldi ZL, low(Text20*2)
        ldi ZH, high(Text20*2)
        rcall Disptext

        ldi ZL, low(Text21*2)
        ldi ZH, high(Text21*2)
        rcall Disptext

        ldi ZL, low(Text22*2)
        ldi ZH, high(Text22*2)
        rcall Disptext

        ldi ZL, low(Text23*2)
        ldi ZH, high(Text23*2)
        rcall Disptext

        ldi ZL, low(Text24*2)
        ldi ZH, high(Text24*2)
        rcall Disptext

        ldi ZL, low(Text25*2)
        ldi ZH, high(Text25*2)
        rcall Disptext

        ldi ZL, low(Text26*2)
        ldi ZH, high(Text26*2)
        rcall Disptext

        ldi ZL, low(Text27*2)
        ldi ZH, high(Text27*2)
        rcall Disptext
```

```
        ldi ZL, low(Text28*2)
        ldi ZH, high(Text28*2)
        rcall Disptext

        ldi ZL, low(Text29*2)
        ldi ZH, high(Text29*2)
        rcall Disptext

        ldi ZL, low(Text30*2)
        ldi ZH, high(Text30*2)
        rcall Disptext

        ldi ZL, low(Text31*2)
        ldi ZH, high(Text31*2)
        rcall Disptext

        ldi ZL, low(Text32*2)
        ldi ZH, high(Text32*2)
        rcall Disptext

        ldi ZL, low(Text33*2)
        ldi ZH, high(Text33*2)
        rcall Disptext

        ldi ZL, low(Text34*2)
        ldi ZH, high(Text34*2)
        rcall Disptext

        ldi ZL, low(Text35*2)
        ldi ZH, high(Text35*2)
        rcall Disptext

        ldi ZL, low(Text36*2)
        ldi ZH, high(Text36*2)
        rcall Disptext

        ldi ZL, low(Text37*2)
        ldi ZH, high(Text37*2)
        rcall Disptext

        ldi ZL, low(Text28*2)
        ldi ZH, high(Text28*2)
        rcall Disptext

        ldi ZL, low(Text39*2)
        ldi ZH, high(Text39*2)
        rcall Disptext

        ldi ZL, low(Text40*2)
        ldi ZH, high(Text40*2)
        rcall Disptext

        ldi ZL, low(Text41*2)
        ldi ZH, high(Text41*2)
        rcall Disptext

        ldi ZL, low(Text42*2)
        ldi ZH, high(Text42*2)
        rcall Disptext

        ldi ZL, low(Text43*2)
        ldi ZH, high(Text43*2)
        rcall Disptext
```

```
        ldi ZL, low(Text44*2)
        ldi ZH, high(Text44*2)
        rcall Disptext

        ldi ZL, low(Text45*2)
        ldi ZH, high(Text45*2)
        rcall Disptext

        ldi ZL, low(Text46*2)
        ldi ZH, high(Text46*2)
        rcall Disptext

        ldi ZL, low(Text47*2)
        ldi ZH, high(Text47*2)
        rcall Disptext


        sei
hier:
        rjmp hier


Disptext:                                       ;Gibt eine Zeile (max. 64 Zeichen) Text aus,
springt in die nächste Zeile
        lpm temp, Z+                        ;Zeichen aus dem String laden
        cpi temp, 0
        breq exit
        push ZL
        push ZH
        ldi ZH, high(CGROM*2)           ;Pointer auf CGROM
        mov ZL, temp

ldi loop, 8                                     ;8 Zeilen senden
SendLines:
        lpm Byte, Z                             ;Grafikdaten aus CGROM laden
        inc ZH
        rcall WriteLine                     ;8 Pixel in den Speiche schreiben
        subi AdressLL, 8
        sbci AdressLH, 0
        adiw AdressHH:AdressHL,1
        dec loop
        brne SendLines

        subi AdressHL, 8
        sbci AdressHH, 0
        adiw AdressLH:AdressLL,8


        inc Color                               ;Nächster Farbwert (zum Testen aller Farben)


        pop ZH
        pop ZL
        rjmp Disptext
exit:
        clr AdressLL
        ldi AdressLH, 192
        adiw AdressHH:AdressHL,8
        cpi AdressHH, 192+2
        brne Skipreset
        ldi AdressHH, 192
SkipReset:

ret
```

```
WriteLine:                                          ;Schreibt 8 Pixel (=1 Byte aus dem CGROM) in
den SDRAM
        ser temp2
        out ddra, temp2
        out AdL, AdressHL
        out AdH, AdressHH
        out Comm, _Activate
        out Comm, _NOP
        out AdH, AdressLH
        out AdL, AdressLL
        clr temp
        sbrc Byte, 7
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 6
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 5
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 4
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdL, AdressLL

        clr temp
        sbrc Byte, 3
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 2
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
```

```
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 1
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        out AdH, AdressLH
        out AdL, AdressLL

        clr temp
        sbrc Byte, 0
        mov temp, Color
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        adiw AdressLH:AdressLL,1
        cpi AdressLH, 192+2
        brne SkipReset2
        ldi AdressLH, 192
Skipreset2:
        out Comm, _Precharge
        out Comm, _NOP
        clr temp2
        out ddra, temp2

ret


WriteByte:
        ser temp2
        out ddra, temp2
        out AdL, AdressHL
        out AdH, AdressHH
        out Comm, _Activate
        out Comm, _NOP
        nop
        out AdL, AdressLL
        out AdH, AdressLH
        out portA, temp
        out Comm, _Write
        out Comm, _NOP
        out Comm, _Precharge
        out Comm, _NOP
        clr temp2
        out ddra, temp2
ret

Init:
        ldi loop, 0                     ;1ms bei 16MHz
        clr temp
StartUpDelay:
        dec temp
        brne StartUpDelay
        dec loop
        brne StartUpDelay

        ldi temp, AutoRefresh   ;8 CBR Cycles
        out Comm, temp
        nop
        nop
```

```
        nop
        nop
        nop
        nop
        nop
        out Comm, _NOP

        clr temp
        out AdL, temp
        ldi temp, Precharge
        ldi temp2, 4                    ;Precharge all Banks
        out AdH, temp2
        out Comm, temp
        out Comm, _NOP

        ldi temp, 7|32          ;CAS Latency:2, Burst Length: Full Page
        out AdL, temp
        ldi temp, 192|2                 ;Burst Read, Single Write
        out AdH, temp
        ldi temp, ModeRegister
        out Comm, temp
        out Comm, _NOP

        cbi portb, RS0          ;Init Data Mask
        sbi portb, RS1
        ser temp
        out portc, temp
        nop
        nop
        cbi portb, WR
        nop
        nop
        sbi portb, WR
        nop

        cbi portb, RS0          ;Reset LUT Adress
        cbi portb, RS1
        clr temp
        out portc, temp
        nop
        nop
        cbi portb, WR
        nop
        nop
        sbi portb, WR
        nop

        sbi portb, RS0          ;Init LUT to SW
        cbi portb, RS1

        clr loop
LUT:
        mov temp, loop
        andi temp, 3
        swap temp
        out portc, temp
        nop
        cbi portb, WR           ;R
        nop
        nop
        sbi portb, WR
        nop
        nop
        nop
```

```
        mov temp, loop
        andi temp, 4+8+16
        lsl temp
        out portc, temp
        nop
        cbi portb, WR           ;G
        nop
        nop
        sbi portb, WR
        nop
        nop
        nop
        mov temp, loop
        andi temp, 32+64+128
        lsr temp
        lsr temp
        out portc, temp
        nop
        cbi portb, WR           ;B
        nop
        nop
        sbi portb, WR
        inc loop
        brne LUT
ret



UART_TXD:                               ; Start transmission of data (r16)
        sbis UCSRA, UDRE
        rjmp UART_TXD
        out UDR, temp
ret

Timer:
        cbi PortE, Blank                        ;Blanking: Low
        sbi PortE, VSync                        ;V-Sync: High

        out Comm, _Precharge            ;Stop Burst Read
        out Comm, _NOP

        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop

;nop

        sbi PortE, HSync

        out AdL, ZeileL
        out AdH, ZeileH
        out Comm, _Activate
        out Comm, _NOP
        ldi temp, 245
        out AdL, temp
        ldi temp, 192+1
        out AdH, temp
        out Comm, _Read
```

```
        out Comm, _NOP

        ldi temp, 1
        add ZeileL, temp                        ;Zeilenzähler
        dec temp
        adc ZeileH, temp

        sbrs ZeileH, 1
        sbi PortE, Blank                        ;Blanking: off

        sbrs ZeileH, 1
        rjmp ExInt
ZeilenC:

        cpi ZeileL,VStart                       ;VSync Beginn ? (bei Zeile 512+VStart)
        brne Sel1
        cbi PortE, VSync                        ;V-Sync: 31,5kHz / 525 Zeilen = 60Hz
        rjmp ExInt
Sel1:
        cpi ZeileL,VStart+VLang         ;VSync Ende ? (bei Zeile 512+VStart+VLang)
        brne Sel2
        sbi PortE, VSync
        rjmp ExInt
Sel2:
        cpi ZeileL,VStart+VLang+Offs;Beginn von neuem Bild ? (bei Zeile 512+VStart+VLang+Offs)
        brne ExInt
        sbi PortE, Blank                        ;Blanking: off
        ldi ZeileH, 192
        clr ZeileL
ExInt:
reti


.org 512        ;2048 Byte CGROM: 256*8*8
CGROM:

.db
0x00,0x7E,0x7E,0x6C,0x10,0x38,0x10,0x10,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x99,0x80,0x02,0x18,0x66,0
x7F,0x3E,0x00,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x6C,0x6C,0x30,0x00,0x38,0x60,0x
18,0x60,0x00,0x00,0x00,0x00,0x00,0x06,0x7C,0x30,0x78,0x78,0x1C,0xFC,0x38,0xFC,0x78,0x78,0x00,0x00,0x1
8,0x00,0x60,0x78
.db
0x7C,0x30,0xFC,0x3C,0xF8,0xFE,0xFE,0x3C,0xCC,0x78,0x1E,0xE6,0xF0,0xC6,0xC6,0x38,0xFC,0x78,0xFC,0x78,0
xFC,0xCC,0xCC,0xC6,0xC6,0xCC,0xFE,0x78,0xC0,0x78,0x10,0x00,0x30,0x00,0xE0,0x00,0x1C,0x00,0x38,0x00,0x
E0,0x30,0x0C,0xE0,0x70,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1
8,0xE0,0x76,0x00
.db
0x78,0x00,0x1C,0x7E,0xCC,0xE0,0x30,0x00,0x7E,0xCC,0xE0,0xCC,0x7C,0xE0,0xC6,0x30,0x1C,0x00,0x3E,0x78,0
x00,0x00,0x78,0x00,0x00,0xC3,0xCC,0x18,0x38,0xCC,0xF8,0x0E,0x1C,0x38,0x00,0x00,0x00,0xFC,0x3C,0x38,0x
30,0x00,0x00,0xC3,0xC3,0x18,0x00,0x00,0x22,0x55,0xDB,0x18,0x18,0x18,0x36,0x00,0x00,0x36,0x36,0x00,0x3
6,0x36,0x18,0x00
.db
0x18,0x18,0x00,0x18,0x00,0x18,0x18,0x36,0x36,0x00,0x36,0x00,0x36,0x00,0x36,0x18,0x36,0x00,0x00,0x36,0
x18,0x00,0x00,0x36,0x18,0x18,0x00,0xFF,0x00,0xF0,0x0F,0xFF,0x00,0x00,0x00,0x00,0xFC,0x00,0x00,0x00,0x
FC,0x38,0x38,0x1C,0x00,0x06,0x38,0x78,0x00,0x30,0x60,0x18,0x0E,0x18,0x30,0x00,0x38,0x00,0x00,0x0F,0x7
8,0x70,0x00,0x00
.db
0x00,0x81,0xFF,0xFE,0x38,0x7C,0x10,0x10,0xFF,0x00,0x00,0x00,0x00,0x00,0x63,0x5A,0xE0,0x0E,0x3C,0x66,0
xDB,0x63,0x00,0x3C,0x3C,0x18,0x18,0x30,0x30,0x30,0x30,0x30,0x00,0x78,0x6C,0x6C,0x7C,0xC6,0x6C,0x60,0x
30,0x30,0x66,0x30,0x00,0x00,0x00,0x0C,0xC6,0x70,0xCC,0xCC,0x3C,0xC0,0x60,0xCC,0xCC,0xCC,0x30,0x30,0x3
0,0x00,0x30,0xCC
.db
0xC6,0x78,0x66,0x66,0x6C,0x62,0x62,0x66,0xCC,0x30,0x0C,0x66,0x60,0xEE,0xE6,0x6C,0x66,0xCC,0x66,0xCC,0
xB4,0xCC,0xCC,0xC6,0xC6,0xCC,0xC6,0x60,0x60,0x18,0x38,0x00,0x30,0x00,0x60,0x00,0x0C,0x00,0x6C,0x00,0x
60,0x00,0x00,0x60,0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x1
```

```
8,0x30,0xDC,0x10
.db
0xCC,0xCC,0x00,0xC3,0x00,0x00,0x30,0x00,0xC3,0x00,0x00,0x00,0xC6,0x00,0x38,0x30,0x00,0x00,0x6C,0xCC,0
xCC,0xE0,0xCC,0xE0,0xCC,0x18,0x00,0x18,0x6C,0xCC,0xCC,0x1B,0x00,0x00,0x1C,0x1C,0xF8,0x00,0x6C,0x6C,0x
00,0x00,0x00,0xC6,0xC6,0x18,0x33,0xCC,0x88,0xAA,0x77,0x18,0x18,0x18,0x36,0x00,0x00,0x36,0x36,0x00,0x3
6,0x36,0x18,0x00
.db
0x18,0x18,0x00,0x18,0x00,0x18,0x18,0x36,0x36,0x00,0x36,0x00,0x36,0x00,0x36,0x18,0x36,0x00,0x00,0x36,0
x18,0x00,0x00,0x36,0x18,0x18,0x00,0xFF,0x00,0xF0,0x0F,0xFF,0x00,0x78,0xFC,0xFE,0xCC,0x00,0x66,0x76,0x
30,0x6C,0x6C,0x30,0x00,0x0C,0x60,0xCC,0xFC,0x30,0x30,0x30,0x1B,0x18,0x30,0x76,0x6C,0x00,0x00,0x0C,0x6
C,0x18,0x00,0x00
.db
0x00,0xA5,0xDB,0xFE,0x7C,0x38,0x38,0x38,0xE7,0x00,0x00,0x00,0x00,0x00,0x7F,0x3C,0xF8,0x3E,0x7E,0x66,0
xDB,0x38,0x00,0x7E,0x7E,0x18,0x0C,0x60,0x60,0x60,0x60,0x60,0x00,0x78,0x6C,0xFE,0xC0,0xCC,0x38,0xC0,0x
60,0x18,0x3C,0x30,0x00,0x00,0x00,0x18,0xCE,0x30,0x0C,0x0C,0x6C,0xF8,0xC0,0x0C,0xCC,0xCC,0x30,0x30,0x6
0,0xFC,0x18,0x0C
.db
0xDE,0xCC,0x66,0xC0,0x66,0x68,0x68,0xC0,0xCC,0x30,0x0C,0x6C,0x60,0xFE,0xF6,0xC6,0x66,0xCC,0x66,0xE0,0
x30,0xCC,0xCC,0xC6,0x6C,0xCC,0x8C,0x60,0x30,0x18,0x6C,0x00,0x18,0x78,0x60,0x78,0x0C,0x78,0x60,0x76,0x
6C,0x70,0x0C,0x66,0x30,0xCC,0xF8,0x78,0xDC,0x76,0xDC,0x7C,0x7C,0xCC,0xCC,0xC6,0xC6,0xCC,0xFC,0x30,0x1
8,0x30,0x00,0x38
.db
0xC0,0x00,0x78,0x3C,0x78,0x78,0x78,0x78,0x3C,0x78,0x78,0x70,0x38,0x70,0x6C,0x00,0xFC,0x7F,0xCC,0x00,0
x00,0x00,0x00,0x00,0x00,0x3C,0xCC,0x7E,0x64,0x78,0xCC,0x18,0x78,0x70,0x00,0x00,0x00,0xCC,0x6C,0x6C,0x
30,0x00,0x00,0xCC,0xCC,0x00,0x66,0x66,0x22,0x55,0xDB,0x18,0x18,0xF8,0x36,0x00,0xF8,0xF6,0x36,0xFE,0xF
6,0x36,0xF8,0x00
.db
0x18,0x18,0x00,0x18,0x00,0x18,0x1F,0x36,0x37,0x3F,0xF7,0xFF,0x37,0xFF,0xF7,0xFF,0x36,0xFF,0x00,0x36,0
x1F,0x1F,0x00,0x36,0xFF,0x18,0x00,0xFF,0x00,0xF0,0x0F,0xFF,0x76,0xCC,0xCC,0x6C,0x60,0x7E,0x66,0xDC,0x
78,0xC6,0xC6,0x18,0x7E,0x7E,0xC0,0xCC,0x00,0xFC,0x18,0x60,0x1B,0x18,0x00,0xDC,0x6C,0x00,0x00,0x0C,0x6
C,0x30,0x3C,0x00
.db
0x00,0x81,0xFF,0xFE,0xFE,0xFE,0x7C,0x7C,0xC3,0x00,0x00,0x00,0x00,0x00,0x63,0xE7,0xFE,0xFE,0x18,0x66,0
x7B,0x6C,0x00,0x18,0x18,0x18,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0x00,0x30,0x00,0x6C,0x78,0x18,0x76,0x00,0x
60,0x18,0xFF,0xFC,0x00,0xFC,0x00,0x30,0xDE,0x30,0x38,0x38,0xCC,0x0C,0xF8,0x18,0x78,0x7C,0x00,0x00,0xC
0,0x00,0x0C,0x18
.db
0xDE,0xCC,0x7C,0xC0,0x66,0x78,0x78,0xC0,0xFC,0x30,0x0C,0x78,0x60,0xFE,0xDE,0xC6,0x7C,0xCC,0x7C,0x70,0
x30,0xCC,0xCC,0xD6,0x38,0x78,0x18,0x60,0x18,0x18,0xC6,0x00,0x00,0x0C,0x7C,0xCC,0x7C,0xCC,0xF0,0xCC,0x
76,0x30,0x0C,0x6C,0x30,0xFE,0xCC,0xCC,0x66,0xCC,0x76,0xC0,0x30,0xCC,0xCC,0xD6,0x6C,0xCC,0x98,0xE0,0x0
0,0x1C,0x00,0x6C
.db
0xCC,0xCC,0xCC,0x06,0x0C,0x0C,0x0C,0xC0,0x66,0xCC,0xCC,0x30,0x18,0x30,0xC6,0x78,0x60,0x0C,0xFE,0x78,0
x78,0x78,0xCC,0xCC,0xCC,0x66,0xCC,0xC0,0xF0,0xFC,0xFA,0x3C,0x0C,0x30,0x78,0xCC,0xF8,0xEC,0x3E,0x38,0x
60,0xFC,0xFC,0xDE,0xDB,0x18,0xCC,0x33,0x88,0xAA,0xEE,0x18,0x18,0x18,0x36,0x00,0x18,0x06,0x36,0x06,0x0
6,0x36,0x18,0x00
.db
0x18,0x18,0x00,0x18,0x00,0x18,0x18,0x36,0x30,0x30,0x00,0x00,0x30,0x00,0x00,0x00,0x36,0x00,0x00,0x36,0
x18,0x18,0x00,0x36,0x18,0x18,0x00,0xFF,0x00,0xF0,0x0F,0xFF,0xDC,0xF8,0xC0,0x6C,0x30,0xD8,0x66,0x18,0x
CC,0xFE,0xC6,0x7C,0xDB,0xDB,0xF8,0xCC,0xFC,0x30,0x30,0x30,0x18,0x18,0xFC,0x00,0x38,0x18,0x00,0x0C,0x6
C,0x60,0x3C,0x00
.db
0x00,0xBD,0xC3,0x7C,0x7C,0xFE,0xFE,0xFE,0xC3,0x00,0x00,0x00,0x00,0x00,0x63,0xE7,0xF8,0x3E,0x18,0x66,0
x1B,0x6C,0x7E,0x7E,0x18,0x7E,0x0C,0x60,0x60,0x60,0x60,0x60,0x00,0x30,0x00,0xFE,0x0C,0x30,0xDC,0x00,0x
60,0x18,0x3C,0x30,0x00,0x00,0x00,0x60,0xF6,0x30,0x60,0x0C,0xFE,0x0C,0xCC,0x30,0xCC,0x0C,0x00,0x00,0x6
0,0x00,0x18,0x30
.db
0xDE,0xFC,0x66,0xC0,0x66,0x68,0x68,0xCE,0xCC,0x30,0xCC,0x6C,0x62,0xD6,0xCE,0xC6,0x60,0xDC,0x6C,0x1C,0
x30,0xCC,0xCC,0xFE,0x38,0x30,0x32,0x60,0x0C,0x18,0x00,0x00,0x00,0x7C,0x66,0xC0,0xCC,0xFC,0x60,0xCC,0x
66,0x30,0x0C,0x78,0x30,0xFE,0xCC,0xCC,0x66,0xCC,0x66,0x78,0x30,0xCC,0xCC,0xFE,0x38,0xCC,0x30,0x30,0x1
8,0x30,0x00,0xC6
.db
0x78,0xCC,0xFC,0x3E,0x7C,0x7C,0x7C,0xC0,0x7E,0xFC,0xFC,0x30,0x18,0x30,0xFE,0xCC,0x78,0x7F,0xCC,0xCC,0
xCC,0xCC,0xCC,0xCC,0xCC,0x66,0xCC,0xC0,0x60,0x30,0xC6,0x18,0x7C,0x30,0xCC,0xCC,0xCC,0xFC,0x00,0x00,0x
C0,0xC0,0x0C,0x33,0x37,0x18,0x66,0x66,0x22,0x55,0xDB,0x18,0xF8,0xF8,0xF6,0xFE,0xF8,0xF6,0x36,0xF6,0xF
```

```
E,0xFE,0xF8,0xF8
.db
0x1F,0xFF,0xFF,0x1F,0xFF,0xFF,0x1F,0x37,0x3F,0x37,0xFF,0xF7,0x37,0xFF,0xF7,0xFF,0xFF,0xFF,0xFF,0x3F,0
x1F,0x1F,0x3F,0xFF,0xFF,0xF8,0x1F,0xFF,0xFF,0xF0,0x0F,0x00,0xC8,0xCC,0xC0,0x6C,0x60,0xD8,0x66,0x18,0x
CC,0xC6,0x6C,0xCC,0xDB,0xDB,0xC0,0xCC,0x00,0x30,0x60,0x18,0x18,0x18,0x00,0x76,0x00,0x18,0x18,0xEC,0x6
C,0x78,0x3C,0x00
.db
0x00,0x99,0xE7,0x38,0x38,0x7C,0x7C,0x7C,0xE7,0x00,0x00,0x00,0x00,0x00,0x67,0x3C,0xE0,0x0E,0x7E,0x00,0
x1B,0x38,0x7E,0x3C,0x18,0x3C,0x18,0x30,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x6C,0xF8,0x66,0xCC,0x00,0x
30,0x30,0x66,0x30,0x30,0x00,0x30,0xC0,0xE6,0x30,0xCC,0xCC,0x0C,0xCC,0xCC,0x30,0xCC,0x18,0x30,0x30,0x3
0,0xFC,0x30,0x00
.db
0xC0,0xCC,0x66,0x66,0x6C,0x62,0x60,0x66,0xCC,0x30,0xCC,0x66,0x66,0xC6,0xC6,0x6C,0x60,0x78,0x66,0xCC,0
x30,0xCC,0x78,0xEE,0x6C,0x30,0x66,0x60,0x06,0x18,0x00,0x00,0x00,0xCC,0x66,0xCC,0xCC,0xC0,0x60,0x7C,0x
66,0x30,0xCC,0x6C,0x30,0xD6,0xCC,0xCC,0x7C,0x7C,0x60,0x0C,0x34,0xCC,0x78,0xFE,0x6C,0x7C,0x64,0x30,0x1
8,0x30,0x00,0xC6
.db
0x18,0xCC,0xC0,0x66,0xCC,0xCC,0xCC,0x78,0x60,0xC0,0xC0,0x30,0x18,0x30,0xC6,0xFC,0x60,0xCC,0xCC,0xCC,0
xCC,0xCC,0xCC,0xCC,0x7C,0x3C,0xCC,0x7E,0xE6,0xFC,0xCF,0x18,0xCC,0x30,0xCC,0xCC,0xCC,0xDC,0x7E,0x7C,0x
CC,0xC0,0x0C,0x66,0x6F,0x18,0x33,0xCC,0x88,0xAA,0x77,0x18,0x18,0x18,0x36,0x36,0x18,0x36,0x36,0x36,0x0
0,0x00,0x00,0x18
.db
0x00,0x00,0x18,0x18,0x00,0x18,0x18,0x36,0x00,0x36,0x00,0x36,0x36,0x00,0x36,0x00,0x00,0x18,0x36,0x00,0
x00,0x18,0x36,0x36,0x18,0x00,0x18,0xFF,0xFF,0xF0,0x0F,0x00,0xDC,0xF8,0xC0,0x6C,0xCC,0xD8,0x7C,0x18,0x
78,0x6C,0x6C,0xCC,0x7E,0x7E,0x60,0xCC,0xFC,0x00,0x00,0x00,0x18,0xD8,0x30,0xDC,0x00,0x00,0x00,0x6C,0x0
0,0x00,0x3C,0x00
.db
0x00,0x81,0xFF,0x10,0x10,0x38,0x38,0x38,0xFF,0x00,0x00,0x00,0x00,0x00,0xE6,0x5A,0x80,0x02,0x3C,0x66,0
x1B,0xCC,0x7E,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x00,0x6C,0x30,0xC6,0x76,0x00,0x
18,0x60,0x00,0x00,0x30,0x00,0x30,0x80,0x7C,0xFC,0xFC,0x78,0x1E,0x78,0x78,0x30,0x78,0x70,0x30,0x30,0x1
8,0x00,0x60,0x30
.db
0x78,0xCC,0xFC,0x3C,0xF8,0xFE,0xF0,0x3E,0xCC,0x78,0x78,0xE6,0xFE,0xC6,0xC6,0x38,0xF0,0x1C,0xE6,0x78,0
x78,0xFC,0x30,0xC6,0xC6,0x78,0xFE,0x78,0x02,0x78,0x00,0x00,0x00,0x76,0xDC,0x78,0x76,0x78,0xF0,0x0C,0x
E6,0x78,0xCC,0xE6,0x78,0xC6,0xCC,0x78,0x60,0x0C,0xF0,0xF8,0x18,0x76,0x30,0x6C,0xC6,0x0C,0xFC,0x1C,0x1
8,0xE0,0x00,0xFE
.db
0x0C,0x7E,0x78,0x3F,0x7E,0x7E,0x7E,0x0C,0x3C,0x78,0x78,0x78,0x3C,0x78,0xC6,0xCC,0xFC,0x7F,0xCE,0x78,0
x78,0x78,0x7E,0x7E,0x0C,0x18,0x78,0x18,0xFC,0x30,0xC6,0xD8,0x7E,0x78,0x78,0x7E,0xCC,0xCC,0x00,0x00,0x
78,0x00,0x00,0xCC,0xCF,0x18,0x00,0x00,0x22,0x55,0xDB,0x18,0x18,0x18,0x36,0x36,0x18,0x36,0x36,0x36,0x0
0,0x00,0x00,0x18
.db
0x00,0x00,0x18,0x18,0x00,0x18,0x18,0x36,0x00,0x36,0x00,0x36,0x36,0x00,0x36,0x00,0x00,0x18,0x36,0x00,0
x00,0x18,0x36,0x36,0x18,0x00,0x18,0xFF,0xFF,0xF0,0x0F,0x00,0x76,0xC0,0xC0,0x6C,0xFC,0x70,0x60,0x18,0x
30,0x38,0xEE,0x78,0x00,0x60,0x38,0xCC,0x00,0xFC,0xFC,0xFC,0x18,0xD8,0x30,0x00,0x00,0x00,0x00,0x3C,0x0
0,0x00,0x00,0x00
.db
0x00,0x7E,0x7E,0x00,0x00,0x7C,0x7C,0x7C,0xFF,0x00,0x00,0x00,0x00,0x00,0xC0,0x99,0x00,0x00,0x18,0x00,0
x00,0x78,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x0
0,0x00,0x00,0x00
.db
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0x
00,0x00,0x78,0x00,0x00,0x00,0x00,0x00,0xF0,0x1E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0x00,0x00,0x0
0,0x00,0x00,0x00
.db
0x78,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0xF8,0x00,0x00,0x18,0x00,0x30,0xC7,0x70,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x0F,0x03,0x00,0x00,0x00,0x88,0xAA,0xEE,0x18,0x18,0x18,0x36,0x36,0x18,0x36,0x36,0x36,0x0
0,0x00,0x00,0x18
.db
0x00,0x00,0x18,0x18,0x00,0x18,0x18,0x36,0x00,0x36,0x00,0x36,0x36,0x00,0x36,0x00,0x00,0x18,0x36,0x00,0
x00,0x18,0x36,0x36,0x18,0x00,0x18,0xFF,0xFF,0xF0,0x0F,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0xC0,0x00,0x
FC,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x70,0x00,0x00,0x00,0x00,0x00,0x1C,0x0
```

```
0,0x00,0x00,0x00


Text1:
.db "ATmega8515 ",0
Text2:
.db "  ",0
Text3:
.db "8-bit Microcontroller with 8K Bytes In-System Programmable Flash",0
Text4:
.db " ",0
Text5:
.db "Features",0
Text6:
.db "High-performance, Low-power AVR®8-bit Microcontroller",0
Text7:
.db "RISC Architecture",0
Text8:
.db "- 130 Powerful Instructions, Most Single Clock Cycle Execution",0
Text9:
.db "- 32 x 8 General Purpose Working Registers",0
Text10:
.db "- Fully Static Operation",0
Text11:
.db "- Up to 16 MIPS Throughput at 16 MHz",0
Text12:
.db "- On-chip 2-cycle Multiplier",0
Text13:
.db "Nonvolatile Program and Data Memories",0
Text14:
.db "- 8K Bytes of In-System Self-programmable Flash",0
Text15:
.db "  Endurance: 10,000 Write/Erase Cycles",0
Text16:
.db "- Optional Boot Code Section with Independent Lock bits",0
Text17:
.db "  In-System Programming by On-chip Boot Program",0
Text18:
.db "  True Read-While-Write Operation",0
Text19:
.db "- 512 Bytes EEPROM",0
Text20:
.db "  Endurance: 100,000 Write/Erase Cycles",0
Text21:
.db "- 512 Bytes Internal SRAM",0
Text22:
.db "- Up to 64K Bytes Optional External Memory Space",0
Text23:
.db "- Programming Lock for Software Security",0
Text24:
.db "Peripheral Features",0
Text25:
.db "- One 8-bit Timer/Counter with Separate Prescaler and",0
Text26:
.db "  Compare Mode",0
Text27:
.db "- One 16-bit Timer/Counter with Separate Prescaler, ",0
Text28:
.db "  Compare Mode, and Capture Mode",0
Text29:
.db "- Three PWM Channels",0
Text30:
.db "- Programmable Serial USART",0
Text31:
.db "- Master/Slave SPI Serial Interface",0
```

```
Text32:
.db "- Programmable Watchdog Timer with Separate On-chip Oscillator",0
Text33:
.db "- On-chip Analog Comparator",0
Text34:
.db "Special Microcontroller Features",0
Text35:
.db "- Power-on Reset and Programmable Brown-out Detection",0
Text36:
.db "- Internal Calibrated RC Oscillator",0
Text37:
.db "- External and Internal Interrupt Sources",0
Text38:
.db "- Three Sleep Modes: Idle, Power-down and Standby",0
Text39:
.db "I/O and Packages",0
Text40:
.db "- 35 Programmable I/O Lines",0
Text41:
.db "- 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad MLF",0
Text42:
.db "Operating Voltages",0
Text43:
.db "- 2.7 - 5.5V for ATmega8515L",0
Text44:
.db "- 4.5 - 5.5V for ATmega8515",0
Text45:
.db "Speed Grades",0
Text46:
.db "- 0 - 8 MHz for ATmega8515L",0
Text47:
.db "- 0 - 16 MHz for ATmega8515",0
```