Tombola: Bingo!!! On Your TV

This is a compilation of an ATMEL AVR AT90S1200 video hardware project documentation.
The project was designed by Alberto Ricci Bitti (a.riccibitti@iname.com).

From README.TXT:

1) You are allowed to build this design for personal, non commercial use only.
2) Any commercial use (included publishing it on CD-ROM, internet etc.) must have an explicit written authorization by the author.
3) This information is given on an "as is" basis, excluding any responsibility for the author: you use it at your own risk.
4) Non-EC users: the unit is designed to work with 50Hz field TV sets (e.g. PAL), that is 25 frames per second, 64 µs (15625Hz) line, non-interlaced display.

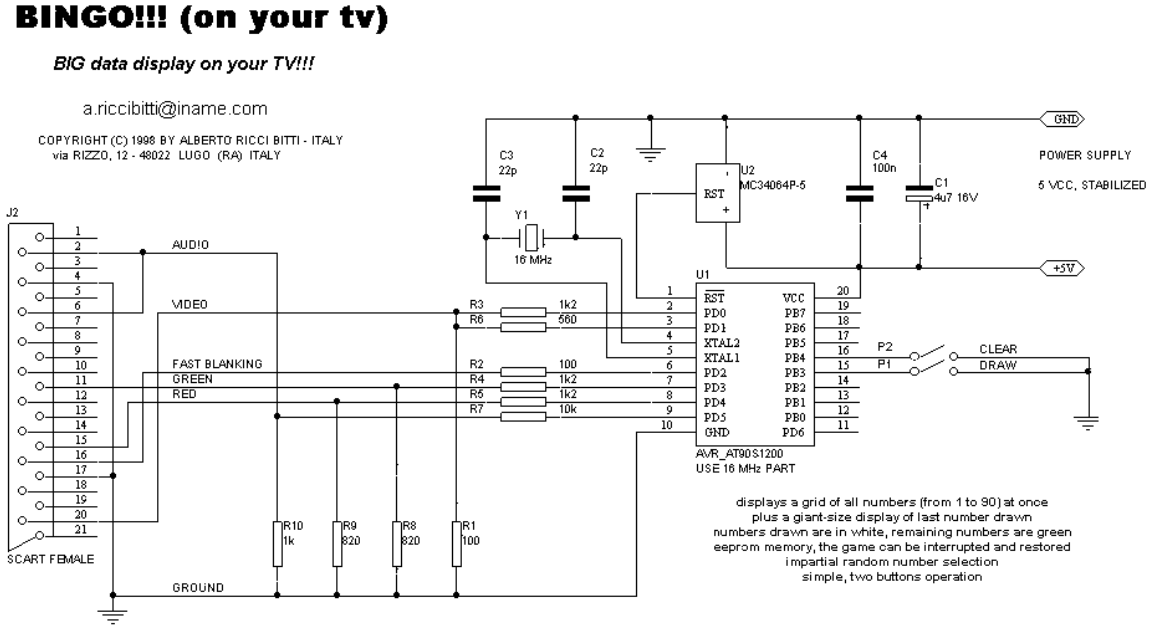
a.riccibitti@iname.com
www.geocities.com/CapeCanaveral/Launchpad/3632

From Alberto Ricci Bitti's Bingo Page:

This design is based on the video DVM and cronograph design. It supports color, so numbers not yet drawn are displayed in green and the others in white.

The programming tech is brought to the extremes, since displaying 90 small numbers on a screen (using a 512 lines assembler program) is a tough task even for a fast microcontroller as the Atmel's AVR AT90S1200. The (admittedly dirty) code and the schematic are derived from the **video-cronograph** project, cutting off the serial port output and the photocells input, so don't be surprised if you find some strange variable names.

Schematics diagram:



TOMBOLA.ASM

```
;*****************************************************************************
;* TOMBOLA.ASM
;*
;* BINGO MACHINE
;* ---> composite video output showing giant digits
;*
;* Copyright (c) 1997 by Alberto Ricci Bitti
;* e-mail: a.riccibitti@iname.com
;*****************************************************************************

.DEVICE AT90S1200

;*****************************************************************************
;* a few, very useful macros
;*****************************************************************************

.LISTMAC
;* definisce il nome passato come il registro r20
.MACRO ARGUMENT1
.def  @0 = r16
.ENDMACRO
.MACRO ARGUMENT2
.def  @0 = r17
.ENDMACRO
.MACRO ARGUMENT3
.def  @0 = r18
.ENDMACRO
.MACRO ARGUMENT4
.def  @0 = r19
.ENDMACRO
.MACRO ARGUMENT5
.def  @0 = r20
.ENDMACRO
.MACRO MAINVARIABLE1
.def  @0 = r21
.ENDMACRO
.MACRO MAINVARIABLE2
.def  @0 = r22
.ENDMACRO
.MACRO MAINVARIABLE3
.def  @0 = r23
.ENDMACRO
.MACRO MAINVARIABLE4
.def  @0 = r24
.ENDMACRO


.MACRO  ADDI
      subi @0, -@1
.ENDMACRO

;skip next instruction if not equal to zero
.MACRO skipne
.SET    _skipne = PC+2
      brne    _skipne
.ENDMACRO

;skip next instruction if carry set
.MACRO skipcs
.SET    _skipcs = PC+2
      brcs    _skipcs
.ENDMACRO
```

```
;skip next instruction if carry clear
.MACRO skipcc
.SET    _skipcc = PC+2
      brcc    _skipcc
.ENDMACRO

;skip next instruction if equal to zero
.MACRO  skipeq
.SET    _skipeq = PC+2
      breq    _skipeq
.ENDMACRO

;skip next instruction if lower
.MACRO skiplo
.SET    _skiplo = PC+2
      brlo    _skiplo
.ENDMACRO

;skip next instruction if half carry set
.MACRO skiphs
.SET    _skiphs = PC+2
      brhs    _skiphs
.ENDMACRO

;skip next instruction if half carry clear
.MACRO skiphc
.SET    _skiphc= PC+2
      brhc   _skiphc
.ENDMACRO

;skip next instruction if T set
.MACRO skipts
.SET    _skipts = PC+2
      brts    _skipts
.ENDMACRO

;skip next instruction if T clear
.MACRO skiptc
.SET    _skiptc = PC+2
      brtc    _skiptc
.ENDMACRO

;skip next instruction if minus
.MACRO skipmi
.SET    _skipmi = PC+2
      brmi    _skipmi
.ENDMACRO

;wait two cycles but waste only one instruction
.MACRO  doublenop
.SET    _doublenop = PC+1
      rjmp    _doublenop
.ENDMACRO

;include AT90S1200 registers definitions
.include "1200def.inc"
.include "fonts.inc"

;***************************************************************************
;* VARIABLE ASSIGNEMENTS
;***************************************************************************
.CSEG

      .def    SaveStatus =    r0      ;status buffer during interrupts
```

```
      .def  NumeriDaEstrarre=r1    ;quanti numeri da estrarre per generatore
casuale
      .def    Random =       r2      ;generatore casuale da 1 a NumeriDaEstrarre
      .def    RunningNumber = r3      ;numero corrente sul tabellone
      .def    AltezzaPixel =  r4
      .def    EstrattiLow=r5      ; bitmap di una decina di numeri estratti
      .def    EstrattiHigh=r6     ; per accensione/spegnimento cifra
      .def    CopiaEstrattiLow=r7  ;copia di lavoro di quanto sopra
      .def    CopiaEstrattiHigh=r8
      .def    x5        =r9
      .def    Delay =         r10
      .def    CopiaPatternDecine =          r11
      .def    X4 =      r12
      .def    EEDriverRegister=r13
      .def    NumeriDaSaltare=r14
      .def    X6   =r15

      .def    Arg1 =          r16     ;Temp variable used by interrupt
      .def    Arg2 =          r17     ; "      "      "          "
      .def    Arg3 =          r18     ; "      "      "          "
      .def    Arg4 =          r19     ; "      "      "          "
      .def  Arg5 =            r20   ; "      "      "          "
      .def    Main1 =         r21     ;Temp variable used by main program
      .def    Main2 =         r22     ; "      "      "       "      "
      .def    Main3 =         r23     ; "      "      "       "      "
      .def    Main4 =         r24     ; "      "      "       "      "
      .def    Decine =        r25
      .def    Unita =         r26    ;  "       "      "     "
      .def    RowsModFour =   r27   ;raster rows counter, split in two for
      .def    RowsDivFour =   r28     ;convenience of use. Counts up to 312
      .def    RowOffset =     r29
      .def    NumeroScansione =r30
      .def    MiscFlags =     r31     ;Assorted flags
                              ;bit 0,1: position (0-3) of interrupt
                              ;identifies wich of the four line position
                              ;we are on (3=end of line, 0=sync pulse) when
                              ;an interrupt occurs
                              ;used by timer interrupt routine
      ;flags bit definition
      .equ  Position1Bit =    0
      .equ  Position2Bit =  1
      .equ    SerialReqBit =  2

      .equ  EEReadReqBit =    3
      .equ  EEWriteReqBit = 4
      .equ  EEWritePendingBit = 5
      .equ  EEDataBit =       6
      .equ  BlankScreenBit = 7

      .equ  POSITIONMASK =  3



;***************************************************************************
;* Constants
;***************************************************************************

      .equ    DisplayRow =    150/4   ;counter value for display start
      .equ    StartRetrace = (312/4)-1;counter value at wich retrace starts
      .equ    StopRetrace = (312/4)   ;maximum counter value


      .equ  STARTBLOCKED = 255      ;time value signalling indefinite
                              ;delay due to jump start
```

```
      .equ  LARGHEZZABIGNUMBERS = 1
      .equ  ALTEZZABIGNUMBERS = 11
      .equ  ALTEZZANUMERITABELLA = 3

;***************************************************************************
;* Port Pin Assignements
;***************************************************************************

      ;port D bit definitions (OUTPUTS)
      .equ    CSyncBit =      0       ;set video output to composite sync level
when low

      .equ    VideoBit =      1       ;set video level to white when high, black
when low
      .equ    FastBlank =     2
      .equ    GreenBit =      3       ;
      .equ    RedBit =        4       ;
      .equ    AudioBit =      5       ;board led
      .equ    RS232Bit =      6       ;1200 baud TTL level serial data

      ;port B bit definitions (INPUTS)
      .equ    KeyStop    =    1       ;Button: forced end
      .equ  KeyLastLap =    2 ;Button: allow automatic stop
      .equ  KeyGo =           3     ;Button: automatic start sequence
      .equ  KeyClear =  4     ;Button: clear counters
      .equ    Photo1 =        5       ;photocell car 1...3
      .equ    Photo2 =        6       ;
      .equ    Photo3 =        7       ;


;***************************************************************************
;* VECTORS
;***************************************************************************

      rjmp    RESET           ;Reset Handle
      rjmp    RESET           ;Ext. interrupt request 0
      rjmp    TIMER           ;Timer




;***************************************************************************
;*
;* MAIN PROGRAM
;*
;***************************************************************************
;* INITAILIZATION
;***************************************************************************


RESET:
            clt     ;clear T bit flag       ;VERY important:
                                ;clear T flag, here used to
                                ;verify if we were sleeping
            ldi     Arg1, 0b01111111        ;set port D bits to outputs
            out     DDRD, Arg1
            ldi     Arg1, 0b00000001        ;preset  output state
            out     PortD, Arg1
            ldi     Arg1, 0b00000000        ;set port B to inputs
            out     DDRB, Arg1
            ldi     Arg1,  0b11111111       ;turn on pullups on inputs
            out     PortB, Arg1
```

```
            ldi   Arg1, 1
            out     TCCR0, Arg1             ;dont'use timer prescaler
            ldi     Arg1, 32
            out     MCUCR, Arg1             ;enable sleep idle mode
            ldi     Arg1, 2
            out     TIMSK, Arg1             ;enable timer interrupt


            ldi   Arg1, 3
            mov   AltezzaPixel, Arg1


            ldi   MiscFlags, 1
            clr     Arg1                ;and clear interrupt variables
            clr     Arg2
            clr     Arg3
            clr     Arg4

            sei                         ;at last, allow interrupts


      ;;;   numeridaestrarre = 0
      ;;;   for numeroscansione = 89 to 0
      ;;;         if numeroscansione is not estratto
      ;;;               numeridaestrarre++

      clr   NumeriDaEstrarre
      ldi   NumeroScansione, 89+1
FOR1: dec   NumeroScansione
      brmi  ENDFOR1
      mov   EEDriverRegister, NumeroScansione  ;set number to read
      rcall READ_EEPROM
      sbrs  MiscFlags, EEDataBit
      inc   NumeriDaEstrarre
      rjmp  FOR1
ENDFOR1:
      ldi   NumeroScansione,90



;************************************************************************
;* MAIN LOOP
;************************************************************************

FOREVER:
MAINVARIABLE2   Ripristino
TESTKEYGO:
      sbic  PinB, KeyGo             ;if GO! pressed, restart time
      rjmp  TESTKEYCLEAR
      tst   NumeriDaEstrarre
      breq  TESTKEYCLEAR

      ;;;   numeridasaltare = random
      ;;;   for numeroscansione = 89 to 0
      ;;;         if numeroscansione IS NOT estratto
      ;;;               if numeridasaltare == 0
      ;;;                     estrai numeroscansione
      ;;;                     exit for
      ;;;               else
      ;;;                     numeridasaltare--
      ;;;               endif
      ;;;         endif


      mov   NumeriDaSaltare, Random
```

```
      ldi   NumeroScansione, 89+1
FOR:  dec   NumeroScansione
      brmi  ENDFOR
      mov   EEDriverRegister, NumeroScansione  ;set number to read
      rcall READ_EEPROM
      sbrc  MiscFlags, EEDataBit
      rjmp  FOR
      dec   NumeriDaSaltare
      brpl  FOR
      ;estrazione!
      dec   NumeriDaEstrarre
      sbr   MiscFlags, EXP2(EEDataBit)         ;set data to write
      mov   EEDriverRegister, NumeroScansione  ;set number to write
      rcall WRITE_EEPROM
ENDFOR:
      rcall Delay
WAITRELEASE:
      sbis  PinB, KeyGo                  ;aspettiamo il rilascio
      rjmp  WAITRELEASE
      rcall Delay                        ;debounce

TESTKEYCLEAR:
      sbic  PinB, KeyClear
      rjmp  ENDKEYS
      ldi   NumeroScansione,90                 ;90 = clear
      ldi   Ripristino, 90
      mov   NumeriDaEstrarre, Ripristino
      ldi   Ripristino, 89
_L30: cbr   MiscFlags, EXP2(EEDataBit)         ;set data to write
      mov   EEDriverRegister, Ripristino       ;set number to write
      rcall WRITE_EEPROM
      dec   Ripristino
      brpl  _L30

ENDKEYS:
      rjmp  FOREVER



;***************************************************************************
;*  MAIN ROUTINES
;***************************************************************************


DELAY:                               ;ritardo dopo pressione tasti
      ldi   Ripristino, 50           ;a me piaceva 5....
      mov   Delay, Ripristino
_L40: tst   Delay
      skipeq
      rjmp  _L40
      ret



READ_EEPROM:
      sbr   MiscFlags, EXP2(EEReadReqBit)      ;send request
_L10: sbrc    MiscFlags, EEReadReqBit               ;read done?
      rjmp  _L10                         ;no, wait
      ret                                ;yes, return

WRITE_EEPROM:
      sbr   MiscFlags, EXP2(EEWriteReqBit)           ;send request
_L20: sbrc    MiscFlags, EEWriteReqBit         ;write done?
      rjmp  _L20                         ;no, wait
      ret                                ;yes, return
```

```
;*************************************************************************
;*
;* Timer Interrupt: occurs four times in each video line.
;*      The Position records which of four interrupt stages we are on.
;*      At the third stage, we go in sleep mode, in order to have constant
;*      servicing time for the fourth, crucial interrupt were we will output
;*      the sync waveform.
;*      This is the very heart of sync generation.
;*
;*************************************************************************


TIMER:
      ;if we were sleeping, then this is a synchronizing cycle: just return
      brtc    TIMER_NOSLEEP   ;if T cleared, do timer routine
      reti                    ;otherwise return (sync sleep)

TIMER_NOSLEEP:
      in      SaveStatus, SREG
      ;if not at line start,simply decrement position counter and return
      ;otherwise do the real thing...
      dec     MiscFlags               ;Position(bits 0 and 1) range from 3 down
to 0
      mov   Arg1, MiscFlags
      andi  Arg1, POSITIONMASK
      breq    WAITNEWLINE             ;if Position = 0 then wait (sleeping) for
new line
TIMER_EXIT:
      out     SREG, SaveStatus        ;otherwise do nothing
      reti
WAITNEWLINE:
      set                             ;set T flag to tell we are sleeping
      sei                             ;re-enable interrupts, otherwise we sleep
forever!
      sleep                           ;wait sleeping the next timer interrupt
                                      ;(during sleep we have constant interrupt
recovery time)
      cli                             ;disable further interrupts

NEWLINE:                                ;when awaken, we will restart here
      ori     MiscFlags, 3          ;reset position index
      in      Arg1, PortD             ;read previous CSync level
      ldi     Arg2, 0b00000001        ;load mask for toggle csync bit

TOGGLESYNC:
      eor     Arg1, Arg2              ;toggle CSync output
      out     PortD, Arg1             ;now we are in the horizontal sync pulse
HOUSEKEEPING:
      ;once in horizontal sync pulse space, we have
      ;enough time to do some housekeeping routines...
      inc     RowsModFour             ;increment row counters
      cpi     RowsModFour, 4          ;if RowsModFour == 4
      skipne
      inc     RowsDivFour             ;then increment RowsDivFour
      andi    RowsModFour, 0b00000011 ;clear anyway bit 2

CHECKLASTLINE:
      ;when at last line, the sync pulse is stretched to half line
            cpi     RowsDivFour, StopRetrace        ;if at last line
            skiplo
      rjmp  LASTLINE                ;do a long pulse
```

```
WAITSYNCEND:                              ;else wait sync pulse end
      in      Arg1, TCNT0               ;note that now we are perfectly in
      cpi     Arg1, 64                  ;sync with timer.        64 = 4 uS
      brlo    WAITSYNCEND

      ldi     Arg2, 0b00000001          ;prepare mask for csync toggle

VERTICALRETRACE:
      ldi     Arg1, StartRetrace
      cpi     RowsModFour, 0            ;if at start of vertical retrace pulse
      cpc     RowsDivFour, Arg1
      brne    HSYNCEND                  ;then leave CSync low
      ldi     Arg2, 0

HSYNCEND:
      in      Arg1, PortD               ;
      eor     Arg1, Arg2                ;toggle CSync output
      out     PortD, Arg1               ;now we are in the color burst space

RANDOMNUMBER:
      inc   Random                    ;increase random counter
      cp    NumeriDaEstrarre, Random      ;modulo (numbers to extract)
      skipne
      clr   Random

EEPROM:     rcall       EEDRIVER

WAITVISIBLEPORTION:
      in      Arg1, TCNT0             ;wait for start of visible area
      cpi     Arg1, 150       ;increasing this moves display right
      brlo    WAITVISIBLEPORTION

      rjmp  BUILDSCREEN       ;go to display building table

LASTLINE:
      ;this is a good time to make slow (1/50 sec.) housekeepings,
      ;or to do things that are made once every screen picture
      clr   Decine
      clr     RowsDivFour               ;reset row counters
      clr     RowsModFour
      tst   Delay
      skipeq
      dec   Delay


WAITHALFLINE:
      in      Arg1, TIFR                ;wait for two timer overflows
      sbrs    Arg1, TOV0                ;(each overflow is a quarter line)
      rjmp    WAITHALFLINE
      ldi     Arg1, EXP2(TOV0)          ;reset timer overflow bit
      out     TIFR, Arg1
SECONDQUARTER:
      in      Arg1, TIFR                ;wait second overflow
      sbrs    Arg1, TOV0                ;
      rjmp    SECONDQUARTER
      ldi     Arg1, EXP2(TOV0)          ;reset timer overflow bit
      out     TIFR, Arg1
      andi  MiscFlags, 0b11111101
      ori   MiscFlags, 0b00000001   ;and align position counter (1)
      sbi     PortD, CSyncBit           ;and end composite sync
      rjmp    TIMER_EXIT


;*************************************************************************
;* SCREEN MAKEUP JUMP STRUCTURE
```

```
;* This compare and jump table determines the appearence of the display
;* at each step, the current line number is examinated to decide
;* which display routine to use
;*****************************************************************************

BUILDSCREEN:
      ;it's time to decide what to display, depending on current line
      ldi   Arg1, ALTEZZABIGNUMBERS
      cpi   RowsDivFour, 220/4
      skipne
      mov   AltezzaPixel, Arg1

      cpi   RowsDivFour, 288/4
      skiplo
      rjmp  VOIDLINE
      mov   Arg1, NumeroScansione
      inc   Arg1
      cpi   RowsDivFour, 224/4
      skiplo
      rjmp  DISPLAYBIGNUMBERS
      cpi     RowsDivFour, 220/4    ;blank display
      skiplo
      rjmp    VOIDLINE
        cpi     RowsDivFour, 35/4
      skiplo
      rjmp    DISPLAYTABLE

      clr   Unita
      clr   RunningNumber
      ldi   Arg1, ALTEZZANUMERITABELLA
      mov   AltezzaPixel, Arg1
        rjmp    VOIDLINE




VOIDLINE:
      clr   RowOffset
      rjmp    TIMER_EXIT



;*****************************************************************************
;* ON SCREEN DISPLAY ROUTINES
;*****************************************************************************
ARGUMENT1 Counter
WHITELINE:
      sbi   PortD, VideoBit
      ldi   Counter, 200
_wh1:
      dec   Counter
      nop
      brne  _wh1
      cbi   PortD,VideoBit
      rjmp  DISPLAYLINERESTART


;*****************************************************************************
;* DISPLAYTABLE
;*****************************************************************************
```

```
ARGUMENT1 Output
ARGUMENT2 Temp
ARGUMENT2 PatternUnita

ARGUMENT3 PatternDecine
ARGUMENT4 PatternUltimaDecina
ARGUMENT5 Counter

DISPLAYTABLE:
      sbic  EECR, EEWE  ;IF WRITE IN PROGRESS, SKIP
      rjmp  VOIDLINE

      sbi   PortD, GreenBit
      in    Output, PortD

      clt               ;separazioni orizzontali
      cpi   Decine, 3
      skipne
      set
      cpi   Decine, 6
      skipne
      set
      ldi   Temp, 3
      cp    AltezzaPixel, Temp
      skipeq
      clt
      cpi   RowOffset, 0
      skipeq
      clt
      bld   Output, VideoBit
      out   PortD, Output



      mov   RunningNumber,Decine
      lsl   RunningNumber
      lsl   RunningNumber
      lsl   RunningNumber
      add   RunningNumber,Decine
      add   RunningNumber,Decine    ;runningnumber=decine*10

      mov   EstrattiLow, CopiaEstrattiLow
      mov   EstrattiHigh, CopiaEstrattiHigh

      rol   EstrattiLow
      rol   EstrattiHigh
      rol   EstrattiLow
      rol   EstrattiHigh
      rol   EstrattiLow
      rol   EstrattiHigh
      rol   EstrattiLow
      rol   EstrattiHigh
      rol   EstrattiLow
      rol   EstrattiHigh

      inc   Unita
      in    Output, PortD           ;read PortB status
      ldi   Counter,8

      mov   Temp, RowOffset
      add   Temp, Decine
      out     EEAR,  Temp             ;set eeprom address register
      sbi     EECR, 0                 ;send read command
      in      CopiaPatternDecine,EEDR ;read font table in EEPROM
```

```
      tst   Decine                  ;suppress leading zero
      skipne
      clr   CopiaPatternDecine

      inc   Temp              ;load pattern for last column
      out     EEAR,  Temp             ;set eeprom address register
      sbi     EECR, 0                 ;send read command
      in      PatternUltimaDecina,EEDR      ;read font table in EEPROM

DISPLAYNUMBER:


      mov   Temp, RowOffset         ;get pattern for the units
      add   Temp, Unita
      out     EEAR, Temp            ;set eeprom address register
      sbi     EECR, 0                 ;send read command
      in      PatternUnita, EEDR      ;read font table in EEPROM
      mov   PatternDecine, CopiaPatternDecine  ;get the pattern for tens


      cpi   Unita, 6
      brne  NOSEPARATORE
      in    Output, PortD
      sbi   PortD, VideoBit
      nop
      out   PortD, Output

NOSEPARATORE:
      cpi   RowOffset, 60           ;from row 6 on, blank display
      skiplo
      clr   PatternDecine
      skiplo
      clr   PatternUnita
      cpi   RowOffset, 0
      brne  BITBANG

TESTESTRATTO:
      mov   Temp, RunningNumber     ;temp=number to examine
      andi  Temp, 0b00111111  ;temp=temp mod 64 to get real address
      out   EEAR, Temp
      sbi   EECR, 0
      cp    Temp, RunningNumber
      in    Temp, EEDR
      skipne
      bst   Temp, 0
      skipeq
      bst   Temp, 1
      bld   CopiaEstrattiLow,0
      rol   CopiaEstrattiLow
      rol   CopiaEstrattiHigh
      inc   RunningNumber
      inc   Unita
      nop
      doublenop
      doublenop
      doublenop
      doublenop
      doublenop
      nop
      clt
      cpi   Decine, 3
      skipne
      set
      cpi   Decine, 6
      skipne
```

```
      set
      ldi   Temp, 3
      cp    AltezzaPixel, Temp
      skipeq
      clt
      bld   Output, VideoBit
      out   PortD, Output

      rjmp  ENDBITBANG



BITBANG:                            ;pixel bitbanging
      rol   EstrattiLow
      rol   EstrattiHigh
      skipcc
      rjmp  BANGBIANCO
      nop

;     skipcs
;     ldi   PatternDecine, 0
;     skipcs
;     ldi   PatternUnita, 0

BANGVERDE:
      bst   PatternDecine, 7
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternDecine, 6
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternDecine, 5
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternDecine, 4
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternDecine, 3
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternDecine, 2
      bld   Output, FastBlank
      out   PortD, Output
      inc   Unita             ;increase now to make black space
      bst   PatternUnita, 7
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternUnita, 6
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternUnita, 5
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternUnita, 4
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternUnita, 3
      bld   Output, FastBlank
      out   PortD, Output
      bst   PatternUnita, 2
      bld   Output, FastBlank
      out   PortD, Output
      rjmp  ENDBITBANG
```

```
BANGBIANCO:
      bst   PatternDecine, 7        ;read bit 7 into T
      bld   Output, VideoBit  ;and copy it in PortB buffer
      out   PortD, Output           ;set PortB accordingly
      bst   PatternDecine, 6        ;repeat for the next 4 bits
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternDecine, 5
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternDecine, 4
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternDecine, 3
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternDecine, 2
      bld   Output, VideoBit
      out   PortD, Output

      ;units digit

      inc   Unita             ;increase now to make black space

      bst   PatternUnita, 7         ;read bit 7 into T
      bld   Output, VideoBit  ;and copy it in output
      out   PortD, Output           ;set hardware
      bst   PatternUnita, 6
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternUnita, 5
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternUnita, 4
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternUnita, 3
      bld   Output, VideoBit
      out   PortD, Output
      bst   PatternUnita, 2
      bld   Output, VideoBit
      out   PortD, Output
      rjmp  ENDBITBANG

ENDBITBANG:
      subi  Counter, 1
      skipmi
      rjmp  DISPLAYNUMBER
      clr   Unita
      mov   CopiaPatternDecine, PatternUltimaDecina
      cpi   Counter, 255
      skipne
      rjmp  DISPLAYNUMBER

      dec   AltezzaPixel
      brne  fine

      ldi   Temp, ALTEZZANUMERITABELLA
      mov   AltezzaPixel, Temp
      addi  RowOffset, 10           ;increment offset
      cpi   RowOffset, 70
      brne  fine
      ldi   RowOffset, 0
      inc   Decine
fine:
```

```
      cbi   PortD, VideoBit
      cbi   PortD, GreenBit
      rjmp    DISPLAYLINERESTART      ;end line




;***************************************************************************
;* DISPLAY BIG NUMBER
;***************************************************************************


ARGUMENT1 Number
ARGUMENT2 Decine
ARGUMENT3 Counter
ARGUMENT4 ShiftPattern
ARGUMENT5 TimeDelay

DISPLAYBIGNUMBERS:
      sbic  EECR, EEWE  ;IF WRITE IN PROGRESS, SKIP
_SKIP:      rjmp  VOIDLINE
      cpi   Number, 90+1
      breq  _SKIP
      cpi   Number, 0
      breq  _SKIP

      clr   Decine
      clr   TimeDelay
BIN2BCD:
      cpi   Number, 10
      brlo  READPATTERNS
      subi  Number, 10
      inc   Decine
      rjmp  BIN2BCD               ;ora Number contiene le unità e decine le
decine

READPATTERNS:
      cpi   Decine, 0         ;skip leading zero
      breq  _LZ1


      add   Decine, RowOffset
      out     EEAR, Decine              ;set eeprom address register
      sbi     EECR, 0                 ;send read command
      in      Decine, EEDR          ;read font table in EEPROM

_LZ1:
      add   Number, RowOffset
      out     EEAR, Number              ;set eeprom address register
      sbi     EECR, 0                 ;send read command
      in      Number, EEDR          ;read font table in EEPROM

      ldi Counter, 161
      rcall ZEBRA

      mov   ShiftPattern, Decine
      ldi   Counter, 6
      rcall SHIFTER

      mov   ShiftPattern, Number
      ldi   Counter, 6
```

```
      rcall SHIFTER

      dec   AltezzaPixel
      brne  END_DISPBIGNUM
      addi  RowOffset, 10
      ldi   Number, ALTEZZABIGNUMBERS
      mov   AltezzaPixel, Number
END_DISPBIGNUM:
      rjmp    DISPLAYLINERESTART      ;end line

SHIFTER:
      rol   ShiftPattern
      skipcc
      sbi   PortD, VideoBit
      skipcs
      cbi   PortD, VideoBit
_L1:  inc   TimeDelay
      cpi   TimeDelay, LARGHEZZABIGNUMBERS
      brlo  _L1
      clr   TimeDelay
      dec   Counter
      brne  SHIFTER
      cbi   PortD, VideoBit
      ret


ZEBRA:
      dec   Counter
      brne  ZEBRA
      ret


;***************************************************************************
;* Official line exit point for all display routines
;***************************************************************************

DISPLAYLINERESTART:
      ldi     Arg1, EXP2(TOV0)        ;clear timer overflow bit
      out     TIFR, Arg1              ;
      rjmp  WAITNEWLINE






;***************************************************************************
*******
;* EEDRIVER:
;* driver per la lettura e scrittura dell'EEprom da parte del main
;* la lettura e scrittura è sempre di un singolo BIT mappato nello spazio virtuale
da 1 a 90
;* usa i flag seguenti di MiscFlags:
;* EEDataBit -> di da leggere o scrivere
;* EEReadRequestBit -> legge all'indirizzo EEDriverRegister
;* EEWriteRequestBit -> scrive il bit EEDataBit all'indirizzo EEDriverRegister
;* il completamento dell'operazione è segnalato dall'azzeramento del flag relativo


ARGUMENT1 EEAddress
ARGUMENT2 Temp

EEDRIVER:
```

```
      sbrs  MiscFlags, EEReadReqBit       ;is read requested?
      rjmp  _EE1                    ;no, go check write request

      ;*******READ*****
      cbr   MiscFlags, EXP2(EEReadReqBit) ;yes, clear service request
      mov   EEAddress, EEDriverRegister   ;copy address
      andi  EEAddress, 0b00111111         ;map into memory
      bst   EEDriverRegister, 6           ;check bit 6 address
      out     EEAR,  EEAddress                ;set eeprom address register

      ldi   TEMP, 1<<0
      out   EECR, TEMP
      out   EECR, TEMP              ;send read command
      in      EEDriverRegister,EEDR       ;read EEPROM contents
      brts    _EE2                        ;if number > 63, go read bit 1
      bst   EEDriverRegister, 0           ;else read bit 0
      bld   MiscFlags, EEDataBit
      ret

_EE2: bst   EEDriverRegister, 1           ;read bit 1 (number greater than 63)
      bld   MiscFlags, EEDataBit
      ret

      ;*******WRITE******

_EE1: cpi   RowsDivFour, 288/4            ;are we near vertical retrace?
      skipne                              ;no, go return
      sbrs  MiscFlags, EEWriteReqBit      ;is write requested?
      ret                           ;no, return

      sbrs  MiscFlags, EEWritePendingBit  ;is write pending?
      rjmp  _EE3                    ;no, go write byte
                                    ;else check for write completion
      sbis  EECR, 1                       ;is write completed?
      cbr   MiscFlags,
EXP2(EEWriteReqBit)+EXP2(EEWritePendingBit)+EXP2(BlankScreenBit);yes, clear
      ret

_EE3:
      mov   EEAddress, EEDriverRegister   ;copy address
      andi  EEAddress, 0b00111111         ;map into memory
      out     EEAR,  EEAddress                ;set eeprom address register

      ldi   TEMP, 1<<0
      out   EECR, TEMP
      out   EECR, TEMP

      in      Temp,EEDR                   ;read EEPROM contents
      bst   MiscFlags, EEDataBit          ;store data to write
      cp    EEDriverRegister, EEAddress   ;if lower 63 bits
      skipne
      bld   Temp, 0                       ;then use bit 0
      skipeq
      bld   Temp, 1                       ;else use bit 1
      out   EEDR, Temp
      sbi     EECR, 1                     ;send write command to store data
      sbr   MiscFlags, EXP2(BlankScreenBit)+EXP2(EEWritePendingBit)    ;blank
screen....
      ret
```

FONTS.INC

```
;*****************************************************************************
;* FONTS.INC
;* fonts shape definitions
;* Copyright (c) 1997 by Alberto Ricci Bitti
;* a.riccibitti@iname.com
;*****************************************************************************


;* include byte values defined as strings for easy font shape declarations
.include "vis_byte.inc"


.ESEG                   ;character set tables stored in EEPROM
                    ;each 5x7 font matrix takes 5 byte, rotated 90 degrees


      ;character set tables stored in EEPROM
      ;each 5x5 font matrix takes 5 bytes



      ;zero
.equ      ee00      =    _ooo____
.equ      ee01      =    o___o___
.equ      ee02      =    o___o___
.equ      ee03      =    o___o___
.equ      ee04      =    _ooo____
;one
.equ      ee05      =    __o_____
.equ      ee06      =    _oo_____
.equ      ee07      =    __o_____
.equ      ee08      =    __o_____
.equ      ee09      =    _ooo____
;two
.equ      ee10      =    oooo____
.equ      ee11      =    ____o___
.equ      ee12      =    __oo____
.equ      ee13      =    _o______
.equ      ee14      =    ooooo___
;three
.equ      ee15      =    oooo____
.equ      ee16      =    ____o___
.equ      ee17      =    _ooo____
.equ      ee18      =    ____o___
.equ      ee19      =    oooo____
;four
.equ      ee20      =    __oo____
.equ      ee21      =    _o_o____
.equ      ee22      =    o__o____
.equ      ee23      =    ooooo___
.equ      ee24      =    ___o____
;five
.equ      ee25      =    ooooo___
.equ      ee26      =    o_______
.equ      ee27      =    oooo____
.equ      ee28      =    ____o___
.equ      ee29      =    oooo____
      ;six
.equ      ee30      =    _oooo___
.equ      ee31      =    o_______
.equ      ee32      =    oooo____
.equ      ee33      =    o___o___
.equ      ee34      =    _ooo____
;seven
.equ      ee35      =    ooooo___
.equ      ee36      =    ____o___
.equ      ee37      =    ___o____
.equ      ee38      =    __o_____
```

```
.equ      ee39      =    _o______
;eight
.equ      ee40      =    _ooo____
.equ      ee41      =    o___o___
.equ      ee42      =    _ooo____
.equ      ee43      =    o___o___
.equ      ee44      =    _ooo____
;nine
.equ      ee45      =    _ooo____
.equ      ee46      =    o___o___
.equ      ee47      =    _oooo___
.equ      ee48      =    ____o___
.equ      ee49      =    oooo____
;
.equ      ee50      =    _________
.equ      ee51      =    _________
.equ      ee52      =    _________
.equ      ee53      =    _________
.equ      ee54      =    _________
;
.equ      ee55      =    _________
.equ      ee56      =    _________
.equ      ee57      =    _________
.equ      ee58      =    _________
.equ      ee59      =    _________
;
.equ      ee60      =    _________
.equ      ee61      =    _________
.equ      ee62      =    _________
.equ      ee63      =    _________


;*ricostruzione dell'eeprom interlacciata
.ORG 0

.DB   0
.DB   0
.DB   0
.DB   0
.DB   0
.DB   0
.DB   0
.DB   0
.DB   0


.DB  ee00
.DB  ee05
.DB  ee10
.DB  ee15
.DB  ee20
.DB  ee25
.DB  ee30
.DB  ee35
.DB  ee40
.DB  ee45

.DB  ee01
.DB  ee06
.DB  ee11
.DB  ee16
.DB  ee21
.DB  ee26
.DB  ee31
.DB  ee36
.DB  ee41
.DB  ee46

.DB  ee02
.DB  ee07
.DB  ee12
```

```
.DB  ee17
.DB  ee22
.DB  ee27
.DB  ee32
.DB  ee37
.DB  ee42
.DB  ee47

.DB  ee03
.DB  ee08
.DB  ee13
.DB  ee18
.DB  ee23
.DB  ee28
.DB  ee33
.DB  ee38
.DB  ee43
.DB  ee48

.DB  ee04
.DB  ee09
.DB  ee14
.DB  ee19
.DB  ee24
.DB  ee29
.DB  ee34
.DB  ee39
.DB  ee44
.DB  ee49

.DB    0
.DB    0
.DB    0
.DB    0
```

VIS_BYTE.INC

```
;*****************************************************************************
;* visual byte .EQUs
;* Copyright (c) 1997 by Alberto Ricci Bitti
;*
;* a.riccibitti@iname.com
;*
;* each byte value is redefined as a string were the dots mean 1s and the
;* underscores mean 0s.
;* very useful when the byte is used to reprent patterns, as in font shape tables
;*****************************************************************************

        .EQU    ________ = 0
        .EQU    _______o = 1
        .EQU    ______o_ = 2
        .EQU    ______oo = 3
        .EQU    _____o__ = 4
        .EQU    _____o_o = 5
        .EQU    _____oo_ = 6
        .EQU    _____ooo = 7
        .EQU    ____o___ = 8
        .EQU    ____o__o = 9
        .EQU    ____o_o_ = 10
        .EQU    ____o_oo = 11
        .EQU    ____oo__ = 12
        .EQU    ____oo_o = 13
        .EQU    ____ooo_ = 14
        .EQU    ____oooo = 15
        .EQU    ___o____ = 16
        .EQU    ___o___o = 17
        .EQU    ___o__o_ = 18
        .EQU    ___o__oo = 19
        .EQU    ___o_o__ = 20
        .EQU    ___o_o_o = 21
        .EQU    ___o_oo_ = 22
        .EQU    ___o_ooo = 23
        .EQU    ___oo___ = 24
        .EQU    ___oo__o = 25
        .EQU    ___oo_o_ = 26
        .EQU    ___oo_oo = 27
        .EQU    ___ooo__ = 28
        .EQU    ___ooo_o = 29
        .EQU    ___oooo_ = 30
        .EQU    ___ooooo = 31
        .EQU    __o_____ = 32
        .EQU    __o____o = 33
        .EQU    __o___o_ = 34
        .EQU    __o___oo = 35
        .EQU    __o__o__ = 36
        .EQU    __o__o_o = 37
        .EQU    __o__oo_ = 38
        .EQU    __o__ooo = 39
        .EQU    __o_o___ = 40
        .EQU    __o_o__o = 41
        .EQU    __o_o_o_ = 42
        .EQU    __o_o_oo = 43
        .EQU    __o_oo__ = 44
        .EQU    __o_oo_o = 45
        .EQU    __o_ooo_ = 46
        .EQU    __o_oooo = 47
        .EQU    __oo____ = 48
        .EQU    __oo___o = 49
        .EQU    __oo__o_ = 50
        .EQU    __oo__oo = 51
        .EQU    __oo_o__ = 52
        .EQU    __oo_o_o = 53
        .EQU    __oo_oo_ = 54
        .EQU    __oo_ooo = 55
        .EQU    __ooo___ = 56
        .EQU    __ooo__o = 57
```

```
.EQU   __ooo_o_ = 58
.EQU   __ooo_oo = 59
.EQU   __oooo__ = 60
.EQU   __oooo_o = 61
.EQU   __ooooo_ = 62
.EQU   __oooooo = 63


.EQU   _o______ = 64
.EQU   _o_____o = 65
.EQU   _o____o_ = 66
.EQU   _o____oo = 67
.EQU   _o___o__ = 68
.EQU   _o___o_o = 69
.EQU   _o___oo_ = 70
.EQU   _o___ooo = 71
.EQU   _o__o___ = 72
.EQU   _o__o__o = 73
.EQU   _o__o_o_ = 74
.EQU   _o__o_oo = 75
.EQU   _o__oo__ = 76
.EQU   _o__oo_o = 77
.EQU   _o__ooo_ = 78
.EQU   _o__oooo = 79
.EQU   _o_o____ = 80
.EQU   _o_o___o = 81
.EQU   _o_o__o_ = 82
.EQU   _o_o__oo = 83
.EQU   _o_o_o__ = 84
.EQU   _o_o_o_o = 85
.EQU   _o_o_oo_ = 86
.EQU   _o_o_ooo = 87
.EQU   _o_oo___ = 88
.EQU   _o_oo__o = 89
.EQU   _o_oo_o_ = 90
.EQU   _o_oo_oo = 91
.EQU   _o_ooo__ = 92
.EQU   _o_ooo_o = 93
.EQU   _o_oooo_ = 94
.EQU   _o_ooooo = 95
.EQU   _oo_____ = 96
.EQU   _oo____o = 97
.EQU   _oo___o_ = 98
.EQU   _oo___oo = 99
.EQU   _oo__o__ = 100
.EQU   _oo__o_o = 101
.EQU   _oo__oo_ = 102
.EQU   _oo__ooo = 103
.EQU   _oo_o___ = 104
.EQU   _oo_o__o = 105
.EQU   _oo_o_o_ = 106
.EQU   _oo_o_oo = 107
.EQU   _oo_oo__ = 108
.EQU   _oo_oo_o = 109
.EQU   _oo_ooo_ = 110
.EQU   _oo_oooo = 111
.EQU   _ooo____ = 112
.EQU   _ooo___o = 113
.EQU   _ooo__o_ = 114
.EQU   _ooo__oo = 115
.EQU   _ooo_o__ = 116
.EQU   _ooo_o_o = 117
.EQU   _ooo_oo_ = 118
.EQU   _ooo_ooo = 119
.EQU   _oooo___ = 120
.EQU   _oooo__o = 121
.EQU   _oooo_o_ = 122
.EQU   _oooo_oo = 123
.EQU   _ooooo__ = 124
.EQU   _ooooo_o = 125
.EQU   _oooooo_ = 126
.EQU   _ooooooo = 127
```

```
.EQU   o_______ =   128
.EQU   o______o =   129
.EQU   o_____o_ =   130
.EQU   o_____oo =   131
.EQU   o____o__ =   132
.EQU   o____o_o =   133
.EQU   o____oo_ =   134
.EQU   o____ooo =   135
.EQU   o___o___ =   136
.EQU   o___o__o =   137
.EQU   o___o_o_ =   138
.EQU   o___o_oo =   139
.EQU   o___oo__ =   140
.EQU   o___oo_o =   141
.EQU   o___ooo_ =   142
.EQU   o___oooo =   143
.EQU   o__o____ =   144
.EQU   o__o___o =   145
.EQU   o__o__o_ =   146
.EQU   o__o__oo =   147
.EQU   o__o_o__ =   148
.EQU   o__o_o_o =   149
.EQU   o__o_oo_ =   150
.EQU   o__o_ooo =   151
.EQU   o__oo___ =   152
.EQU   o__oo__o =   153
.EQU   o__oo_o_ =   154
.EQU   o__oo_oo =   155
.EQU   o__ooo__ =   156
.EQU   o__ooo_o =   157
.EQU   o__oooo_ =   158
.EQU   o__ooooo =   159
.EQU   o_o_____ =   160
.EQU   o_o____o =   161
.EQU   o_o___o_ =   162
.EQU   o_o___oo =   163
.EQU   o_o__o__ =   164
.EQU   o_o__o_o =   165
.EQU   o_o__oo_ =   166
.EQU   o_o__ooo =   167
.EQU   o_o_o___ =   168
.EQU   o_o_o__o =   169
.EQU   o_o_o_o_ =   170
.EQU   o_o_o_oo =   171
.EQU   o_o_oo__ =   172
.EQU   o_o_oo_o =   173
.EQU   o_o_ooo_ =   174
.EQU   o_o_oooo =   175
.EQU   o_oo____ =   176
.EQU   o_oo___o =   177
.EQU   o_oo__o_ =   178
.EQU   o_oo__oo =   179
.EQU   o_oo_o__ =   180
.EQU   o_oo_o_o =   181
.EQU   o_oo_oo_ =   182
.EQU   o_oo_ooo =   183
.EQU   o_ooo___ =   184
.EQU   o_ooo__o =   185
.EQU   o_ooo_o_ =   186
.EQU   o_ooo_oo =   187
.EQU   o_oooo__ =   188
.EQU   o_oooo_o =   189
.EQU   o_ooooo_ =   190
.EQU   o_oooooo =   191


.EQU   oo______ =   192
.EQU   oo_____o =   193
.EQU   oo____o_ =   194
.EQU   oo____oo =   195
.EQU   oo___o__ =   196
```

```
.EQU   oo___o_o =   197
.EQU   oo___oo_ =   198
.EQU   oo___ooo =   199
.EQU   oo__o___ =   200
.EQU   oo__o__o =   201
.EQU   oo__o_o_ =   202
.EQU   oo__o_oo =   203
.EQU   oo__oo__ =   204
.EQU   oo__oo_o =   205
.EQU   oo__ooo_ =   206
.EQU   oo__oooo =   207
.EQU   oo_o____ =   208
.EQU   oo_o___o =   209
.EQU   oo_o__o_ =   210
.EQU   oo_o__oo =   211
.EQU   oo_o_o__ =   212
.EQU   oo_o_o_o =   213
.EQU   oo_o_oo_ =   214
.EQU   oo_o_ooo =   215
.EQU   oo_oo___ =   216
.EQU   oo_oo__o =   217
.EQU   oo_oo_o_ =   218
.EQU   oo_oo_oo =   219
.EQU   oo_ooo__ =   220
.EQU   oo_ooo_o =   221
.EQU   oo_oooo_ =   222
.EQU   oo_ooooo =   223
.EQU   ooo_____ =   224
.EQU   ooo____o =   225
.EQU   ooo___o_ =   226
.EQU   ooo___oo =   227
.EQU   ooo__o__ =   228
.EQU   ooo__o_o =   229
.EQU   ooo__oo_ =   230
.EQU   ooo__ooo =   231
.EQU   ooo_o___ =   232
.EQU   ooo_o__o =   233
.EQU   ooo_o_o_ =   234
.EQU   ooo_o_oo =   235
.EQU   ooo_oo__ =   236
.EQU   ooo_oo_o =   237
.EQU   ooo_ooo_ =   238
.EQU   ooo_oooo =   239
.EQU   oooo____ =   240
.EQU   oooo___o =   241
.EQU   oooo__o_ =   242
.EQU   oooo__oo =   243
.EQU   oooo_o__ =   244
.EQU   oooo_o_o =   245
.EQU   oooo_oo_ =   246
.EQU   oooo_ooo =   247
.EQU   ooooo___ =   248
.EQU   ooooo__o =   249
.EQU   ooooo_o_ =   250
.EQU   ooooo_oo =   251
.EQU   oooooo__ =   252
.EQU   oooooo_o =   253
.EQU   ooooooo_ =   254
.EQU   oooooooo =   255
```

1200DEF.INC

```
;*****************************************************************************
;* A P P L I C A T I O N   N O T E   F O R   T H E   A V R   F A M I L Y
;*
;* Number           :AVR000
;* File Name        :"1200def.inc"
;* Title            :Register/Bit Definitions for the AT90S1200
;* Date                     :99.01.28
;* Version          :1.30
;* Support telephone        :+47 72 88 43 88 (ATMEL Norway)
;* Support fax              :+47 72 88 43 99 (ATMEL Norway)
;* Support E-Mail   :avr@atmel.com
;* Target MCU       :AT90S1200
;*
;* DESCRIPTION
;* When including this file in the assembly program file, all I/O register
;* names and I/O register bit names appearing in the data book can be used.
;*
;* The Register names are represented by their hexadecimal addresses.
;*
;* The Register Bit names are represented by their bit number (0-7).
;*
;* Please observe the difference in using the bit names with instructions
;* such as "sbr"/"cbr" (set/clear bit in register) and "sbrs"/"sbrc"
;* (skip if bit in register set/cleared). The following example illustrates
;* this:
;*
;* in  r16,PORTB            ;read PORTB latch
;* sbr r16,(1<<PB6)+(1<<PB5)      ;set PB6 and PB5 (use masks, not bit#)
;* out  PORTB,r16           ;output to PORTB
;*
;* in  r16,TIFR             ;read the Timer Interrupt Flag Register
;* sbrc       r16,TOV0             ;test the overflow flag (use bit#)
;* rjmp       TOV0_is_set          ;jump if set
;* ...                      ;otherwise do something else
;*****************************************************************************

;***** Specify Device
.device AT90S1200

;***** I/O Register Definitions
.equ   SREG   =$3f
.equ   GIMSK  =$3b
.equ   TIMSK  =$39
.equ   TIFR   =$38
.equ   MCUCR  =$35
.equ   TCCR0  =$33
.equ   TCNT0  =$32
.equ   WDTCR  =$21
.equ   EEAR   =$1e
.equ   EEDR   =$1d
.equ   EECR   =$1c
.equ   PORTB  =$18
.equ   DDRB   =$17
.equ   PINB   =$16
.equ   PORTD  =$12
.equ   DDRD   =$11
.equ   PIND   =$10
.equ   ACSR   =$08

;***** Bit Definitions

.equ   INT0   =6

.equ   TOIE0  =1

.equ   TOV0   =1

.equ   SE     =5
.equ   SM     =4
```

```
.equ   ISC01  =1
.equ   ISC00  =0

.equ   CS02   =2
.equ   CS01   =1
.equ   CS00   =0

.equ   WDE    =3
.equ   WDP2   =2
.equ   WDP1   =1
.equ   WDP0   =0

.equ   EEWE   =1
.equ   EERE   =0

.equ   PB7    =7
.equ   PB6    =6
.equ   PB5    =5
.equ   PB4    =4
.equ   PB3    =3
.equ   PB2    =2
.equ   PB1    =1
.equ   PB0    =0

.equ   DDB7   =7
.equ   DDB6   =6
.equ   DDB5   =5
.equ   DDB4   =4
.equ   DDB3   =3
.equ   DDB2   =2
.equ   DDB1   =1
.equ   DDB0   =0

.equ   PINB7  =7
.equ   PINB6  =6
.equ   PINB5  =5
.equ   PINB4  =4
.equ   PINB3  =3
.equ   PINB2  =2
.equ   PINB1  =1
.equ   PINB0  =0

.equ   PD6    =6
.equ   PD5    =5
.equ   PD4    =4
.equ   PD3    =3
.equ   PD2    =2
.equ   PD1    =1
.equ   PD0    =0

.equ   DDD6   =6
.equ   DDD5   =5
.equ   DDD4   =4
.equ   DDD3   =3
.equ   DDD2   =2
.equ   DDD1   =1
.equ   DDD0   =0

.equ   PIND6  =6
.equ   PIND5  =5
.equ   PIND4  =4
.equ   PIND3  =3
.equ   PIND2  =2
.equ   PIND1  =1
.equ   PIND0  =0

.equ   ACD    =7
.equ   ACO    =5
.equ   ACI    =4
.equ   ACIE   =3
.equ   ACIS1  =1
.equ   ACIS0  =0
```

```
.equ   XRAMEND =0
.equ   E2END  =3F
.equ   FLASHEND=1FF

.equ   INT0addr=$001;External Interrupt0 Vector Address
.equ   OVF0addr=$002;Overflow0 Interrupt Vector Address
.equ   ACIaddr =$003;Analog Comparator Interrupt Vector Address

.def   ZL     =r30
```