FRC Evaluation Software (8086 Assembly source code)

I implemented FRC (Frame Rate Control) logic on 82C455, 82C456, and 82C457 to achieve displaying multiple gray levels on monochrome flat panels or multiple colors on monochrome flat panels with RGB color masks/filters (color flat panels).

8086 assembly source code (FRCEVA.ASM) attached below was assembled by using MASM.EXE, LINK.EXE and LIB.EXE under MSDOS (MicroSoft Disk Operating System) console.

The object code (FRCEVA.EXE or FRCEVA.COM) running on IBM PC MSDOS console provided multiple gray level display on flat panel and allowed evaluating the display result with ease to see if the flicker made by FRC is less noticeable. Then, the best FRC scheme obtained was reflected to actual logic design at next stage.

Therefore, FRCEVA.ASM exhibits how the FRC is actually working by the logic implemented on 82C455, 82C456, and 82C457.

In addition to the FRC,

To provide multiple gray levels on Plasma and EL (Electro-Luminescent) flat panels which emit themselves, PWM (Pulse Width Modulation) logic was implemented and applied to Plasma and EL flat panels.

```
;*****************************************************************************
;Program name:   FRCEVA
;Module name:    FRCEVA.ASM
;Description:    The program evaluates FRC (Frame Rate Control) scheme on
;                various panels such as monochrome STN panel, color STN panel,
;                and color TFT panel. To emulate FRC scheme, the display memory
;                stores many frame images created depending on specified FRC
;                scheme or panel type (monochrome or color). The program
;                includes various kind of panel drivers.
;
;Usage:          FRCEVA <space>
;                 for each inquiry, specify operation mode one by one.
;
;Version:        1.6
;Date:           August,1 1990
;
;Programmer:     Tetsuji Oguchi
;
;(c) Chips and Technologies Inc, 1990   (c) ASCII Corporation 1990
;*****************************************************************************


name        FRCscheme
pgroup      group prog
prog  segment     byte public 'prog'
      assume      cs:pgroup,ds:pgroup
      ;
      org   100h
      ;
main  proc  near
      ;
eva:  mov   sp,offset mem_max
      mov   ax,cs
      mov   ss,ax
      mov   ds,ax
restart:    call  setup
      call  sel_FRC                 ;select FRC scheme
      call  sel_image               ;select image
      call  buf_cl                  ;Clear image buffer
      call  FRC_pre                 ;FRC preprocessing
      call  load_mono               ;load display memory data
      call  FRC_exec                ;sweep 3/16/64 frames
      call  FRC_dump                ;dump if necessary
      mov   byte ptr sch_64,0
      cmp   byte ptr scheme,0ffh
      jnz   restart
      mov   ax,4c00h
      int   21h                     ;Exit to DOS
      ;
;************************************
;* Subroutines  {FRC scheme selection} *
;*  [sel_FRC]                         *
;************************************
      ;
;[0]; Basic FRC pattern on Type-0 ("old" scheme)
; | #1 offset 0 | #2 offset 0 | #1 offset 2 | #2 offset 2 |
; | #3 offset 0 | #4 offset 0 | #3 offset 2 | #4 offset 2 |
; | #1 offset 3 | #2 offset 3 | #1 offset 1 | #2 offset 1 |
; | #3 offset 3 | #4 offset 3 | #3 offset 1 | #4 offset 1 |  fixed
;
;[1]; Basic FRC pattern on Type-1 ("Arun's offset")
; | #1 offset 0 | #2 offset 0 | #3 offset 0 | #4 offset 0 |
; | #3 offset 0 | #4 offset 0 | #1 offset 0 | #2 offset 0 |  fixed
;
;[2]; Basic FRC pattern on Type-2 (New-0)
; | #1 offset 0 | #2 offset 0 |
; | #3 offset 0 | #4 offset 0 |  variable
```

```
;
;[3]; Basic FRC pattern on Type-3 (New-1)
; | #1 offset 0 | #2 offset 0 | #1 offset 2 | #2 offset 2 |
; | #3 offset 0 | #4 offset 0 | #3 offset 2 | #4 offset 2 |
; | #1 offset 3 | #2 offset 3 | #1 offset 1 | #2 offset 1 |
; | #3 offset 3 | #4 offset 3 | #3 offset 1 | #4 offset 1 |  variable
;
;[4]; Basic FRC pattern on Type-4 (New-2 should be improved)
; | #1 offset 0 | #2 offset 0 | #3 offset 0 | #4 offset 0 |
; | #3 offset 0 | #4 offset 0 | #1 offset 0 | #2 offset 0 |  variable
;
      ;
sel_FRC:    mov   dx,offset mes_21
      mov   bh,2
      call  pr_key                  ;"mono or color LCD"
      mov   col_mono,ah             ;save panel
      mov   dx,offset mes_05
      mov   bh,5
      call  pr_key                  ;"FRC scheme"
      mov   scheme,ah               ;save selected FRC scheme
      cmp   ah,0
      jnz   sel_f1                  ;except FRC type-0
      mov   byte ptr hoff,0         ;HOFF%=0
      mov   byte ptr voff,0         ;VOFF%=0
      mov   byte ptr hoff1_cal,2    ;HOFF1_CAL%=2
      mov   byte ptr voff1_cal,2    ;VOFF1_CAL%=2
      mov   word ptr hoff1_cl,4     ;HOFF1_CL%=4
      mov   word ptr voff1_cl,4     ;VOFF1_CL%=4
      ret
sel_f1:     cmp   byte ptr scheme,1
      jnz   new_FRC                 ;FRC type-1
      mov   byte ptr hoff1_cal,4    ;HOFF1_CAL%=4
      mov   byte ptr voff1_cal,2    ;VOFF1_CAL%=2
      mov   word ptr hoff1_cl,31    ;HOFF1_CL%=31
      mov   word ptr voff1_cl,15    ;VOFF1_CL%=15
      jmp   short sel_ae
new_FRC:    mov   dx,offset mes_06
      call  pr_key_1
      mov   hoff1_cal,ah            ;HOFF1_CAL%=(?)
      mov   dx,offset mes_07
      call  pr_key_1
      mov   voff1_cal,ah            ;VOFF1_CAL%=(?)
      mov   dx,offset mes_08
      call  pr_key_4
      mov   hoff1_cl,cx             ;HOFF1_CL%=(?)
      mov   dx,offset mes_09
      call  pr_key_4
      mov   voff1_cl,cx             ;VOFF1_CL%=(?)
sel_ae:     mov   dx,offset mes_02
      call  pr_key_1
      mov   hoff,ah                 ;HOFF%=(?)
      mov   dx,offset mes_03
      call  pr_key_1
      mov   voff,ah                 ;VOFF%=(?)
      ret
      ;
;**************************************
;* Subroutines  {sorce image selection} *
;*  [sel_image]                       *
;**************************************
      ;
;
;(1)"select VGA mode; VGA mode 12 or VGA mode 13"
;(2)if VGA mode 12; "select source image", "select tile pattern", "select test pattern"
;   draw test pattern or store image
;(3)if VGA mode 13; "select 64 shades FRC scheme", "select source image"
;   draw test pattern or store image, convert color to monochrome
```

```
;
      ;
sel_i12:    xor   ah,ah
      cmp   byte ptr col_mono,0
      jz    sel_i136              ;mono LCD
      inc   ah
      inc   ah
      jmp   short sel_i12c
      ;
sel_image:  call  save_xr
      mov   dx,offset mes_11      ;"select VGA mode"
      mov   bh,2
      call  pr_key
      add   ah,12h
      mov   vmode,ah              ;vmode=12h, 13h
      cmp   ah,12h
      jz    sel_i12               ;vmode=12
      cmp   byte ptr col_mono,0
      jz    sel_i135              ;mono LCD
      mov   dx,offset mes_22      ;"select 16/64/4/no FRC"
      mov   bh,9
      call  pr_key
sel_i12c:   or    ah,10h          ;set color flag
      push  ax
      mov   dx,offset mes_26      ;"select color dot position"
      mov   bh,3
      call  pr_key
      mov   rgb_seq,ah
      pop   ax
      jmp   short sel_i136
sel_i135:   mov   dx,offset mes_16  ;"select 16/64 FRC"
      mov   bh,3
      call  pr_key
sel_i136:   mov   sch_64,ah
      call  scr_cont
      mov   ah,9bh
      cmp   byte ptr sch_64,0
      jz    sel_i132
      mov   ah,8bh
sel_i132:   cmp   byte ptr scheme,0
      jnz   sel_i133
      and   ah,0f7h
sel_i133:   mov   al,6dh
      call  xr_reg_w              ;XR6D=9bh,8bh,93h,83h
      mov   ah,voff
      shl   ah,1
      shl   ah,1
      shl   ah,1
      shl   ah,1
      and   ah,0f0h
      mov   al,hoff
      and   al,0fh
      or    ah,al
      mov   al,6eh
      call  xr_reg_w              ;XR6E=voff*16+hoff
sel_i131:   mov   dx,offset mes_15  ;"select source"
      mov   bh,3
      call  pr_key
      cmp   ah,1
      jnz   v_1302
      ret                         ;image on screen
v_1302:     cmp   ah,0
      jnz   v13_im                ;image on file
      cmp   byte ptr vmode,13h
      jz    v_1303
      call  v12_tst
      ret
```

```
v_1303:     cmp   byte ptr col_mono,0
      jz    m13_mono_1              ;mono LCD
m13_col_1:  mov   dx,offset mes_25  ;"external pal."
      mov   bh,3
      call  pr_key
      mov   pal13_z,ah              ;store pal mode
      cmp   ah,2
      jnz   m13_c21                 ;except palette input
m13_c22:    mov   dx,offset mes_29  ;"palette value"
      call  pr_key_2
      cmp   cl,40h
      jnc   m13_c22
      jmp   short m13_c23
m13_c21:    mov   dx,offset mes_23  ;"select pattern"
      call  pr_key_2
      cmp   cl,0ffh
      jz    m13_ctst
m13_c23:    mov   ax,13h            ;fill pattern
      int   10h                     ;VGA mode 13
      call  m13_cfill
      ret
m13_ctst:   mov   dx,offset mes_24
      mov   bh,0ch
      call  pr_key                  ;"color repetition"
      push  ax
      mov   ax,13h                  ;test pattern
      int   10h                     ;VGA mode 13
      pop   ax
      call  m13_cbar
      ret
      ;
m13_mono_1: mov   dx,offset mes_19  ;"select pattern"
      call  pr_key_2
      cmp   cl,0ffh
      jz    m13_mtst
      cmp   cl,40h
      jnc   m13_mono_1
      mov   ax,13h                  ;fill pattern
      int   10h                     ;VGA mode 13
      call  m13_mfill
      ret
m13_mtst:   mov   ax,13h            ;test pattern
      int   10h                     ;VGA mode 13
      call  m13_mbar
      ret
      ;
v13_im:     mov   dx,offset mes_14  ;"select source image on file"
      cmp   byte ptr vmode,12h
      jz    sel_23im1
      mov   dx,offset mes_13        ;"select source image on file"
sel_23im1:  mov   bh,3
      call  pr_key
sel_23im:   mov   cl,4
      shl   ah,cl
      mov   al,ah
      xor   ah,ah
      push  ax
      mov   ax,ds
      mov   es,ax
      mov   bx,offset mem_max
      mov   cl,4
      shr   bx,cl
      inc   bx
      mov   ah,4ah
      int   21h
      pop   bx
      jc    rest_xr_e
```

```
      mov   ax,offset c_tail_13
      cmp   byte ptr vmode,13h
      jz    sel_23_1
      mov   ax,offset c_tail_12
sel_23_1:   add   ax,bx
      mov   block+2,ax
      mov   ax,ds
      mov   block+4,ax
      mov   block+8,ax
      mov   block+12,ax
      mov   ah,4bh
      mov   dx,offset path
      mov   bx,offset block
      push  ds
      pop   es
      xor   al,al
      int   21h
      jc    rest_xr_e
      mov   dl,0ah
      mov   ah,2
      int   21h
      mov   dl,0dh
      mov   ah,2
      int   21h
      mov   ah,4dh
      int   21h
rest_xr_e:  call  rest_xr
      ret
      ;
m13_mfill:  call  fill_m13
      jmp   short m13_mdac
m13_mbar:   mov   ax,0a000h
      mov   es,ax
      mov   bl,72
      xor   di,di
v13t_1:     call  ul_draw
      dec   bl
      jnz   v13t_1
      xor   ax,ax
v13t_2:     mov   cx,80
      rep   stosw
      not   ax
      and   ax,3f3fh
      mov   cx,80
      rep   stosw
      not   ax
      and   ax,3f3fh
      add   ax,101h
      cmp   ax,4040h
      jnz   v13t_2
      mov   ax,3f3fh
v13t_3:     mov   cx,80
      rep   stosw
      not   ax
      and   ax,3f3fh
      mov   cx,80
      rep   stosw
      not   ax
      and   ax,3f3fh
      sub   ax,101h
      cmp   al,0ffh
      jnz   v13t_3
m13_mdac:   mov   ax,ds             ;set DAC for mode 13 monochrome
      mov   es,ax
      mov   di,offset epal_w
      xor   ax,ax
v13t_4:     stosw
```

```
      stosb
      add   ax,101h
      jnc   v13t_4
wr_epal:    mov   ax,1012h
      xor   bx,bx
      mov   cx,100h
      mov   dx,offset epal_w
      int   10h
      ret
      ;
fill_m13:   mov   ax,0a000h
      mov   es,ax
      xor   di,di
      mov   al,cl
      mov   cx,0fa00h
      rep   stosb
      ret
      ;
m13_cbar:   inc   ah
      mov   cl,ah
      mov   ch,ah                   ;CH=x pitch, CL=temporary x pitch
      mov   dx,cx                   ;DH=y pitch, DL=temporary y pitch
      xor   di,di
      mov   ax,0a000h
      mov   es,ax
      xor   ax,ax                   ;AH=data, AL=temporary data
m13_cb3:    mov   bx,320
m13_cb1:    stosb
      dec   bx
      jz    m13_cb2
      dec   cl
      jnz   m13_cb1
      mov   cl,ch
      inc   al
      test  al,0fh
      jnz   m13_cb1
      mov   al,ah
      jmp   short m13_cb1
m13_cb2:    cmp   di,0fa00h
      jz    m13_cdac
      mov   cl,ch
      mov   al,ah
      dec   dl
      jnz   m13_cb3
      mov   dl,dh
      add   al,10h
      mov   ah,al
      jmp   short m13_cb3
      ;
m13_cfill:  mov   bx,cx             ;BX=CX=pal data
      call  fill_m13
m13_cdac:   mov   ax,ds
      mov   es,ax
      cmp   pal13_z,1
      jz    m13_c9                ;epal mode 1
      cmp   pal13_z,0
      jz    m13_c16               ;epal mode 0
      mov   bh,bl
      mov   di,offset epal_b
      xor   ax,ax
      mov   cx,80h
      rep   stosw
      mov   ax,bx
      mov   cx,80h
      rep   stosw
      xor   ax,ax
      mov   cx,80h
```

```
      rep   stosw
      jmp   m13_c10
m13_c16:    mov   di,offset epal_b
      mov   ch,10h
m13_c2:     mov   al,3
      mov   cl,10h
m13_c1:     stosb
      add   al,4
      dec   cl
      jnz   m13_c1
      dec   ch
      jnz   m13_c2
      mov   di,offset epal_g
      mov   bh,10h
      mov   ax,303h
m13_c3:     mov   cx,8
      rep   stosw
      add   ax,404h
      dec   bh
      jnz   m13_c3
      mov   di,offset epal_r
      mov   ch,10h
      mov   ax,3f3fh
m13_c5:     mov   bx,4              ;BH=0:add BH=1:sub
      mov   cl,10h
m13_c4:     stosb
      cmp   al,3fh
      jnz   m13_c7
      inc   bh
m13_c7:     add   al,bl
      test  bh,1
      jz    m13_c6
      sub   al,bl
      sub   al,bl
m13_c6:     dec   cl
      jnz   m13_c4
      sub   ah,bl
      mov   al,ah
      dec   ch
      jnz   m13_c5
      jmp   short m13_c10
m13_c9:     mov   di,offset epal_b
      mov   ah,16
      mov   bl,15h
m13_c12:    xor   al,al
m13_c11:    mov   cx,8
      rep   stosb
      add   al,bl
      cmp   al,54h
      jnz   m13_c11
      dec   ah
      jnz   m13_c12
      mov   di,offset epal_g
      mov   ah,64
      mov   bx,4809h
m13_c14:    xor   al,al
m13_c13:    stosb
      add   al,bl
      cmp   al,bh
      jnz   m13_c13
      dec   ah
      jnz   m13_c14
      mov   di,offset epal_r
      xor   al,al
m13_c15:    mov   cx,32
      rep   stosb
      add   al,bl
```

```
      cmp   al,bh
      jnz   m13_c15
m13_c10:    mov   si,offset epal_b
      mov   di,offset epal_w+2
      mov   bl,3
m13_c8:     mov   cx,100h
m13_ca:     movsb
      add   di,2
      loop  m13_ca
      sub   di,301h
      dec   bl
      jnz   m13_c8
      call  wr_epal
      ret
      ;
scr_cont:   cmp   ah,2
      jnz   scr_1
      dec   ah
scr_1:      test  ah,10h
      jz    scr_2
      sub   ah,0eh
scr_2:      mov   cl,3
      shl   ah,cl
      xor   bx,bx
      mov   bl,ah
      mov   cx,4
      mov   ax,ds
      mov   es,ax
      mov   si,offset sch64_0_dat
      add   si,bx
      mov   di,offset esize
      rep   movsw
      ret
      ;
ul_draw:    mov   al,3fh
uld_1:      stosb
      dec   al
      cmp   al,0ffh
      jnz   uld_1
      inc   al
uld_2:      stosb
      inc   al
      cmp   al,40h
      jnz   uld_2
      dec   al
uld_3:      mov   cx,3
      rep   stosb
      dec   al
      cmp   al,0ffh
      jnz   uld_3
      ret
      ;
cl_dm:      mov   ax,0f01h
      jmp   cl_dm_1
      ;
;
;Test pattern generation for VGA mode 12
;
      ;
v12_tst:    mov   dx,offset mes_04  ;"select tile pattern"
      mov   bh,4
      call  pr_key
      cmp   ah,3
      jz    sel_tpat                ;filled bar
      mov   si,offset t_pat
      shl   ah,1
      mov   bl,ah
```

```
      xor   bh,bh
      mov   al,[si]+[bx]           ;store tile foreground pattern for 1st line
      mov   tile_l1,al
      not   al
      mov   tile_t1,al             ;store background tile pattern for 1st line
      mov   al,[si]+[bx]+1         ;store tile foreground pattern for 2nd line
      mov   tile_l2,al
      not   al
      mov   tile_t2,al             ;store background tile pattern for 2nd line
sel_im1:    mov   dx,offset mes_10 ;"select tile brightness"
      call  pr_key_1
      mov   tile_br,ah             ;store foreground tile brightness
      call  cl_dm                  ;clear display memory
      call  v_t_bar_b              ;draw background
      call  v_t_bar_f              ;draw foreground
      ret
      ;
sel_tpat:   mov   dx,offset mes_01
      call  mes_pr                 ;"select test pattern"
      call  key_in
      push  ax
      call  cl_dm                  ;clear display memory
      pop   ax
      mov   ah,al
      cmp   ah,"h"
      jz    h_bar
      cmp   ah,"H"
      jz    h_bar
      cmp   ah,"v"
      jz    v_bar
      cmp   ah,"V"
      jz    v_bar
      call  num_tran
      cmp   al,0ffh
      jz    sel_tpat
      xor   ah,0fh
      mov   al,1                   ;calculate plane mask
cl_dm_1:    call  gr_reg_w         ;GR01=?
      mov   ax,0a000h
      mov   es,ax
      mov   cx,8000h
      mov   ax,0ffffh
      repz  stosw
      jmp   rec_gr1
      ;
v_bar:      mov   tile_t1,0ffh
      mov   tile_t2,0ffh
v_t_bar_b:  mov   ax,0a000h
      mov   es,ax                  ;Set memory window
      mov   bx,0fc8h               ;Set write plane select "F"
                                   ;Set 200 lines repetition
      xor   di,di                  ;Set written address "0000h"
v_bar_1:    mov   ah,bh            ;Load plane select
      mov   al,1
      call  gr_reg_w               ;GR01=?
      mov   al,tile_t1             ;Set fill data
      mov   cx,5                   ;Set 5 bytes repetition
      rep   stosb                  ;Write 5 byte data
      dec   bh
      cmp   bh,0ffh
      jnz   v_bar_1                ;Check horizontal repetition
      mov   bh,0fh                 ;Load initial value "F"
      dec   bl
v_bar_2:    mov   ah,bh            ;Load plane select
      mov   al,1
      call  gr_reg_w               ;GR01=?
      mov   al,tile_t2             ;Set fill data
```

```
      mov   cx,5                  ;Set 5 bytes repetition
      rep   stosb                 ;Write 5 byte data
      dec   bh
      cmp   bh,0ffh
      jnz   v_bar_2               ;Check horizontal repetition
      mov   bh,0fh                ;Load initial value "F"
      dec   bl
      jnz   v_bar_1               ;Check vertical repetition
      jmp   rec_gr1
      ;
h_bar:      mov   ax,0a000h
      mov   es,ax                 ;Set memory window
      mov   bh,0fh                ;Set write plane select "F"
      xor   di,di                 ;Set written address "0000h"
h_bar_1:    mov   ah,bh           ;Load plane select
      mov   al,1
      call  gr_reg_w              ;GR01 write
      mov   ax,0ffffh             ;Set written data "0FFh"
      mov   cx,208h               ;Set repetition words
      rep   stosw                 ;Write 5 byte data
      dec   bh
      cmp   bh,0ffh
      jnz   h_bar_1               ;Check horizontal repetition
rec_gr1:    mov   ax,1
      call  gr_reg_w              ;Recover GR01
      ret
      ;
v_t_bar_f:  mov   ax,0a000h
      mov   es,ax                 ;Set memory window
      mov   ax,100h               ;ah=SR02, al=GR04
      test  tile_br,ah
      jz    vtbf_1
      call  t_bar_f
vtbf_1:     mov   ax,201h         ;ah=SR02, al=GR04
      test  tile_br,ah
      jz    vtbf_2
      call  t_bar_f
vtbf_2:     mov   ax,402h         ;ah=SR02, al=GR04
      test  tile_br,ah
      jz    vtbf_3
      call  t_bar_f
vtbf_3:     mov   ax,803h         ;ah=SR02, al=GR04
      test  tile_br,ah
      jz    vtbf_4
      call  t_bar_f
vtbf_4:     mov   ax,0f02h        ;recover SR02, GR04
      call  sr_reg_w
      mov   ax,0004h
      call  gr_reg_w
      ret
      ;
t_bar_f:    mov   bh,al
      mov   al,02h
      call  sr_reg_w              ;SR02=(?)
      mov   ah,bh
      mov   al,04h
      call  gr_reg_w              ;GR04=(?)
      mov   bx,0fc8h              ;Set write plane select "F"
                                  ;Set 200 lines repetition
      xor   di,di                 ;Set written address "0000h"
t_bar_f1:   mov   al,es:[di]
      or    al,tile_l1
      mov   cx,5                  ;Set 5 bytes repetition
      rep   stosb                 ;Write 5 byte data
      dec   bh
      cmp   bh,0ffh
      jnz   t_bar_f1              ;Check horizontal repetition
```

```
      mov   bh,0fh                  ;Load initial value "F"
      dec   bl
t_bar_f2:   mov   al,es:[di]
      or    al,tile_l2
      mov   cx,5                    ;Set 5 bytes repetition
      rep   stosb                   ;Write 5 byte data
      dec   bh
      cmp   bh,0ffh
      jnz   t_bar_f2                ;Check horizontal repetition
      mov   bh,0fh                  ;Load initial value "F"
      dec   bl
      jnz   t_bar_f1                ;Check vertical repetition
      ret
      ;
;***********************************
;* Subroutines  {FRC preprocessing} *
;***********************************
      ;
FRC_pre:    cmp   byte ptr sch_64,13h
      jnz   frc_p3                  ;except 0 FRC
      jmp   frc0_dith
frc_p3:     xor   ax,ax
      mov   line_c,al               ;LC%=0
      mov   line1_c,al              ;LC1%=0
      mov   line2_c,ax              ;LC2%=0
      mov   voff1,al                ;VOFF1%=0
      call  get_org                 ;get source address
      call  get_ipal
      cmp   byte ptr col_mono,0
      jnz   sv13_yy                 ;color LCD
      call  set_epal_gray
sv13_yy:    call  get_epal
      cmp   byte ptr vmode,12h
      jz    FRC_v12                 ;mode 12
      call  FRC_v13
      ret
FRC_v12:    mov   dx,offset mes_28
      mov   bh,2
      call  pr_key                  ;"D/D emulation"
      mov   dd_emu,ah               ;save selected condition
f12_line:   xor   ax,ax
      mov   chr_c,al                ;CHR%=0
      mov   dot_c,al                ;DC%=0
      mov   dot1_c,al               ;DC1%=0
      mov   dot2_c,ax               ;DC2%=0
      mov   byte ptr dotc_c,2       ;DCC%=2 * color position adjust *
      mov   hoff1,al                ;HOFF1%=0
f12_byte:   push  si
      call  c_b_conv12              ;convert color to brightness
      cmp   byte ptr sch_64,0
      jz    f12_dot                 ;mono
f12_dotc:   call  sel_pat12
      pop   si
      mov   bx,5000h
      call  px_sto16c
      push  si
      cmp   byte ptr rgb_seq,2
      jz    f12_d5
      dec   byte ptr dotc_c         ;* color position adjust *
f12_d5:     dec   byte ptr dotc_c   ;* color position adjust *
      mov   si,offset g_buf
      call  sel_pat121
      pop   si
      mov   bx,6000h
      call  px_sto16c
      push  si
      dec   byte ptr dotc_c         ;* color position adjust *
```

```
      cmp   byte ptr rgb_seq,2
      jz    f12_d6
      inc   byte ptr dotc_c         ;* color position adjust *
      inc   byte ptr dotc_c         ;* color position adjust *
f12_d6:     mov   si,offset r_buf
      call  sel_pat121
      pop   si
      mov   bx,7000h
      call  px_sto16c
      add   byte ptr dotc_c,5       ;* color position adjust *
      cmp   byte ptr rgb_seq,2
      jz    f12_d7
      dec   byte ptr dotc_c         ;* color position adjust *
f12_d7:     call  FRC_cal23h
      mov   al,dot_c
      and   al,7
      jz    f12_a                   ;encouter 8 pixel boundary
      push  si
      jmp   short f12_dotc
f12_dot:    call  sel_pat12         ;select FRC pattern
      pop   si
      call  px_sto16                ;write 1 pixel into image buffer
      call  FRC_cal23h
      mov   al,dot_c
      and   al,7
      jz    f12_a                   ;encounter 8 pixel boundary
      push  si
      jmp   short f12_dot
f12_a:      test  byte ptr dd_emu,1
      jz    f12_a3
      xor   byte ptr dd_emu,8
      cmp   byte ptr dd_emu,9       ;D/D emu+lower panel
      jz    f12_a2
f12_a3:     inc   si
      inc   byte ptr chr_c          ;CHR%=CHR%+1
      cmp   byte ptr chr_c,50h      ;check if 80 character boundary
      jnz   f12_a2                  ;restart inter-8 pixel processing
      call  FRC_cal23v
      mov   ax,1f40h
      test  byte ptr dd_emu,1
      jnz   f12_a4
      mov   ax,3e80h
f12_a4:     cmp   si,ax
      jc    f12_a1                  ;restart 1 line pixel processing
      ret
f12_a1:     jmp   f12_line
f12_a2:     jmp   f12_byte
      ;
frc0_dith:  call  get_ipal
      call  get_epal
      xor   si,si
      xor   di,di
      mov   bx,0a000h
      mov   es,bx
frc0_d3:    mov   cx,50h
frc0_d2:    push  cx
      mov   ah,80h                  ;pixel pointer
frc0_d1:    mov   bl,es:[si]
      xor   bh,bh
      mov   cx,bx                   ;CX=BX=8 bit video
      push  si
      mov   si,offset epal_b
      call  frc0_pat
      mov   bx,5000h
      call  draw_dith
      mov   bx,cx
      mov   si,offset epal_g
```

```
      call  frc0_pat
      mov   bx,6000h
      call  draw_dith
      mov   bx,cx
      mov   si,offset epal_r
      call  frc0_pat
      mov   bx,7000h
      call  draw_dith
      pop   si
      inc   si
      shr   ah,1
      shr   ah,1
      jnc   frc0_d1
      pop   cx
      inc   di
      loop  frc0_d2
      add   di,50h
      cmp   si,0fa00h
      jnz   frc0_d3
      jmp   sv13_q
      ;
set_epal_gray:    mov   ax,101bh
      xor   bx,bx
      mov   cx,100h
      int   10h                   ;set gray scale for monochrome LCD
      ret
      ;
get_epal:   push  si
      mov   ax,ds
      mov   es,ax
      mov   ax,1017h
      xor   bx,bx
      mov   cx,100h
      mov   dx,offset epal_w
      int   10h
      mov   si,offset epal_w+2
      mov   di,offset epal_b
      mov   bl,3
g_epal2:    mov   cx,100h
g_epal1:    movsb
      add   si,2
      loop  g_epal1
      sub   si,301h
      dec   bl
      jnz   g_epal2
      pop   si
      ret
      ;
get_ipal:   mov   ax,ds
      mov   es,ax
      mov   dx,offset ipal
      mov   ax,1009h
      int   10h                   ;get internal color palette
      ret
      ;
get_org:    xor   si,si
      cmp   byte ptr vmode,12h
      jz    get_or1               ;mode 12
      mov   dx,offset mes_12      ;"select image cut Y"
      mov   bh,6
      call  pr_key
;     mov   si,offset sty_m12
;     cmp   byte ptr vmode,12h
;     jz    get_or2               ;mode 12
      mov   si,offset sty_m13
get_or2:    mov   bl,ah
      xor   bh,bh
```

```
      mov   ah,[si]+[bx]
      xor   al,al
      cmp   byte ptr esize,0
      jz    get_or3                 ;640x200
      push  ax
      mov   dx,offset mes_17        ;"select image cut X"
      mov   bh,5
      call  pr_key
      mov   si,offset stx_m13
      mov   bl,ah
      pop   ax
      xor   bh,bh
      mov   al,[si]+[bx]
get_or3:    mov   si,ax             ;SI=source byte address
get_or1:    ret
      ;
draw_dith:  mov   ds,bx
      shr   al,1
      jnc   draw_d1
      or    [di],ah
draw_d1:    shr   al,1
      jnc   draw_d2
      or    [di]+50h,ah
draw_d2:    shr   ah,1
      shr   al,1
      jnc   draw_d3
      or    [di],ah
draw_d3:    shr   al,1
      jnc   draw_d4
      or    [di]+50h,ah
draw_d4:    shl   ah,1
      mov   bx,cs
      mov   ds,bx
      ret
      ;
frc0_pat:   mov   dl,[si]+[bx]      ;DL=6 bit color brightness
      xor   bl,bl
      cmp   dl,8
      jc    frc0_p1
      inc   bl
      cmp   dl,10h
      jc    frc0_p1
      inc   bl
      cmp   dl,20h
      jc    frc0_p1
      inc   bl
      cmp   dl,30h
      jc    frc0_p1
      inc   bl
frc0_p1:    mov   si,offset dith_pat
      mov   al,[si]+[bx]            ;BX=dither pattern
      ret
      ;
FRC_v13:    xor   di,di             ;DI=destination byte address
sv13_st:    xor   ax,ax
      mov   chr_c,al                ;CHR%=0
      mov   dot_c,al                ;DC%=0
      mov   dot1_c,al               ;DC1%=0
      mov   dot2_c,ax               ;DC2%=0
      mov   byte ptr dotc_c,2       ;DCC%=2 * color position adjust *
      mov   hoff1,al                ;HOFF1%=0
sv13_0:     mov   bx,0a000h
      mov   es,bx
      mov   bl,es:[si]
      xor   bh,bh                   ;BX=8 bit video
      push  si
      mov   si,offset epal_b
```

```
      mov   dl,[si]+[bx]            ;DL=6 bit brightness or blue
      cmp   byte ptr col_mono,0
      jz    v13_mfrc                ;mono LCD
      push  bx
      mov   ax,5000h
      cmp   byte ptr sch_64,16h
      jz    FRC_5_dith              ;4 frames 5FRC with dither
      cmp   byte ptr sch_64,17h
      jz    FRC_5_dith              ;2 frames 3FRC with dither
      cmp   byte ptr sch_64,18h
      jz    FRC_5_dith              ;4 frames 3FRC with dither
      call  rgb_sto
      pop   bx
      mov   si,offset epal_g
      mov   dl,[si]+[bx]            ;DL=6 bit green
      push  bx
      call  rgb_sto
      pop   bx
      mov   si,offset epal_r
      mov   dl,[si]+[bx]            ;DL=6 bit red
      call  rgb_sto
      jmp   sv13_e
      ;
FRC_5_dith: add   ah,10h
      cmp   dl,4
      jc    FRC_5_d1
      sub   ah,10h
      call  rgb_sto4
FRC_5_d1:   pop   bx
      mov   si,offset epal_g
      mov   dl,[si]+[bx]            ;DL=6 bit green
      push  bx
      add   ah,10h
FRC_5_d3:   cmp   dl,4
      jc    FRC_5_d2
      sub   ah,10h
      call  rgb_sto4
FRC_5_d2:   pop   bx
      mov   si,offset epal_r
      mov   dl,[si]+[bx]            ;DL=6 bit red
      cmp   dl,4
      jc    sv13_e
      call  rgb_sto4
      jmp   short sv13_e
      ;
v13_mfrc:   cmp   dl,4
      jc    sv13_e                  ;check if less than 3/64
      call  dith_make16
      cmp   byte ptr sch_64,0
      jz    sv13_6                  ;16FRC+dither
      mov   si,offset sub_64
      mov   dh,[si]+[bx]            ;DH=subtraction data
      mov   ax,5000h                ;destination segment
      call  sel_sto64               ;1-16 frame
      call  sel_sto64               ;17-32 frame
      call  sel_sto64               ;33-48 frame
      call  sel_sto64               ;49-64 frame
      jmp   short sv13_e
      ;
sv13_6:     mov   si,offset dither_m
      sub   dl,[si]+[bx]            ;DL=final 4 bit brightness
      mov   bl,dl                   ;BL
      call  sel_pat13
      xchg  si,di
      call  px_sto16
      xchg  si,di
sv13_e:     call  FRC_cal23h
```

```
      pop   si
      mov   al,dot_c
      test  al,1
      jnz   sv13_4
      inc   si
sv13_4:     and   al,7
      jz    sv13_44                 ;within 8 pixels
      jmp   sv13_0
sv13_44:    inc   di
      mov   al,50h                  ;AL=destination bytes of 1 H
      cmp   byte ptr esize,0
      jz    sv13_8                  ;640x200
      shr   al,1
sv13_8:     inc   byte ptr chr_c    ;CHR%=CHR%+1
      cmp   chr_c,al
      jz    sv13_x                  ;end of scan line
      jmp   sv13_0                  ;restart inter-8 pixel processing
sv13_x:     call  FRC_cal23v
      mov   ax,140h                 ;AX=source bytes of 1 H
      mov   bx,3e80h                ;BX=end of destination
      cmp   byte ptr esize,0
      jz    sv13_9                  ;640x200
      shr   ax,1
      shr   bx,1
      shr   bx,1
sv13_9:     test  byte ptr line_c,1
      jz    sv13_3
      sub   si,ax                   ;scan first line again
sv13_5:     jmp   sv13_st           ;restart 1 line pixel processing
sv13_3:     cmp   byte ptr esize,0
      jz    sv13_p                  ;640x200
      add   si,ax                   ;adjust source byte address
sv13_p:     cmp   di,bx
      jc    sv13_5                  ;restart 1 line pixel processing
sv13_q:     mov   ax,12h
      int   10h                     ;set vgamode 12
ld_setup:   mov   ah,80h
ld_setup_1: mov   dx,46e8h
      mov   al,18h
      out   dx,al
      mov   dx,103h
      mov   al,ah
      out   dx,al
      mov   dx,46e8h
      mov   al,8
      out   dx,al
      ret
      ;
rel_setup:  xor   ah,ah
      jmp   short ld_setup_1
      ;
FRC_cal23h: inc   byte ptr dot_c    ;DC%=DC%+1
      inc   byte ptr dot1_c         ;DC1%=DC1%+1
      inc   word ptr dot2_c         ;DC2%=DC2%+1
      mov   al,dot1_c
      cmp   al,hoff1_cal
      jnz   f_c_23h1                ;check HOFF1% calculation timing
      mov   byte ptr dot1_c,0       ;DC1%=0
      mov   ah,hoff
      add   hoff1,ah                ;HOFF1%=HOFF%+HOFF1%
f_c_23h1:   mov   ax,dot2_c
      cmp   ax,hoff1_cl
      jnz   f_c_23he                ;check HOFF1% clear timing
      xor   ax,ax
      mov   dot2_c,ax               ;DC2%=0
      mov   hoff1,al                ;HOFF1%=0
f_c_23he:   ret
```

```
      ;
FRC_cal23v: inc   byte ptr line_c   ;LC%=LC%+1
      inc   byte ptr line1_c        ;LC1%=LC1%+1
      inc   word ptr line2_c        ;LC2%=LC2%+1
      mov   al,line1_c
      cmp   al,voff1_cal
      jnz   f_c_23v1                ;check VOFF1% calculation timing
      mov   byte ptr line1_c,0      ;LC1%=0
      mov   ah,voff
      add   voff1,ah                ;VOFF1%=VOFF%+VOFF1%
f_c_23v1:   mov   ax,line2_c
      cmp   ax,voff1_cl
      jnz   f_c_23ve                ;check VOFF1% clear timing
      xor   ax,ax
      mov   line2_c,ax              ;LC2%=0
      mov   voff1,al                ;VOFF1%=0
f_c_23ve:   ret
      ;
dith_make4: cmp   byte ptr sch_64,16h
      jz    dith_m3                 ;4 frames 5FRC with dither
      add   dl,20h
      shr   dl,1
      jmp   short dith_m4
dith_m3:    add   dl,10h
dith_m4:    shr   dl,1
      shr   dl,1
dith_make16:      mov   al,dl
      and   al,3
      shr   dl,1
      shr   dl,1                    ;DL=4/2 bit brightness, AL=DL mod 3
      xor   bx,bx
      test  byte ptr dot_c,1
      jz    dith_m1
      inc   bl
dith_m1:    test  byte ptr line_c,1
      jz    dith_m2
      inc   bl                      ;   11     ;   10     ;   01    ;   00    ;
      inc   bl                      ;  X   X   ; X-1  X   ; X-1  X  ;  X  X-1 ;
dith_m2:    shl   al,1              ;  X   X   ;  X   X   ;  X  X-1 ; X-1 X-1 ;
      shl   al,1
      add   bl,al
      ret
      ;
;
;Entry label;     [sel_pat12], [sel_pat121], [sel_pat13]
;  Input;   BL=Brightness (sel_pat13 only)
;  Output;  AX=aligned FRC pattern
;     CL=destination pixel address
;
      ;
sel_pat12:  mov   si,offset b_buf
sel_pat121: mov   bl,dot_c
      and   bx,7
      mov   bl,[si]+[bx]            ;Select brightness
sel_pat13:  mov   dl,dot_c
      cmp   byte ptr col_mono,0
      jz    dot_cal6                ;mono LCD
      cmp   byte ptr rgb_seq,0
      jz    dot_cal6                ;don't care for color position
      mov   dl,dotc_c
dot_cal6:   mov   dot_buf,dl
      xor   dl,dl                   ;Clear dot position
      test  byte ptr line_c,1
      jz    dot_cal1
      mov   dl,2
dot_cal1:   test  byte ptr dot_buf,1
      jz    dot_cal2
```

```
      inc   dl                      ;Int. pattern offset address
dot_cal2:   cmp   byte ptr scheme,1
      jz    dot_cal5                ;FRC type-1
      cmp   byte ptr scheme,4
      jnz   dot_cal3                ;except FRC type-4
dot_cal5:   test  byte ptr dot_buf,2      ;** type-1,4 **
      jz    dot_cal3                ;** type-1,4 **
      test  byte ptr line_c,1       ;** type-1,4 **
      jz    dot_cal4                ;** type-1,4 **
      dec   dl                      ;** type-1,4 **
      dec   dl                      ;** type-1,4 **
      jmp   short dot_cal3          ;** type-1,4 **
dot_cal4:   inc   dl                ;** type-1,4 **
      inc   dl                      ;** type-1,4 **
dot_cal3:   shl   dl,1
      shl   bl,1
      shl   bl,1
      shl   bl,1
      add   bl,dl
      xor   bh,bh                   ;BX=FRC pattern select offset address
      mov   si,offset FRC_pat34     ;Set FRC pattern pointer
      cmp   byte ptr sch_64,12h
      jz    d_cal1                  ;3 frames 4FRC col
      mov   si,offset FRC_pat23     ;Set FRC pattern pointer
      cmp   byte ptr sch_64,17h
      jz    d_cal1                  ;2 frames 3FRC col
      mov   si,offset FRC_pat43     ;Set FRC pattern pointer
      cmp   byte ptr sch_64,18h
      jz    d_cal1                  ;4 frames 3FRC col
      mov   si,offset FRC_pat45     ;Set FRC pattern pointer
      cmp   byte ptr sch_64,14h
      jz    d_cal1                  ;4 frames 5FRC col
      cmp   byte ptr sch_64,15h
      jz    d_cal1                  ;4 frames 5FRC col
      cmp   byte ptr sch_64,16h
      jz    d_cal1                  ;4 frames 5FRC col with dither
      mov   si,offset FRC_pat16     ;Set FRC pattern pointer
d_cal1:     mov   ax,[si]+[bx]      ;Take FRC pattern
      xor   cl,cl
      cmp   byte ptr sch_64,15h
      jz    d_cal22                 ;4 frames 5FRC col without offset
      cmp   byte ptr sch_64,16h
      jz    d_cal22                 ;4 frames 5FRC col without offset with dither
      cmp   byte ptr sch_64,17h
      jz    d_cal22                 ;2 frames 5FRC col without offset with dither
      cmp   byte ptr sch_64,18h
      jz    d_cal22                 ;4 frames 5FRC col without offset with dither
      cmp   byte ptr sch_64,12h
      jz    d_cal2                  ;3 frmaes 4FRC col
      mov   cl,hoff1
      add   cl,voff1                ;OFF%=HOFF1%+VOFF1%
d_cal2:     cmp   byte ptr scheme,2
      jnz   pat_al                  ;except FRC type-2
      and   cl,3                    ;** type-2 **
      jmp   short pat_al1
pat_al:     cmp   byte ptr scheme,0
      jnz   pat_al6
      jmp   short pat_al5           ;FRC type-0
pat_al6:    cmp   byte ptr scheme,3
      jnz   pat_al2                 ;except FRC type-3
pat_al5:    test  byte ptr line_c,2 ;** type-0,3 **
      jz    pat_al3                 ;** type-0,3 **
      inc   cl                      ;** type-0,3 **
      test  byte ptr dot_buf,2      ;** type-0,3 **
      jnz   pat_al2                 ;** type-0,3 **
      jmp   short pat_al4           ;** type-0,3 **
pat_al3:    test  byte ptr dot_buf,2      ;** type-0,3 **
```

```
      jz    pat_al2                 ;** type-0,3 **
pat_al4:    inc   cl                ;** type-0,3 **
      inc   cl                      ;** type-0,3 **
pat_al2:    and   cl,0fh
pat_al1:    cmp   byte ptr sch_64,12h
      jz    pat_al7
d_cal22:    rol   ax,cl             ;Allign FRC pattern from frame0
pat_al8:    mov   si,offset p_mask
      mov   bl,dot_c
      and   bx,7
      mov   cl,[si]+[bx]            ;CL=destination pixel address
      ret                           ;AX=FRC pattern
pat_al7:    xor cl,cl
pat_al10:   jz    pat_al8
      xor   bl,bl
      test  al,4
      jz    pat_al9
      inc   bl
pat_al9:    shl   al,1
      or    al,bl
      dec   cl
      jmp   short pat_al10
      ;
;
;Entry label;     [c_b_conv12]
;  Input;   SI=source byte address
;  Output;  [bright]-[bright+8]=brightness for 8 pixels
;
      ;
c_b_conv12: cmp   byte ptr dd_emu,9
      jnz   c_bc2                   ;except D/D emu+lower panel
      add   si,1f40h
c_bc2:      mov   di,offset b_buf
      call  col_br
c_bc1:      call  sto_cb
      call  col_br_1
      jnc   c_bc1
sto_cb:     push  di
      mov   di,offset ipal
      mov   bp,cx
      mov   cl,ds:[di]+[bp]         ;CL=video output
      mov   di,offset epal_b
      mov   bp,cx
      call  c_b_chk
      pop   di
      mov   [di],cl                 ;blue or brightness
      push  di
      mov   di,offset epal_g
      call  c_b_chk
      pop   di
      mov   [di]+8,cl               ;green
      push  di
      mov   di,offset epal_r
      call  c_b_chk
      pop   di
      mov   [di]+16,cl              ;red
      inc   di
      ret
      ;
c_b_chk:    mov   cl,ds:[di]+[bp]   ;CL=6 bit video
      shr   cl,1
      shr   cl,1                    ;CL=4 bit
      cmp   byte ptr sch_64,12h
      jz    c_b_c2                  ;3 frames 4FRC
      cmp   byte ptr sch_64,14h
      jz    c_b_c2                  ;4 frames 5FRC with offset
      cmp   byte ptr sch_64,15h
```

```
      jz    c_b_c2                  ;4 frames 5FRC without offset
      cmp   byte ptr sch_64,16h
      jz    c_b_c2                  ;4 frames 5FRC without offset with dither
      cmp   byte ptr sch_64,17h
      jz    c_b_c2                  ;2 frames 3FRC without offset with dither
      cmp   byte ptr sch_64,18h
      jz    c_b_c2                  ;4 frames 3FRC without offset with dither
      ret
c_b_c2:     shr   cl,1
      shr   cl,1                    ;CL=2 bit
      ret
      ;
col_br:     push  ds
      mov   ax,0a000h
      mov   ds,ax
      mov   ax,4
      call  gr_reg_w                ;GR04=0
      mov   dl,[si]                 ;Blue      --> DL
      mov   ax,104h
      call  gr_reg_w                ;GR04=1
      mov   dh,[si]                 ;Green     --> DH
      mov   ax,204h
      call  gr_reg_w                ;GR04=2
      mov   bl,[si]                 ;Red       --> BL
      mov   ax,304h
      call  gr_reg_w                ;GR04=3
      mov   bh,[si]                 ;Intensity --> BH
      pop   ds
      mov   ah,80h                  ;AH=pixel pointer
col_br_1:   xor   cx,cx
      test  dl,ah
      jz    br_1
      or    cl,1
br_1: test  dh,ah
      jz    br_2
      or    cl,2
br_2: test  bl,ah
      jz    br_3
      or    cl,4
br_3: test  bh,ah
      jz    br_4
      or    cl,8
br_4: shr   ah,1                    ;CX=brightness
      ret
      ;
;
;Entry label; [sel_sto64]
;  Input;   AX=destination segment
;     DH=subtraction data
;     DL=brightness
;
      ;
sel_sto64:  mov   bl,dl
      shr   dh,1
      push  dx                      ;push BL;brightness, BH;subtraction
      push  ax                      ;push destination segment
      jnc   sv13_7
      dec   bl
sv13_7:     call  sel_pat13
      pop   bx                      ;pop segment
      xchg  si,di
      call  px_sto64
      xchg  si,di
      pop   dx
      add   bx,60h                  ;goto next segment
      mov   ax,bx
      ret
```

```
      ;
;
;Entry label; [rgb_sto], [rgb_sto4]
;  Input;   AX=destination segment
;     DL=brightness
;
      ;
rgb_sto4:   push  ax
      call  dith_make4
      mov   si,offset dither_m
      sub   dl,[si]+[bx]           ;DL=final 2 bit brightness
      mov   bl,dl                  ;BL
      jmp   short rgb_s5
      ;
rgb_sto:    cmp   byte ptr sch_64,11h
      jz    rgb_s1                 ;64FRC col
      shr   dl,1
      shr   dl,1                   ;ignore lower 2 bits
      cmp   byte ptr sch_64,10h
      jz    rgb_s1                 ;16FRC col
      shr   dl,1
      mov   bl,dl
      shr   dl,1                   ;ignore lower 2 bits
      cmp   byte ptr sch_64,12h
      jz    rgb_s1                 ;3 frames 4 FRC
      inc   dl
      or    bl,bl
      jnz   rgb_s1
      xor   dl,dl
rgb_s1:     mov   bl,dl
      push  ax                     ;push destination segment
rgb_s5:     call  sel_pat13
      pop   bx                     ;pop segment
      xchg  si,di
      call  px_sto64
      xchg  si,di
      add   bh,10h
      and   bx,0f000h
      mov   ax,bx
      cmp   ah,60h
      jz    rgb_s3                 ;"blue" plane
      cmp   ah,70h
      jz    rgb_s4                 ;"green" plane
      add   byte ptr dotc_c,4      ;* color position adjust *
      ret
rgb_s3:     sub   byte ptr dotc_c,2 ;* color position adjust *
      ret
rgb_s4:     inc   byte ptr dotc_c   ;* color position adjust *
      ret
      ;
;
;Entry label;     [px_sto16]
;  Input;   AX=aligned FRC pattern
;     SI=destination byte  address
;     CL=destination pixel address
;  Output;  16 frames of 640X200 monochrome screen
;
      ;
px_sto16:   push  si
      cmp   byte ptr dd_emu,9
      jnz   sto16_4                ;except D/D emu+lower panel
      add   si,1f40h               ;goto lower panel
sto16_4:    push  ds
      mov   bx,5000h
      mov   ch,4                   ;loop count
sto16_3:    mov   dh,4             ;loop count
sto16_2:    sar   ax,1
```

```
      jnc   sto16_1
      mov   ds,bx
      or    [si],cl
sto16_1:    add   bx,3e8h
      dec   dh
      jnz   sto16_2
      add   bx,60h
      dec   ch
      jnz   sto16_3
      pop   ds
      pop   si
      ret
      ;
;
;Entry label;     [px_sto16c]
;  Input;   AX=aligned FRC pattern
;     SI=destination byte  address
;     CL=destination pixel address
;  Output;  3 frames of 640X200 monochrome screen
;
      ;
px_sto16c:  mov   dh,4              ;loop count
sto16c_2:   sar   ax,1
      jnc   sto16c_1
      mov   es,bx
      or    es:[si],cl
sto16c_1:   add   bx,3e8h
      dec   dh
      jnz   sto16c_2
      ret
      ;
;
;Entry label;     [px_sto64]
;  Input;   AX=aligned FRC pattern
;     BX=destination segment
;     SI=destination byte address
;     CL=destination pixel address
;  Output;  16 frames of 320X100 monochrome screen for 16FRC
;     3 frames of 640X200 monochrome screen for 4FRC
      ;
px_sto64:   mov   dh,sloop_cnt      ;DH=loop count
sto64_2:    sar   ax,1
      jnc   sto64_1
      mov   es,bx
      or    es:[si],cl
sto64_1:    add   bx,fscr_seg
      dec   dh
      jnz   sto64_2
      ret
      ;
;*************************************************
;* Subroutines  {load FRC result to display memory} *
;*************************************************
      ;
load_mono:  cld
      mov   ax,0a000h
      mov   es,ax                   ;display window
      mov   bx,5000h
      mov   ax,0102h                ;SR02=1
load_m1:    mov   ds,bx
      call  sr_reg_w                ;SR02=(?)
      mov   cx,8000h
      xor   si,si
      xor   di,di
      repz  movsw                   ;64KB data load
      add   bx,1000h
      shl   ah,1
```

```
      cmp   ah,10h
      jnz   load_m1
      dec   ah
      call  sr_reg_w              ;SR02=0fh
      mov   ax,cs
      mov   ds,ax
      ret
      ;
;***********************************************
;* Subroutines  {FRC result display sequencing} *
;***********************************************
      ;
FRC_exec:   call  wr_ipal         ;change internal palette
      cmp   byte ptr sch_64,13h
      jnz   fexe_7                ;except 0FRC
fexe_8:     call  chk_keyi
      jz    fexe_8
      jmp   fexe_4
      ;
fexe_7:     call  set_dparam
      cmp   byte ptr col_mono,0
      jz    fexe_3                ;mono LCD
      mov   ax,308h
      call  xr_reg_w              ;XR08=3
      mov   ax,2d2dh              ;ACDCLK 00101101b
      mov   bx,808h               ;ACDCLK period
      cmp   byte ptr sch_64,14h
      jz    fexe_a
      cmp   byte ptr sch_64,15h
      jz    fexe_a
      mov   ax,06666h             ;ACDCLK waveform
      mov   bx,0c0ch              ;ACDCLK period
      cmp   byte ptr sch_64,16h
      jz    fexe_a
      mov   ax,06666h             ;ACDCLK waveform
      mov   bx,808h               ;ACDCLK period
      cmp   byte ptr sch_64,12h
      jz    fexe_a
      cmp   byte ptr sch_64,17h
      jz    fexe_a
      cmp   byte ptr sch_64,18h
      jz    fexe_a
      mov   ax,5555h              ;ACDCLK waveform
      mov   bx,808h               ;ACDCLK period
fexe_a:     mov   acd_wwork,ax
      mov   acd_worg,ax
      mov   word ptr acd_cwork,bx
      call  vrtc_chk
fexe_5:     cli
      call  fr_scan
      sti
      call  chk_keyi
      jz    fexe_5                ;no key interrupt
      jmp   short fexe_4
      ;
fexe_3:     call  vrtc_chk
FRC_disp:   cli
      mov   ax,132h               ;AR12=1
      call  fr_scan_m
      mov   ax,232h               ;AR12=2
      call  fr_scan_m
      mov   ax,432h               ;AR12=4
      call  fr_scan_m
      mov   ax,832h               ;AR12=8
      call  fr_scan_m
      sti
      call  chk_keyi
```

```
      jz    FRC_disp                ;no key interrupt
fexe_4:     cmp   al,1bh
      jnz   FRC_disp_e              ;check "ESC"
      mov   byte ptr scheme,0ffh
FRC_disp_e: mov   ax,208h
      call  xr_reg_w                ;XR08=2
      call  rest_dparam
      mov   si,offset ar_org
      call  wr_ipal_e
      ret
      ;
wr_ipal:    mov   si,offset ar_cont_m
      cmp   byte ptr col_mono,0
      jz    wr_ipal_e               ;mono LCD
      mov   si,offset ar_cont_c
wr_ipal_e:  xor   bx,bx
rec_ar:     mov   ah,[si]+[bx]
      mov   al,bl
      call  ar_reg_w                ;write AR
      inc   bl
      cmp   bl,10h
      jnz   rec_ar
ar_turn_on: mov   dx,3dah
      in    al,dx
      mov   dx,3c0h
      mov   al,20h
      out   dx,al
      ret
      ;
set_dparam: mov   ax,8f12h
      call  cr_reg_w                ;CR12=8Fh
      mov   ax,1302h
      call  xr_reg_w                ;XR02=13h
      mov   al,50h
      call  xr_reg_r
      and   al,0fch
      mov   ah,al
      mov   al,50h
      call  xr_reg_w                ;XR50 D1 and D0 = 0
      mov   al,51h
      call  xr_reg_r
      test  al,2
      jnz   set_dp3                 ;Dual panel
      mov   ax,675bh                ;Single panel
      call  xr_reg_w                ;XR5B=68h
      mov   ax,406bh
      call  xr_reg_w                ;XR6B=40h
      mov   ax,9f5ah
      jmp   short set_dp4
set_dp3:    mov   ax,0ef5bh
      call  xr_reg_w                ;XR5B=F0h
      mov   ax,6bh
      call  xr_reg_w                ;XR6B=0
      mov   ax,275ah
set_dp4:    call  xr_reg_w          ;XR5A=27h, 9Fh
      cmp   byte ptr esize,0
      jz    set_dp1                 ;640X200
      mov   ax,1413h
      call  cr_reg_w                ;CR13=14h
      mov   al,51h
      call  xr_reg_r
      test  al,2
      jnz   set_dp5                 ;Dual panel
      mov   ax,35bh                 ;Single panel
      jmp   short set_dp6
set_dp5:    mov   ax,8b5ah
set_dp6:    call  xr_reg_w          ;XR5A=8bh, XR5B=4
```

```
      mov   ax,1ch
      call  xr_reg_w                ;XR1C=00h
      mov   al,51h
      call  xr_reg_r
      mov   ah,52h
      test  al,1
      jnz   set_dp2                 ;LCD D/D panel
      mov   ah,2bh
set_dp2:    mov   al,1dh
      call  xr_reg_w                ;XR1D=52h (D/D,S/S), 2bh (D/S)
set_dp1:    ret
      ;
rest_dparam:      mov   ax,000dh
      call  cr_reg_w                ;CR0D=00h
      mov   ax,000ch
      call  cr_reg_w                ;CR0C=00h
      mov   ax,0f12h
      call  ar_reg_w                ;AR12=0Fh
      mov   ax,0302h
      call  xr_reg_w                ;XR02=03h
      call  rest_xr                 ;XR18-6E=(?)
      mov   ax,0df12h
      call  cr_reg_w                ;CR12=DFh
      mov   ax,2813h
      call  cr_reg_w                ;CR13=28h
      ret
      ;
fr_scan_m:  mov   dx,3c0h
      out   dx,ax                   ;AR12=(?)
fr_scan:    mov   cl,sloop_cnt      ;CL=loop count
      mov   di,lscr_off             ;DI=last frame segment
      mov   si,scr_off              ;SI=1st frame offset
      mov   bx,si
fr_s4:      mov   dx,3d4h
      mov   ah,bl
      mov   al,0dh
      out   dx,ax                   ;CR0D=(?)
      mov   ah,bh
      dec   al
      out   dx,ax                   ;CR0C=(?)
fr_s2:      add   bx,si
      cmp   bx,di
      jnz   fr_s3
      xor   bx,bx
fr_s3:      call  vrtc_chk
      dec   cl
      jnz   fr_s4
      ret
      ;
vrtc_chk:   mov   dx,3dah
vr_1: in    al,dx
      test  al,8
      jnz   vr_1
      mov   dx,3d6h
      mov   ax,109h
      shr   word ptr acd_wwork,1
      jc    vr_3
      dec   ah
vr_3: out   dx,ax                   ;XR09=(?)
      dec   byte ptr acd_cwork
      jnz   vr_4
      mov   ax,acd_worg
      mov   acd_wwork,ax
      mov   al,acd_corg
      mov   acd_cwork,al
vr_4: mov   dx,3dah
vr_2: in    al,dx
```

```
      test  al,8
      jz    vr_2
      ret
      ;
;*****************************
;* Subroutines  {dump to file} *
;*****************************
      ;
FRC_dump:   mov   dx,offset mes_18  ;"dump"
      call  mes_pr
      call  key_in
      cmp   al,"0"
      jz    FRC_dall
      cmp   al,"1"
      jz    FRC_dasc
      ret
FRC_dasc:   push  ds
FRC_da1:    mov   dx,offset mes_20  ;"lines to be taken"
      call  pr_key_2
      cmp   cl,40h
      jnc   FRC_da1
      mov   dump_mline,cl
      mov   ah,3ch
      mov   dx,offset path_dasc
      mov   cx,0
      int   21h                   ;create
      jc    wr_13
      mov   handle,ax             ;file handle
      mov   byte ptr dump_fc,0
      mov   ax,5000h
      call  wr_exe
      mov   ax,6000h
      call  wr_exe
      mov   ax,7000h
      call  wr_exe
      mov   ax,8000h
      call  wr_exe
      jmp   short wr_13
FRC_dall:   push  ds
      mov   ah,3ch
      mov   dx,offset path_dall
      mov   cx,0
      int   21h                   ;create
      jc    wr_13
      mov   handle,ax             ;file handle
      mov   bx,ax
      mov   cx,8000h
      mov   ax,5000h
      call  dump_64k
      mov   ax,6000h
      call  dump_64k
      mov   ax,7000h
      call  dump_64k
      mov   ax,8000h
      call  dump_64k
wr_13:      pop   ds
      mov   bx,handle
      mov   ah,3eh
      int   21h                   ;file close
FRC_de:     ret
      ;
wr_exe:     mov   es,ax             ;clear segment
      push  ax
      xor   si,si                 ;clear byte address
wr_e3:      push  si
      mov   byte ptr dump_lc,0
      mov   al,dump_fc
```

```
      mov   di,offset dump_ty1+8
      call  b_h_tran2
      mov   al,hoff
      mov   di,offset dump_ty1+22
      call  b_h_tran2
      mov   al,voff
      mov   di,offset dump_ty1+36
      call  b_h_tran2
      mov   dx,offset dump_ty1
      mov   bx,handle
      mov   cx,40
      mov   ah,40h
      int   21h                     ;write tytle-1
      mov   dx,offset dump_ty2
      mov   cx,71
      mov   ah,40h
      int   21h                     ;write tytle-2
      mov   dx,offset dump_ty3
      mov   cx,73
      mov   ah,40h
      int   21h                     ;write tytle-3
wr_e4:      call  wr_1line
      inc   byte ptr dump_lc
      mov   ax,50h
      mov   cl,dump_mline
      xor   ch,ch
      xor   dx,dx
wr_e41:     add   dx,ax
      loop  wr_e41
      mov   cx,dx                   ;max. lines
      mov   dx,3e8h
      cmp   byte ptr esize,0
      jz    wr_e1                   ;640X200
      shr   ax,1
      shr   cx,1
      shr   dx,1
      shr   dx,1
wr_e1:      pop   si
      add   si,ax
      cmp   si,cx
      jnz   wr_e2
      inc   byte ptr dump_fc
      pop   ax
      add   ax,dx
      mov   dx,ax
      and   dx,0fffh
      cmp   dx,0fa0h
      jz    wr_ee
      jmp   wr_exe
wr_ee:      ret
wr_e2:      push  si
      jmp   short wr_e4
      ;
wr_1line:   mov   al,dump_lc
      mov   di,offset dump_buf
      call  b_h_tran2               ;convert line count
      mov   di,offset dump_buf+5
      mov   cl,8
wr_l2:      mov   dl,es:[si]
      mov   ch,8
wr_l4:      mov   al,"0"
      shl   dl,1
      jnc   wr_l1
      mov   al,"1"
wr_l1:      mov   [di],al
      inc   di
      dec   ch
```

```
      jnz   wr_l4
      inc   si
      dec   cl
      jnz   wr_l2
      mov   bx,handle
      mov   dx,offset dump_buf
      mov   cx,71
      mov   ah,40h
      int   21h                     ;write
      ret
      ;
dump_64k:   mov   ds,ax
      xor   dx,dx
      mov   ah,40h
      int   21h                     ;write
      mov   dx,8000h
      mov   ah,40h
      int   21h                     ;write
      ret
      ;
;*******************************
;* Subroutines  {buffer clear} *
;*******************************
      ;
buf_cl:     cld
      xor   ax,ax                   ;clear data=0
      mov   bx,5000h                ;plane buffer #0-3
      call  buf_cl1
      mov   bx,6000h                ;plane buffer #4-7
      call  buf_cl1
      mov   bx,7000h                ;plane buffer #8-B
      call  buf_cl1
      mov   bx,8000h                ;plane buffer #C-F
buf_cl1:    mov   es,bx
      mov   cx,8000h
      xor   di,di
      repz  stosw                   ;64KB data clear
      ret
      ;
;*****************************************
;* Subroutines  {miscellaneous translation} *
;*****************************************
      ;
b_d_tran2:  push  si
      mov   ah,0ffh
b_d_t1:     inc   ah
      sub   al,10
      jnc   b_d_t1
      add   al,10                   ;AH=upper decimal N, AL=lower decimal N
      mov   si,offset b_d_num
      mov   bl,ah
      xor   bh,bh
      mov   bl,[si]+[bx]
      cmp   bl,'0'
      jnz   b_d_t2
      mov   bl,' '
b_d_t2:     mov   [di],bl
      inc   di
      mov   bl,al
      xor   bh,bh
      mov   bl,[si]+[bx]
      mov   [di],bl
      pop   si
      ret
      ;
;
;Entry label; [b_h_tran2]
```

```
;  Input;   AL=binary data
;     DI=destination address
;
b_h_tran2:  push  si
      mov   ah,al
      shr   ah,1
      shr   ah,1
      shr   ah,1
      shr   ah,1
      and   ah,0fh
      and   al,0fh                ;AH=upper hex, AL=lower hex
      mov   si,offset b_h_num
      mov   bl,ah
      xor   bh,bh
      mov   bl,[si]+[bx]
      cmp   bl,'0'
      jnz   b_h_t2
      mov   bl,' '
b_h_t2:     mov   [di],bl
      inc   di
      mov   bl,al
      xor   bh,bh
      mov   bl,[si]+[bx]
      mov   [di],bl
      pop   si
      ret
      ;
;*******************************
;* Subroutines  {miscellaneous} *
;*  [setup], [mes_pr], [key_in] *
;*******************************
      ;
setup:      call  ld_setup          ;set VGA mode
      mov   dx,offset mes_00
      call  mes_pr
      mov   dx,offset mes_27
      mov   bh,3
      call  pr_key                ;"clock select"
      mov   bx,54h
setup_2:    cmp   ah,0
      jz    setup_1
      add   bh,4
      dec   ah
      jmp   short setup_2
setup_1:    mov   ax,bx
      call  xr_reg_w              ;XR54=4  28MHz (LCD-DS works only below 30MHz)
      ret
      ;
mes_pr:     mov   ah,9
      int   21h
      ret
      ;
chk_keyi:   mov   ah,6
      mov   dl,0ffh
      int   21h
      ret
      ;
key_in:     mov   ah,1
      int   21h
      cmp   al,1bh
      jz    end_p
      push  ax
      mov   dx,offset mes_cr
      call  mes_pr
      pop   ax
      ret
end_p:      mov   ax,4c00h
```

```
      int   21h
      ;
save_xr:    mov   ax,ds             ;save XR registers
      mov   es,ax
      mov   di,offset xr_buf
      mov   bl,18h
save_xr1:   mov   al,bl
      call  xr_reg_r
      stosb
      inc   bl
      cmp   bl,1fh
      jnz   save_xr1
      mov   bl,50h
save_xr2:   mov   al,bl
      call  xr_reg_r
      stosb
      inc   bl
      cmp   bl,6fh
      jnz   save_xr2
      ret
      ;
rest_xr:    call  ld_setup          ;set VGA mode
      mov   si,offset xr_buf
      mov   al,18h
rest_xr1:   mov   ah,[si]
      call  xr_reg_w
      inc   si
      inc   al
      cmp   al,1fh
      jnz   rest_xr1
      mov   al,50h
rest_xr2:   mov   ah,[si]
      call  xr_reg_w
      inc   si
      inc   al
      cmp   al,6fh
      jnz   rest_xr2
      ret
      ;
xr_reg_r:   mov   dx,3d6h
reg_r:      out   dx,al
      inc   dx
      in    al,dx
      ret
      ;
xr_reg_w:   push  dx
      mov   dx,3d6h               ;Set XR Index
      jmp   short reg_w
gr_reg_w:   push  dx
      mov   dx,3ceh               ;Set GR Index
      jmp   short reg_w
sr_reg_w:   push  dx
      mov   dx,3c4h               ;Set SR Index
      jmp   short reg_w
cr_reg_w:   push  dx
      mov   dx,3d4h               ;Set CR Index
reg_w:      out   dx,ax
      pop   dx
      ret
      ;
ar_reg_w:   push  ax
      mov   dx,3dah
      in    al,dx
      pop   ax
      mov   dx,3c0h
      out   dx,al
      mov   al,ah
```

```
      out   dx,al
      ret
      ;
pr_key_1:   mov   bh,10h
pr_key:     call  mes_pr
      push  dx
      call  key_in
      pop   dx
      mov   ah,al
      call  num_tran
      cmp   al,0ffh
      jz    pr_key
      cmp   ah,bh
      jnc   pr_key
      ret
      ;
;
;Entry label; [pr_key_2], [pr_key_4]
;  Input;   DX=message offset address
;  Output;  CL=2 key code, CX=4 key code
;
      ;
pr_key_4:   mov   bh,5              ;key count
      jmp   short pr_k_24
pr_key_2:   mov   bh,3              ;key count
pr_k_24:    mov   si,offset key_buf
      mov   [si],bh
pr_k_241:   call  mes_pr
      push  dx
      mov   dx,si
      mov   ah,0ah
      int   21h                     ;key buffer in
      pop   dx
      mov   dx,offset mes_cr
      call  mes_pr
      mov   bl,[si]+1
      cmp   bl,0
      jz    pr_k_241
      call  key_enc
      cmp   al,0ffh
      jz    pr_k_241
      mov   ah,5
      cmp   [si],ah
      jz    pr_k_242
      ret
pr_k_242:   or    bl,bl
      jnz   pr_k_243
      ret
pr_k_243:   push  cx
      call  key_enc
      mov   ah,cl
      pop   cx
      cmp   al,0ffh
      jz    pr_k_241
      mov   ch,cl
      mov   cl,ah
      ret
      ;
key_enc:    xor   bh,bh
      dec   bl
      mov   ah,[si]+[bx]+2
      call  num_tran
      cmp   al,0ffh
      jz    key_enc_e
      xor   ch,ch
      mov   cl,ah                   ;store 2nd key
      or    bl,bl
```

```
      jz    pr_k_21                 ;1 key depression
      dec   bl
      mov   ah,[si]+[bx]+2
      call  num_tran
      cmp   al,0ffh
      jz    key_enc_e
      mov   ch,ah                   ;store 1st key
pr_k_21:    shl   ch,1
      shl   ch,1
      shl   ch,1
      shl   ch,1
      or    cl,ch
      xor   ch,ch
      xor   al,al
key_enc_e:  ret
      ;
num_tran:   cmp   ah,"0"
      jc    num_er
      cmp   ah,":"
      jc    num_0
      cmp   ah,"A"
      jc    num_er
      cmp   ah,"G"
      jc    num_l
      cmp   ah,"a"
      jc    num_er
      cmp   ah,"g"
      jc    num_s
num_er:     mov   al,0ffh
      ret
      ;
num_s:      sub   ah,20h
num_l:      sub   ah,7h
num_0:      sub   ah,30h
      ret
      ;
;******************************
;* Data tables  {FRC pattern} *
;******************************
      ;
FRC_pat23   dw     0000h,  0000h,  0000h,  0000h      ;BR=0
      dw     5555h, 0aaaah, 0aaaah,  5555h           ;BR=1
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=2
FRC_pat43   dw     0000h,  0000h,  0000h,  0000h      ;BR=0
      dw     5555h, 0aaaah, 0aaaah,  5555h           ;BR=1
;     dw     6666h,  9999h,  9999h,  6666h           ;BR=1
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=2
FRC_pat45   dw     0000h,  0000h,  0000h,  0000h      ;BR=0
;     dw     4924h,  2492h,  2492h,  4924h           ;BR=2
      dw     5555h, 0aaaah, 0aaaah,  5555h           ;BR=1
;     dw    0bb6dh, 076dbh, 0cdb6h, 0bb6dh           ;BR=2
      dw    0f7dfh, 07df7h, 0befbh, 0efbeh           ;BR=2
;     dw    0d75dh, 0aebah, 0aebah, 0d75dh           ;BR=2
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=4
FRC_pat34   dw     0000h,  0000h,  0000h,  0000h      ;BR=0
      dw     0004h,  0002h,  0002h,  0001h           ;BR=1,,(0,0),(1,0),(0,1),(1,1)
      dw     0003h,  0005h,  0005h,  0006h           ;BR=2
      dw     0007h,  0007h,  0007h,  0007h           ;BR=3
FRC_pat16   dw     0000h,  0000h,  0000h,  0000h      ;BR=0; MSB=#15, LSB=#0 frame
      dw     0101h,  1010h,  0404h,  4040h           ;BR=1,,(0,0),(1,0),(0,1),(1,1)
      dw     0421h,  2104h,  4210h,  1042h           ;BR=2
      dw     1111h,  4444h,  2222h,  8888h           ;BR=3
      dw     2491h,  4912h,  9124h,  1249h           ;BR=4
      dw     2525h,  5252h,  9494h,  4949h           ;BR=5
      dw     2a55h, 0a552h,  52a5h,  552ah           ;BR=6
      dw     3333h, 0cccch,  3333h, 0cccch           ;BR=7
```

```
      dw    0d5aah,  5aadh, 0ad5ah, 0aad5h          ;BR=8
      dw    0dadah, 0adadh,  6b6bh, 0b6b6h          ;BR=9
      dw    0db6eh, 0b6edh,  6edbh, 0edb6h          ;BR=A
      dw    0eeeeh, 0bbbbh, 0ddddh,  7777h          ;BR=B
      dw    0fbdeh, 0defbh, 0bdefh, 0efbdh          ;BR=C
      dw    0fefeh, 0efefh, 0fbfbh, 0bfbfh          ;BR=D
      dw    0fffeh, 0ffefh, 0feffh, 0efffh          ;BR=E
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh          ;BR=F
      ;
FRC_PAT64   dw     0000h,  0000h,  0000h,  0000h     ;BR=00 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=01 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=02 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=03 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=04 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=05 (0,0)
      dw     0000h,  0000h,  0000h,  0000h          ;BR=06 (0,0)
      dw     0101h,  0101h,  0101h,  0101h          ;BR=07*(0,0)
      dw     4081h,  1020h,  0208h,  0102h          ;BR=08 (0,0)
      dw     2041h,  0208h,  2041h,  0208h          ;BR=09 (0,0)
      dw     1041h,  2104h,  8208h,  0820h          ;BR=0A (0,0)
      dw     0421h,  0421h,  0421h,  0421h          ;BR=0B*(0,0)
      dw     8421h,  4210h,  2108h,  1084h          ;BR=0C (0,0)
      dw     4221h,  1084h,  4221h,  1084h          ;BR=0D (0,0)
      dw     2111h,  4222h,  8444h,  0888h          ;BR=0E (0,0)
      dw     1111h,  1111h,  1111h,  1111h          ;BR=0F*(0,0)
      dw     9111h,  4888h,  2444h,  2222h          ;BR=10 (0,0)
      dw     4891h,  2224h,  4891h,  2224h          ;BR=11 (0,0)
      dw     2489h,  4892h,  9124h,  1244h          ;BR=12 (0,0)
      dw     1249h,  1249h,  1249h,  1249h          ;BR=13*(0,0)
      dw     9249h,  4924h,  2492h,  1249h          ;BR=14 (0,0)
      dw     9249h,  4924h,  9249h,  4924h          ;BR=15 (0,0)
      dw     4a49h,  9292h,  2494h,  4925h          ;BR=16 (0,0)
      dw     4949h,  4949h,  4949h,  4949h          ;BR=17*(0,0)
      dw    0a529h,  94a4h,  5252h,  494ah          ;BR=18 (0,0)
      dw     94a5h,  2a52h,  94a5h,  2a52h          ;BR=19 (0,0)
      dw     52a5h,  9529h, 0a94ah,  2a54h          ;BR=1A (0,0)
      dw     2a95h,  2a95h,  2a95h,  2a95h          ;BR=1B*(0,0)
      dw    0aa55h,  9552h,  54aah,  2aa5h          ;BR=1C (0,0)
      dw     9555h,  2aaah,  9555h,  4aaah          ;BR=1D (0,0)
      dw     5555h,  9555h, 0aaaah,  4aaah          ;BR=1E (0,0)
      dw     5555h,  5555h,  5555h,  5555h          ;BR=1F*(0,0)
      dw    0aaaah,  6aaah,  5555h, 0b555h          ;BR=20 (0,0)
      dw     6aaah, 0d555h,  6aaah, 0b555h          ;BR=21 (0,0)
      dw     55aah,  6aadh, 0ab55h, 0d55ah          ;BR=22 (0,0)
      dw    0d56ah, 0d56ah, 0d56ah, 0d56ah          ;BR=23*(0,0)
      dw    0ad5ah,  6ad6h,  56b5h, 0d5abh          ;BR=24 (0,0)
      dw     6b5ah, 0d5adh,  6b5ah, 0d5adh          ;BR=25 (0,0)
      dw     5ad6h,  6b5bh, 0adadh, 0b6b5h          ;BR=26 (0,0)
      dw    0b6b6h, 0b6b6h, 0b6b6h, 0b6b6h          ;BR=27*(0,0)
      dw    0b5b6h,  6d6dh, 0db6bh, 0b6dah          ;BR=28 (0,0)
      dw     6db6h, 0b6dbh,  6db6h, 0b6dbh          ;BR=29 (0,0)
      dw     6db6h, 0b6dbh, 0db6dh, 0edb6h          ;BR=2A (0,0)
      dw    0edb6h, 0edb6h, 0edb6h, 0edb6h          ;BR=2B*(0,0)
      dw    0db76h, 0b76dh,  6edbh, 0edbbh          ;BR=2C (0,0)
      dw    0b76eh, 0dddbh, 0b76eh, 0dddbh          ;BR=2D (0,0)
      dw     6eeeh, 0b777h, 0dbbbh, 0ddddh          ;BR=2E (0,0)
      dw    0eeeeh, 0eeeeh, 0eeeeh, 0eeeeh          ;BR=2F*(0,0)
      dw    0deeeh, 0bdddh,  7bbbh, 0f777h          ;BR=30 (0,0)
      dw    0bddeh, 0ef7bh, 0bddeh, 0ef7bh          ;BR=31 (0,0)
      dw     7bdeh, 0bdefh, 0def7h, 0ef7bh          ;BR=32 (0,0)
      dw    0fbdeh, 0fbdeh, 0fbdeh, 0fbdeh          ;BR=33*(0,0)
      dw    0efbeh, 0defbh,  7df7h, 0f7dfh          ;BR=34 (0,0)
      dw    0dfbeh, 0fdf7h, 0dfbeh, 0fdf7h          ;BR=35 (0,0)
      dw    0bf7eh, 0efdfh, 0fdf7h, 0fefdh          ;BR=36 (0,0)
      dw    0fefeh, 0fefeh, 0fefeh, 0fefeh          ;BR=37*(0,0)
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh          ;BR=38 (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh          ;BR=39 (0,0) temporary
```

```
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3A (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3B (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3C (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3D (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3E (0,0) temporary
      dw    0ffffh, 0ffffh, 0ffffh, 0ffffh           ;BR=3F (0,0)
      ;
;*********************************
;* Data tables  {miscellaneous} *
;*********************************
      ;
p_mask      db     80h, 40h, 20h, 10h, 08h, 04h, 02h, 01h  ;pixel mask data
ar_cont_m   db     00h, 3fh, 3fh, 00h, 3fh, 00h, 00h, 00h  ;AR00-07h mono
      db     3fh, 00h, 00h, 00h, 00h, 00h, 00h, 00h        ;AR08-0fh mono
ar_cont_c   db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h  ;AR00-07h color
      db     00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h        ;AR08-0fh color
ar_org      db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h  ;AR00-07h of origin
      db     38h, 39h, 3ah, 3bh, 3ch, 3dh, 3eh, 3fh        ;AR08-0fh of origin
t_pat       db    0aah, 55h,0aah,0aah,0ffh, 00h            ;tile pattern
sty_m12     db     00h, 06h, 0ch, 12h, 19h, 1fh            ;conv. start upper address
sty_m13     db     00h, 19h, 32h, 4bh, 64h, 7dh            ;conv. start upper address
stx_m13     db     00h, 28h, 50h, 78h,0a0h                 ;conv. start lower address
dither_m    db     00h, 01h, 01h, 01h, 01h, 00h, 00h, 01h  ;00(0,0),(1,0),(0,1),(1,1)
;     db     01h, 00h, 00h, 00h, 00h, 00h, 00h, 00h        ;10(0,0),(1,0),(0,1),(1,1)
;dither_mt  db     01h, 01h, 00h, 01h, 00h, 00h, 01h, 01h  ;00(0,0),(1,0),(0,1),(1,1)
      db     01h, 00h, 00h, 00h, 00h, 00h, 00h, 00h        ;10(0,0),(1,0),(0,1),(1,1)
sub_64      db     0eh, 07h, 0dh, 0bh, 05h, 0ah, 0ah, 05h  ;11(0,0),(1,0),(0,1),(1,1) 10(same)
      db     01h, 08h, 02h, 04h, 00h, 00h, 00h, 00h        ;01(0,0),(1,0),(0,1),(1,1) 00(same)
dith_pat    db     00h, 04h, 09h, 07h, 0fh                 ;0FRC=0,1,2,3,4
sch64_0_dat dw     1000h, 3e8h, 3e80h, 0fa00h   ;e_size,sloop_cnt,fscr_seg,scr_off,lscr_off
sch64_1_dat dw     4001h, 0fah, 0fa0h, 0fa00h
sch64_10_dat      dw     1001h, 0fah, 0fa0h, 0fa00h
sch64_11_dat      dw     4002h,  3eh, 0fa0h, 0fa00h
sch64_12_dat      dw      300h, 3e8h, 3e80h, 0bb80h
sch64_13_dat      dw        0h, 3e8h, 3e80h, 0fa00h
sch64_14_dat      dw      400h, 3e8h, 3e80h, 0fa00h
sch64_15_dat      dw      400h, 3e8h, 3e80h, 0fa00h
;sch64_16_dat     dw      400h, 3e8h, 3e80h, 0fa00h
sch64_16_dat      dw     0c01h, 0fah, 0fa0h, 0bb80h
sch64_17_dat      dw      200h, 3e8h, 3e80h, 0fa00h
;sch64_18_dat     dw      400h, 3e8h, 3e80h, 0fa00h
sch64_18_dat      dw     0c01h, 0fah, 0fa0h, 0bb80h
      ;
;*************************
;* Data tables  {message} *
;*************************
      ;
mes_cr  db  0dh,0ah,'$'
mes_00  db  0dh,0ah,'FRCEVA.COM - Let you evaluate FRC scheme on 82C455/456 DK board',0dh,0ah
        db  '(C) Chips and Technologies, Inc. 1990.  Version 1.6'
        db  0dh,0ah,0dh,0ah
        db  'Direction:  Type desired key following menu.',0dh,0ah
        db  '             -Restart; During FRC result observation, type any key',0dh,0ah
        db  '             -Exit;    Type "Esc" at any time',0dh,0ah,0dh,0ah,'$'
      db  'Designed by notorious T.Oguchi'
mes_01  db  '?? Test pattern to be drawn                                ??',0dh,0ah
        db  '   Entire screen filled by gray level 0-F                    ,,,   [0-F]',0dh,0ah
        db  '   Bar chart with 16 40X200 with different gray levels       ,,,   [H,h]',0dh,0ah
        db  '   Bar chart with 16 640X13 with different gray levels       ,,,   [V,v]   @ $'
mes_02  db  '   Horizontal offset                                         ,,,   [0-F]   @ $'
mes_03  db  '   Vertical   offset                                         ,,,   [0-F]   @ $'
mes_04  db  '?? Tile pattern to be drawn                                ??',0dh,0ah
        db  '   1st line = 10101010b     2nd line = 01010101b             ,,,    [0]',0dh,0ah
        db  '   1st line = 10101010b     2nd line = 10101010b             ,,,    [1]',0dh,0ah
        db  '   1st line = 11111111b     2nd line = 00000000b             ,,,    [2]',0dh,0ah
        db  '   1st line = 11111111b     2nd line = 11111111b             ,,,    [3]    @ $'
mes_05  db  '?? FRC scheme to be evaluated                              ??',0dh,0ah
```

```
        db  '   Old FRC scheme implemented on all 82C455 revisions    ,,,    [0]',0dh,0ah
        db  '   Aruns offset scheme implemented on 82C456 Rev.1,2     ,,,    [1]',0dh,0ah
        db  '   Mixed old and Aruns offset                            ,,,    [2]',0dh,0ah
        db  '   Improved mixed old and Aruns offset                   ,,,    [3]',0dh,0ah
        db  '   New FRC scheme for future use (temporary)             ,,,    [4]    @ $'
mes_06  db  '   Horizontal offset calculation boundary                ,,,   [0-F]   @ $'
mes_07  db  '   Vertical   offset calculation boundary                ,,,   [0-F]   @ $'
mes_08  db  '   Horizontal offset clear boundary    (press enter key) ,,, [0-FFFF]  @ $'
mes_09  db  '   Vertical   offset clear boundary    (press enter key) ,,, [0-FFFF]  @ $'
mes_10  db  '   Tile pattern background brightness                    ,,,   [0-F]   @ $'
mes_11  db  '?? VGA mode                                           ??',0dh,0ah
        db  '   VGA mode 12;  [0]    VGA mode 13;  [1]                                @ $'
mes_12  db  '?? Conversion origin;  co-ordinate Y                  ??',0dh,0ah
        db  '   0; [0]  20; [1]  40; [2]  60; [3]  80; [4]  100; [5]                  @ $'
mes_13  db  '?? VGA mode 13 source image on file                   ??',0dh,0ah
        db  '   CLOWN; [0]   ZOE  ; [1]   BARS ; [2]                                 @ $'
mes_14  db  '?? VGA mode 12 source image on file                   ??',0dh,0ah
        db  '   BARS ; [0]   BARS2; [1]   LINE ; [2]                                 @ $'
mes_15  db  '?? Source image to be processed                       ??',0dh,0ah
        db  '   Test pattern; [0]  Image on screen; [1]  Image on file; [2]           @ $'
mes_16  db  '   16 FRC + dither;  [0]   64 FRC current  method;  [1]',0dh,0ah
        db  '                           64 FRC improved method;  [2]                  @ $'
mes_17  db  '?? Conversion origin;  co-ordinate X                  ??',0dh,0ah
        db  '   0; [0]  40; [1]  80; [2]  120; [3]  160 [4]                           @ $'
mes_18  db  '?? FRC result dump format                             ??',0dh,0ah
        db  '   Binary (bit file compatible)  default : FRCdump.bit   ,,,    [0]',0dh,0ah
        db  '   ASCII  (binary notation)      default : FRCdump.dat   ,,,    [1]',0dh,0ah
        db  '   No    file dump                                      ,,, [any key]  @ $'
mes_19  db  '?? Test pattern to be drawn                           ??',0dh,0ah
        db  '   Screen filled by gray level 0-3F   (press enter key)  ,,,   [0-3F]',0dh,0ah
        db  '   Bar chart with various gray levels (press enter key)  ,,,    [FF]   @ $'
mes_20  db  '   Number of maximum lines per frame  (press enter key)  ,,,   [0-3F]  @ $'
mes_21  db  '   monochrome LCD panel                                  ,,,    [0]',0dh,0ah
        db  '   color      LCD panel                                  ,,,    [1]    @ $'
mes_22  db  '   16 frames 16 FRC;  [0]      64 frames 64 FRC;           [1]',0dh,0ah
        db  '    3 frames  4 FRC;  [2]       0 FRC with Dither;         [3]',0dh,0ah
        db  '    4 frames  5 FRC;  [4]       4 frames  5 FRC no offset; [5]',0dh,0ah
        db  '    4 frames  5 FRC no offset with dither;                 [6]',0dh,0ah
        db  '    2 frames  3 FRC no offset with dither;                 [7]',0dh,0ah
        db  '    4 frames  3 FRC no offset with dither;                 [8]         @ $'
mes_23  db  '?? Test pattern to be drawn                           ??',0dh,0ah
        db  '   Screen filled by color             (press enter key)  ,,,   [0-FE]',0dh,0ah
        db  '   Bar chart with various colors      (press enter key)  ,,,    [FF]   @ $'
mes_24  db  '?? 256 color block size                               ??',0dh,0ah
        db  '    16x16 ; [0]   32x32 ; [1]   48x48 ; [2]   64x64 ; [3]',0dh,0ah
        db  '    80x80 ; [4]   96x96 ; [5]  112x112; [6]  128x128; [7]',0dh,0ah
        db  '   144x144; [8]  160x160; [9]  176x176; [A]  192x192; [B]               @ $'
mes_25  db  '?? External palette contents to change color          ??',0dh,0ah
        db  '    Realistic;  [0]    Test;  [1]    Value assigned;  [2]               @ $'
mes_26  db  '?? Treatment for color dot position on panel          ??',0dh,0ah
        db  '    Dont care; [0]   <GRB>(HITACHI TFT); [1]   <RGB>(SANYO STN); [2]   @ $'
mes_27  db  '?? Dot clock selection                                ??',0dh,0ah
        db  '    25.175MHz; [0]   28.432MHz; [1]   panel clock; [2]                  @ $'
mes_28  db  '?? Dual Panel / Double Drive Emulation                ??',0dh,0ah
        db  '    Other Emulations; [0]     D/D Emulation; [1]                       @ $'
mes_29  db  '    Palette value for Green            (press enter key)  ,,,  [0-3F]   @ $'
     ;
;*********************
;* Working registers *
;*********************
     ;
hoff        db    (?)             ;Horizontal offset
hoff1       db    (?)             ;Intermediate horizontal offset
hoff1_cal   db    (?)             ;HOFF1% calculation timing
hoff1_cl    dw    (?)             ;HOFF1% clear       timing
voff        db    (?)             ;Vertical offset
voff1       db    (?)             ;Intermediate vertical offset
```

```
voff1_cal   db    (?)              ;VOFF1% calculation timing
voff1_cl    dw    (?)              ;VOFF1% clear       timing
dot_c       db    (?)              ;Dot counter
dot1_c      db    (?)              ;Dot counter to detect HOFF1% calculation timing
dot2_c      dw    (?)              ;Dot counter to detect HOFF1% clear timing
dotc_c      db    (?)              ;Dot counter for color LCD
dot_buf     db    (?)              ;Dot counter buffer for color LCD
line_c      db    (?)              ;Line counter
line1_c     db    (?)              ;Line counter to detect VOFF1% calculation timing
line2_c     dw    (?)              ;Line counter to detect VOFF1% clear timing
chr_c       db    (?)              ;Character counter
b_buf       db    8 dup (?)        ;brightness or blue for pixel0-7
g_buf       db    8 dup (?)        ;green for pixel0-7
r_buf       db    8 dup (?)        ;red   for pixel0-7
col_mono    db    (?)              ;color/monochrome
scheme      db    (?)              ;FRC scheme 0;old 1;aruns 2:new
sch_64      db    (?)   ;0;16FRC+dither 1;64FRC old 2;64FRC new 8;16 color FRC 9;64 color FRC
tile_l1     db    (?)              ;tile pattern for 1st line
tile_l2     db    (?)              ;tile pattern for 2nd line
tile_br     db    (?)              ;tile pattern foreground brightness
tile_t1     db    (?)              ;tile pattern for temporary use
tile_t2     db    (?)              ;tile pattern for temporary use
xr_buf      db    38 dup (?)       ;XR18-1E, XR50-6E
vmode       db    (?)              ;mode12=12h, mode13=13h
ipal        db    17 dup (?)       ;internal palette
epal_b      db    256 dup (?)      ;external palette green
epal_g      db    256 dup (?)      ;external palette red
epal_r      db    256 dup (?)      ;external palette blue
epal_w      db    256*3 dup (?)    ;external palette work area
dump_lc     db    (?)              ;line count for dump
dump_fc     db    (?)              ;frame count for dump
dump_ty1    db    0dh,0ah,'FRAME=  ','   H_OFFSET=  ','   V_OFFSET=  ',0dh,0ah     ;l=40
dump_ty2    db    21 dup (' '),16 dup ('1'),16 dup ('2'),16 dup ('3'),0dh,0ah      ;l=71
dump_ty3    db     5 dup (' '),4 dup ('0123456789ABCDEF'),0dh,0ah,0dh,0ah          ;l=73
dump_buf    db     5 dup (' '),64 dup (' '),0dh,0ah                          ;length=71
dump_mline  db     (?)              ;maximum lines to be dumped
b_d_num     db     '0123456789'           ;binary to decimal conversion table
b_h_num     db     '0123456789ABCDEF'     ;binary to hexadecimal conversion table
key_buf     db     7 dup(' ')       ;key buffer
pal13_z     db     (?)              ;external palette select mode
rgb_seq     db     (?)              ;0=don't care, 1="GRB", 2="RGB"
dd_emu      db     (?)              ;0=no D/D, 1=D/D, bit3=toggle upper/lower
acd_wwork   dw     (?)              ;temporary acdclk waveform
acd_worg    dw     (?)              ;original acdclk waveform
acd_cwork   db     (?)              ;acdclk period
acd_corg    db     (?)              ;originl acdclk period
esize       db     (?)              ;0=640X200, 1=320X100
sloop_cnt   db     (?)              ;screen loop count
fscr_seg    dw     (?)              ;1st screen segment
scr_off     dw     (?)              ;screen offset
lscr_off    dw     (?)              ;last screen offset
      ;
;*********************************
;* Data tables {file descripter} *
;*********************************
      ;
handle      dw    (?)
block       dw    0,?,?,5ch,?,6ch,?
path        db    'picgen.com',0
c_tail_12   db    14,' barsm12      ',0dh
            db    14,' bars2m12     ',0dh
            db    14,' linem12      ',0dh
c_tail_13   db    14,' clownm13     ',0dh
            db    14,' zoem13       ',0dh
            db    14,' barsm13      ',0dh
path_dall   db    'FRCdump.bit',0
path_dasc   db    'FRCdump.dat',0
```

```
      ;
;**************
;* Stack area *
;**************
      ;
      dw    80 dup (?)
      ;
mem_max     label word
      ;
;
main  endp
prog  ends
      end main
```