V. PERENCANAAN PERANGKAT LUNAK

Sesuai dengan pembahasan bab IV, perangkat keras terus menerus menyampling data dan menuliskan ke memori eksternal sampai terdeteksinya penekanan tombol 'capture'. Penekanan tombol ini menghentikan proses sampling gambar berikutnya dan data ditransfer dari memori eksternal ke memori komputer. Selanjutnya data ditampilkan ke monitor dengan resolusi 640x480.

Untuk menampilkan gambar dalam resolusi 640x480, baik 256 warna ataupun 16 juta warna, tak dapat menggunakan mode VGA standard (interrupt 10H, fungsi 00H, subfungsi 13H), melainkan harus memanfaatkan mode SVGA. Pada Tugas Akhir ini Super VGA card yang dipakai adalah Tseng 4000.

Pada bab V ini pembahasan meliputi hal-hal sebagai berikut :

- Perencanaan perangkat lunak secara umum.
- Format file PCX.
- Driver Super VGA card Tseng 4000.
- Pengambilan dan transfer data biner dari alat (memori eksternal) ke memori komputer.
- Penampilan gambar ke monitor Super VGA.
- Konversi data binari ke file PCX.
- Listing program.

1. PERENCANAAN PERANGKAT LUNAK SECARA UMUM

Bagian ini diawali dengan menancapkan alat ke salah satu slot komputer. Kemudian sinyal video komposit dimasukkan ke input peralatan ini. Sinyal video yang dipakai bisa berasal dari televisi, kaset video atau kamera (handycam). Mulai dari awal program dijalankan alat akan terus menerus mengambil data gambar medan genap dari input sinyal video komposit. Gambar yang ingin di-'capture', diatur agar terfokus dengan baik. Kemudian tekan tombol tertentu (tombol untuk perintah capture) maka perangkat lunak akan mulai mengambil data gambar biner dari peralatan (memori luar) dan ditransfer ke memori komputer. Setelah beberapa saat gambar akan ditampilkan pada layar. Gambar ini dapat disimpan pada media disk.

Pada proses penampilan gambar ke monitor, program berhubungan dengan driver Super VGA card. Pada Tugas Akhir ini Super VGA card yang dibahas adalah Tseng 4000 yang mampu menampilkan gambar 256 warna maupun 16 juta warna dalam resolusi 640x480.

Adapun langkah-langkah algoritma perangkat lunak adalah sebagai berikut :

a. Inisialisasi peralatan.
b. Inisialisasi driver Super VGA card.
c. Ambil data gambar dari input sinyal video komposit dan tulis ke RAM eksternal.
d. Deteksi tombol untuk transfer gambar (capture). Jika tombol ditekan mulai proses berikut ini.

e. Ambil data merah dan simpan ke memori komputer.

f. Ambil data hijau dan simpan ke memori komputer.

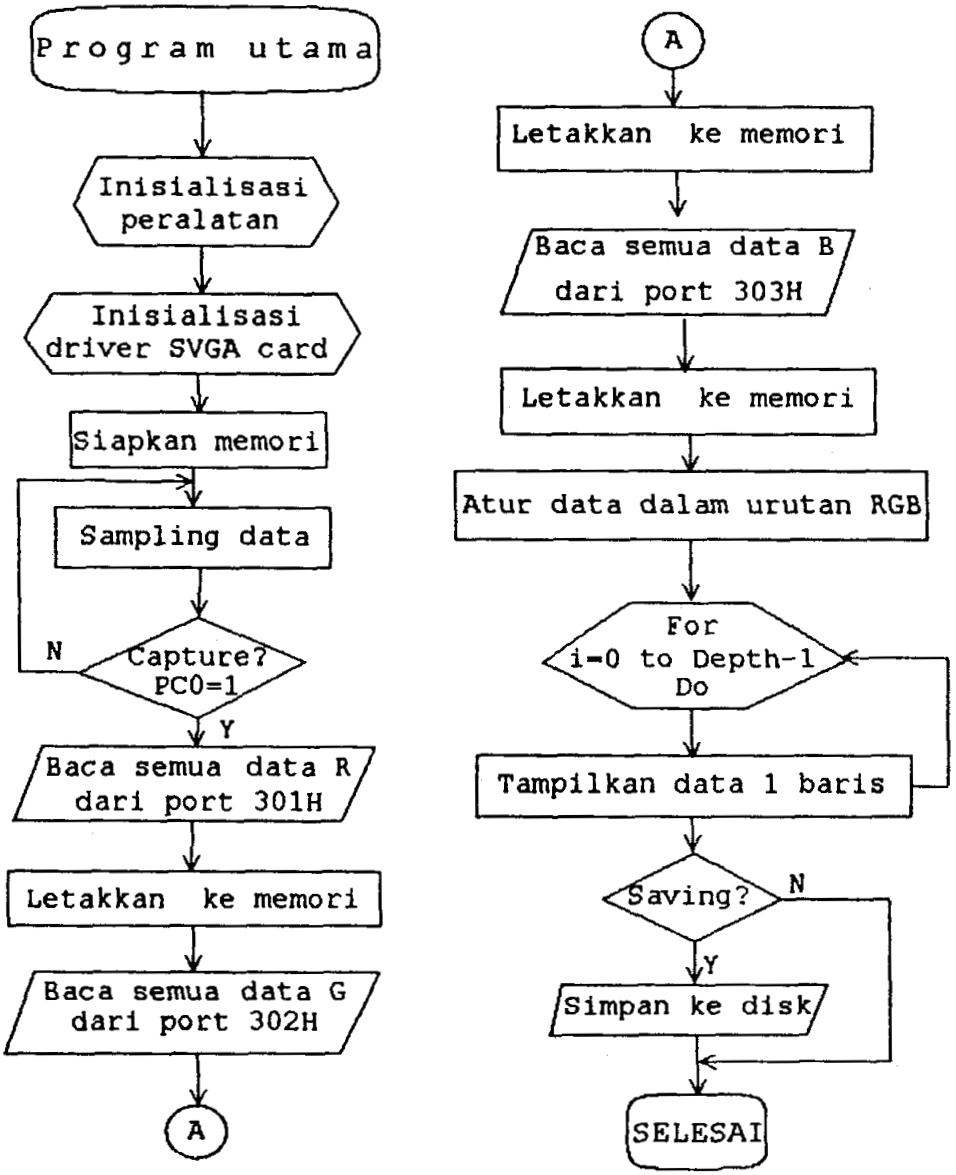
g. Ambil data biru dan simpan ke memori komputer.

h. Atur data-data biner pada memori dalam urutan RGB.

i. Tampilkan data RGB ke monitor.

j. Deteksi tombol penyimpanan data. Jika tombol ditekan pindahkan blok data dari memori komputer ke disk.

k. Selesai.



GAMBAR 5-1

DIAGRAM ALIR PROGRAM UTAMA

2. FORMAT FILE PCX

Pengolahan gambar lebih lanjut, misalnya pengeditan dan pencetakan pada printer dilakukan dengan menggunakan paket program yang telah banyak beredar di pasaran misalnya Corel Draw atau Paintbrush. Karena itu data gambar yang diambil harus dalam bentuk standar format file tertentu, misalnya GIF, PCX atau BMP. Pada perencanaan ini dipakai standar format file PCX 256 warna dan 16.7 juta warna (true color).

Format file PCX secara umum terdiri dari 3 bagian, yaitu :

- Header
- Data gambar
- Palette warna

Masing-masing bagian tersebut akan dijelaskan berikut ini.

2.1 Header

Header terletak pada bagian awal file PCX dengan panjang 128 byte. Header mengandung beberapa informasi yang menentukan tampilan data obyek gambar pada layar. Berikut ini adalah uraian dari ke-128 byte informasi pada header.

- Byte 0 : Kode format file PCX, selalu A0H.
- Byte 1 : Nomor versi dari PC Paintbrush.
  - 2 = PC Paintbrush 2.8 dengan palette.
  - 5 = PC Paintbrush 3.0 atau yang lebih baru.

- Byte 2 : Encoding = 1, artinya memakai teknik RLE (run length encoding).
- Byte 3 : Bit/pixel/plane = 8 untuk format 256 dan 16 juta warna.
- Byte 4-5 : Koordinat x minimum (Xmin).
- Byte 6-7 : Koordinat y minimum (Ymin).
- Byte 8-9 : Koordinat x maksimum (Xmaks).
- Byte 10-11 : Koordinat y maksimum (Ymaks).
- Byte 12-13 : Resolusi horisontal dalam dot/inchi.
- Byte 14-15 : Resolusi vertikal dalam dot/inchi.
- Byte 16-63 : Palette header, hanya dipakai untuk file 16 warna atau kurang.
- Byte 64 : Byte cadangan, biasanya = 0.
- Byte 65 : Banyaknya plane. Untuk file 256 warna plane = 1, sedangkan untuk file 16 juta warna plane = 3.
- Byte 66-67 : Byte/garis horisontal/plane.
- Byte 68-69 : Tipe palette.

  1 = gambar berwarna atau monokrom.

  2 = gambar dengan skala keabuan (greyscale).
- Byte 70-127: Kosong, tak dipakai.

2.2 Data Gambar

Data gambar terletak pada bagian tengah file PCX, tepat di bawah header. Berarti awal data gambar dimulai pada byte ke-129 (atau byte 128). Sedang akhir data gambar bergantung pada resolusi dan dimensi gambar. Adapun data gambar file PCX menggunakan teknik pemampatan RLE (run length encoding) sebagai

berikut :

- Data berurutan yang bernilai sama dapat diartikan terjadi pengulangan data. Pengulangan dinyatakan dengan indeks dan data itu sendiri.
- Indeks terletak di depan data, menyatakan banyaknya pengulangan data.
- Indeks dibedakan dari data dengan cara membuat bit 6 dan bit 7 berharga 1.
- Pada indeks, jumlah pengulangan dinyatakan pada bit 0 sampai bit 5.
- Jumlah pengulangan yang lebih besar dari 63 kali dinyatakan dengan lebih dari satu urutan indeks dan data.
- Data gambar yang lebih besar dari 3FH (63), meskipun tidak terjadi pengulangan tetap dinyatakan dengan indeks dan data itu sendiri.
- Data yang tidak berulang dan lebih kecil atau sama dengan 3FH, dinyatakan dengan data itu sendiri tanpa indeks.
- Pemadatan data dilakukan pada tiap baris. Baris yang satu tak berkaitan dengan baris berikutnya, meskipun ada data yang sama pada akhir suatu baris dengan awal baris berikutnya.

Pada file PCX 256 warna data gambar merupakan penunjuk (pointer) ke data palette yang bersesuaian. Sedangkan pada file PCX 16 juta warna data gambar merupakan harga komponen R,G dan B yang sebenarnya. Pemampatan data gambar 16 juta warna dilakukan untuk komponen merah pada satu garis, kemudian komponen

hijau pada garis yang sama dan terakhir untuk komponen biru. Demikian seterusnya untuk garis-garis berikutnya sampai semua data termampatkan.

2.3 Palette Warna

Palette terletak pada bagian akhir dari file PCX. Pada file PCX 256 warna terdapat palette, sedang pada file PCX 16 juta warna tak terdapat palette. Palette warna merupakan komposisi campuran warna yang dapat ditampilkan dalam satu layar pada waktu yang bersamaan. Setiap warna yang ditampilkan merupakan campuran dari ketiga warna primer. Komposisi warna tersebut disimpan secara berurutan yakni merah, hijau dan biru (RGB). Palette warna yang terletak pada akhir dari file PCX ini memiliki panjang 768 byte. Hal ini menunjukkan ada 256 campuran warna. Awal dari bagian ini ditandai dengan byte 0CH.

3. DRIVER SUPER VGA CARD TSENG 4000

Pada bagian ini akan diuraikan stuktur dan cara kerja Super VGA card Tseng 4000 secara umum. Adapun program driver untuk Super VGA card dibuat terpisah dari program utama supaya program lebih fleksibel dan dapat dikembangkan untuk berbagai driver SVGA card lainnya. Pada saat program utama dijalankan, maka program driver akan di-'load' ke memori sebagai kode-kode biner.

Seperti yang telah kita ketahui komputer menyediakan memori untuk buffer layar dimulai pada alamat

A000:0000H, sebesar satu segment atau 64Kbyte. Untuk gambar 256 warna resolusi VGA standar 320x200 titik, memori sebesar itu masih mencukupi. Tetapi resolusi 640x480 membutuhkan memori sebesar 640x480x1byte atau sebesar 300Kbyte untuk gambar 256 warna dan memori 600Kbyte untuk gambar true color. Untuk mengatasi hal ini maka Super VGA card mempunyai memori yang terdiri dari beberapa page atau bank.

Jika data gambar yang dikirim ke buffer layar telah melebihi alamat A000:FFFFH (batas 1 segment sebesar 64K) maka super VGA card Tseng 4000 akan mengganti page layar (screen paging) dengan page berikutnya. Pada page baru ini data dikirim kembali ke alamat A000:0000H.

Program driver lengkap dapat dilihat pada akhir dari bab V. Program ini terdiri dari beberapa fungsi yang dapat dipanggil oleh program utama. Fungsi-fungsi tersebut adalah seperti berikut ini:

- Inisialisasi.
- RGB On, RGB Line dan RGB Off.
- VGA On, VGA Line, VGA Palette dan VGA Off.

3.1 Inisialisasi

Bagian inisialisasi berisi alamat (offset dan segment) dari fungsi-fungsi yang terdapat pada driver. Harga segment ini hanya dapat ditentukan setelah program di-'load' ke memori. Informasi tentang resolusi layar juga terdapat pada bagian ini.

3.2 RGB On dan VGA On

SVGA card Tseng mempunyai nomor mode 002EH untuk mengaktifkan layar dengan resolusi 640x480 (256 warna dan 16 juta warna). Bagian ini membuat sebuah tabel alamat awal dari tiap baris yang akan ditampilkan. Karena terdapat 480 baris maka tabel terdiri dari 480 input. Tiap-tiap alamat yang disimpan terdiri dari 2 word (4 byte). 2 buah byte pertama menunjukkan offset awal baris yang dihitung dari page yang bersangkutan. Byte ke-3 menandakan nomor page layar sedangkan byte terakhir merupakan suatu flag. Jika dalam satu baris terdapat pergantian page maka flag diberi harga FFH,sebaliknya flag diberi harga 00H.

3.3 RGB Line dan VGA Line

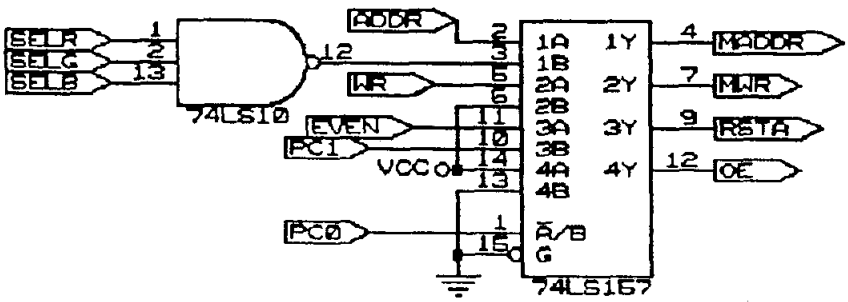
Bagian ini berfungsi untuk menampilkan data gambar satu baris ke layar. Pada gambar 256 warna data tiap titik yang ditulis ke buffer layar diwakili data 1 byte. Sedangkan pada gambar 16 juta warna data RGB 24 bit ditulis ke buffer layar setelah dipadatkan menjadi data 2 byte. Bagian ini akan mendeteksi flag dari tabel alamat. Jika flag = 0H data satu garis akan langsung ditulis ke page layar yang bersesuaian. Jika flag = FFH maka sebagian data ditulis ke buffer, setelah melampaui batas 64Kbyte terjadi pergantian page dan sisa data yang ada ditulis ke page yang baru.

3.4 RGB Off dan VGA Off

Bagian ini berfungsi mengembalikan layar ke mode teks setelah gambar ditampilkan.

4. TRANSFER DATA KE MEMORI KOMPUTER

Seperti telah dibahas pada bab IV, peralatan akan terus menerus menyampling data medan genap suatu gambar. Gambar yang telah disampling sebelumnya akan dihapus dan diisi dengan gambar yang baru, sehingga gambar yang diambil akan selalu real time.Pada saat tombol untuk capture ditekan, data-data dari memori eksternal akan ditransfer ke memori komputer.

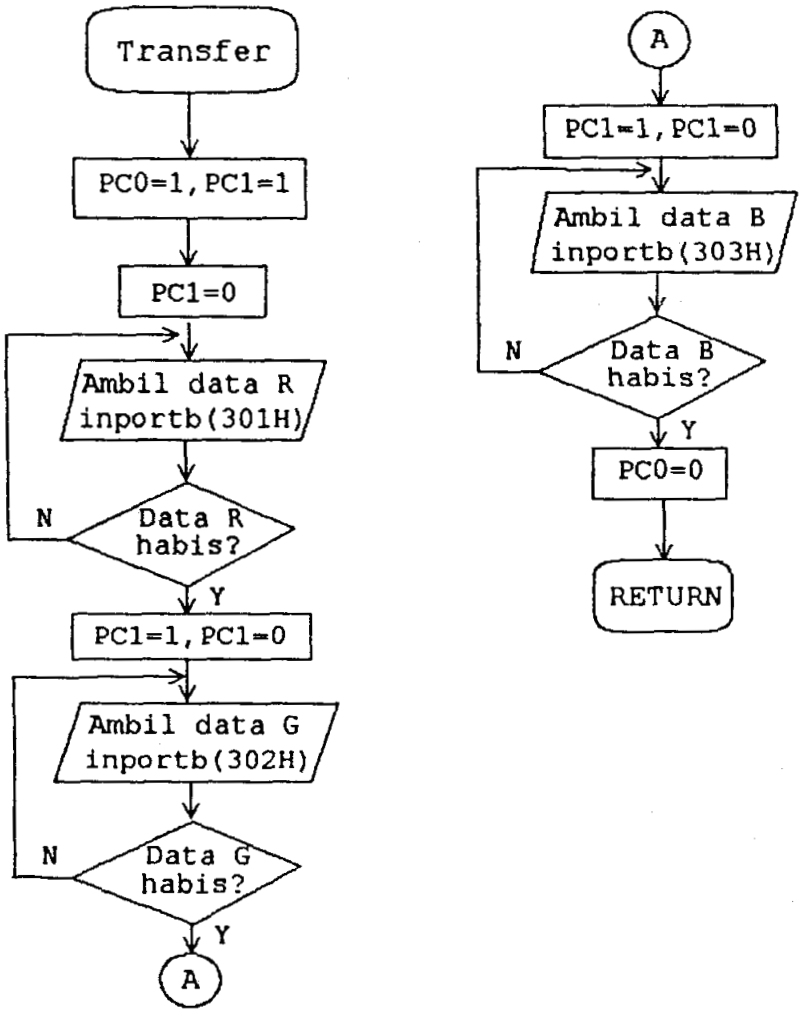


GAMBAR 5-2

MULTIPLEKSER PENGONTROL PROSES BACA & TULIS

Gambar 5-2 menunjukkan sebuah multiplekser 2 input yang berfungsi untuk mengontrol proses baca dan tulis pada memory eksternal. Buffer PC0 dan PC1 terletak pada alamat 300H, data merah (SELR), data hijau (SELG) dan data biru (SELB) berturut-turut terletak pada alamat 301H, 302H dan 303H.

Pada proses sampling PC0 diberi harga 0, yang berarti semua input A akan diteruskan ke output dan data RGB akan ditulis dari ADC ke memori eksternal. Pada proses transfer, PC0 diberi harga 1, berarti semua input B akan diteruskan ke output dan data RGB dibaca dari memori eksternal ke memori komputer.



GAMBAR 5-3

DIAGRAM ALIR PROSES TRANSFER

Diagram alir proses transfer dapat dilihat pada gambar 5-3. Sebelum proses transfer dimulai PC1 diberi harga 1 diikuti harga 0, yang menyebabkan address coun-

ter 74LS393 di-'clear'. Selanjutnya data merah mulai dibaca dari alamat 301H. Setiap kali instruksi inportb(0x301H) dilaksanakan akan memberikan pulsa trigger (pulsa MADDR) ke address counter 74LS393. Dengan demikian data merah akan dibaca sebanyak jumlah instruksi inportb yang dilaksanakan. Setelah semua data merah terbaca, address conter di-'clear' lagi (PC1=1 lalu PC1=0) dan proses yang sama terjadi untuk data hijau dan data biru. Setelah semua data RGB terbaca PC0 akan diberi harga 0 dan proses sampling berlanjut.

5. MENAMPILKAN DATA GAMBAR KE MONITOR

Setelah terdeteksi perintah untuk 'capture', maka data-data pada memori eksternal akan ditransfer ke memori komputer. Sebelum ditampilkan data-data ini harus diolah lagi, sehingga letak data (dalam bentuk RGB) harus berurutan untuk tiap-tiap titik. Untuk gambar dengan ukuran 416x312 letak data yang dimaksud ditunjukkan pada gambar 5-4.

$R_1G_1B_1$ $R_2G_2B_2$ $R_3G_3B_3$ ..... $R_{416}G_{416}B_{416}$ baris 1
$R_1G_1B_1$ $R_2G_2B_2$ $R_3G_3B_3$ ..... $R_{416}G_{416}B_{416}$ baris 2
$R_1G_1B_1$ $R_2G_2B_2$ $R_3G_3B_3$ ..... $R_{416}G_{416}B_{416}$ baris 3
.....
$R_1G_1B_1$ $R_2G_2B_2$ $R_3G_3B_3$ ..... $R_{416}G_{416}B_{416}$ baris 312

GAMBAR 5-4

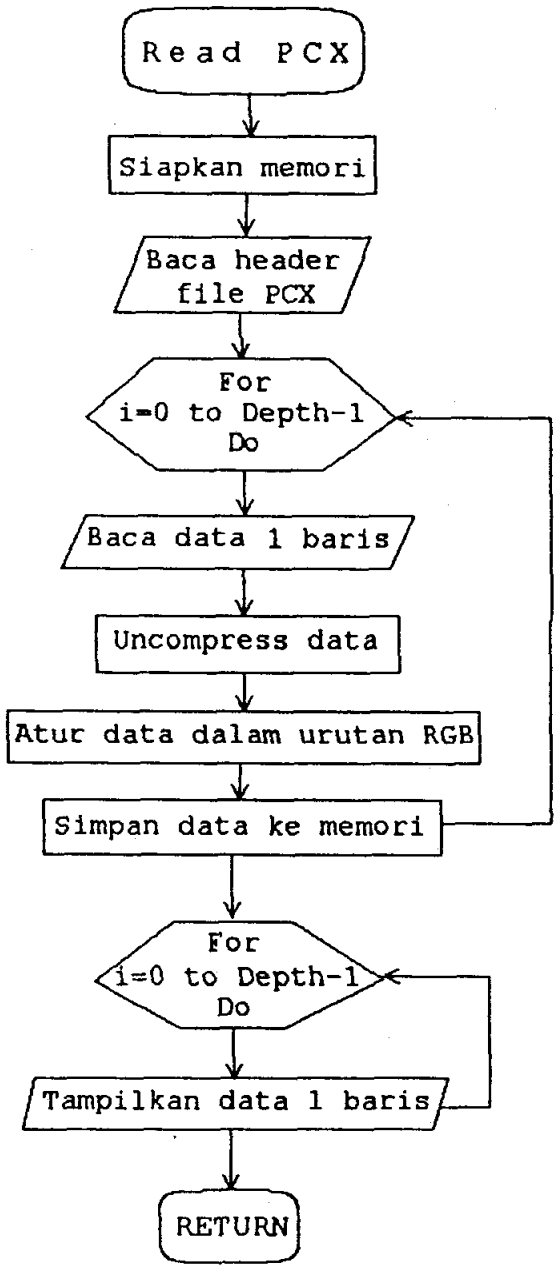
DATA GAMBAR YANG SIAP DITAMPILKAN

Sebelum data dikirimkan ke buffer layar, driver SVGA card harus di-'load' ke memori. Jika proses ini berhasil, maka akan dihasilkan alamat dari fungsi RGB On, RGB Line, RGB Off, VGA On dan fungsi-fungsi lainnya. Dengan demikian fungsi-fungsi tersebut siap dipanggil oleh program utama. Alamat segment yang pada program driver dibiarkan kosong, ditentukan pada saat loading ini. Proses loading dikerjakan oleh fungsi LoadDriver pada program utama.

Setelah proses loading berhasil, program utama akan memanggil fungsi RGB Line untuk menampilkan satu baris data ke layar (untuk gambar 16 juta warna). Hal ini terus berulang sampai tercapai sejumlah baris yang ingin ditampilkan pada layar monitor. Hal yang sama berlaku untuk gambar 256 warna dengan fungsi VGA Line. Jika dimensi gambar lebih kecil dari resolusi layar, gambar akan ditampilkan tepat di tengah. Sedangkan jika dimensi gambar lebih besar dari resolusi layar, gambar akan ditampilkan mulai kiri atas dan selanjutnya dapat digeser-geser. Penekanan tombol 'escape' menyebabkan layar kembali ke mode teks.

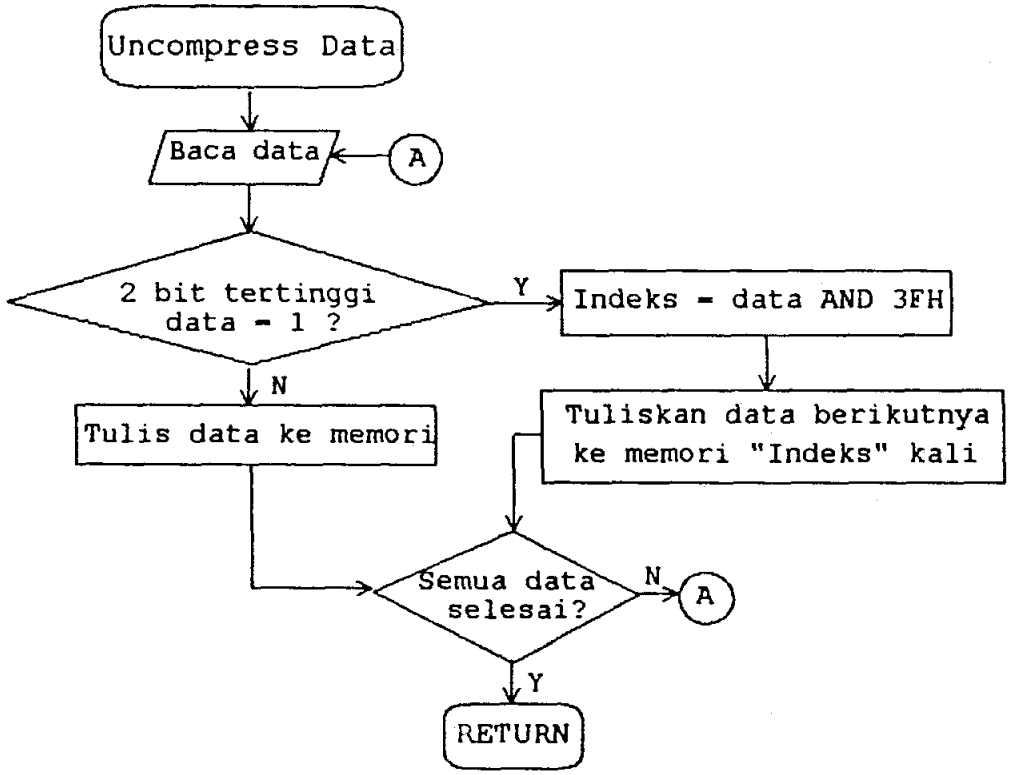
Selain bisa menampilkan gambar yang di-'capture' dari alat, program ini juga dapat menampilkan file gambar PCX dari media disk. Pada kasus ini, mula-mula disiapkan sejumlah memori sesuai dengan dimensi gambar. Kemudian baca data tiap baris dari file, atur sesuai urutan RGB, dan letakkan ke memori. Setelah semua data termuat di memori, data gambar siap ditampilkan dengan cara yang sama seperti yang telah dijelaskan sebelumnya.

Diagram alir untuk menampilkan file PCX dapat dilihat pada gambar 5-5. Fungsi yang bersesuaian dengan hal ini adalah fungsi UnpackPCXFile pada program utama. Fungsi ReadPCXLine berfungsi untuk membaca data satu baris dari file dan mendekode data yang tersimpan dengan teknik RLE menjadi data biner RGB. Diagram alir fungsi ini dapat dilihat pada gambar 5-6.



GAMBAR 5-5

DIAGRAM ALIR PEMBACAAN FILE PCX



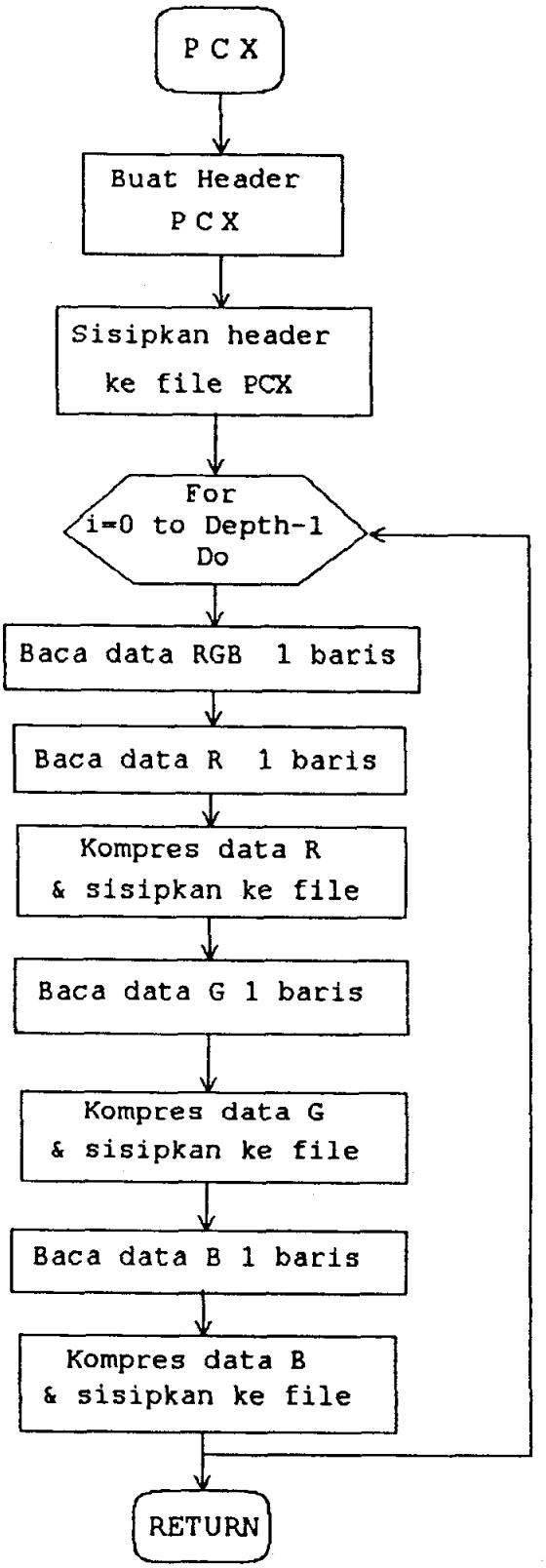
GAMBAR 5-6

DIAGRAM ALIR UNCOMPRESS DATA

6. KONVERSI DATA BINER KE FILE PCX

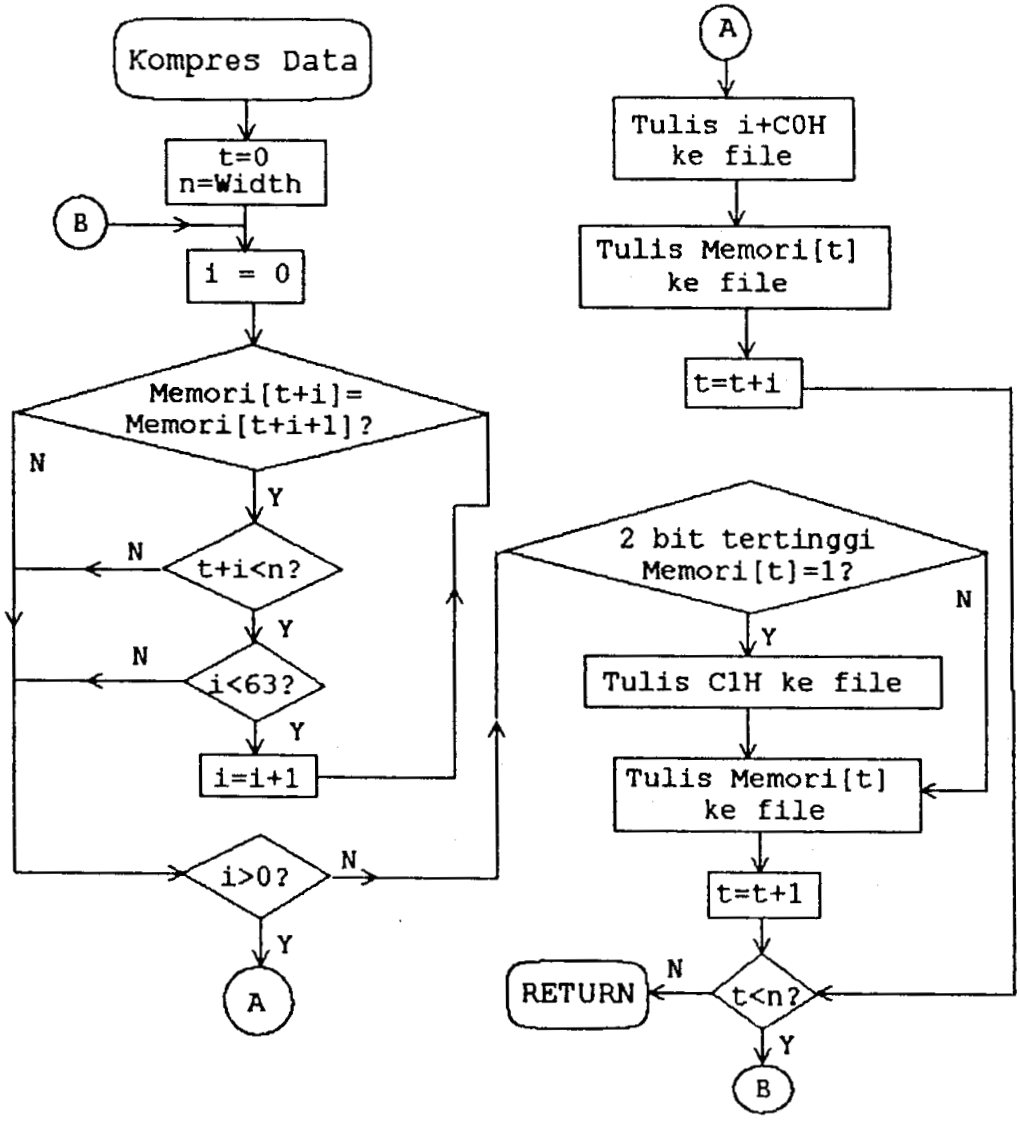
Data-data biner yang tersimpan di memori komputer dalam urutan RGB, dapat disimpan ke media disk. Untuk menyimpan file gambar digunakan standar format file PCX.

Langkah-langkah pembuatan file PCX dapat dilihat pada gambar 5-7. Diagram alir ini diwujudkan dengan fungsi WritePCXFile pada program utama. Sedangkan diagram alir pemampatan data pada file PCX ditunjukkan pada gambar 5-8. Fungsi WritePCXLine memadatkan data dengan metode RLE.



GAMBAR 5-7

DIAGRAM ALIR PEMBUATAN FORMAT FILE PCX



GAMBAR 5-8

DIAGRAM ALIR KOMPRESI DATA DENGAN TEKNIK RLE

7. LISTING PROGRAM

```
;     Program driver untuk Super VGA Card Tseng 4000
;     Mode layar : 640x480 16,7 juta warna (24 bit)
;                  640x480   256 warna     ( 8 bit)
;                  320x200   256 warna     ( 8 bit)
;     Author  : Maria Imelda N.  (23491047)
;     Version : 1.20   Date : 13 Desember 1995
;     Electrical Engineering Department
;     Petra Christian University

RGBWide       EQU  640
RGBDeep       EQU  480
RGBScreenSeg  EQU  0A000H
VGAWide       EQU  640
VGADeep       EQU  480
VGAScreenSeg  EQU  0A000H

RGB2AX   MACRO
         MOV  DX,0
         LODSB
         MOV  CL,3
         SHR  AL,CL
         AND  AX,001FH
         MOV  CL,10
         SHL  AX,CL
         OR   DX,AX
         LODSB
         MOV  CL,3
         SHR  AL,CL
         AND  AX,001FH
         MOV  CL,5
         SHL  AX,CL
         OR   DX,AX
         LODSB
         MOV  CL,3
         SHR  AL,CL
         AND  AX,001FH
         OR   DX,AX
         MOV  AX,DX
         ENDM

TSengReg MACRO
         MOV  DX,03C4H
         MOV  AL,0
         OUT  DX,AL
         MOV  DX,03C5H
         IN   AL,DX
         AND  AL,0FEH
         OUT  DX,AL
         MOV  DX,03CEH
         MOV  AL,6
         OUT  DX,AL
         MOV  DX,03CFH
         IN   AL,DX
         XOR  AL,6
         OR   AL,2
         OUT  DX,AL
         ENDM

CODE  SEGMENT PARA PUBLIC 'CODE'
      ASSUME  CS:CODE
      ORG     0000H
      DB      '23491047'

Initialize   PROC FAR
             DW   RGBOn
             DW   ?
             DW   RGBLine
             DW   ?
             DW   RGBOff
             DW   ?
             DW   VGAOn
             DW   ?
             DW   VGALine
             DW   ?
             DW   VGAOff
             DW   ?
             DW   VGAPalette
             DW   ?
V_RGBWide    DW   RGBWide
V_RGBDeep    DW   RGBDeep
V_VGAWide    DW   VGAWide
V_VGADeep    DW   VGADeep
V_VGAScrnSeg DW   VGAScreenSeg
Initialize   ENDP

RGBOn PROC NEAR
      PUSH BP
      PUSH DS
      PUSH ES
      MOV  CS:[V_RGBWide],640
      MOV  CS:[V_RGBDeep],480
      MOV  CX,480
      SUB  BX,BX
      MOV  SI,OFFSET Table
RGB1: PUSH CX
      MOV  AX,1280
      MUL  BX
      AND  DX,0FFH
      CMP  AX,(0FFFEH-1280)
      JB   RGB2
      MOV  DH,0FFH
RGB2: MOV  CS:[SI],AX
      MOV  CS:[SI+2],DX
      ADD  SI,4
      INC  BX
      POP  CX
      LOOP RGB1
      TSengReg
      MOV  AX,10F0H
      MOV  BX,2EH
      INT  10H
      MOV  CS:[Bank],0
      POP  ES
      POP  DS
      POP  BP
      RETF
```

```
RGBOn ENDP

RGBLine   PROC NEAR
          PUSH BP
          MOV  BP,SP
          PUSH DS
          PUSH ES
          MOV  SI,[BP+6]
          MOV  DS,[BP+8]
          MOV  CX,[BP+10]
          CMP  CX,CS:[V_RGBWide]
          JL   RGBLine1
          MOV  CX,CS:[V_RGBWide]
RGBLine1: MOV  AX,0A000H
          MOV  ES,AX
          MOV  BX,[BP+12]
          SHL  BX,1
          SHL  BX,1
          MOV  DI,CS:[BX+Table]
          MOV  AX,CS:[BX+Table+2]
          CLD
          CALL TSengBank
          CMP  AH,0
          JNE  RGBLine3
          MOV  AX,[BP+14]
          SHL  AX,1
          ADD  DI,AX
RGBLine2: PUSH CX
          RGB2AX
          STOSW
          POP  CX
          LOOP RGBLine2
          JMP  ExitRGB
          NOP
RGBLine3: MOV  AX,[BP+14]
          SHL  AX,1
          ADD  DI,AX
          CMP  DI,1280
          JB   RGBLine6
RGBLine4: PUSH CX
          RGB2AX
          STOSW
          POP  CX
          CMP  DI,0
          JNE  RGBLine5
          MOV  AX,CS:[Bank]
          INC  AX
          CALL TSengBank
RGBLine5: LOOP RGBLine4
          JMP  ExitRGB
RGBLine6: MOV  AX,CS:[Bank]
          ADD  AX,1
          CALL TSengBank
RGBLine7: PUSH CX
          RGB2AX
          STOSW
          POP  CX
          LOOP RGBLine7
ExitRGB:  POP  ES
          POP  DS
          POP  BP
          RETF
RGBLine   ENDP

RGBOff    PROC NEAR
          PUSH BP
          PUSH DS
          PUSH ES
          MOV  AX,1200H
          MOV  BX,0031H
          INT  10H
          MOV  AX,0003H
          INT  10H
          POP  ES
          POP  DS
          POP  BP
          RETF
RGBOff    ENDP

VGAOn     PROC NEAR
          PUSH BP
          MOV  BP,SP
          PUSH DS
          CMP  WORD PTR [BP+6],320
          JG   SVGAOn1
          CMP  WORD PTR [BP+8],200
          JG   SVGAOn1
          MOV  CS:[V_VGAWide],320
          MOV  CS:[V_VGADeep],200
          MOV  CX,200
          SUB  DX,DX
          MOV  SI,OFFSET Table
VGAOn1:   PUSH DX
          MOV  AX,320
          MUL  DX
          MOV  CS:[SI],AX
          ADD  SI,2
          POP  DX
          INC  DX
          LOOP VGAOn1
          MOV  AX,0013H
          INT  10H
          JMP  ExitVGA
SVGAOn1:  MOV  CS:[V_VGAWide],640
          MOV  CS:[V_VGADeep],480
          MOV  CX,480
          SUB  BX,BX
          MOV  SI,OFFSET Table
SVGAOn2:  PUSH CX
          MOV  AX,640
          MUL  BX
          AND  DX,00FFH
          CMP  AX,(0FFFFH-639)
          JB   SVGAOn3
          MOV  DH,0FFH
SVGAOn3:  MOV  CS:[SI],AX
          MOV  CS:[SI+2],DX
          ADD  SI,4
          INC  BX
          POP  CX
          LOOP SVGAOn2
          TSengReg
          MOV  AX,2EH
          INT  10H
          MOV  CS:[Bank],0000H
ExitVGA:  POP  DS
          POP  BP
          RETF
```

```
VGAOn     ENDP

VGALine   PROC NEAR
          PUSH BP
          MOV  BP,SP
          PUSH DS
          PUSH ES
          CMP  CS:[V_VGAWide],320
          JNE  SVGALine1
          MOV  SI,[BP+6]
          MOV  DS,[BP+8]
          MOV  BX,[BP+12]
          CMP  BX,200
          JG   ExitVGA1
          SHL  BX,1
          MOV  DI,CS:[Table+BX]
          MOV  AX,[BP+14]
          ADD  DI,AX
          CLD
          MOV  CX,[BP+10]
          CMP  CX,0
          JE   ExitVGA1
          CMP  CX,320
          JL   VGALine1
          MOV  CX,320
VGALine1: MOV  AX,0A000H
          MOV  ES,AX
REPNE     MOVSB
ExitVGA1: JMP  ExitVGA2
SVGALine1:CMP WORD PTR [BP+12],480
          JL   SVGALine2
          JMP  ExitVGA2
SVGALine2:MOV  SI,[BP + 6]
          MOV  DS,[BP+8]
          MOV  CX,[BP+10]
          CMP  CX,640
          JL   SVGALine3
          MOV  CX,640
SVGALine3:MOV  BX,[BP+12]
          SHL  BX,1
          SHL  BX,1
          MOV  DI,CS:[Table+BX]
          MOV  AX,[BP+14]
          ADD  DI,AX
          MOV  AX,CS:[Table+BX+2]
          CALL TSengBank
          CMP  AH,00H
          JNE  SVGALine4
          MOV  AX,0A000H
          MOV  ES,AX
          CLD
REPNE     MOVSB
          JMP  ExitVGA2
SVGALine4:MOV  AX,0A000H
          MOV  ES,AX
          CMP  DI,640
          JB   SVGALine6
          PUSH CX
          MOV  AX,DI
          MOV  CX,0000H
          SUB  CX,AX
          MOV  BX,CX
          CMP  CX,[BP+10]
          JL   SVGALine5
          MOV  CX,[BP+10]
          CLD
REPNE     MOVSB
          POP  CX
          JMP  ExitVGA2
SVGALine5:CLD
REPNE     MOVSB
          POP  CX
          SUB  CX,BX
          CMP  CX,0
          JLE  ExitVGA2
          MOV  BX,[BP+12]
          SHL  BX,1
          SHL  BX,1
          MOV  AX,CS:[Table+BX+2]
          INC  AX
          CALL TSengBank
          MOV  AX,0A000H
          MOV  ES,AX
          MOV  DI,0
          CLD
REPNE     MOVSB
          JMP  ExitVGA2
SVGALine6:MOV  AX,CS:[BX+2+Table]
          INC  AX
          CALL TSengBank
REPNE     MOVSB
ExitVGA2: POP  ES
          POP  DS
          POP  BP
          RETF
VGALine   ENDP

VGAOff    PROC NEAR
          MOV  AX,1200H
          MOV  BX,0031H
          INT  10H
          MOV  AX,0003H
          INT  10H
          RETF
VGAOff    ENDP

TSengBank PROC NEAR
          CMP  AX,CS:[Bank]
          JNE  Bank1
          RET
Bank1:    MOV  CS:[Bank],AX
          PUSH AX
          PUSH DX
          MOV  AH,AL
          MOV  DX,3BFH
          MOV  AL,3
          OUT  DX,AL
          MOV  DL,0D8H
          MOV  AL,0A0H
          OUT  DX,AL
          AND  AH,15
          MOV  AL,AH
          SHL  AL,1
          SHL  AL,1
          SHL  AL,1
          SHL  AL,1
          OR   AL,AH
          MOV  DL,0CDH
```

```
          OUT  DX,AL
          POP  DX
          POP  AX
          RET
TSengBank ENDP

VGAPalette  PROC NEAR
            PUSH BP
            MOV  BP,SP
            PUSH DS
            PUSH ES
            MOV  SI,[BP + 6]
            MOV  DS,[BP + 8]
            MOV  CX,[BP + 10]
            CMP  CX,0
            JG   Palette1
            JMP  ExitPalette
Palette1:   MOV  DX,03C6H
            MOV  AL,0FFH
            OUT  DX,AL
            MOV  BX,0
Palette2:   PUSH CX
            MOV  DX,03C8H
            MOV  AL,BL
            INC  BX
            OUT  DX,AL
            INC  DX
            LODSB
            SHR  AL,1
            SHR  AL,1
            OUT  DX,AL
            LODSB
            SHR  AL,1
            SHR  AL,1
            OUT  DX,AL
            LODSB
            SHR  AL,1
            SHR  AL,1
            OUT  DX,AL
            POP  CX
            LOOP Palette2
ExitPalette:POP  ES
            POP  DS
            POP  BP
            RETF
VGAPalette  ENDP

Table   DW   960 DUP(?)
Bank    DW   ?
CODE    ENDS
        END
```

```
/*      Hak Cipta oleh : Maria Imelda (23491047)                        */
/*      Program ini dibuat untuk ujian kesarjanaan                      */
/*      Pada Fakultas Teknik Jurusan Teknik Elektro                     */
/*      Universitas Kristen Petra  Surabaya                             */
/*      Version : 1.40    Date :  3  Januari 1995                       */
/*      Judul Tugas Akhir :                                             */
/*      PERENCANAAN DAN PEMBUATAN INTERFACE KAMERA KE IBM PC            */
/*      UNTUK MENGAMBIL GAMBAR,MENAMPILKAN KE MONITOR                   */
/*      DAN MENYIMPAN KE DISK                                           */

#include "alloc.h"
#include "conio.h"
#include "dos.h"
#include "memmangr.h"
#include "stdio.h"

#define      PanahKiri    0x4b00
#define      PanahKanan   0x4d00
#define      PanahAtas    0x4800
#define      PanahBawah   0x5000
#define      Home         0x4700
#define      End          0x4f00

#define      GOOD_READ          0
#define      BAD_FILE           1
#define      BAD_READ           2
#define      MEMORY_ERROR       3
#define      WRONG_BITS         4
#define      NO_DRIVER          5
#define      Step              32

typedef struct
{       char ch,at;
} pixel;

pixel *display=
{ (pixel far*)0xb8000000
},stack_disp[16000];

typedef struct
{     int Width,Depth,Xdisp,Ydisp,bytes,bits;
      char Palette[768];
} FILEINFO;

typedef struct
{     char Manufacturer,Version,Encoding,Bits_per_Pixel;
      int  Xmin,Ymin,Xmax,Ymax,Hres,Vres;
      char Palette[48],Reserved,Colour_Planes;
      int  Bytes_per_Line,Palette_Type;
      char Filler[58];
} PcxHeader;

typedef struct
{       int (*RGBOn)();
        int (*RGBLine)();
        int (*RGBOff)();
        int (*VGAOn)();
        int (*VGALine)();
        int (*VGAOff)();
        int (*VGAPalette)();
        int RGBWide;
        int RGBDeep;
        int VGAWide;
        int VGADeep;
```

```
        int VGAScreenSeg;
} GRAFDRIVER;

GRAFDRIVER *LoadDriver(char *s);
GRAFDRIVER *gd;
FILEINFO fi;

main()
{    int  ff,e,k;
     char ch,opt1,opt2,Display[80];
     char *vector[5]={"1. Setup          ",
                      "2. Capture        ",
                      "3. Display        ",
                      "4. About          ",
                      "5. Exit           "};

     char *setup[2]={"1. SVGA Card Driver",
                     "2. Capture         "};

     char *about[2]={"1. This Program   ",
                     "2. Author         "};

     FILE *fcfg;

/*   textbackground(RED);*/
     clrscr();
     opt1=0;
     do
     {   MakeBox(5,2,75,17,0x10,1,1,"COPYRIGHT BY : MARIA IMELDA
         (23491047)");
         window(1,1,80,25);
         ff=BarMenu(vector,30,6,opt1,0x30,0x21,5," Menu Utama");
         opt1=ff;
         SaveScreen(1);
         switch(ff)
         {  case 1 :
            {   opt2=0;
                do
                {  e=BarMenu(setup,35,8,opt2,0x30,0x21,2,"Setup");
                   opt2=e;
                   switch(e)
                   {  case 1 :
                      {  SaveScreen(2);
                         MakeBox(5,19,75,23,BLUE,1,1,"Driver setup");
                         ViewCursor();
                         if ((fcfg = fopen("MARIA.CFG","wb"))!=NULL)
                         {  gotoxy(10,21);
                            gets(Display);
                            fputs(Display,fcfg);
                            fclose(fcfg);
                         }
                       /*  do {} while(getch()!=13);*/
                         LoadScreen(2);
                         break;
                      }
                      case 2 :
                      {  SaveScreen(2);
                         MakeBox(5,18,75,24,BLUE,1,1,"Capture setup");
                         ViewCursor();
                         do {} while(getch()!=13);
                         LoadScreen(2);
                         break;
                      }
                   }
```

```
            LoadScreen(1);
        }
        while(e!=0);
        break;
    }
    case 2 :
    {   SaveScreen(2);
        MakeBox(5,18,75,24,LIGHTGRAY,1,1,"Capture");
        ViewCursor();
        Sampling();
        getch();
        LoadScreen(2);
        break;
    }
    case 3 :
    {   SaveScreen(2);MakeBox(5,18,75,24,CYAN,1,1,"Display");
        ViewCursor();
        DisplayFilePCX();
        getch();
        LoadScreen(2);
        break;
    }
    case 4 :
    {   opt2=0;
        do
        {   k=BarMenu(about,35,12,opt2,0x30,0x21,2,"About");
            opt2=k;
            switch(k)
            {   case 1 :
                {   SaveScreen(2);
                    MakeBox(5,10,75,24,BLUE,1,1,"About This
                    Program");
                    window(5,12,74,23);
                    textcolor(RED);
                    gotoxy(8,2);
                    cputs("Program ini dibuat untuk Skripsi/
                    Tugas Akhir yang berjudul\n");
                    gotoxy(11,3);
                    cputs("PERENCANAAN DAN PEMBUATAN INTERFACE
                    KAMERA KE IBM PC\n");
                    gotoxy(3,4);
                    cputs("UNTUK MENGAMBIL GAMBAR,MENAMPILKAN
                    KE MONITOR DAN MENYIMPAN KE DISK\n");
                    gotoxy(27,5);
                    cputs("Oleh :\n");
                    gotoxy(25,6);
                    cputs("Maria Imelda (23491047) \n");
                    gotoxy(18,7);
                    cputs("Pada Ujian Sarjana Jurusan Teknik
                    Elektro\n");
                    gotoxy(18,8);
                    cputs("Fakultas Teknik Universitas Kristen
                    Petra\n");
                    getch();
                    LoadScreen(2);
                    break;
                }
                case 2 :
                {   SaveScreen(2);
                    MakeBox(5,18,75,24,BLUE,1,1,"Author");
                    ViewCursor();
                    do {}while(getch()!=13);
                    LoadScreen(2);
                    break;
```

```
                    }
                }
                LoadScreen(1);
            }
            while(k!=0);
            break;
        }
    }
}
while (ff!=5);
ViewCursor();
textattr(0x0F);
clrscr();
return 0;
}

BarMenu(char **vector,int x,int y,char option,int boxwarna,
        int barwarna,int max,char *judul)
{   int  i;
    char ch;

    HideCursor();
    if(option<=0) option=1;
    MakeBox(x,y,x+strlen(*vector)+1,y+max+3,boxwarna,0,1,judul);
    textattr(boxwarna);
    window(1,1,80,25);
    for(i=0;i<max;++i)
    {   gotoxy(x+1,y+3+i);
        cputs(*(vector+i));
    }
    textattr(barwarna);
    gotoxy(x+1,y+2+option);
    cputs(*(vector+(option-1)));
    do
    {   ch=getch();
        switch(ch)
        { case 72 :
            textattr(boxwarna);
            gotoxy(x+1,y+2+option);
            cputs(*(vector+(option-1)));
            if(option==1) option=max;
            else --option;
            textattr(barwarna);
            gotoxy(x+1,y+2+option);
            cputs(*(vector+(option-1)));
            break;
          case 80 :
            textattr(boxwarna);
            gotoxy(x+1,y+2+option);
            cputs(*(vector+(option-1)));
            if(option==max) option=1;
            else ++option;
            textattr(barwarna);
            gotoxy(x+1,y+2+option);
            cputs(*(vector+(option-1)));
            break;
        }
    }
    while(ch!=13 && ch!=27);
    if(ch==27)option=0;
    return(option);
}
```

```
MakeBox(int x1,int y1,int x2,int y2,int warna,int jenis,
        int varjudul,char *judul)
{    int  i,tengah;
     char leftup [4] = {'+','+','+','+'};
     char linehor[4] = {'-','-','-','-'};
     char rightup[4] = {'+','+','+','+'};
     char linever[4] = {'¦','¦','¦','¦'};
     char leftdn [4] = {'+','+','+','+'};
     char rightdn[4] = {'+','+','+','+'};
     char lefttl [4] = {'+','¦','¦','¦'};
     char righttl[4] = {'¦','¦','¦','¦'};

     if (jenis<0 || jenis>3) jenis=0;
     textattr(warna);
     window(x1,y1,x2,y2);
     clrscr();
     window(1,1,80,25);
     for (i=x1+1;i<=x2-1;++i)
     {   gotoxy(i,y1);
         cprintf("%c",linehor[jenis]);
         gotoxy(i,y2);
         cprintf("%c",linehor[jenis]);
     }
     for (i=y1+1;i<=y2-1;++i)
     {    gotoxy(x1,i);
          cprintf("%c",linever[jenis]);
          gotoxy(x2,i);
          cprintf("%c",linever[jenis]);
     }
     gotoxy(x1,y1);
     cprintf("%c",leftup[jenis]);
     gotoxy(x2,y1);
     cprintf("%c",rightup[jenis]);
     gotoxy(x1,y2);
     cprintf("%c",leftdn[jenis]);
     gotoxy(x2,y2);
     cprintf("%c",rightdn[jenis]);
     if (varjudul==1)
     {    for(i=x1+1;i<=x2-1;++i)
          {    gotoxy(i,y1+2);
               cprintf("%c",linehor[jenis]);
          }
          gotoxy(x1,y1+2);
          cprintf("%c",lefttl[jenis]);
          gotoxy(x2,y1+2);
          cprintf("%c",righttl[jenis]);
          window(x1,y1,x2,y2);
          tengah=((int)((x2-x1+1)/2))-((int)(strlen(judul)/2));
          gotoxy(tengah,2);
          cputs(judul);
          window(x1+1,y1+3,x2-1,y2-1);
     }
     else
     {    gotoxy(x1,y1+2);
          cprintf("%c",linever[jenis]);
          gotoxy(x2,y1+2);
          cprintf("%c",linever[jenis]);
          window(x1+1,y1+1,x2-1,y2-1);
     }
}

SaveScreen(int layarke)
{    pixel far *source,*dest;
     int i;
```

```
     if(layarke<0) layarke=0;
     if(layarke>10) layarke=10;
     source=display;
     if(layarke<=3) dest=&display[layarke*2048];
     else dest=&stack_disp[(layarke-4)*2000];
     for(i=0;i<=1999;i++) *(dest++)=*(source++);
}

LoadScreen(int layarke)
{    pixel far *source,*dest;
     int   i;

     if(layarke<0) layarke=0;
     if(layarke>10) layarke=10;
     source=display;
     if(layarke<=3) dest=&display[layarke*2048];
     else dest=&stack_disp[(layarke-4)*2000];
     for(i=0;i<=(1999);i++) *(source++)=*(dest++);
}

HideCursor()
{    union REGS r;

     r.h.ah=1;
     r.h.ch=7;
     r.h.cl=0;
     int86(0x10,&r,&r);
}

ViewCursor()
{    union REGS r;

     r.h.ah=1;
     r.h.ch=6;
     r.h.cl=7;
     int86(0x10,&r,&r);
}

DisplayFilePCX()
{    FILE *fp,*fcfg;
     char NamaFile[80],Display[80];
     static char results[6][16] = {  "Ok",
                                     "Bad file",
                                     "Bad read",
                                     "Memory error",
                                     "Too few colours",
                                     "No driver mode" };
     char path[80];
     int r;

     if((fcfg=fopen("maria.cfg","rb"))!=NULL)
     {   fgets(Display,80,fcfg);
         fclose(fcfg);
         if((gd=LoadDriver(Display))== NULL)
         {    printf("Error loading driver %s\n",Display);
              return(1);
         }
         window(7,21,74,23);
         gotoxy(1,2);puts("File Name ?");
         gotoxy(13,2);
         gets(NamaFile);
         if((fp = fopen(NamaFile,"rb")) != NULL)
         {    SaveScreen(3);
              r=UnpackPCXFile(fp,&fi);
```

```
            MakeBox(1,1,40,4,MAGENTA,2,0,"");
            printf("\%s",results[r]);getch();
            LoadScreen(3);
            fclose(fp);
         }
         else printf("Error opening %s",NamaFile);
      }
      else printf("Please Setup");
}

UnpackPCXFile(fp,fi)
      FILE *fp;
      FILEINFO *fi;
{     PcxHeader pcx;
      char *p,*pr;
      int i,j,bytes;

      if(fread((char *)&pcx,1,sizeof(PcxHeader),fp) == sizeof(PcxHeader)
      &&  pcx.Manufacturer==10)
      {  if(pcx.Bits_per_Pixel==8)
         {  fi->Width=pcx.Xmax-pcx.Xmin+1;
            fi->Depth=pcx.Ymax-pcx.Ymin+1;
            bytes=pcx.Bytes_per_Line;
            if (pcx.Colour_Planes==3)
            {  fi->bits=24;
               fi->bytes=bytes*3;
               if((p=dosalloc(fi->bytes)) == NULL) return(MEMORY_ERROR);
               if((pr=dosalloc(fi->bytes)) == NULL)
               {  dosfree(p);
                  return(MEMORY_ERROR);
               }
               if(Init(fi) != GOOD_READ)
               {  dosfree(pr);
                  dosfree(p);
                  return(MEMORY_ERROR);
               }
               for(i=0;i<fi->Depth;++i)
               {  for(j=0;j<3;++j)
                  {  if(ReadPCXLine(p+(j*bytes),fp,bytes) != bytes)
                     {  freebuffer();
                        dosfree(pr);
                        dosfree(p);
                        return(BAD_READ);
                     }
                  }
                  for(j=0;j<fi->Width;++j)
                  {     pr[j*3]=p[j];
                        pr[j*3+1]=p[bytes+j];
                        pr[j*3+2]=p[2*bytes+j];
                  }
                  putline(pr,i);
               }
               DisplayPicture(fi);
               freebuffer();
               dosfree(pr);
               dosfree(p);}
               else
               {  fi->bits=8;
                  fi->bytes=bytes;
                  if(!fseek(fp,-769L,SEEK_END))
                  {  if(fgetc(fp)!=0x0c ||
                     fread(fi->Palette,1,768,fp)!=768))
                     { puts ("Error reading palette");exit(1);
                     }
```

```
                }
                else
                { puts("Error seeking palette");
                  exit(1);
                }
                if((p=dosalloc(bytes))==NULL) return(MEMORY_ERROR);
                if(Init(fi)!=GOOD_READ)
                {  dosfree(p);
                   return(MEMORY_ERROR);
                }
                fseek(fp,128L,SEEK_SET);
                for (i=0;i<fi->Depth;++i)
                {  if(ReadPCXLine(p,fp,bytes)!=bytes)
                   {  freebuffer();
                      dosfree(p);
                      return(BAD_READ);
                   }
                   putline(p,i);
                }
                DisplayPicture(fi);
                freebuffer();
                dosfree(p);
             }
          }
          else return(WRONG_BITS);
       }
       else return(BAD_READ);
       return(GOOD_READ);
}

ReadPCXLine(p,fp,bytes)
char *p;
FILE *fp;
int bytes;
{    int n=0,c,i;

     do
     {    c=fgetc(fp) & 0xff;
          if((c & 0xc0) == 0xc0)
          {    i=c & 0x3f;
               c=fgetc(fp);
               while(i--) p[n++]=c;
          }
          else p[n++]=c;
     }
     while(n < bytes);
     return(n);
}

Init(fi)
FILEINFO *fi;

{    if(!getbuffer((long)fi->bytes*(long)fi->Depth,fi->bytes,fi->Depth))
     return(MEMORY_ERROR);
     if(gd->RGBOn==NULL || gd->VGAOn==NULL)
     {    freebuffer();
          return(NO_DRIVER);
     }
     if (fi->bits==24)(gd->RGBOn)(fi->Width,fi->Depth);
     else if (fi->bits==8)
     { (gd->VGAOn)(fi->Width,fi->Depth);
       (gd->VGAPalette)(fi->Palette,256);
     }
     return(GOOD_READ);
```

```
}

DisplayPicture(fi)
     FILEINFO *fi;
{    int screenwide,screendeep;
     int c,i,n,m,x=0,y=0;

     if (fi->bits==24)
     {  screenwide=gd->RGBWide;
        screendeep=gd->RGBDeep;
     }
     else
     {  screenwide=gd->VGAWide;
        screendeep=gd->VGADeep;
     }
     if(fi->Width < screenwide)
     {    fi->Xdisp=(screenwide-(fi->Width))/2;
          n=fi->Width;
     }
     else
     {    fi->Xdisp=0;
          n=screenwide;
     }
     if(fi->Depth < screendeep)
     {    fi->Ydisp=(screendeep-(fi->Depth))/2;
          m=fi->Depth;
     }
     else
     {    fi->Ydisp=0;
          m=screendeep;
     }
     do
     {  for(i=0;i<m;++i)
        {    c=y+i;
             if(c>=fi->Depth) break;
             if (fi->bits==24)(gd->RGBLine) (getline(y+i)+(x*3), n,
             i+fi->Ydisp,fi->Xdisp);
             else (gd->VGALine)(getline(y+i)+x,n, i+fi->Ydisp,
             fi->Xdisp);
        }
        c=GetKey();
        switch(c)
        {  case PanahKiri:
             if((x-Step) > 0) x-=Step;
             else x=0;
             break;
           case PanahKanan:
             if((x+Step+screenwide) < fi->Width) x+=Step;
             else if(fi->Width > screenwide)
             x=fi->Width-screenwide;
             else x=0;
             break;
           case PanahAtas:
             if((y-Step) > 0) y-=Step;
             else y=0;
             break;
           case PanahBawah:
             if((y+Step+screendeep) < fi->Depth) y+=Step;
             else if(fi->Depth > screendeep)
             y=fi->Depth-screendeep;
             else y=0;
             break;
           case Home:
             x=y=0;
```

```
            break;
         case End:
            if(fi->Width > screenwide)  x=fi->Width-screenwide;
            else x=0;
            if(fi->Depth > screendeep)  y=fi->Depth-screendeep;
            else y=0;
            break;
      }
   }
   while(c != 27);
   freebuffer();
   if (fi->bits==24)(gd->RGBOff)();
   else (gd->VGAOff)();
   return(GOOD_READ);
}

GetKey()
{    int c;

   c = getch();
   if(!(c & 0x00ff)) c = getch() << 8;
   return(c);
}

GRAFDRIVER *LoadDriver(s)
char *s;
{    GRAFDRIVER *gd;
     char *grafDriver=NULL;
     FILE *fp;
     long l;
     char b[8];
     unsigned int seg;

     if((fp=fopen(s,"rb")) != NULL)
     {   fseek(fp,0L,SEEK_END);
         l=ftell(fp);
         rewind(fp);
         fread(b,1,8,fp);
         if(!memcmp(b,"23491047",8))
         {   l-=8L;
             if(l < 0xffffL &&
             (grafDriver=dosalloc((unsigned int)l)) != NULL)
             {   if(fread(grafDriver,1,(unsigned int)l,fp)
                 ==(unsigned int)l)
                 {   gd=(GRAFDRIVER *)grafDriver;
                     seg=FP_SEG(grafDriver);
                     if(gd->RGBOn != NULL)
                     {   grafDriver[2]=seg;
                         grafDriver[3]=(seg >> 8);
                         grafDriver[6]=seg;
                         grafDriver[7]=(seg >> 8);
                         grafDriver[10]=seg;
                         grafDriver[11]=(seg >> 8);
                     }
                     if(gd->VGAOn != NULL)
                     {   grafDriver[14]=seg;
                         grafDriver[15]=(seg >> 8);
                         grafDriver[18]=seg;
                         grafDriver[19]=(seg >> 8);
                         grafDriver[22]=seg;
                         grafDriver[23]=(seg >> 8);
                         grafDriver[26]=seg;
                         grafDriver[27]=(seg >> 8);
                     }
```

```
                    }
                    else
                    {    dosfree(grafDriver);
                         grafDriver=NULL;
                    }
               }
          }
          fclose(fp);
     }
     return((GRAFDRIVER *)grafDriver);
}

Sampling()
{    FILE *fcfg,*Output;
     char Display[80],FileName[60],path[80],a;
     int r;

      if((fcfg=fopen("maria.cfg","rb"))!=NULL)
      {   fgets(Display,80,fcfg);
          fclose(fcfg);
          if((gd=LoadDriver(Display))== NULL)
          {    printf("Error loading driver %s\n",Display);
               return(1);
          }
          outportb(0x300,0);
          puts("Press C or c to capture");
          do
          {   a=getch();
          }
          while(a !='c' && a !='C');
          r=capture(&fi);
          outportb(0x300,0);
          printf("%d",r);
          puts("Save picture?");
          a=getch();
          if(a=='Y' || a=='y')
          {   puts("Nama file?");
              gets(path);
              FileExt(path,FileName,"PCX");
              strupr(FileName);
              if((Output = fopen(FileName,"wb")) !=NULL)
              {    WritePcxFile(Output,fi.Width,fi.Depth,fi.bits);
                   fclose(Output);
              }
          }
          freebuffer();
      }
      else printf("Please setup");
}

capture(fi)
FILEINFO *fi;
{     char *pr,*pg,*pb,*p1,*p2,*p3;
      int i,j,k,bytes;

       bytes=416;
       fi->Width=416;
       fi->Depth=312;
       fi->bits=24;
       if(fi->Width < gd->RGBWide)
       fi->Xdisp=(gd->RGBWide-(fi->Width))/2;
       else fi->Xdisp=0;
       if(fi->Depth < gd->RGBDeep)
       fi->Ydisp=(gd->RGBDeep-(fi->Depth))/2;
```

```
        else fi->Ydisp=0;
        fi->bytes=bytes*3;
        if((p1=dosalloc(fi->bytes)) == NULL)
        {    return(2);
        }
        if((p2=dosalloc(fi->bytes)) == NULL)
        {    dosfree(p1);
             return(3);
        }
        if((p3=dosalloc(fi->bytes)) == NULL)
        {    dosfree(p1);
             dosfree(p2);
             return(4);
        }
        if(Init(fi) != GOOD_READ)
        {    dosfree(p1);
             dosfree(p2);
             dosfree(p3);
             return(5);
        }
        for(k=0;k<3;++k)
        {   outportb(0x0300,3);
            outportb(0x0300,1);
            for(i=0;i<fi->Depth/3;++i)
            {  for(j=0;j<fi->Width*3;++j)
               {    p1[j]=inportb(0x301+k);
               }
               putline(p1,3*i+k);
            }
        }
        for(i=0;i<fi->Depth/3;++i)
        {    pr=getline(3*i);
             pg=getline(3*i+1);
             pb=getline(3*i+2);
             for(j=0;j<fi->Width;++j)
             {    p1[3*j]=pr[j];
                  p1[3*j+1]=pg[j];
                  p1[3*j+2]=pb[j];
                  p2[3*j]=pr[bytes+j];
                  p2[3*j+1]=pg[bytes+j];
                  p2[3*j+2]=pb[bytes+j];
                  p3[3*j]=pr[2*bytes+j];
                  p3[3*j+1]=pg[2*bytes+j];
                  p3[3*j+2]=pb[2*bytes+j];
             }
             putline(p1,3*i);
             putline(p2,3*i+1);
             putline(p3,3*i+2);
        }
        dosfree(p1);
        dosfree(p2);
        dosfree(p3);
        DisplayPicture(fi);
        return(GOOD_READ);
}

FileExt(Name1,Name2,Ext)
char *Name1,*Name2,*Ext;
{    while(*Name1 != 0 && *Name1 != '.')
     *Name2++=*Name1++;     *Name2++='.';
     while(*Ext) *Name2++=*Ext++;
     *Name2=0;
}
```

```
#pragma warn -par
WritePcxFile(Out,Width,Depth,Bits)
FILE          *Out;
unsigned int Width,Depth,Bits;

{     PcxHeader PcxH;
      int       d,w,Bytes;
      char      *MemBytes,*MemWidth;

      if(Bits != 24)
      {     puts("This isn't a 24 bit picture");
            return;
      }
      if((MemWidth=dosalloc(Width)) == NULL)
      {     puts("Memory error");
      return;
      }
      memset((char *)&PcxH,0,sizeof(PcxHeader));
      PcxH.Manufacturer=10;
      PcxH.Version=5;
      PcxH.Encoding=1;
      PcxH.Bits_per_Pixel=8;
      PcxH.Xmin=fi.Xdisp;
      PcxH.Ymin=fi.Ydisp;
      PcxH.Xmax=fi.Xdisp+Width-1;
      PcxH.Ymax=fi.Ydisp+Depth-1;
      PcxH.Colour_Planes=3;
      PcxH.Bytes_per_Line=Width;
      fwrite((char *)&PcxH,1,sizeof(PcxHeader),Out);
      for(d=0;d<Depth;++d)
      {     MemBytes=getline(d);
            for(w=0;w<Width;++w) MemWidth[w]=MemBytes[w*3];
            WritePcxLine(MemWidth,Width,Out);
            for(w=0;w<Width;++w) MemWidth[w]=MemBytes[w*3+1];
            WritePcxLine(MemWidth,Width,Out);
            for(w=0;w<Width;++w) MemWidth[w]=MemBytes[w*3+2];
            WritePcxLine(MemWidth,Width,Out);
      }
      dosfree(MemWidth);
      if(ferror(Out)) puts("Error writing file");
}

WritePcxLine(p,n,fp)
char *p;
int  n;
FILE *fp;
{     unsigned int i=0,j=0,t=0;

      do
      {     i=0;
            while((p[t+i]==p[t+i+1]) && ((t+i) < n) && (i < 63))++i;
            if(i>0)
            {     fputc(i | 0xc0,fp);
                  fputc(p[t],fp);
                  t+=i;
                  j+=2;
            }
            else
            {     if(((p[t]) & 0xc0)==0xc0)
                  {     fputc(0xc1,fp);
                        ++j;
                  }
            fputc(p[t++],fp);
            ++j;
```

```
        }
    }
    while(t<n);
    return(ferror(fp));
}
```

Penjelasan program :

```
BarMenu(char **vector,int x,int y,char option,int boxwarna,
        int barwarna,int max,char *judul)
 - Membuat pull down menu.

MakeBox(int x1,int y1,int x2,int y2,int warna,int jenis,
        int varjudul,char *judul)
 - Membuat kotak pada koordinat tertentu.

SaveScreen(int layarke)
 - Menyimpan tampilan layar ke memori.

LoadScreen(int layarke)
 - Mengembalikan layar pada tampilan sebelumnya.

HideCursor()
 - Cursor tidak ditampilkan.

ViewCursor()
 - Menampilkan kembali cursor.

DisplayFilePCX()
 - Menampilkan file PCX 256 warna dan 16.7 juta warna dari disk.

UnpackPCXFile(fp,fi)
 - Men-decode file PCX.

ReadPCXLine(p,fp,bytes)
 - Membuka kompresi data 1 baris menjadi data dalam urutan RGB.

Init(fi)
 - Menyiapkan monitor dengan mode layar yang diinginkan.

DisplayPicture(fi)
 - Menampilkan gambar yang sudah di-load ke memori komputer.

GRAFDRIVER *LoadDriver(s)
 - Me-load driver SVGA card (Tseng ET 4000).

Sampling()
 - Data RGB analog disampling oleh ADC dan ditulis ke RAM ekster-
   nal (HM628128).

capture(fi)
 - Proses sampling dihentikan, data pada RAM eksternal ditransfer
   ke memori komputer.

FileExt(Name1,Name2,Ext)
 - Secara otomatis menambahkan ekstension PCX ke nama file.

WritePcxFile(Out,Width,Depth,Bits)
 - Menulis file PCX.

WritePcxLine(p,n,fp)
 - Memadatkan data 1 baris dengan teknik RLE.
```