III. PERENCANAAN

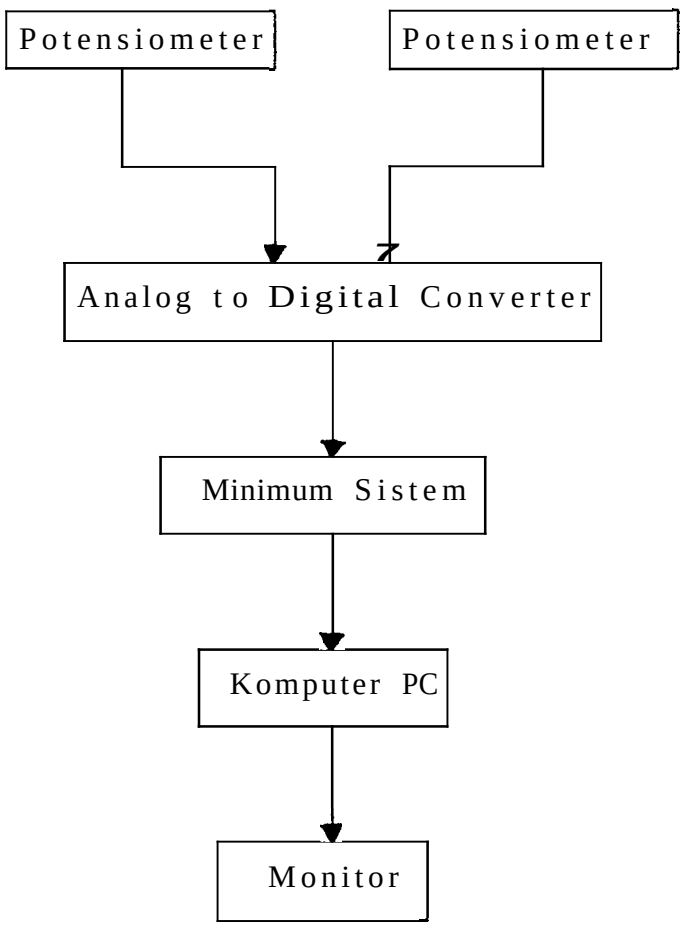
1. PERENCANAAN SISTEM

Pada perencanaan pembuatan alat ini, secara umum dibagi menjadi dua bagian, yaitu perencanaan perangkat keras (hardware) dan perencanaan perangkat lunak (software).

Perencanaan perangkat keras meliputi perencanaan minimum sistem 80C31, perencanaan rangkaian Analog to Digital Converter (ADC) dan perencanaan mekanik dalam hubungannya dengan sensor potensiometernya. Blok diagram dari perangkat keras dapat dilihat pada gambar **3-1.**

Sensor potensiometer yang berfungsi untuk menentukan posisi pena berjumlah dua buah dimasukkan ke dalam 'rangkaian ADC. Rangkaian ADC ini kemudian dihubungkan ke minimum sistem 80C31 untuk memperoleh data digital dari potensiometer tersebut. Selanjutnya dengan komunikasi serial data dari minimum sistem dikirim ke komputer untuk diolah. Hasil pengolahan data

ini akan diperoleh posisi pena. Kemudian berdasarkan data tersebut maka dapat ditampilkan pada monitor komputer dengan posisi yang sesuai dengan posisi pena pada kertas gambar.



GAMBAR 3-1

BLOK DIAGRAM PERANGKAT KERAS

Perencanaan perangkat lunak meliputi perencanaan program mikrokontroller 80C31 untuk mengambil data dari Analog to Digital Converter (ADC) dan kemudian dikirim ke komputer melalui port serial yang terdapat pada

komputer (COM1 atau COM2). Selanjutnya dengan menggunakan bahasa pemrograman Turbo Pascal, data dari port serial tersebut diambil kemudian diolah sehingga menjadi gambar pada monitor komputer. Tersedia pula fasilitas penggambaran berwarna sampai dengan maksimum 16 warna dan penyimpanan gambar dalam bentuk file.

2. PERENCANAAN PERANGKAT KERAS

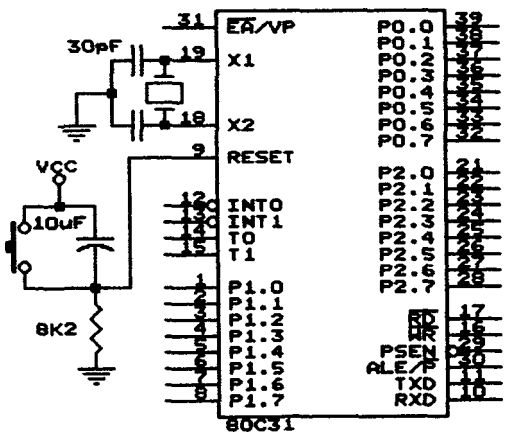
2.1 Perencanaan Minimum Sistem 8ØC31

Perencanaan minimum sistem 8ØC31 meliputi rangkaian pembangkit sinyal clock dan rangkaian reset.

2.1.1 Rangkaian Clock. Rangkaian clock yang digunakan dalam minimum sistem 8ØC31 ini merupakan rangkaian clock yang dianjurkan oleh Intel Co. selaku pembuat mikrokontroller. Rangkaian clock ini hanya membutuhkan satu buah kristal dan dua buah kapasitor untuk membentuk rangkaian clock yang akan digunakan oleh mikrokontroller 8ØC31.

Nilai kapasitansi dari kapasitor dipakai 3ØpF, nilai ini diambil dari buku data yang menyatakan bahwa nilai kapasitansi yang diperkenankan untuk oscillator kristal adalah 20 pF - 40 pF. Kristal yang digunakan untuk minimum sistem 8ØC31 bernilai 11,059 MHz. Nilai ini

diambil karena frekuensi oscillator juga dipakai sebagai pernbangkit 'baud rate' sehingga harus dipilih suatu frekuensi yang merupakan kelipatan dari nilai 'baud rate' (lihat gambar 2-7). Dengan menggunakan kristal yang tepat 11,Ø59MHz maka 'baud rate' yang dihasilkan juga tepat sesuai dengan yang diinginkan.



GAMBAR 3-2

RANGKAIAN CLOCK DAN RESET

2.1.2 Rangkaian Reset. Rangkaian reset untuk minimum sistem 8ØC31 direncanakan agar mempunyai kemampuan 'power on reset' disamping reset secara manual dengan menggunakan saklar tombol tekan (push button). Untuk dapat memenuhi kedua

hal diatas, harus dipenuhi beberapa syarat, yaitu:

1. Lebar pulsa untuk sinyal reset harus lebih besar dari 24 periode osilator (2 machine cycle ≈ 2μS).
2. 'Rise time' dari VCC tidak melebihi 1 ms.
3. Waktu 'start up' dari oscillator tidak melebihi 10 ms.

Syarat 2 dan 3 hanya berlaku bila 'power on reset´ digunakan.

Rangkaian reset yang digunakan untuk minimum sistem 8ØC31 seperti pada gambar 3-2. Cara kerja dari rangkaian adalah sebagai berikut, bila tegangan supply Vcc mulai diberikan, maka ada arus yang melewati kapasitor sehingga menimbulkan beda tegangan pada tahanan. Tegangan pada pin RST merupakan beda tegangan antara Vcc dengan tegangan pada kapasitor. Karena sifat dari kapasitor maka tegangan pada pin RST akan menurun sehingga akhirnya mencapai tegangan `lower threshold' dari internal schmitt trigger (2,5V). Untuk bisa memberikan efek 'power on reset' maka waktu penurunan tegangan dari Vcc sampai mencapai tegangan `lower threshold´ dari schmitt trigger harus lebih besar dari 2 siklus

rnesin ditambah dengan waktu 'start up' dari osilator.

Saklar tombol tekan digunakan untuk keperluan reset secara manual, bila saklar ini ditekan maka pin RST akan rnendapat tegangan yang setara dengan VCC dan hal ini menyebabkan mikrokontroller berada dalarn keadaan reset.

Dengan asumsi waktu 'start up' dari oscillator 10 ms dan frekuensi oscillator 11,059 **MHz** maka lebar pulsa untuk pin reset harus lebih besar dari (10^{-2} + 24.10^{-6}/11,059) detik atau 10,00217 ms. Dengan menggunakan ketentuan bahwa tegangan 'lower threshold' dari pin RST sebesar 2,5 volt, t(lebar pulsa) sebesar 10,00217 ms akan diperoleh nilai tahanan dan kapasitor dengan menggunakan rurnus :

V_{reset} = Vcc(1-e-t/RC)

2,5 = $5(1-e^{-0.01000217/RC})$

$5e^{-0.01000217/RC}$ = 2,5

$\ln(e^{-0.01000217/RC})$ = ln 0.5

0,01000217/RC = 0,69314718

RC = 14,43 ms.

Terdapat cukup banyak nilai tahanan dan kapasitor yang dapat memenuhi nilai konstanta RC tersebut diatas, tetapi dengan memperhatikan nilai-nilai komponen yang ada di pasaran maka

nilai untuk kapasitor diambil 10µF dan nilai untuk tahanan diambil 8,2 KR. Pengambilan nilai tahanan dan kapasitor tersebut masih dibenarkan karena dengan kedua nilai tersebut didapatkan lebar pulsa reset sebesar 82 ms dan nilai ini lebih besar dari lebar pulsa reset minimum.

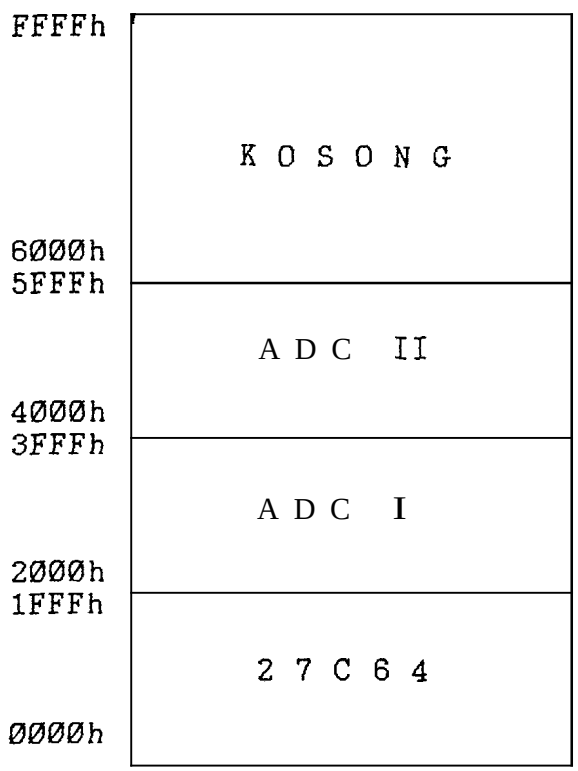
2.2 Rangkaian Dekoder

Untuk menghubungkan mikrokontroller 80C31 dengan piranti-piranti pendukung yang lain, seperti memori, adc, maka dibutuhkan suatu rangkaian dekoder.

Mikrokontroller 80C31 mempunyai dua buah peta alamat memori, yaitu program memori dan data memori. Kedua peta alamat memori tersebut diakses secara terpisah oleh program pengontrol. Pembentukan peta alamat program memori tidak memakai dekoder, karena peta alamat program memori hanya ditempati satu buah EPROM dengan kapasitas 8 Kbyte. EPROM 27C64 ditempatkan pada alamat 0000H-1FFFH.

Dekoder digunakan untuk memberi peta alamat data memori. Dekoder yang digunakan yaitu satu buah IC 74LS138. Untuk memberikan interval 8 Kbyte pada masing-masing outputnya, ketiga input alamat A,B dan C dari IC dekoder 74LS138 masing-

masing dihubungkan dengan A_{13}, A_{14} dan A_{15} dari rnikrokontroller 80C31. Agar IC dekoder selalu berada dalam keadaan aktif maka input `enable' G_1 dihubungkan dengan Vcc, sedangkan kedua input 'enable' yang lain yaitu G_{2A} dan G_{2B} dihubungkan dengan 'ground'. Output dari dekoder (Y_0-Y_2) dihubungkan dengan masing-masing pin CS/CE dari piranti.



GAMBAR 3-3

PETA ALAMAT HEHORI

Dari hubungan ini, ***output*** Y_1 ***akan mengaktifkan*** ADC yang pertama pada **alamat** 2000h-3FFFh,

sedangkan output Y_2 akan rnengaktifkan ADC yang kedua pada alamat 4000h-5FFFh.

TABEL 3.1

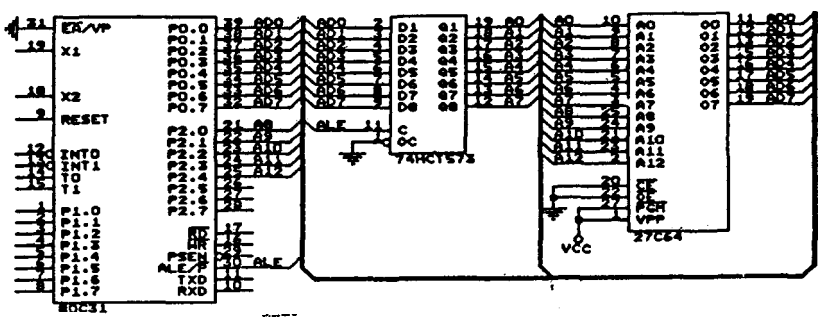
TABEL KEBENARAN DEKODER

			74LS138								
A15	A14	A13	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	DEVICE
0	0	0	1	1	1	1	1	1	1	0	27C64
0	0	1	1	1	1	1	1	1	0	1	ADC I
0	1	0	1	1	1	1	1	0	1	1	ADCII
0	1	1	1	1	1	1	0	1	1	1	
1	0	0	1	1	1	0	1	1	1	1	
1	0	1	1	1	0	1	1	1	1	1	KOSONG
1	1	0	1	0	1	1	1	1	1	1	
1	1	1	0	1	1	1	1	1	1	1	

2.3 Hubungan 80C31 dengan EPROM 27C64

Dalam mode pengalamatan eksternal, port 0 dipakai sebagai jalur alamat bagian bawah (A0-A7) dan jalur data secara bergantian, Sinyal ALE dipakai untuk menandai saat port 0 berfungsi sebagai jalur alamat. Untuk memisahkan data dengan alamat yang ada pada port 0 digunakan sebuah IC 'octal latch' dengan tipe 74HCT573. Dengan rnendapatkan sinyal sinkronisasi ALE dari

mikrokontroller, maka keluaran dari IC latch 74HCT573 akan berisi alamat bagian bawah yaitu byte rendah dari jalur alamat yang bersesuaian dengan EPROM 27C64.



GAMBAR 3-4

HUBUNGAN 8ØC31 DENGAN EPROM 27C64

Informasi jalur alamat byte tinggi (A_8-A_{15}) disediakan oleh mikrokontroller melalui port 2. Sebagian dari jalur alamat tersebut (A_8-A_{12}) dihubungkan dengan jalur alamat 27C64 yang bersesuaian, sedangkan sisanya dihubungkan dengan input dari IC dekoder 74LS138.

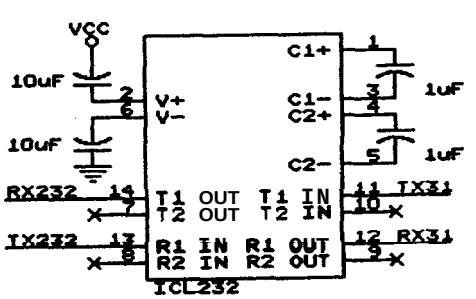
Karena port 0 merupakan jalur alamat sekaligus sebagai jalur data yang prosesnya dilakukan secara multipleks, maka port 0 dapat

langsung dihubungkan dengan jalur data dari 27C64.

Pin CE dari 27C64 dihubungkan langsung ke ground karena hanya menggunakan satu buah EPROM. Dengan demikian EPROM–nya selalu aktif. Pin Vpp dan **PGM** dihubungkan ke **Vcc.**

2.4 Hubungan Mikrokontroller dengan MAX232

Sinyal transmit dan receive dari mikrokontroller 8ØC31 menggunakan level +5V untuk logika '1' dan ØV untuk logika 'O'. **Hal** ini tidak sesuai dengan RS232 dari komputer yang menggunakan tegangan negatif (-12V) untuk logika '1' dan tegangan positif (+12V) untuk logika Θ Untuk itu diperlukan rangkaian yang dapat mengkonversikan sinyal RS232 menjadi level TTL.



GAMBAR 3–5

KONVERSI LEVEL RS232 KE LEVEL TTL

Dengan menggunakan IC MAX232 maka dengan mudah hal tersebut dapat dilakukan. Rangkaian konversinya seperti terlihat pada gambar 3-5.

2.5 Analog To Digital Converter ICL7109CPL

Untuk mengaktifkan ADC ICL-7109CPL perlu perencanaan komponen pendukungnya sebagai berikut:

- Membutuhkan catu daya ± 5 Volt.
- Tegangan Referensi = $\frac{Vin}{2}$.. Untuk tegangan input full scale 4 volt berarti tegangan referensi diatur sehingga menghasilkan V_{ref} = 2 Volt.

Untuk menghasilkan konversi yang linier perlu ditentukan nilai R_{INT} yang sesuai dengan rumus :

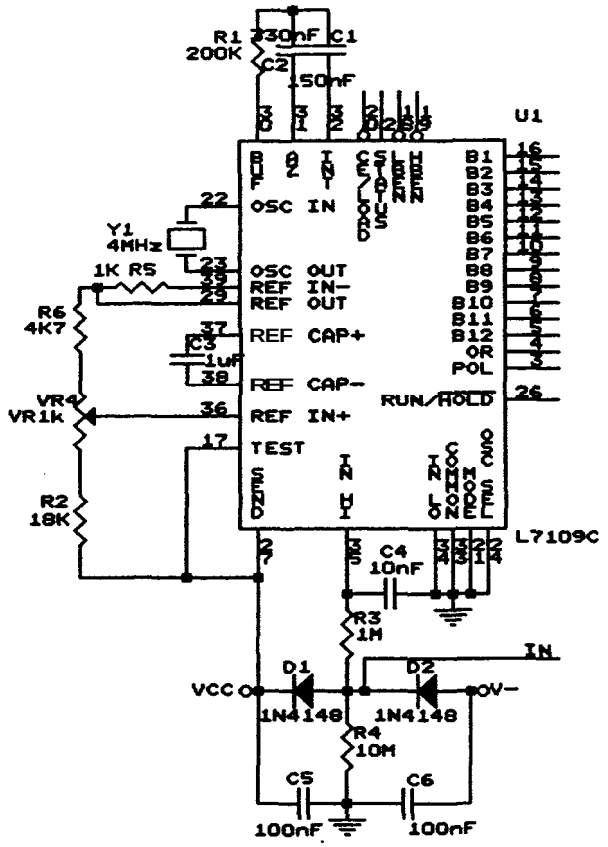
$$R_{INT} = \frac{\text{full scale voltage}}{20\,\mu A}$$

Dalam perencanaan ini digunakan tegangan full scale 4 volt sehingga diperoleh R_{INT} = 200kΩ.

Untuk menghasilkan tegangan V_{ref} = 2 volt digunakan variabel resitor yang diserikan dengan tahanan tetap sehingga tegangannya dapat diatur agar tepat 2 volt.

Fungsi dioda di sini adalah untuk memproteksi tegangan input agar tidak lebih dari Vcc+Vd atau -Vcc-Vd (Vd=$V_{forward}$ dioda = 0.7 volt). Sedangkan fungsi resistor 1MΩ dan 10MΩ adalah

untuk menjaga agar impedansi input analognya sama dengan impedansi input pada ADC itu sendiri.



GAHBAR 3-6

RANGKAIAN ADC ICL7109CPL

Untuk memaksimumkan kekebalan terhadap **noise,** maka semua input-input digital pada ADC yaitu pin-pin CE,LBEN,HBEN dan pin RUN/$\overline{\text{HOL}}$ diberikan tahanan 'pull-up' yang nilainya 4K7Ω. Rangkaian

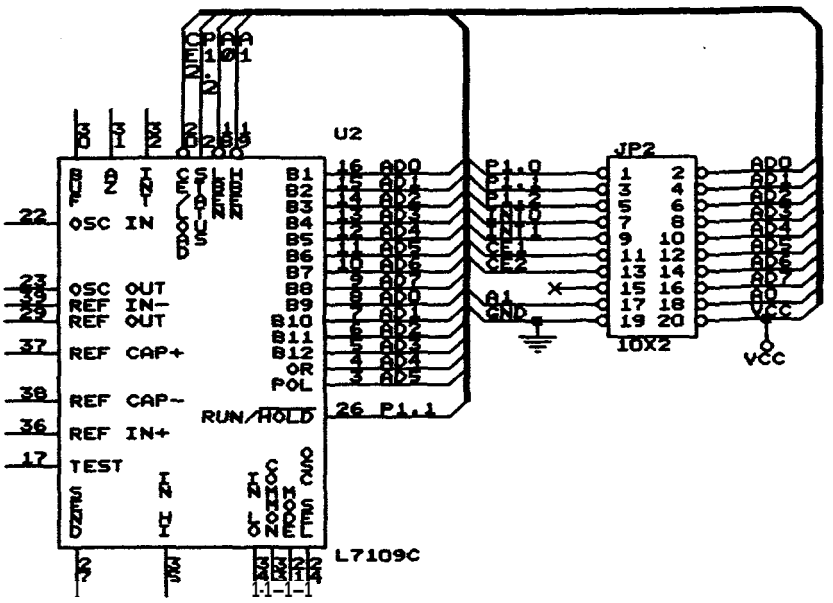
dari ADC ICL7109CPL ini dapat dilihat pada garnbar 3-6.

2.6 Hubungan Mikrokontroller dengan ADC

Untuk mengontrol proses konversinya dilakukan pengecekan pada pin RUN dan pin STATUS dari ADC. Apabila output pin STATUS berlogika 0 berarti proses konversi analog ke digital telah selesai dan data siap untuk dibaca. Apabila output STATUS masih berlogika '1' berarti proses konversi belum selesai. Untuk itu pin STATUS dihubungkan dengan port 1 dari 89C31 yaitu P1.0 dan P1.2.

Untuk menjalankan ADC ini yaitu dengan cara mengatur masukan pada pin $\overline{\text{RUN/HOL}}$. Dengan memberi logika '1' pada pin ini maka proses konversi akan mulai berjalan. Apabila proses konversi telah selesai, yang ditandai dengan output pada pin STATUS menjadi 0 maka pin $\overline{\text{RUN/HOL}}$ diberi logika '0' untuk menghentikan sementara proses konversinya. Setelah itu pembacaan data digital dapat dilakukan dengan mengatur pin CE dan LBEN/HBEN sesuai dengan alamat yang diberikan pada masing-masing ADC tersebut. Pengontrolan pada pin $\overline{\text{RUN/HOL}}$ dilakukan oleh port 1.1 dari 89C31. Sedangkan

untuk membaca data **ADC** tersebut digunakan CE1 dan CE2 serta A0 dan A1 untuk memilih byte atas atau byte bawah.



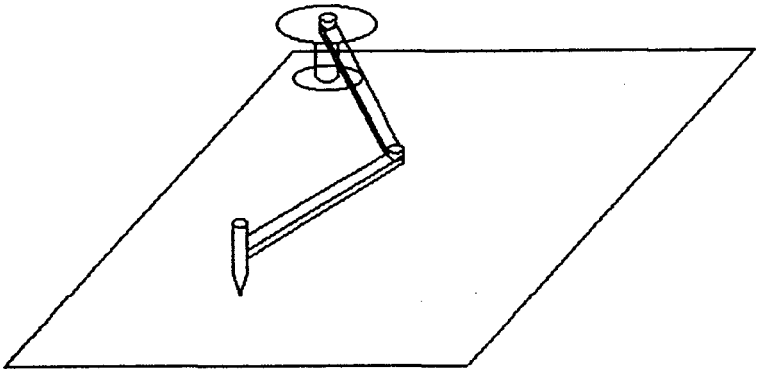
GAMBAR 3-7

HUBUNGAN 8ØC31 DENGAN ADC ICL71Ø9CPL

2.7 Perencanaan Sistem Mekanik

Pada perencanaan sistem mekanik mula-mula ditentukan terlebih dahulu luas bidang gambar yang diinginkan. Agar memudahkan perhitungan maka luas bidang gambar disesuaikan dengan luas bidang gambar yang digunakan pada monitor (640 x 480 pixel) yaitu 64Øcm x 48Øcm. Sistem mekanik yang dibuat berbentuk seperti lengan yang dapat

dilihat pada gambar 3-8. Sistem lengan ini dibandingkan dengan sistem X-Y jauh lebih sederhana karena tidak memerlukan roda gigi-roda gigi seperti yang dibutuhkan pada model X-Y. Namun sistem lengan mempunyai kekurangan dalam hal perhitungan karena menggunakan sistem koordinat polar dimana hal ini sangat bergantung dari ketelitian panjang lengan dan sensor potensiometer.



GAMBAR 3-8

SISTEU MEKANIK ALAT MENGGAMBAR

Panjang lengan merupakan jari-jari yang besarnya sudah tertentu. Putaran potensio yang merubah nilai tahanan dari potensiometer tersebut sebanding dengan perubahan sudut putarannya. Tegangan keluaran dari potensiometer merupakan data posisi analog yang kemudian dimasukkan ke rangkaian ADC sebagai perubah bentuk sinyal dari analog ke bentuk digital.

Perhitungan posisi pena gambar adalah sebagai berikut: mula-mula ditentukan terlebih dahulu posisi dari lengan yang pertama di mana panjangnya adalah R1 dan besar sudutnya adalah A (lihat gambar 3-9). Maka akan diperoleh

Xl = R1 Sin A

Y1 = R1 Cos A

Setelah itu posisi lengan kedua dapat dihitung dengan titik pusat lengan kedua pada ujung dari lengan yang pertama. Putaran potensio pada titik tersebut akan dapat diketahui besar sudut B, maka sudut C dapat dihitung yaitu

C = 180 - (90-A) - B

C = 9Ø + A - B

Dengan transformasi akan diperoleh

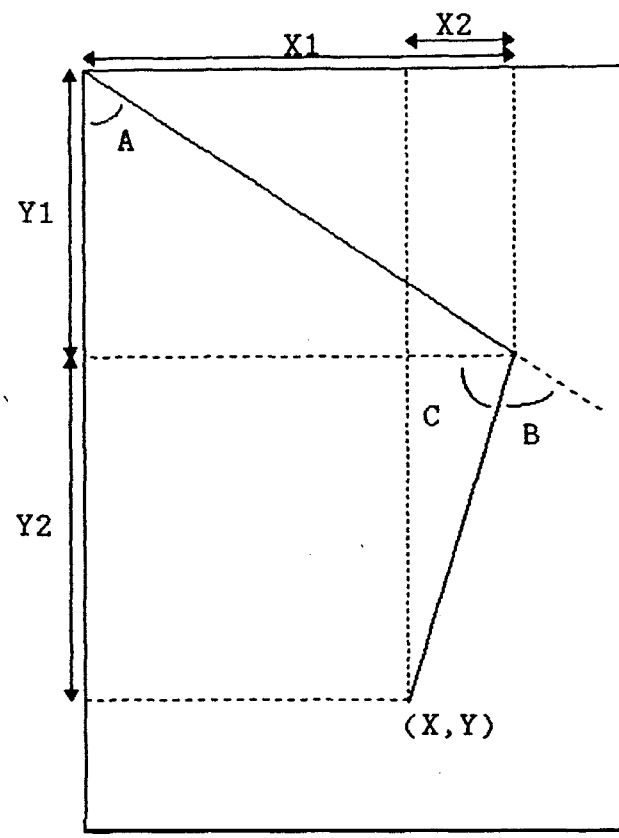
X2 □ R2 Cos C

Y2 = R2 Sin C

Dengan demikian posisi pena dapat diketahui dalam bentuk X-Y sebagai berikut:

X = X1 - X2

Y = Y1 + Y2



GAMBAR 3-9

KONVERSI POLAR KE CARTESIAN

Perencanaan yang diinginkan bisa mendeteksi gambar seluas 640mm X 480mm. Untuk itu dapat dihitung panjang lengan yang merupakan jari-jari lengan penggambar sebagai berikut:

$$R = \sqrt{(640)^2 + (480)^2}$$

$$R = 800\text{mm}$$

Untuk memudahkan penggambaran, posisi lengan diletakkan diatas bidang gambar sehingga panjang

dari lengan harus lebih besar dari 800mm. Dari hasil ini digunakan panjang lengan masing-masing 500mm dan panjang lengan ini cukup untuk menjangkau keseluruhan bidang gambar.

Sistem dirancang agar dapat mendeteksi perubahan satu pixel pada monitor. Dengan menggunakan rumus konversi polar ke X-Y diatas diperoleh :

$X1 = R1 \text{ Sin } A$

$1 = 500 \text{ Sin } A$

$A = \text{Arc Sin } (1/500)$

$A = 0{,}114591635^{o}$

Putaran maksimun potensio adalah 300^{0} sehingga maksimum data digital dari ADC adalah

$\text{Count} = 300/0{,}114591635$

$= 2618$

Untuk dapat mencapai maksimum data digital tersebut diperlukan resolusi ADC minimum sebesar

$2^{n} = 2618$

$\text{Ln } 2^{n} = \text{Ln } 2618$

$n\text{Ln } 2 = \text{Ln } 2618$

$$n = \frac{\text{Ln } 261}{\text{Ln } 2}$$

$n = 11{,}3542$

Jadi, dibutuhkan ADC yang lebih besar dari 11 bit, Sehingga dalam perencanaan diambil ADC dengan resolusi 12 bit.

3. PERENCANAAN PERANGKAT LUNAK

Untuk mengendalikan perangkat keras maka diperlukan perangkat lunak, dalam hal ini perangkat lunak dibuat dalam bahasa assembly 80C31 dan Turbo Pascal.

3.1 Perangkat Lunak Minimum Sistem 80C31

Program yang bekerja pada minimum sistem dengan mikrokontroller 80C31 secara garis besar dapat dibagi menjadi beberapa bagian, yaitu :

- Program inisialisasi dari mikrokontroller 80C31. Program inisialisasi ini meliputi inisialisasi komunikasi serial dengan baud rate, pemindahan stack pointer alamat yang aman dari alamat-alamat yang digunakan oleh interrupt dan register.
- Program untuk menjalankan ADC 7109CPL dan pengambilan data digital dari kedua ADC tersebut.
- Program pengecekan status penggambaran. Program ini melakukan pengecekan apakah data yang diperoleh dari ADC merupakan data untuk menggambar atau memindahkan pena gambar.
- Program komunikasi dengan komputer PC. Program ini digunakan untuk komunikasi secara

serial dengan komputer PC. Data-data digital dari ADC tersebut beserta dengan status penggambarannya dikirim secara serial ke komputer PC. Sistem pengiriman datanya adalah mula-mula dikirim terlebih dahulu kode penggambaran yang diinginkan barulah kemudian data dari ADC. Karena menggunakan ADC dengan resolusi 12 bit maka setiap data ADC dikirim sebanyak 2 byte. Untuk setiap pengiriman data diadakan pengecekan kembali data tersebut sudah benar atau tidak dengan mengirim kembali data tersebut ke minimum sistem. Apabila data yang diterima komputer sudah benar maka akan dikirim kode ØØØ1h, apabila data tersebut masih salah maka akan dikirim kode FFFEh dan akan dikirim kembali data yang salah tersebut dari minimum sistem.

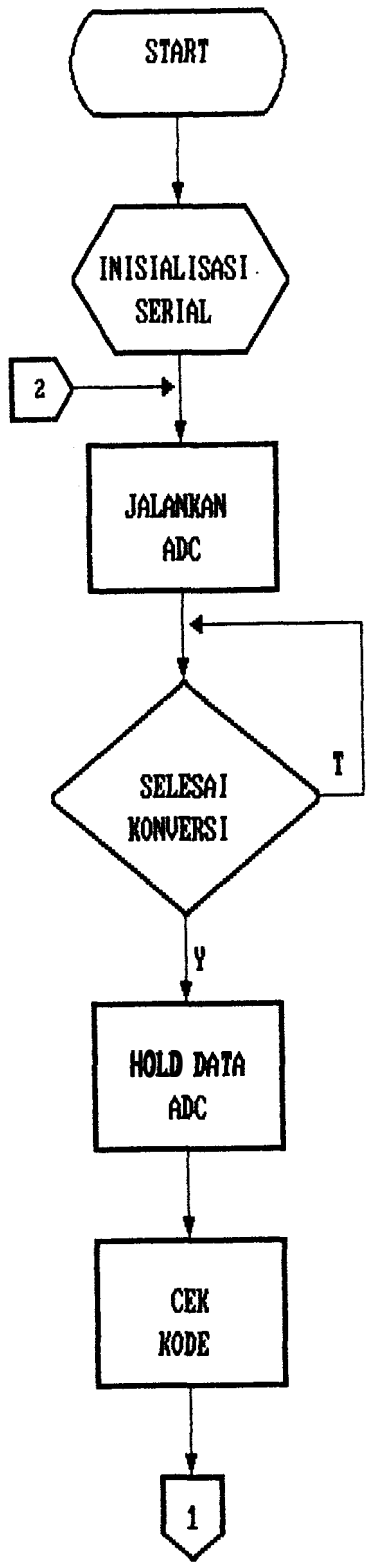
3.2 Perangkat Lunak untuk Komputer PC

Program pada komputer juga dibagi menjadi beberapa bagian yaitu :

- Program inisialisasi komunikasi. Program ini melakukan inisialisasi mengenai baud rate yang digunakan, alamat jalur komunikasi yang diinginkan. Inisialisasi ini harus sesuai dengan inisialisasi pada minimum sistem,

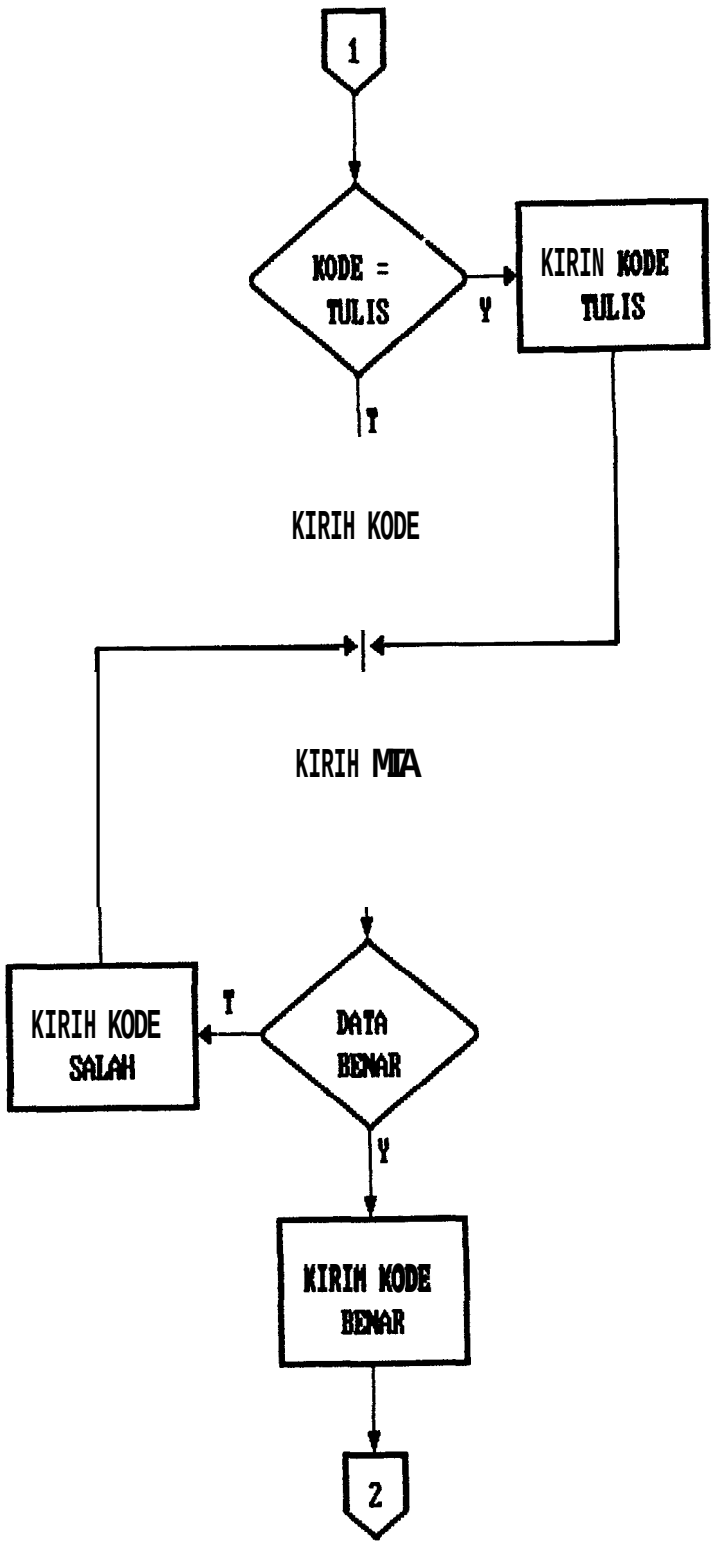
karena apabila tidak sesuai maka penerimaan data akan terjadi kesalahan.

- Program komunikasi. Program ini melakukan proses komunikasi dua arah. Setelah menerima data dari mikrokontroller 8ØC31, program mengirim balik data tersebut yang menandakan bahwa data telah diterima dengan benar dan siap untuk menerima data berikutnya.
- Program konversi data. Setelah data diterima, data tersebut masih berupa data counter 12 bit. Untuk itu dilakukan pemrosesan data tersebut sehingga menjadi besaran sudut dan jari-jari. Kemudian besaran tersebut kemudian diubah lagi menjadi koordinat X–Y. Data inilah yang kemudian siap untuk ditampilkan pada monitor.
- Program penampilan gambar. Data X-Y tersebut kemudian dengan fasilitas grafik dari Turbo Pascal ditampilkan ke monitor VGA. Disamping itu disediakan pula kemampuan penyimpanan gambar ke file dan penggambaran berwarna dengan maksimum jumlah warna adalah 16.
- Program peyimpanan gambar. Hasil gambar pada monitor ini kemudian dapat disimpan ke dalam bentuk file untuk kemudian nantinya dapat ditampilkan kembali.



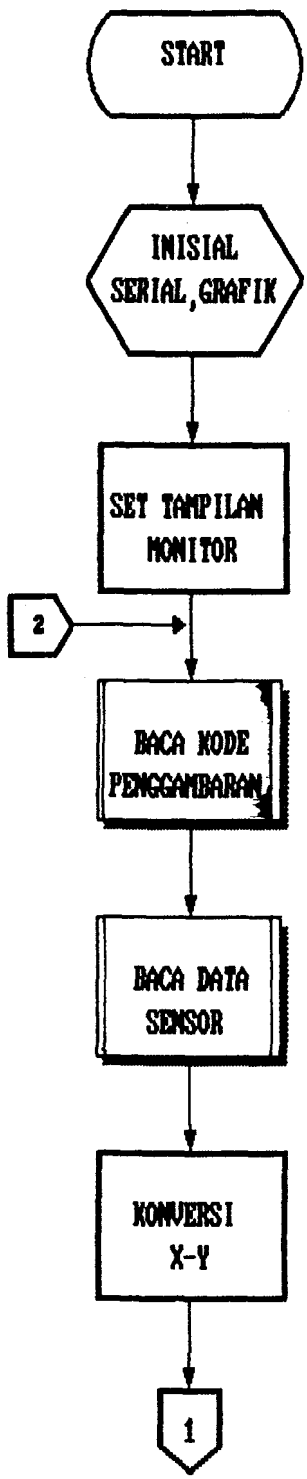
GAMBAR 3-10

FLOWCHART MINIMUM SISTEM



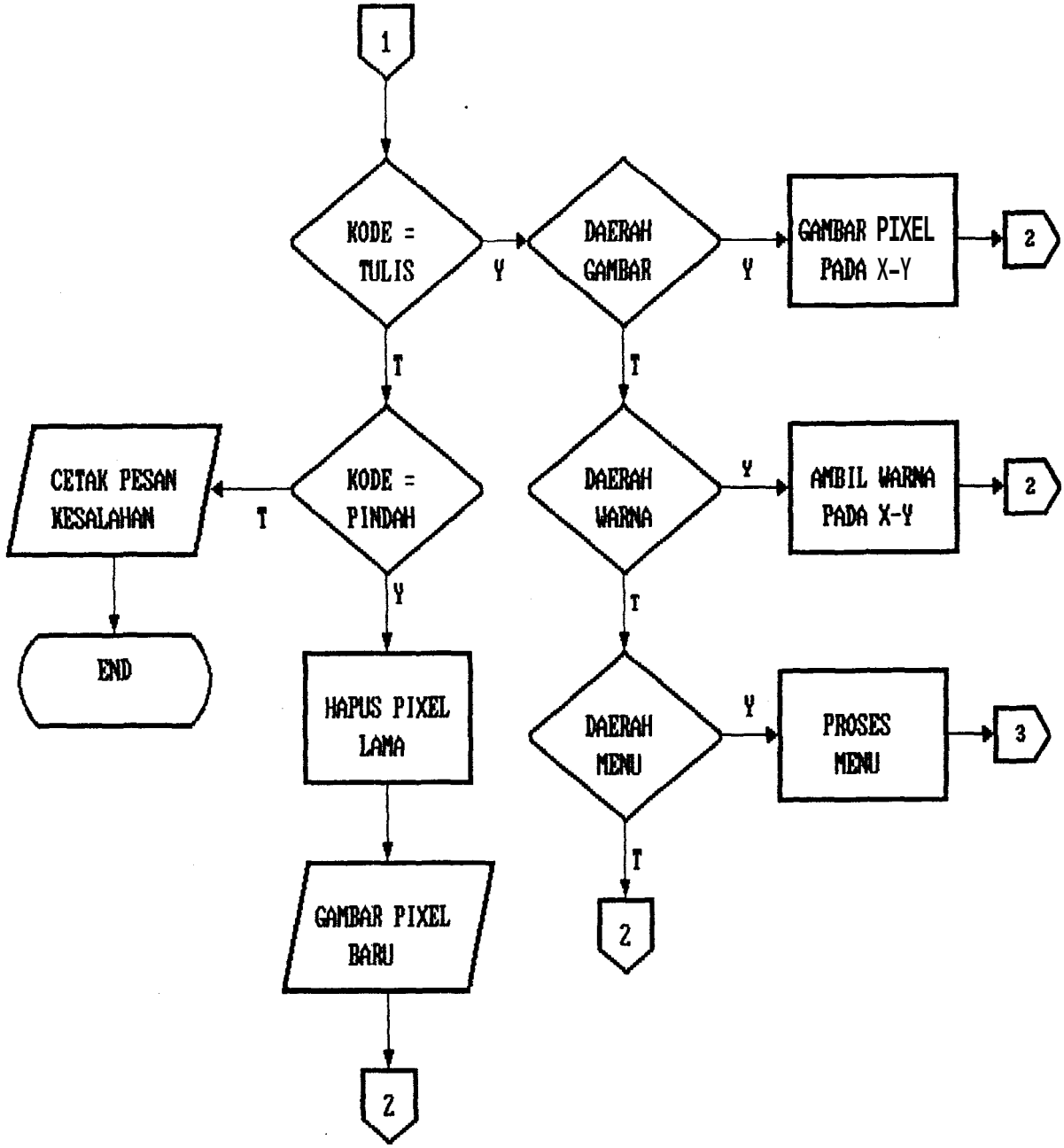
GAMBAR 3-11

FLOWCHART MINIMUM SISTEM



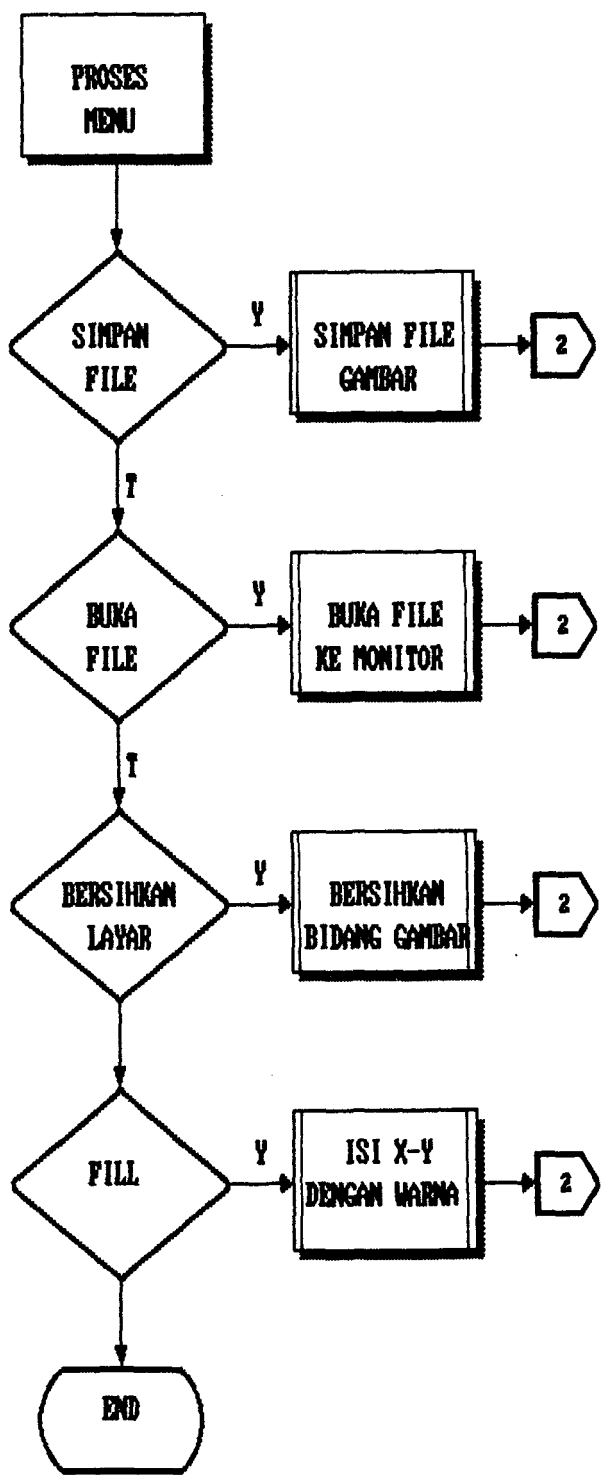
GAMBAR 3-12

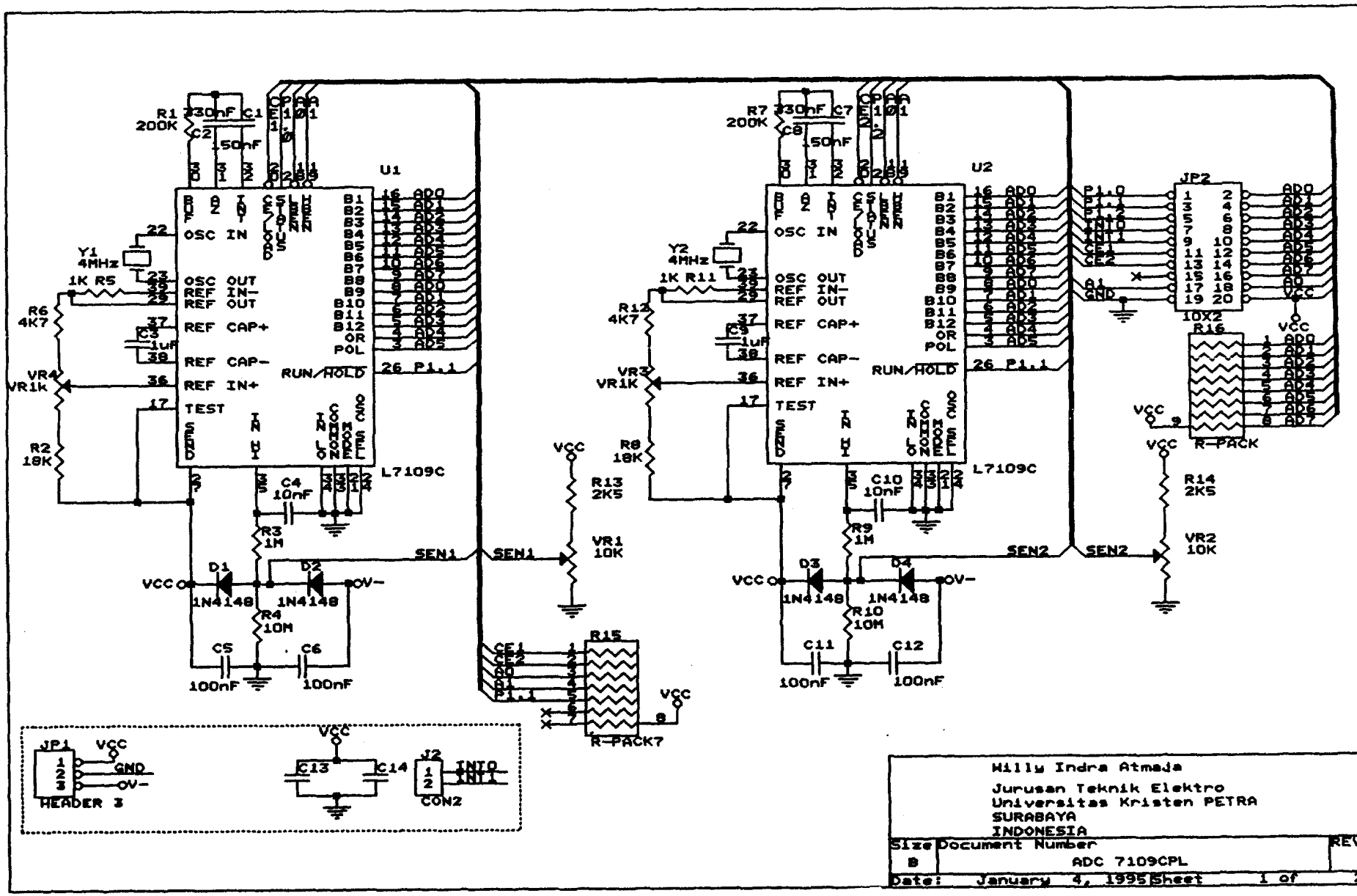
FLOWCHART KOMPUTER



GAMBAR 3-13

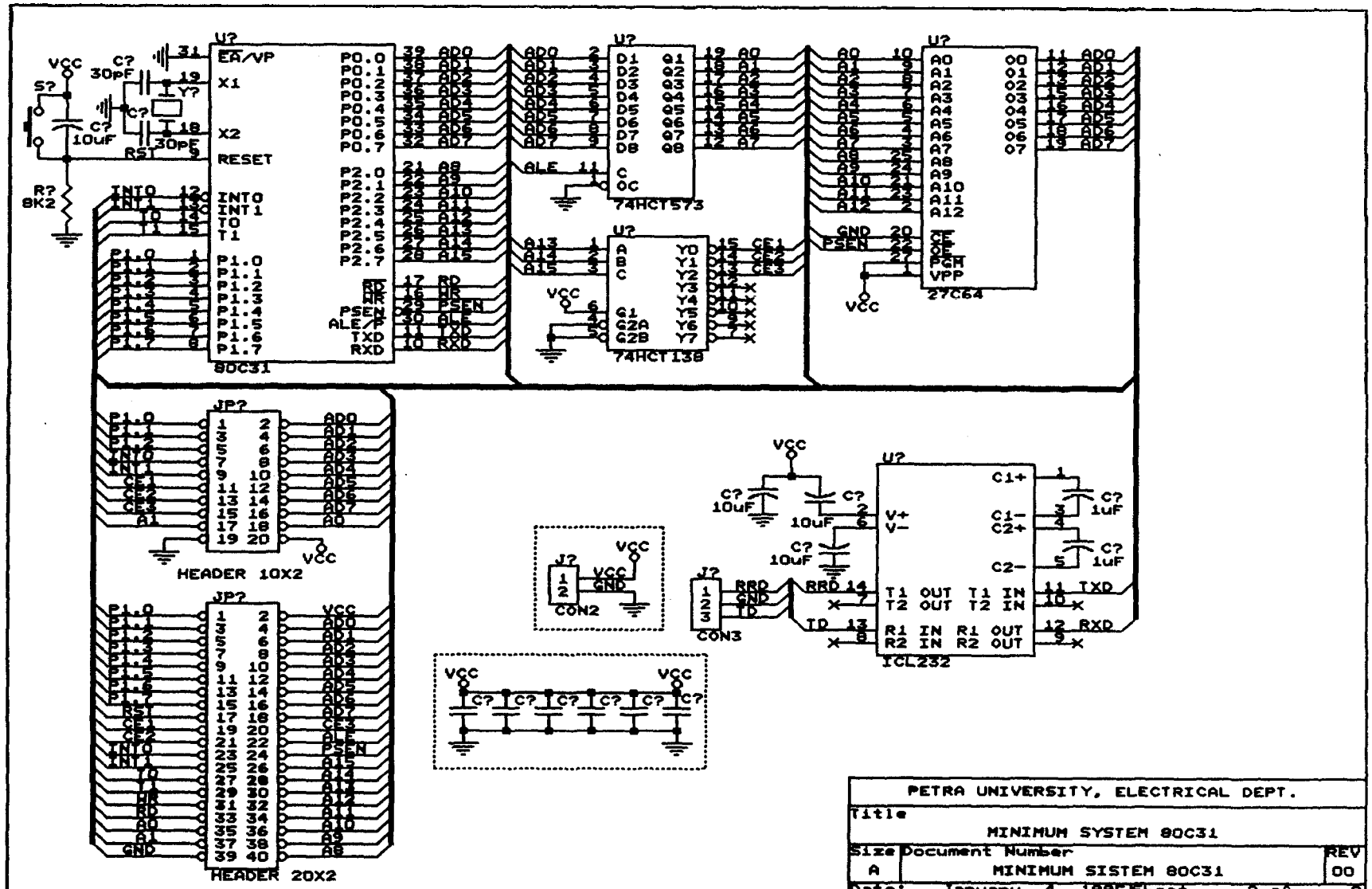
FLOWCHART KOMPUTER


PROSES MENU
SIMPAN FILE
Y
SIMPAN FILE GAMBAR
2
T
BUKA FILE
Y
BUKA FILE KE MONITOR
2
T
BERSIHKAN LAYAR
Y
BERSIHKAN BIDANG GAMBAR
2
FILL
Y
ISI X-Y DENGAN WARNA
2
END



GAMBAR 3-15

RANGKAIAN **ADC ICL7109CPL**



GAMBAR 3-16

RANGKAIAN MINIMUM SISTEM 8ØC31

```
1   ADC1_LO          EQU     4002H
2   ADC1_HI          EQU     4001H
3   ADC2_LO          EQU     2002H
4   ADC2_HI          EQU     2001H
5
6                    ORG     0000H
7                    LJMP    START
8
9                    om      0100H
10  INIT-SERIAL:     MOV     SCON,#01010010B
11                   MOV     TMOD,#00100000B
12                   MOV     TH1,#0FDH
13                   MOV     PCON,#10000000B;baud rate 19200
14                   SETB    TR1
15                   RET
16
17  TRANSMIT:        JNB     TI,$
18                   CLR     TI
19                   MOV     SBUF,A
20                   RET
21
22  RECEIVE:         JNB     RI,$
23                   CLR     RI
24                   MOV     A, SBUF
25                   RET
26
27  ADC1:            MOV     DPTR, #ADC1_HI
28                   MOVX    A,@DPTR
29                   ANL     A,#0FH
30                   LCALL   TRANSMIT
31                   LCALL   RECEIVE
32                   MOV     DPTR,#ADC1_LO
33                   MOVX    A,@DPTR
34                   LCALL   TRANSMIT
35                   LCALL   RECEIVE
36                   RET
37
38  ADC2:            MOV     DPTR, #ADC2_HI
39                   MOVX    A,@DPTR
40                   ANL     A,#0FH
41                   LCALL   TRANSMIT
42                   LCALL   RECEIVE
43                   MOV     DF'TR, #ADC2_LO
44                   MOVX    A,@DPTR
45                   LCALL   TRANSMIT
46                   LCALL   RECEIVE
47                   RET
48
49  KODE_WRITE:      MOV     A,#0FH        ;kode write = 0ffffh
50                   LCALL   TRANSMIT
                     LCALL   RECEIVE
                     MOV     A,#0FFH
53                   LCALL   TRANSMIT
54                   LCALL   RECEIVE
```

```
55                      RET
56
57  KODE_MOVE:          MOV         A,#ØØH              ;kode move = ØØØØh
58                      LCALL       TRANSMIT
59                      LCALL       RECEIVE
60                      MOV         A,#ØØH
61                      LCALL       TRANSMIT
62                      LCALL       RECEIVE
63                      RET
64
65  delay:              mov         r3,#Ø1h
66  delay_1:            mov         r4,#Øffh
67                      djnz        r4,$
68                      djnz        r3,delay_1
69                      ret
70
71  START:              MOV         SP,#3ØH
72                      LCALL       INIT-SERIAL
73  BEGIN:              SETB        P3.2                ;SET INTØ
74                      SETB        P1.1          ;AKTIFKAN KEDUA ADC
75                      JB          P1.Ø,$              ;CHECK STATUS ADC2
76                      JB          P1.2,$              ;CHECK STATUS ADC1
77                      CLR         P1.1                ;HOLD DATA ADC
78                      JB          P3.2,DATA_MOVE      ;DATA MOVE
79
80  DATA_WRITE:         LCALL       KODE_WRITE
81                      LJMP        LANJUT
82
83  DATA_MOVE:          LCALL       KODE_MOVE
84
85  LANJUT:             LCALL       ADC1
86                      LCALL       ADC2
87                      LJMP        BEGIN
88                      END
89
```

```
1   Program Prog11;
2   Uses Crt,Graph,Serial,WIAGraph,IrGraph;
3   Const
4     MaxMenu = 4;
5     MenuLabel: Array[Ø..MaxMenu-1] Of String =
6     ('Save', 'Open ','Clear', 'Quit' );
7   Type
8     Kp     = Record
9                X,Y  : Array[1..2] Of Word;
10               Flag : Boolean;
11             End;
12    Master = Array[Ø..MaxMenu-1] Of Kp;
13  Var
14    Count,Sensor1,Sensor2,Size       : Word;
15    Kode,KeyFlag,MouseFlag           : Boolean;
16    DataHi,DataLo,PixelColor         : Byte;
17    X,Y,XOld,YOld,CounterKey,Prev    : Integer;
18    Kursor                           : Pointer;
19    MenuPos                          : Master;
20    MouseComm,MouseKey,MouseX,MouseY : Integer;
21    Ch                               : Char;
22
23  Procedure SetAwal;
24  Begin
25    XOld := 2;
26    YOld := 2;
27    PixelColor := White;
28    KeyFlag := False;
29    CounterKey := 0;
30    Prev := 0;
31    Size := ImageSize(Ø,Ø,4,4);
32    GetMem(Kursor,Size);
33  End;
34
35  Procedure InitMouse;
36  Begin
37    MouseComm := Ø;
38    MouseKey  := 0;
39    MouseX    := Ø;
40    MouseY    := 0;
41    Mouse(MouseComm,MouseKey,MouseX,MouseY);
42    MouseShow;
43    MouseOn;
44  End;
45
46  Procedure HighLightOn(X1,Y1,X2,Y2 : Integer);
47  Begin
48    SetColor(LightRed);
49    Rectangle(X1,Y1,X2,Y2);
50  End;
51
52  Procedure HighLightOff(X1,Y1,X2,Y2:Integer);
53  Begin
54      SetColor(DarkGray);
```

```
55        Rectangle(X1,Y1,X2,Y2);
56    End;
57
58    Procedure BuatMenuBar;
59    Begin
60      For Count := 0 To MaxMenu-1 Do Begin
61      Button(8+(Count)*72,5,75+(Count)*72,29,2,2,17,8,
62      MenuLabel[Count]);
63        InnerBox(296,5,538,29,White,LightGray);
64        MenuPos[Count].X[1] :=  8+(Count)*72;
65        MenuPos[Count].X[2] := 75+(Count)*72;
66        MenuPos[Count].Y[1] := 5;
67        MenuPos[Count].Y[2] := 29;
68        MenuPos[Count].Flag := False;
69      End;
70      InitMouse;
71      HighLightOn(MenuPos[Prev].X[1],MenuPos[Prev].Y[1],
72                  MenuPos[Prev].X[2],MenuPos[Prev].Y[2]);
73    End;
74
75    Procedure BuatColorBox;
76    Var
77      X,Y : Word;
78    Begin
79      X := 543;
80      Y := 10;
81      For Count := 0 To 15 Do Begin
82        SetFillStyle(SolidFill,Count);
83        Bar(X,Y,X+40,Y+52);
84        SetColor(White);
85        Rectangle(X-1,Y-1,X+41,Y+53);
86        If Count =(15  div 2) Then Begin
87          X := 590;
88          Y := 10;
89        End
90        Else
91          Inc(Y,58);
92      End;
93    End;
94
95    Procedure InitMenu;              (* Mengatur daerah pada monitor *)
96    Begin
97      Box(0,0,639,479,3,3);
98      BuatMenuBar;
99      BuatColorBox;
100     OuterBox(8,31,538,471,White,Black);
101                                  (* Bidang Gambar 530 x 440 *)
102   End;
103
104   Procedure AmbilData(Var DataHi,DataLo : Byte);
105   Begin
106     DataHi := Receive;
107     Transmit(DataHi);
108     DataLo := Receive;
```

```
109     Transmit(DataLo);
110   End;
111
112   Procedure AmbilKodeADC(Var Kode : Boolean);
113   Begin
114     AmbilData(DataHi,DataLo);
115     If (DataHi = $0f) And (DataLo = $ff) Then Kode := True
116                                                (* Tombol Ditekan *)
117     Else If (DataHi = $00) And (DataLo = $00) Then Kode := False
118                                                (* Tombol Dilepas *)
119     Else Begin
120       OutText('Kode Salah !');
121       ReadLn;
122       CloseGraph;
123       Halt;
124     End;
125   End;
126
127   Procedure AmbilDataSensor (Var Sensor1,Sensor2 : Word);
128   Begin
129     AmbilData(DataHi,DataLo);
130     Sensor1 := (DataHi*$100)+DataLo;
131     AmbilData(DataHi,DataLo);
132     Sensor2 := (DataHi*$100)+DataLo;
133   End;
134
135   Procedure KonversiXY(Sensor1,Sensor2 : Word; Var X,Y : Integer);
136   Var
137     Sudut1,Sudut2 : Real;
138     X1,X2,Y1,Y2   : Integer;
139   Const
140     R1 = 500;
141     R2 = 490;
142
143   Begin
144     Sudut1 := Sensor1/4024*300;
145     Sudut2 := Sensor2/4022*300;
146     X1 := Round(R1*Sin(Sudut1/180.0*Pi));
147     Y1 := Round(R1*Cos(Sudut1/180.0*Pi));
148     X2 := Round(R2*Cos((90+Sudut1-Sudut2)/180.0*Pi));
149     Y2 := Round(R2*Sin((90+Sudut1-Sudut2)/180.0*Pi));
150     X  := X1-X2-13;
151     Y  := Y1+Y2-226;
152     If X < 0  Then X := 0;
153     If Y < 0  Then Y := 0;
154     If X > 639 Then X := 639;
155     If Y > 479 Then Y := 479;
156   End;
157
158   Procedure HapusPixel(XOld,YOld : Integer);
159   Begin
160     PutImage(XOld-2,YOld-2,Kursor^,NormalPut);
161   End;
162
```

```
163  Procedure GambarPixel(X,Y : Integer; PixelColor : Byte; Var
164  XOld,YOld : Integer);
165  Begin
166    GetImage(X-2,Y-2,X+2,Y+2,Kursor^);
167    MoveTo(X,Y);
168    SetColor(PixelColor);
169    Line(X-2,Y,X+2,Y);
170    Line(X,Y-2,X,Y+2);
171    PutPixel(X,Y,PixelColor);
172    Xold := X;
173    YOld := Y;
174  End;
175
176  Procedure KeyBoardMouse(Var Ch1:Char;Var ScanCode:Integer;
177                          MaxKey:Integer;GenPos:Master);
178
179    Function LeftMouse(Var CodeCount:Integer):Char;
180    Var MC,J,X1,X2     : Integer;
181    Begin
182     OldCursorX := MouseX; OldCursorY := MouseY;
183     Repeat
184       Mouse(MouseComm,MouseKey,MouseX,MouseY);
185     Until MouseKey=0;
186     For J := 0 to MaxKey-1 Do
187      Begin
188        If (OldCursorY>=GenPos[J].Y[1]) AND
189        (OldCursorY<=GenPos[J].Y[2]) Then
190         If ((OldCursorX>=GenPos[J].X[1]) AND
191         (OldCursorX<=GenPos[J].X[2])) Then Begin
192              CodeCount := J;
193              MouseFlag := True;LeftMouse := #13;
194        End;
195      End;
196    End;
197
198    Function RightMouse:Char;
199    Begin
200     Repeat
201       Mouse(MouseComm,MouseKey,MouseX,MouseY);
202     Until MouseKey=0;
203     MouseFlag := True;RightMouse := #27;
204    End;
205
206  Begin
207       MouseComm := 3;MouseFlag := False;
208       Mouse(MouseComm,MouseKey,MouseX,MouseY);
209       While((Not  MouseFlag) AND(Not  KeyPressed)) Do
210        Begin
211          Mouse(MouseComm,MouseKey,MouseX,MouseY);
212         If (MouseKey AND 1) =1 Then Chl := LeftMouse(ScanCode)
213          Else
214           If (MouseKey AND 2) = 2 Then Begin Chl := RightMouse;End;
215        End;
216       If Not(MouseFlag) Then
```

```
217         Begin
218           Ch1 := Readkey;
219           If Ch1 = #0 Then Ch1 := Readkey;
220         End;
221   End;
222
223   Procedure MenuProcess(KeyC :Integer);
224   Begin
225        MenuPos[KeyC].Flag := True;
226        MouseHide;
227        ButtonPress(MenuPos[KeyC].X[1],MenuPos[KeyC].Y[1],
228                    MenuPos[KeyC].X[2],MenuPos[KeyC].Y[2],
229                    17,8,MenuLabel[KeyC]);
230        Case KeyC Of
231         0  : Begin Delay(500);End;
232         1  : Begin Delay(500);End;
233         2  : Begin Delay(500);ClearDevice; InitMenu; End;
234         3  : Begin Delay(500);KeyFlag := True End;
235        End;
236        Button(MenuPos[KeyC].X[1],MenuPos[KeyC].Y[1],
237               MenuPos[KeyC].X[2],MenuPos[KeyC].Y[2],
238               2,2,16,8,MenuLabel[CounterKey]);
239        HighLightOn(MenuPos[KeyC].X[1],MenuPos[KeyC].Y[1],
240                    MenuPos[KeyC].X[2],MenuPos[KeyC].Y[2]);
241        MouseShow;
242        MenuPos[KeyC].Flag := False;
243   End;
244
245   Procedure CheckMenu;
246   Begin
247     KeyboardMouse(Ch,CounterKey,MaxMenu,MenuPos);
248     Case Ch Of
249          Chr(75) : Begin
250                     HighLightOff(8+(Prev)*72,5,75+(Prev)*72,29);
251                     If CounterKey = 0 Then CounterKey := 3
252                      Else Dec(CounterKey);Prev:= CounterKey;
253
254   HighLightOn(8+(CounterKey)*72,5,75+(CounterKey)*72,29);
255                    End;
256          Chr(77) : Begin
257                     HighLightOff(8+(Prev)*72,5,75+(Prev)*72,29);
258                     If CounterKey = 3 Then CounterKey := 0
259                      Else Inc(CounterKey);Prev:= CounterKey;
260
261   HighLightOn(8+(CounterKey)*72,5,75+(CounterKey)*72,29);
262                    End;
263          #13     : Begin
264                     HighLightOff(8+(Prev)*72,5,75+(Prev)*72,29);
265                     MenuProcess(CounterKey);
266                     Prev:= CounterKey;
267                    End;
268        End;
269   End;
270
```

```
271  Begin
272    ClrScr;
273    InitSerial(3,2);
274    InitGrp;
275    SetAwal;
276    InitMenu;
277    Repeat
278      AmbilKodeADC(Kode);
279      AmbilDataSensor(Sensor1,Sensor2);
280      KonversiXY(Sensor1,Sensor2,X,Y);
281      If Kode Then Begin
282        If ((X>8) And (X<538)) And ((Y>8) And (Y<471)) Then Begin
283          SetColor(PixelColor);
284          LineTo(X,Y);
285        End
286        Else If (X > 543) Then PixelColor := GetPixel(X,Y)
287        Else If (Y < 30) Then Begin
288        End;
289      End
290      Else Begin
291        If Not((X = XOld) And (Y=YOld)) Then Begin
292          HapusPixel(XOld,YOld);
293          GambarPixel(X,Y,PixelColor,XOld,YOld);
294        End;
295      End;
296      CheckMenu;
297    Until (Ch=#27) Or KeyFlag;
298    CloseGraph;
299  End.
300
```

```
1   Unit Serial;
2   Interface
3
4   Uses Crt,WIA;
5
6   Var
7     SerialAdd     : Integer;
8     LST,Data      : Word;
9
10  Procedure InitSerial(X,Y : Word);
11  Function Receive : Byte;
12  Procedure Transmit(Var Data : Byte);
13
14  Implementation
15  Procedure InitSerial(X,Y : Word);
16  Var
17    BaudRate : Integer;
18    I : Word;
19  Const
20    Teks : String = 'Serial  Address pada COM [1/2] ? ';
21  Begin
22    Repeat
23      GoToXY(X,Y);Write(Teks);
24      BacaI(X+Length(Teks),Y,1,15,Ø,SerialAdd);
25      Case SerialAdd Of
26        1 : SerialAdd := $3f8;
27        2 : SerialAdd := $2f8;
28        Else Begin
29           GoToXY(X,Y+2);WriteLn('Masukkan 1 untuk COM1 atau 2 untuk
30  COM2');
31        End;
32      End;
33    Until (SerialAdd=$3f8) Or (Ser ialAdd=$2f8);
34    GoToXY(X,Y+2);ClrEol;
35    Teks :='Baud  Rate yang digunakan ? ';
36    Repeat
37      GotoXY(X,Y+1);Write(Teks);
38      BacaI(X+Length(Teks),Y+1,5,15,Ø,BaudRate);
39      Case BaudRate Of
40         2400 : I := 48;
41         4800 : I := 24;
42         9600 : I := 12;
43        19200 : I :=  6;
44        Else Begin
45          GoToXY(X,Y+2);
46          WriteLn('Masukkan baud rate 2400, 4800, 9600 atau 19200');
47        End;
48      End;
49    Until (BaudRate=2400) Or (BaudRate=19200) Or (BaudRate=9600);
50    GoToXY(X,Y+2);ClrEol;
51    Port[SerialAdd+3] := $80;
52    Port[SerialAdd]   := Lo(I);
53    Port[SerialAdd+1] := Hi(I);
54    Port[SerialAdd+3] := $03;
```

```
55      Port[SerialAdd+1] := $00;
56    End;
57
58    Function Receive : Byte;
59    Begin
60      Repeat
61        LST := Port[SerialAdd+5];
62        LST := LST And $01;
63      Until LST = $01;
64      Receive := Port[SerialAdd];
65    End;
66
67    Procedure Transmit(Var Data : Byte);
68    Begin
69      Port[SerialAdd] := Data;
70      Repeat
71        LST := Port[SerialAdd+5];
72        LST := LST And $20;
73      Until LST = $20;
74    End;
75    End.
76
```

```
1   unit WIAGraph;
2
3   interface
4   uses Crt,Graph;
5   var  ErrorCode : integer;
6
7   procedure InitGrp;
8   procedure GWriteLn(X,Y : word; NumKar : real;
9                      Lebar, Desimal : byte);
10  function GReadLn(X,Y : word) : string;
11
12  implementation
13
14  procedure InitGrp;
15  var
16    GrafDrv, GrafMode : integer;
17  begin
18    GrafDrv := detect;
19    initGraph(GrafDrv,GrafMode,'c:\app\bp\bgi');
20    ErrorCode := graphResult;
21    if ErrorCode <> grOk then begin
22      Writeln('Kesalahan graphics: ',GraphErrorMsg(ErrorCode));
23      WriteLn('Program dihentikan...');
24      Halt(1);
25    end;
26  end;
27
28  procedure GWriteLn(X,Y : word; NumKar : real;
29                     Lebar, Desimal : byte);
30  var S : string;
31  begin
32    str(NumKar:Lebar:Desimal,S);
33    outTextXY(X,Y,S);
34  end;
35
36  function GReadLn(X,Y : word) : string;
37  const Hidup     = true;
38        Mati      = false;
39        Enter     = #13;
40        Kiri      = #75;
41        Kanan     = #77;
42        Backspace = #08;
43
44  var  C,C1 : char; Posisi : integer;
45       Panjang : integer; Kata : string;
46       Lebar,Tinggi : byte;
47
48  procedure Kursor(X,Y : word; Status : boolean);
49  begin
50    if Status then setFillStyle(solidFill,white)
51    else setFillStyle(emptyFill,white);
52    bar(X,Y-2,X+Lebar-1,Y+Tinggi);
53  end;
54
```

```
55  begin
56    setWriteMode(xorPut);
57    setTextStyle(defaultFont,horizDir,1);
58    setTextJustify(leftText,topText);
59    Lebar := textWidth('H');
60    Tinggi := textHeight('H');
61    Kursor(X,Y,Hidup);
62    Posisi := 1;
63    Panjang := 0;
64    Kata := '';
65
66    repeat
67      C := readKey;
68      case C of
69        #00: begin
70               C1 := readKey;
71               if C1 = Kiri then
72                 begin
73                   Kursor(X,Y,Mati);
74                   Kursor(X-Lebar,Y,Hidup);
75                   setColor(getBkColor);
76                   outTextXY(X-Lebar,Y,Kata[Posisi-1]);
77                   if Posisi < Panjang+1 then
78                     begin
79                       setColor(White);
80                       outTextXY(X,Y,Kata[Posisi]);
81                     end;
82                   dec(X,Lebar);
83                   dec(Posisi);
84                   setColor(white);
85                 end
86                 else if C1 = Kanan then
87                   begin
88                     if (Posisi<Panjang+1) then
89                       begin
90                         Kursor(X,Y,Mati);
91                         setColor(white);
92                         {outTextXY(X,Y,Kata[Posisi]);}
93                         Kursor(X+Lebar,Y,Hidup);
94                         setColor(getBkColor);
95                         if Posisi<Panjang then
96                           outTextXY(X+Lebar,Y,Kata[Posisi+1]);
97                         inc(X,Lebar);
98                         inc(Posisi);
99                       end;
100                  end;
101           end;
102       Backspace: begin
103                    Kursor(X,Y,Mati);
104                    setColor(getBkColor);
105                    outTextXY(X-Lebar,Y,Kata[Panjang]);
106                    setColor(White);
107                    Kursor(X-Lebar,Y,Hidup);
108                    dec(X,Lebar);
```

```
109                     dec(Panjang);
110                     dec(Posisi);
111                   end;
112         else
113           begin
114             Kursor(X,Y,Mati);
115             outTextXY(X,Y,C);
116             Kursor(X+Lebar,Y,Hidup);
117             inc(X,Lebar);
118             inc(Panjang);
119             inc(Posisi);
120             Kata := Kata + C;
121           end;
122       end;
123     until C = Enter;
124     GReadLn := Kata;
125   end;
126   end.
127
```

```
 1  Unit WIA;
 2  Interface
 3  Uses Crt;
 4  Type
 5    String79= String[79];
 6  Procedure Bersih(B1,B2,B3,B4 : Byte);
 7  Procedure Kotak(X1,Y1,X2,Y2 : Byte);
 8  Procedure BacaI(X,Y,P,WH,WD : Byte; Var Bil : Integer);
 9  Procedure BacaR(X,Y,P,WH,WD : Byte; Var Bil : Real);
10  Procedure BacaS(X,Y,P,WH,WD : Byte; Var A : String79);
11
12  Implementation
13  Procedure Bersih(B1,B2,B3,B4 : Byte);
14  Begin
15    Window(B1,B2,B3,B4);
16    ClrScr;
17    Window(1,1,80,24);
18  End;
19
20  Procedure Kotak(X1,Y1,X2,Y2 : Byte);
21  Var
22    I : Byte;
23  Begin
24    GoToXY(X1,Y1);
25    Write(Chr(201));
26    For I := X1+1 To X2-1 Do Write(Chr(205));
27    Write(Chr(187));
28    For I := Y1+1 To Y2-1 Do
29    Begin
30      GoToXY(X1,I);Write(Chr(186));
31      GoToXY(X2,I);Write(Chr(186));
32    End;
33    GoToXY(X1,Y2);
34    Write(Chr(200));
35    For I := X1+1 To X2-1 Do Write(Chr(205));
36    Write(Chr(188));
37  End;
38
39  Procedure BacaI(X,Y,P,WH,WD : Byte; Var Bil : Integer);
40  var
41    R : Real;
42    A : String[79];
43    In-Char : Char;
44    ValCode : Integer;
45  Begin
46    Repeat
47      If X+P > 80 Then Exit;
48      Window(X,Y,X+P-1,Y);
49      TextColor(WH);
50      TextBackGround(WD);
51      ClrScr;
52      Window(X,Y,X+P,Y);
53      In-Char := ReadKey;
54      A :='';
```

```
55        While In-Char <> Chr(13) Do
56        Begin
57          If In-Char = Chr(8) Then
58          Begin
59            If Length(A) > 0 Then
60            Begin
61              A[0] := Chr(Length(A)-1);
62              Write(Chr(8));
63              Write(' ');
64              Write(Chr(8));
65            End
66            Else
67              Write(Chr(7));
68          End
69          Else
70            If In-Char = Chr(27) Then Write(Chr(7))
71            Else
72            Begin
73              If(Length(A) < P) And (In-Char In ['-','0'..'9']) Then
74              Begin
75                A[0] := Chr(Length(A)+1);
76                A[Ord(A[0])] := In-Char;
77                Write(In_Char);
78              End
79              Else
80                Write(Chr(7));
81            End;
82            In-Char := ReadKey;
83          End;
84        If Length(A) = 0 Then A:='';
85        Val(A,R,ValCode);
86        If (R > MaxInt) Or (R < -MaxInt) Or (ValCode <> 0) Then
87        Begin
88          Write(Chr(7));
89          ValCode := 1;
90        End;
91      Until ValCode=0;
92      Window(1,1,80,25);
93      Bil:=Round(R);
94    End;
95
96    Procedure BacaR(X,Y,P,WH,WD : Byte; Var Bil : Real);
97    Var
98      A : String[79];
99      In-Char : Char;
100     ValCode : Integer;
101   Begin
102     Repeat
103       If X+P > 80 Then Exit;
104       Window(X,Y,X+P-1,Y);
105       TextColor(WH);
106       TextBackGround(WD);
107       ClrScr;
108       Window(X,Y,X+P,Y);
```

```
109       In-Char := ReadKey;
110       A :='';
111       While In_Char <> Chr(13) Do
112       Begin
113         If In-Char = Chr(8) Then
114         Begin
115           If Length(A) > 0 Then
116           Begin
117             A[Ø] := Chr(Length(A)-1);
118             Write(Chr(8));
119             Write(' ');
120             Write(Chr(8));
121           End
122         .Else
123           Write(Chr(7));
124           End
125         Else
126           If In-Char = Chr(27) Then Write(Chr(7))
127           Else
128           Begin
129             If(Length(A) < P) And (In-Char In ['-','0'. .'9']) Then
130             Begin
131               A[Ø] := Chr(Length(A)+1);
132               A[Ord(A[Ø])] := In-Char;
133               Write(In_Char);
134             End
135             Else
136             Write(Chr(7));
137           End;
138       In-Char := ReadKey;
139       End;
140       If Length(A) = 0 Then A:='Ø';
141       Val(A,Bil,ValCode);
142       If ValCode <> 0 Then Write(Chr(7));
143     Until ValCode=Ø;
144     Window(1,1,8Ø,25);
145     End;
146
147   Procedure BACAS(X,Y,P,WH,WD : Byte; Var A : String78);
148   var
149     In-Char : Char;
150   Begin
151     If X+P > 8Ø Then Exit;
152     Window(X,Y,X+P-1,Y);
153     TextColor(WH);
154     TextBackGround(WD);
155     ClrScr;
156     Window(X,Y,X+P,Y);
157     In-Char := ReadKey;
158     A := '';
159     While In_Char <> Chr(13) Do
160     Begin
161       If In-Char = Chr(8) Then
162       Begin
```