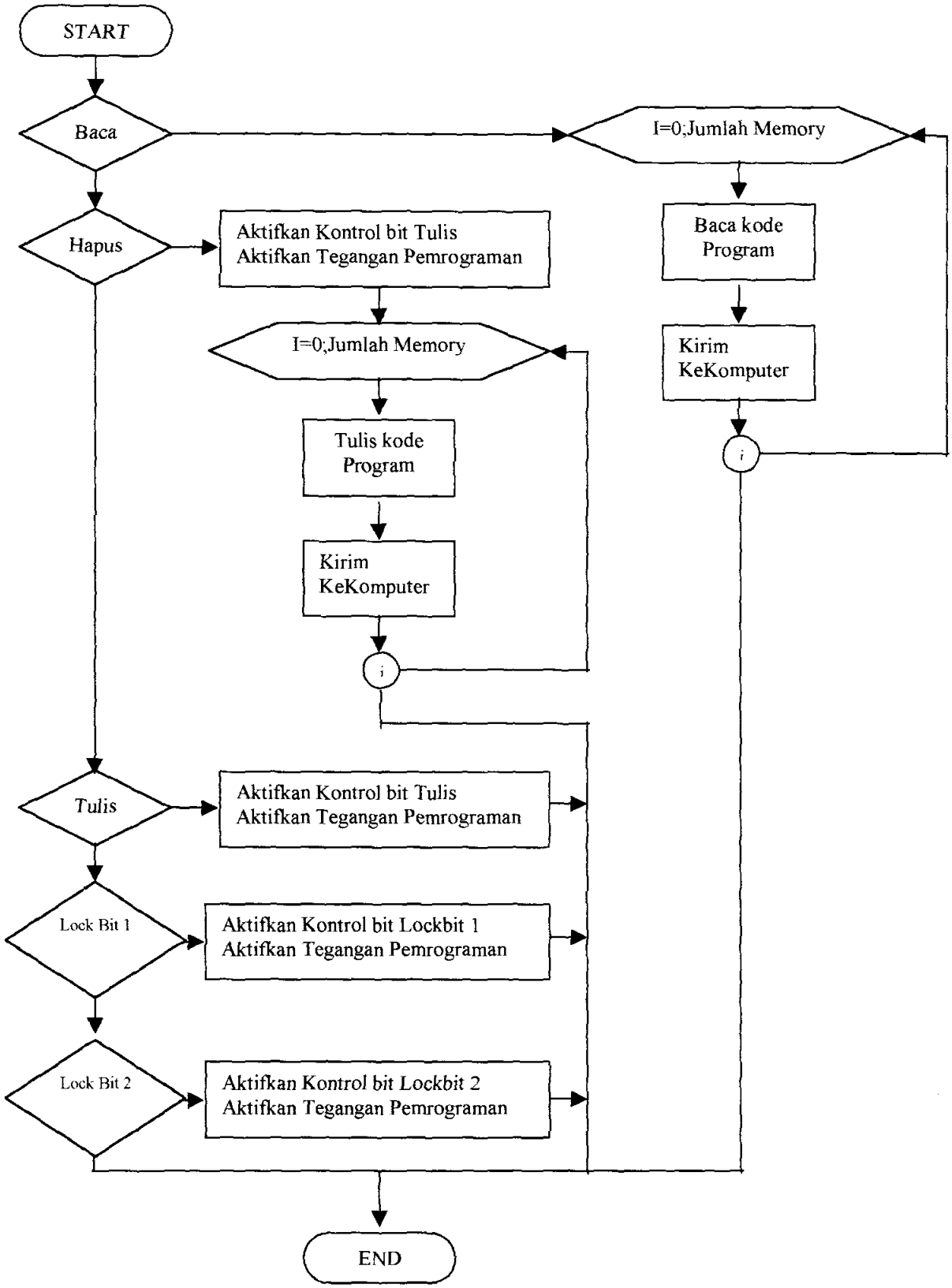
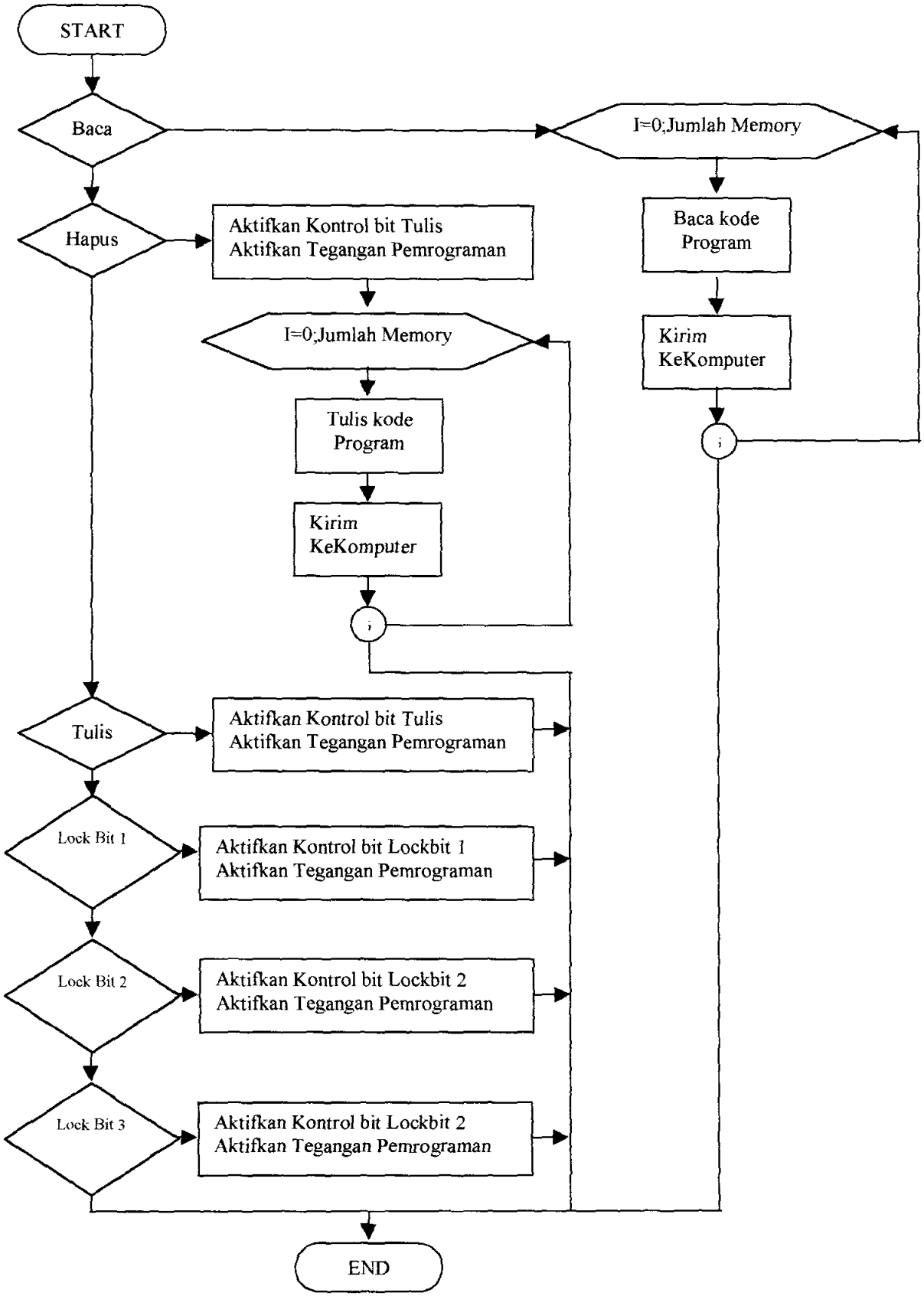
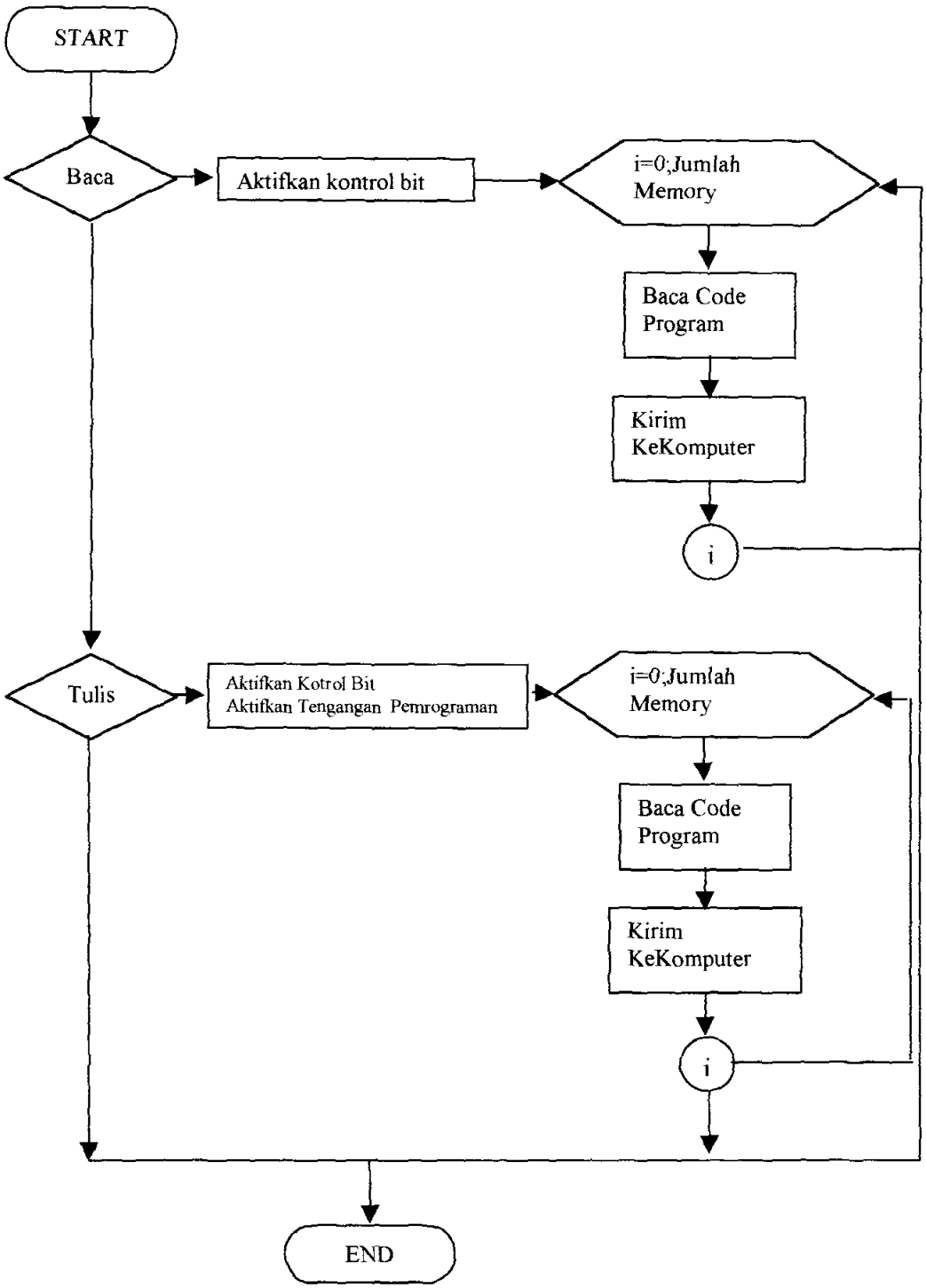

START
Baca
Hapus
Aktifkan Kontrol bit Tulis
Aktifkan Tegangan Pemrograman
I=0;Jumlah Memory
Tulis kode
Program
Kirim
KeKomputer
i
I=0;Jumlah Memory
Baca kode
Program
Kirim
KeKomputer
i
Tulis
Aktifkan Kontrol bit Tulis
Aktifkan Tegangan Pemrograman
Lock Bit 1
Aktifkan Kontrol bit Lockbit 1
Aktifkan Tegangan Pemrograman
Lock Bit 2
Aktifkan Kontrol bit Lockbit 2
Aktifkan Tegangan Pemrograman
END

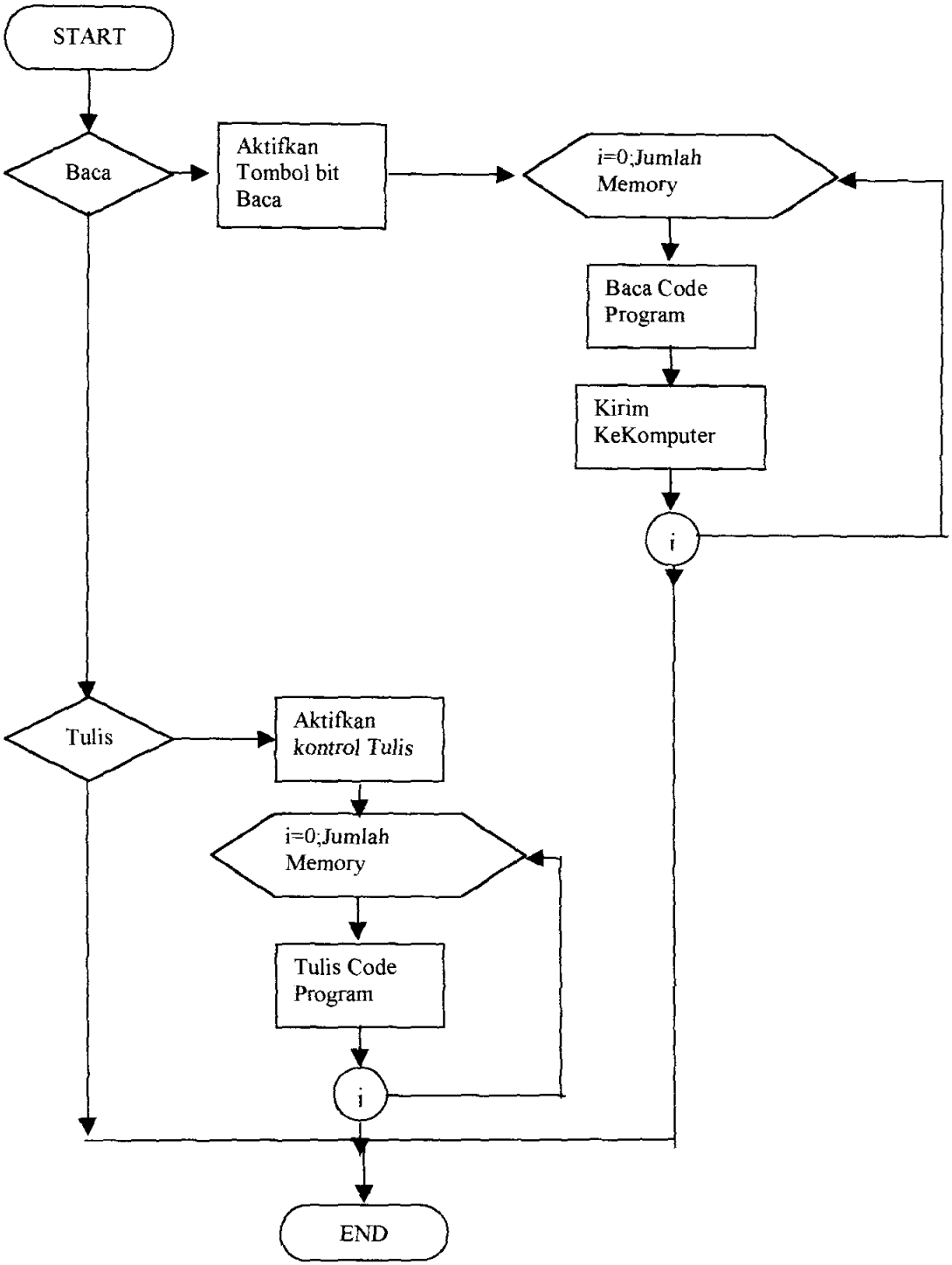
FLOW CHART PROGRAM 89CX051

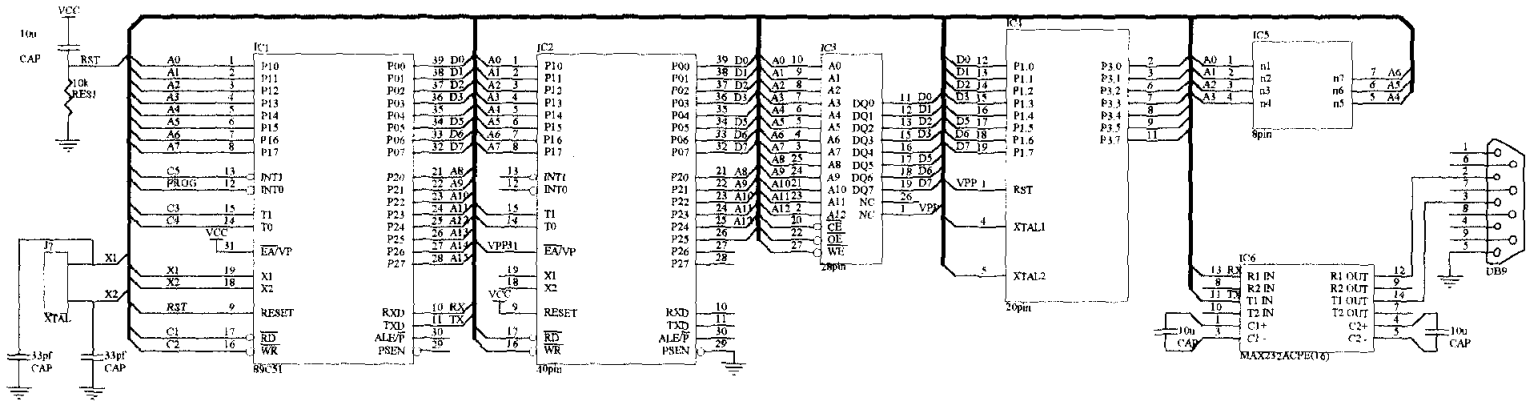


FLOW CHART 27CXX



FLOW CHART 28CXX, 93CXX, DAN 24CXX





Title		
Size: B	Number	Revision
Date: 21-Jan-2002		Sheet of
File: D:\yudi\TA\SKEMA.sch		Drawn By:

```
uses dos,mouse,crt;
type rec_command=record
     command:byte;
     lock   :array[0..10]of boolean;
     end;
const pos_file=4;
      pos_command=11;
      pos_device=21;
      pos_help=64;
      TX  = $3F8;
      RX  = $3F8;
      LCR = $3FB;
      MSB = $3F9;
      LSB = $3F8;
      IER = $3F9;
      LSR = $3FD;
      MCR = $3FC;
      IIR = $3FA;
      AOO = $20;
      AO1 = $21;
      PASSWORD:STRING='   5103098008';
      ADDR=$9000;

var  r:registers;
     hexa:string[4];
     tot_file,tot_verify,tot_memory,tombol:word;
     file_menu,command_menu,device_menu,
     help_menu,lockbit_menu,menu,sub_menu:byte;
     file_open,error,read_ok,press_ok,alt_ok:boolean;
     SERIAL,code_ic:byte;
     command:rec_command;
     rev:pointer;
     buffer:array[0..64000]of byte;
     xmouse,ymouse:integer;

function keypress:boolean;
var r:registers;
    begin
    r.ah:=$11;
    intr($16,r);
    if (r.flags and fzero)=0 then keypress:=true
    else keypress:=false;
    end;

function IntToStr(I: Longint): String;
{ Convert any integer type to a string }
var
 S: string[11];
begin
 Str(I, S);
```

```
 IntToStr := S;
end;

function rdkey:word;
var r:registers;
    begin
    r.ah:=$10;
    intr($16,r);
    rdkey:=r.ax;
    end;

procedure text_xy(a:string; x,y:byte; back,text:byte);
var temp     :word;
    long,n   :byte;
    begin
    if(x<=80)and(y<=27)then
        begin
        long:=length(a);
        for n:=1 to long do
            begin
            temp:=(x-1)*2+(y-1)*160;
            mem[$b800:temp]:=ord(a[n]);
            mem[$b800:temp+1]:=((back and $0f) shl 4)or(text
and $0f) ;
            inc(x);
            end;
        end;
    end;

procedure text_background(x1,y1,x2,y2:byte;back:byte);
var temp      :word;
    long,n,x,y :byte;
    begin
    if(x2<=80)and(y2<=27)then
        begin
        for x:=x1 to x2 do
            for y:=y1 to y2 do
                begin
                temp:=(x-1)*2+(y-1)*160;
                mem[$b800:temp+1]:=((back and $0f) shl 4)or(m
em[$b800:temp+1]and $0f) ;
                end;
        end;
    end;

procedure line_x(x1,x2,y,t:byte);
var x:byte;
    temp:integer;
    begin
    for x:=x1 to x2 do
```

```
            begin
            temp:=(x-1)*2+(y-1)*160;
            end;
    end;

procedure line_y(x,y1,y2,t:byte);
var y:byte;
    temp:integer;
    begin
    for y:=y1 to y2 do
            begin
            temp:=(x-1)*2+(y-1)*160;
            end;
    end;

procedure back_ground(c:char;x1,y1,x2,y2,b,t:byte);
var x,y :byte;
          begin
          for x:=x1 to x2 do
              for y:=y1 to y2 do text_xy(c,x,y,b,t)
          end;

function byte_ascii(a:byte):char;
         begin
         byte_ascii:=char(a);
         end;

function byte_hex(a:byte):string;
const CONV  :ARRAY[0..15] OF CHAR=('0','1','2','3','4','5','6
','7',
                                   '8','9','A','B','C','D','E
','F');
         begin
         byte_hex:=CONV[((a)MOD 256)DIV 16]+CONV[((a)MOD 256)
mod 16];
         end;

function word_hex(a:word):string;
const CONV  :ARRAY[0..15] OF CHAR=('0','1','2','3','4','5','6
','7',
                                   '8','9','A','B','C','D','E
','F');
          begin
          word_hex:=CONV[(a)DIV 4096]+CONV[((a)MOD 4096)DIV 2
56]+
          CONV[((a)MOD 256)DIV 16]+CONV[((a)MOD 256)mod 16];
          end;

PROCEDURE CLEAR;
VAR X,Y:BYTE;
```

```
          BEGIN
          FOR X:=1 TO 80 DO
          FOR Y:=1 TO 25 DO TEXT_XY(' ',X,Y,0,15);
          END;

PROCEDURE INIT_SER;
          BEGIN
          PORT[LCR]:=108;
          PORT[MSB]:=0;
          PORT[LSB]:=50;
          PORT[LCR]:=1;
          PORT[IER]:=3;
          PORT[AO1]:=32;
          PORT[LSR]:=0;
          PORT[MCR]:=$e;
          END;

FUNCTION DATA_IN:BOOLEAN;
         BEGIN
          IF PORT[IIR]=4 THEN DATA_IN:=TRUE ELSE DATA_IN:=FAL
SE;
          END;
FUNCTION TX_KOSONG:BOOLEAN;
         BEGIN
         IF PORT[LSR]=96 THEN TX_KOSONG:= TRUE ELSE TX_KOSONG
:=FALSE;
         END;

PROCEDURE SEND(DATA:BYTE);
          BEGIN
          IF TX_KOSONG THEN PORT[TX]:=DATA;
          END;

PROCEDURE RECEIVER;INTERRUPT;
          BEGIN
          IF DATA_IN THEN
             BEGIN
             READ_OK:=TRUE;
             SERIAL:=PORT[RX];
             END;
          PORT[AOO]:=$20;
          END;

PROCEDURE LOAD_PASSWORD;
var n:byte;
          BEGIN
          FOR n:=1 TO LENGTH(PASSWORD) DO
              BEGIN
              SEND(ORD(PASSWORD[N]));
              DELAY(10);
```

```
            END;
        END;

procedure beep_true;
        begin
        sound(1000);
        delay(200);
        nosound;
        end;

procedure beep_false;
var n:byte;
        begin
        for n:=1 to 3 do
            begin
            sound(1000);
            delay(200);
            nosound;
            delay(100);
            end;
        end;

procedure background;
        begin
        back_ground('°',1,2,80,18,7,1);
        back_ground('°',1,19,49,24,7,1);
        text_xy('
                                        ',1,1,7,0);
        text_xy(' File ',pos_file,1,7,0);
        text_xy(' Command ',pos_command,1,7,0);
        text_xy(' Device ',pos_device,1,7,0);
        text_xy(' Help ',pos_help,1,7,0);
        text_xy(' F',pos_file,1,7,4);
        text_xy(' C',pos_command,1,7,4);
        text_xy(' D',pos_device,1,7,4);
        text_xy(' H',pos_help,1,7,4);
        text_xy('                                    ',50,19,4,1
0);
        text_xy('         TUGAS AKHIR                ',50,20,4,1
5);
        text_xy('    by Silasilu@yahoo.com           ',50,21,4,1
0);
        text_xy('       Yudi Hariyanto               ',50,22,4,1
0);
        text_xy('         5103098008                 ',50,23,4,1
0);
        text_xy('                                    ',50,24,4,1
0);
        text_xy('É',50,19,4,0);
        text_xy('È',50,24,4,0);
```

```
          text_xy('»',80,19,4,0);
          text_xy('¼',80,24,4,0);
          line_x(51,79,19,0);
          line_x(51,79,24,0);
          line_y(50,20,23,0);
          line_y(80,20,23,0);
          text_background(1,25,80,25,7);
          text_xy('  Alt+X ',1,25,7,4);
          text_xy('Exit ',9,25,7,0);
          end;

procedure block_menu_file;
          begin
          background;
          sub_menu:=file_menu;
          text_xy(' File ',pos_file,1,3,0);
          text_xy(' F',pos_file,1,3,4);
          back_ground(' ',pos_file-3,2,pos_file+20,9,7,15);
          text_xy('É',pos_file-3,2,7,0);
          text_xy('È',pos_file-3,9,7,0);
          text_xy('»',pos_file+20,2,7,0);
          text_xy('¼',pos_file+20,9,7,0);
          line_x(pos_file-2,pos_file+19,2,0);
          line_x(pos_file-2,pos_file+19,9,0);
          line_y(pos_file-3,3,8,0);
          line_y(pos_file+20,3,8,0);
          text_xy('Load              F3',pos_file,3,7,0);
          text_xy('Save              F2',pos_file,4,7,0);
          text_xy('Display Buffer     B',pos_file,5,7,0);
          text_xy('Display File       D',pos_file,6,7,0);
          text_xy('Download          F9',pos_file,7,7,0);
          text_xy('Exit             F10',pos_file,8,7,0);
          text_background(pos_file-1,3,pos_file+22,10,0);
          end;

procedure load_file;
const xfile=11;
      yfile=11;
var filename:string[50];
    f:file;
    n:word;
          begin
          background;
          back_ground(' ',10,11,71,12,0,15);
          text_xy('File Name : ',xfile,yfile,0,15);
          gotoxy(xfile+12,yfile);
          textcolor(white);
          readln(filename);
          assign(f,filename+'.bin');
          {$i-}
```

```
          reset(f);
          {$i+}
          if ioresult=0 then
             begin
             reset(f,1);
             if filesize(f)<=tot_memory then
                begin
                for n:=0 to 65535 do mem[addr:n]:=0;
                reset(f,1);
                blockread(f,buffer,filesize(f));
                for n:=0 to filesize(f) do mem[addr:n]:=buffe
r[n];
                text_xy('Filesize  : '+inttostr(filesize(f)),
xfile,12,0,15);
                text_xy('Filesize  : '+inttostr(tot_memory),x
file+20,12,0,15);
                tot_file:=filesize(f);
                file_open:=true;
                end
             else
                begin
                background;
                text_xy('File to big size',32,12,0,3);
                end;
             close(f);
             rdkey;
             end
          else
             begin
             background;
             text_xy('File not found',33,12,0,3);
             rdkey;
             end;
          background;
          alt_ok:=false;
          menu:=0;
          end;

procedure save;
var filename:string[50];
    tombol:char;
    n:word;
    f:file of byte;
          begin
          background;
          back_ground(' ',10,8,70,14,0,15);
          text_xy('Filename : ',12,10,0,15);
          gotoxy(23,10);
          readln(filename);
          assign(f,filename);
```

```
          {$i-}
          reset(f);
          {$i+}
          if ioresult<>0 then
             begin
             rewrite(f);
             for n:=1 to tot_memory do write(f,buffer[n-1]);
             end
          else
             begin
             repeat
             text_xy('Do you want to replace (y/n) ?
                    ',12,12,0,15);
             gotoxy(42,12);
             readln(tombol);
             until (upcase(tombol)='Y')or(upcase(tombol)='N')
;
             if upcase(tombol)='Y' then
                begin
                rewrite(f);
                for n:=1 to tot_memory do write(f,buffer[n-1]
);
                end;
             end;
          close(f);
          background;
          alt_ok:=false;
          menu:=0;
          end;

procedure disp(memory:word);
const x_off=11;
      y_off=6;
      x_str=61;
      x_memory=5;
var x,y:byte;
          begin
          texT_xy('DISPLAY BUFFER',33,3,0,10);
          text_xy('É',31,2,0,15);
          text_xy('È',31,4,0,15);
          text_xy('»',48,2,0,15);
          text_xy('¼',48,4,0,15);
          line_x(32,47,2,15);
          line_x(32,47,4,15);
          line_y(31,3,3,15);
          line_y(48,3,3,15);
          text_xy('É',3,5,0,15);
          text_xy('È',3,22,0,15);
          text_xy('»',78,5,0,15);
          text_xy('¼',78,22,0,15);
```

```
          line_x(4,77,5,15);
          line_x(4,77,22,15);
          line_y(3,6,21,15);
          line_y(78,6,21,15);
          for x:=0 to 255 do
              begin
              text_xy(byte_hex(memory)+byte_hex(x div 16 shl
4),x_memory,y_off+x div 16,0,15);
              text_xy(byte_hex(buffer[memory*256+x]),x_off+3*
(x mod 16),y_off+x div 16,0,15);
              if (buffer[memory*256+x]=0)or(buffer[memory*256
+x]=$ff)then
                  text_xy('.',x_str+x mod 16,y_off+x div 16,0,
14)
              else text_xy(byte_ascii(buffer[memory*256+x]),x
_str+x mod 16,y_off+x div 16,0,14);
              end;
          end;

procedure disp_mem(memory:word);
const x_off=11;
      y_off=6;
      x_str=61;
      x_memory=5;
var x,y:byte;
          begin
          texT_xy(' DISPLAY FILE ',33,3,0,10);
          text_xy('É',31,2,0,15);
          text_xy('È',31,4,0,15);
          text_xy('»',48,2,0,15);
          text_xy('¼',48,4,0,15);
          line_x(32,47,2,15);
          line_x(32,47,4,15);
          line_y(31,3,3,15);
          line_y(48,3,3,15);
          text_xy('É',3,5,0,15);
          text_xy('È',3,22,0,15);
          text_xy('»',78,5,0,15);
          text_xy('¼',78,22,0,15);
          line_x(4,77,5,15);
          line_x(4,77,22,15);
          line_y(3,6,21,15);
          line_y(78,6,21,15);
          for x:=0 to 255 do
              begin
              text_xy(byte_hex(memory)+byte_hex(x div 16 shl
4),x_memory,y_off+x div 16,0,15);
              text_xy(byte_hex(mem[addr:memory*256+x]),x_off+
3*(x mod 16),y_off+x div 16,0,15);
              if (mem[addr:memory*256+x]=0)or(mem[addr:memory
```

```
*256+x]=$ff)then
                text_xy('.',x_str+x mod 16,y_off+x div 16,0,
14)
              else text_xy(byte_ascii(mem[addr:memory*256+x])
,x_str+x mod 16,y_off+x div 16,0,14);
              end;
          end;

procedure display_buffer;
var tombol:word;
    n:byte;
          begin
          clear;
          n:=0;
          disp(n);
          repeat
          if keypressed then
             begin
             tombol:=rdkey;
             case tombol of
                  $51e0:begin
                        inc(n);
                        disp(n);
                        end;
                  $49e0:begin
                        dec(n);
                        disp(n);
                        end;
                  end;
             end;
          until tombol=$011b;
          background;
          alt_ok:=false;
          menu:=0;
          end;

procedure display_file;
var tombol:word;
    n:byte;
          begin
          clear;
          n:=0;
          disp_mem(n);
          repeat
          if keypressed then
             begin
             tombol:=rdkey;
             case tombol of
                  $51e0:begin
                        inc(n);
```

```
                        disp_mem(n);
                        end;
                  $49e0:begin
                        dec(n);
                        disp_mem(n);
                        end;
                  end;
            end;
         until tombol=$011b;
         background;
         alt_ok:=false;
         menu:=0;
         end;

procedure read(memory:word);
var n:word;
    tom:boolean;
         begin
         delay(20);
         if (memory mod 256=0)then
            begin
            send(memory div 256 );
            delay(20);
            end
         else
            begin
            send(memory div 256+1);
            delay(20);
            end;
         send($ff);
         delay(20);
         tom:=false;
         n:=0;
         repeat
             read_ok:=false;
             send($ff);
             repeat
             if keypressed then tom:=true;
             until (read_ok)or(tom);
             buffer[n]:=serial;
             inc(n);
         until (tom)or(n=memory);
         end;

procedure prog;
var n:word;
    m:longint;
    tom:boolean;
         begin
         text_xy('Filesize : '+inttostr(tot_file),33,6,0,15)
```

```
;
          if (tot_file mod 256=0)then
             begin
             send(tot_file mod 256 );
             delay(20);
             end
          else
             begin
             send(tot_file mod 256);
             delay(20);
             end;
          send($ff);
          delay(100);
          tom:=false;
          n:=0;
          repeat
              read_ok:=false;
              send(mem[addr:n]);
              repeat
              if keypressed then tom:=true;
              until (read_ok)or(tom);
              inc(n);
              m:=n;
              m:=100*m div tot_file;
              text_xy('Programing : '+inttostr(m)+'%',12,7,0,
15);
          until (n=tot_file)or(tom);
          text_xy('OK       ',25,7,0,10);
          beep_true;
          text_xy(    'Verify     : Wait...',12,9,0,15);
          read(tot_file);
          error:=false;
          for n:=1 to tot_file do if buffer[n-1]<>mem[addr:n-
1]then error:=true;
          if error then
             begin
             text_xy('Error     ',25,9,0,12);
             beep_false;
             end
          else
              begin
              text_xy('OK     ',25,9,0,10);
              beep_true;
              end;
          end;

procedure erase;
var tom:boolean;
          begin
          read_ok:=false;
```

```
          repeat
          if keypressed then tom:=true;
          until (read_ok)or(tom);
          text_xy('Erase Completely',32,10,0,10);
          beep_true;
          end;

procedure lockbit1;
          begin
          text_xy('Lock Bit 1',12,11,0,15);
          beep_true;
          end;

procedure lockbit2;
          begin
          text_xy('Lock Bit 2',12,11,0,15);
          beep_true;
          end;

procedure lockbit3;
          begin
          text_xy('Lock Bit 3',12,11,0,15);
          beep_true;
          end;

procedure download;
          begin
          background;
          if (command.lock[command.command])then
             begin
             back_ground(' ',10,5,70,12,0,15);
             case command.command of
                  1:begin
                    text_xy('Reading   Wait...',31,9,0,15);
                    read(tot_memory);
                    beep_true;
                    text_xy('     Reading  OK       ',28,9,0,
10);
                    end;
                  2:erase;
                  3:if(tot_file<=tot_memory)and(file_open) th
en prog
                    else text_xy('Error to big size OR file n
ot open',23,10,0,15);
                  4:if(tot_file<=tot_memory)and(file_open)the
n lockbit1
                    else text_xy('Error to big size OR file n
ot open',23,10,0,15);
                  5:if(tot_file<=tot_memory)and(file_open)the
n lockbit2
```

```
                    else text_xy('Error to big size OR file n
ot open',23,10,0,15);
                  6:if(tot_file<=tot_memory)and(file_open)the
n lockbit3
                    else text_xy('Error to big size OR file n
ot open',23,10,0,15);
                  end;
               text_xy('PRESS ANY KEY',33,11,8,15);
               rdkey;
               end;
            background;
            alt_ok:=false;
            menu:=0;
            end;

procedure rutin_sub_file;
            begin
            case file_menu of
                 0:load_file;
                 1:save;
                 2:display_buffer;
                 3:display_file;
                 4:download;
                 5:tombol:=$2d00;  {exit}
                 end;
            end;

procedure sub_file;
            begin
            press_ok:=true;
            if sub_menu=255 then sub_menu:=5;
            if sub_menu>5 then sub_menu:=0;
            file_menu:=sub_menu;
            text_background(pos_file-3,2,pos_file+20,9,7);
            text_background(pos_file-2,3+sub_menu,pos_file+19,3
+sub_menu,2);
            if ((tombol and $ff)=13)and    {if menu file  and p
ress Enter}
               (press_ok)and(menu=1)then
               begin
               press_ok:=false;
               rutin_sub_file;
               end;
            end;

procedure load_sub_command;
            begin
            case command_menu of
                 0:command.command:=3;
                 1:command.command:=1;
```

```
                2:command.command:=2;
                3:command.command:=4;
                4:command.command:=5;
                5:command.command:=6;
                end;
            end;

procedure block_menu_command;
            begin
            background;
            sub_menu:=command_menu;
            load_sub_command;
            text_xy(' Command ',pos_command,1,3,0);
            text_xy(' C',pos_command,1,3,4);
            back_ground(' ',pos_command-3,2,pos_command+20,9,7,
15);
            text_xy('É',pos_command-3,2,7,0);
            text_xy('È',pos_command-3,9,7,0);
            text_xy('»',pos_command+20,2,7,0);
            text_xy('¼',pos_command+20,9,7,0);
            line_x(pos_command-2,pos_command+19,2,0);
            line_x(pos_command-2,pos_command+19,9,0);
            line_y(pos_command-3,3,8,0);
            line_y(pos_command+20,3,8,0);
            if command.lock[3]then text_xy('Prog',pos_command,3
,7,0);
            if command.lock[1]then text_xy('Read',pos_command,4
,7,0);
            if command.lock[2]then text_xy('Erase',pos_command,
5,7,0);
            if command.lock[4]then text_xy('LB 1',pos_command,6
,7,0);
            if command.lock[5]then text_xy('LB 2',pos_command,7
,7,0);
            if command.lock[6]then text_xy('LB 3',pos_command,8
,7,0);
            text_background(pos_command-1,3,pos_command+22,10,0
);
            end;

procedure sub_command;
            begin
            press_ok:=true;
            if sub_menu=255 then sub_menu:=5;
            if sub_menu>5 then sub_menu:=0;
            command_menu:=sub_menu;
            load_sub_command;
            text_background(pos_command-3,2,pos_command+20,9,7)
;
            text_background(pos_command-2,3+sub_menu,pos_comman
```

```
d+19,3+sub_menu,2);
          end;

procedure block_menu_device;
          begin
          background;
          sub_menu:=device_menu;
          text_xy(' Device ',pos_device,1,3,0);
          text_xy(' D',pos_device,1,3,4);
          back_ground(' ',pos_device-2,2,pos_device+20,20,7,1
5);
          text_xy('É',pos_device-2,2,7,0);
          text_xy('È',pos_device-2,20,7,0);
          text_xy('»',pos_device+20,2,7,0);
          text_xy('¼',pos_device+20,20,7,0);
          line_x(pos_device-1,pos_device+19,2,0);
          line_x(pos_device-1,pos_device+19,20,0);
          line_y(pos_device-2,3,19,0);
          line_y(pos_device+20,3,19,0);
          text_xy('AT89C1051',pos_device,3,7,0);
          text_xy('AT89C2051',pos_device,4,7,0);
          text_xy('AT89C4051',pos_device,5,7,0);
          text_xy('AT89C51',pos_device,6,7,0);
          text_xy('AT89C52',pos_device,7,7,0);
          text_xy('AT89C53',pos_device,8,7,0);
          text_xy('AT89C8252',pos_device,9,7,0);
          text_xy('28C64',pos_device,10,7,0);
          text_xy('28C128',pos_device,11,7,0);
          text_xy('27C64',pos_device,12,7,0);
          text_xy('27C128',pos_device,13,7,0);
          text_xy('AT24C02',pos_device,14,7,0);
          text_xy('AT24C04',pos_device,15,7,0);
          text_xy('AT24C08',pos_device,16,7,0);
          text_xy('AT93C46',pos_device,17,7,0);
          text_xy('AT93C56',pos_device,18,7,0);
          text_xy('AT93C66',pos_device,19,7,0);
          text_background(pos_device,3,pos_device+22,21,0);
          end;

procedure load_sub_device;
          begin
          case device_menu of
               0:begin
                 code_ic:=10;
                 tot_memory:=1024;
                 command.lock[1]:=true;
                 command.lock[2]:=true;
                 command.lock[3]:=true;
                 command.lock[4]:=true;
                 command.lock[5]:=true;
```

```
  command.lock[6]:=false;
  end;
1:begin
  code_ic:=11;
  tot_memory:=2048;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
  command.lock[6]:=false;
  end;
2:begin
  code_ic:=12;
  tot_memory:=4096;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
  command.lock[6]:=false;
  end;
3:begin
  code_ic:=1;
  tot_memory:=4096;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
  command.lock[6]:=true;
  end;
4:begin
  code_ic:=3;
  tot_memory:=8192;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
  command.lock[6]:=true;
  end;
5:begin
  code_ic:=4;
  tot_memory:=12288;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
```

```
  command.lock[6]:=true;
  end;
6:begin
  code_ic:=6;
  tot_memory:=10240;
  command.lock[1]:=true;
  command.lock[2]:=true;
  command.lock[3]:=true;
  command.lock[4]:=true;
  command.lock[5]:=true;
  command.lock[6]:=true;
  end;
7:begin
  code_ic:=20;
  tot_memory:=8192;
  command.lock[1]:=true;
  command.lock[2]:=false;
  command.lock[3]:=true;
  command.lock[4]:=false;
  command.lock[5]:=false;
  command.lock[6]:=false;
  end;
8:begin
  code_ic:=21;
  tot_memory:=16384;
  command.lock[1]:=true;
  command.lock[2]:=false;
  command.lock[3]:=true;
  command.lock[4]:=false;
  command.lock[5]:=false;
  command.lock[6]:=false;
  end;
9:begin
  code_ic:=30;
  tot_memory:=8192;
  command.lock[1]:=true;
  command.lock[2]:=false;
  command.lock[3]:=true;
  command.lock[4]:=false;
  command.lock[5]:=false;
  command.lock[6]:=false;
  end;
10:begin
   code_ic:=31;
   tot_memory:=16384;
   command.lock[1]:=true;
   command.lock[2]:=false;
   command.lock[3]:=true;
   command.lock[4]:=false;
   command.lock[5]:=false;
```

```
    command.lock[6]:=false;
    end;
11:begin
    code_ic:=40;
    tot_memory:=256;
    command.lock[1]:=true;
    command.lock[2]:=false;
    command.lock[3]:=true;
    command.lock[4]:=false;
    command.lock[5]:=false;
    command.lock[6]:=false;
    end;
12:begin
    code_ic:=41;
    tot_memory:=512;
    command.lock[1]:=true;
    command.lock[2]:=false;
    command.lock[3]:=true;
    command.lock[4]:=false;
    command.lock[5]:=false;
    command.lock[6]:=false;
    end;
13:begin
    code_ic:=42;
    tot_memory:=1024;
    command.lock[1]:=true;
    command.lock[2]:=false;
    command.lock[3]:=true;
    command.lock[4]:=false;
    command.lock[5]:=false;
    command.lock[6]:=false;
    end;
14:begin
    code_ic:=50;
    tot_memory:=128;
    command.lock[1]:=true;
    command.lock[2]:=false;
    command.lock[3]:=true;
    command.lock[4]:=false;
    command.lock[5]:=false;
    command.lock[6]:=false;
    end;
15:begin
    code_ic:=51;
    tot_memory:=256;
    command.lock[1]:=true;
    command.lock[2]:=false;
    command.lock[3]:=true;
    command.lock[4]:=false;
    command.lock[5]:=false;
```

```
                command.lock[6]:=false;
                end;
            16:begin
                code_ic:=52;
                tot_memory:=512;
                command.lock[1]:=true;
                command.lock[2]:=false;
                command.lock[3]:=true;
                command.lock[4]:=false;
                command.lock[5]:=false;
                command.lock[6]:=false;
                 end;
            end;
        end;

procedure sub_device;
        begin
        if sub_menu=17 then sub_menu:=0;
        if sub_menu=255 then sub_menu:=16;
        device_menu:=sub_menu;
        load_sub_device;
        text_background(pos_device-2,2,pos_device+20,20,7);
        text_background(pos_device-1,3+sub_menu,pos_device+
19,3+sub_menu,2);
        end;

procedure block_menu_help;
        begin
        background;
        text_xy(' Help ',pos_help,1,3,0);
        text_xy(' H',pos_help,1,3,4);
        text_background(pos_help-18,3,80,7,0);
        back_ground(' ',pos_help-20,2,80,6,7,15);
        text_xy('É',pos_help-20,2,7,0);
        text_xy('È',pos_help-20,6,7,0);
        text_xy('»',80,2,7,0);
        text_xy('¼',80,6,7,0);
        line_x(pos_help-19,79,2,0);
        line_x(pos_help-19,79,6,0);
        line_y(pos_help-20,3,5,0);
        line_y(80,3,5,0);
        text_xy(' Contact: Sin Fu          ',pos_help-17,3,
7,0);
        text_xy(' Phone: 031-5679993       ',pos_help-17,4,
7,0);
        text_xy(' Email: Silasilu@yahoo.com',pos_help-17,5,
7,0);
        end;

procedure default_menu;
```

```
          begin
          case menu of
               1:block_menu_file;
               2:block_menu_command;
               3:block_menu_device;
               4:block_menu_help;
               end;
          end;

procedure rutin_menu;
          begin
          case menu of
               1:sub_file;
               2:sub_command;
               3:sub_device;
               end;
          end;

procedure main_routine;
          begin
          if (tombol and $ff)=0 then
            begin
            alt_ok:=true;
            case tombol of                    {cek 'Alt'}
               $2100:menu:=1;
               $2E00:menu:=2;
               $2000:menu:=3;
               $2300:menu:=4;
               end;
            default_menu;
            end;
          if alt_ok then
            begin
            case ((tombol and $ff00)shr 8) of
                 $48:dec(sub_menu);
                 $50:inc(sub_menu);
                 $4d:begin
                     inc(menu);
                     if menu>4 then menu:=1;
                     default_menu;
                     end;
                 $4b:begin
                     dec(menu);
                     if menu=0 then menu:=4;
                     default_menu;
                     end;
                 end;
            end;
          rutin_menu;
          if (tombol and $ff)=27 then     {press Esc}
```

```
            begin
            background;
            alt_ok:=false;
            menu:=0;
            end;
         case tombol of
              $3b00:begin
                    block_menu_help;     {F1}
                    menu:=4;
                    alt_ok:=true;
                    end;
              $3c00:save;                {F2}
              $3d00:load_file;           {F3}
              $4300:download;            {F9}
              $4400:tombol:=$2D00;       {F10}
              $3042:display_buffer;
              $3062:display_buffer;
              $2044:display_file;
              $2064:display_file
              end;
         end;

begin     {MAIN PROGRAM}
mouseg.init;
mouseg.aturbatas(0,639,0,199);
mouseg.tampilkanmouse;
getintvec($c,rev);
setintvec($c,@receiver);
init_ser;
alt_ok:=false;
error:=false;
file_open:=false;
background;
device_menu:=3;
load_sub_device;
command_menu:=1;
load_sub_command;
repeat
   if keypress then
      begin
      tombol:=rdkey;
      main_routine;
      end;
   if mouseg.kejadian then
      begin
      mouseg.perolehxy(xmouse,ymouse);
      case mouseg.perolehtombol of
           1:begin
             if(xmouse>=24)and(xmouse<=71)and(ymouse=0)then
                begin
```

```
   tombol:=$2100;
   main_routine;
   end;
if(xmouse>=80)and(xmouse<=151)and(ymouse=0)then
   begin
   tombol:=$2E00;
   main_routine;
   end;
if(xmouse>=160)and(xmouse<=223)and(ymouse=0)then
   begin
   tombol:=$2000;
   main_routine;
   end;
if(xmouse>=504)and(xmouse<=552)and(ymouse=0)then
   begin
   tombol:=$2300;
   main_routine;
   end;
if alt_ok then
   begin
   case menu of
        1:begin
          if(xmouse>=8)and(xmouse<=183) then
            begin
            case ymouse of
                 16:begin
                    tombol:=$3d00;
                    file_menu:=0;
                    block_menu_file;
                    end;
                 24:begin
                    tombol:=$3c00;
                    file_menu:=1;
                    block_menu_file;
                    end;
                 32:begin
                    tombol:=$3042;
                    file_menu:=2;
                    block_menu_file;
                    end;
                 40:begin
                    tombol:=$2044;
                    file_menu:=3;
                    block_menu_file;
                    end;
                 48:begin
                    tombol:=$4300;
                    file_menu:=4;
                    block_menu_file;
                    end;
```

```
                                56:begin
                                   tombol:=$2d00;
                                   file_menu:=4;
                                   block_menu_file;
                                   end;
                                end;
                             main_routine;
                             end;
                          end;
                        2:begin
                          if(xmouse>=64)and(xmouse<=239) then
                            begin
                            case ymouse of
                                 16:command_menu:=0;
                                 24:command_menu:=1;
                                 32:command_menu:=2;
                                 40:command_menu:=3;
                                 48:command_menu:=4;
                                 56:command_menu:=5;
                                 end;
                            block_menu_command;
                            sub_command;
                            end;
                          end;
                        3:if(xmouse>=160)and(xmouse<=328)and(ymo
use>=16) then
                            begin
                            device_menu:=(ymouse-16) div 8;
                            if device_menu>16 then device_menu:=
16;
                            block_menu_device;
                            sub_device;
                            end;
                        end;
                    end;
              end;
            2:begin
              background;
              alt_ok:=false;
              menu:=0;
              end;
            end;
      repeat
      mouseg.tomboldilepas
      until not(mouseg.kejadian);
      end;
until (tombol=$2d00);
mouseg.done;
setintvec($c,rev);
back_ground(' ',1,1,80,25,0,7);
```

```
end.

(*          text_xy('  ',10,20,0,15);
          text_xy(inttostr(code_ic),10,20,0,15);
          text_xy('  ',10,22,0,15);
          if command.lock[command.command] then text_xy(inttos
tr(command.command),10,22,0,15);
          text_xy(word_hex(tombol),10,23,0,15);
                                    *)
```

```
;-------------------------------------------------------------
---------
;                    UNIVERSAL PROGRAMER BY YUDI
;                  UNIKA WIDYA MANDALA SURABAYA
;                         TUGAS AKHIR
;-------------------------------------------------------------
---------
                ORG     00H
                LJMP    MULAI

                ORG     28H
;-----------------------------
;          INISIALISASI
;-----------------------------
Init            MOV     TMOD,#00100010B         ;T1 mode2, T0
 mode2
                MOV     TL1,#0FFH               ;T1 mode2 gen
erate BR=9600 Bps
                MOV     TH1,#0FFH
                MOV     TCON,#01000000B         ;T1 on,T0 off
                MOV     SCON,#01010000B         ;Mode1,REN
                MOV     PCON,#80H               ;SMOD=0
                MOV     COUNT_PASSWORD,#0
                RET

;-----------------------------
;      DELAY RISE VOLTAGE
;-----------------------------
DELAY_RISE      MOV     A,#50
                DJNZ    A,$
                RET

;-----------------------------
;        DELAY SERMEM ON
;-----------------------------
DEL_SERMEM      MOV     R6,#255
DELAY_ON1       MOV     R7,#255
                DJNZ    R7,$
                DJNZ    R6,DELAY_ON1
                RET

;-----------------------------
;        CEK PASSWORD
;-----------------------------
CEK_PASSWORD    JNB     RI,$
                CLR     RI
                MOV     R5,SBUF
                MOV     A,COUNT_PASSWORD
                MOV     DPTR,#PASSWORD
                MOVC    A,@A+DPTR
```

```
                CJNE    A,5,PASSWORD_FALSE
                INC     COUNT_PASSWORD
                MOV     A,COUNT_PASSWORD
                CJNE    A,#10,CEK_PASSWORD   ;sudah 10 char/p
assword correct?
                MOV     COUNT_PASSWORD,#0
                MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                RET

PASSWORD_FALSE  MOV     SBUF,#0
                JNB     TI,$
                CLR     TI
                MOV     COUNT_PASSWORD,#0
                AJMP    CEK_PASSWORD
PASSWORD        DB      '5103098008'

;---------------------------
;       CONTROL BYTE
;---------------------------
CONTROL_BYTE    MOV     TEMP1,A
                ANL     A,#0F0H
                ORL     A,#0FH
                MOV     P2,A
                MOV     A,TEMP1
                SWAP    A
                ANL     A,#0F0H
                ORL     A,#0BH
                MOV     P3,A
                RET

;---------------------------
;       READ 89C5X
;---------------------------
READ_89C5X      MOV     A,#0
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                MOV     P1,#0FFH
                MOV     TEMP2,ADDR_MSB
                MOV     TEMP3,ADDR_LSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0

RD_89C5X_AGAIN  JNB     RI,$
                CLR     RI
                MOV     A,ADDR_MSB
                MOV     C,ACC.4
                MOV     P2.5,C
                MOV     C,ACC.5
```

```
                MOV     P3.4,C
                MOV     P0,ADDR_LSB
                ANL     P2,#0F0H
                MOV     A,ADDR_MSB
                ANL     A,#0FH
                ORL     P2,A
                MOV     A,#20
                DJNZ    A,$
                MOV     SBUF,P1
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PASS_89C5X_RD
                INC     ADDR_MSB
PASS_89C5X_RD   DJNZ    TEMP3,RD_89C5X_AGAIN
                DJNZ    TEMP2,RD_89C5X_AGAIN
                RET

;---------------------------
;       ERASE 89C5X
;---------------------------
ERASE_89C5X     MOV     A,#0
                MOV     DPTR,#READ_89C5X_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                MOV     A,#0
                MOV     DPTR,#ERASE_89C5X_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                ACALL   DELAY_RISE

                CLR     P3.7            ;PROG LOW 10 mS
                MOV     A,#15
DEL_ERASE_895X  MOV     R7,#250
                DJNZ    R7,$
                DJNZ    A,DEL_ERASE_895X
                MOV     A,#0
                MOV     DPTR,#READ_89C5X_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE

                MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                RET

;---------------------------
;       WRITE 89C5X
;---------------------------
```

```
WRITE_89C5X      ACALL   ERASE_89C5X
                 MOV     A,TEMP2
                 ACALL   CONTROL_BYTE
                 ACALL   DELAY_RISE
                 MOV     TEMP2,ADDR_LSB
                 MOV     TEMP3,ADDR_MSB
                 MOV     ADDR_LSB,#0
                 MOV     ADDR_MSB,#0
WR_89C5X_AGAIN   JNB     RI,$
                 CLR     RI
                 MOV     TEMP1,SBUF
                 MOV     A,ADDR_MSB
                 MOV     C,ACC.4
                 MOV     P2.5,C
                 MOV     C,ACC.5
                 MOV     P3.4,C
                 MOV     P0,ADDR_LSB
                 ANL     P2,#0F0H
                 MOV     A,ADDR_MSB
                 ANL     A,#0FH
                 ORL     P2,A
                 MOV     A,#5
                 DJNZ    A,$
                 MOV     P1,TEMP1
                 CLR     P3.7            ;PULSE PROG LOW
                 MOV     A,#2
DEL_PROG_89C5X   MOV     R7,#250         ;DELAY PULSE PROG
                 DJNZ    R7,$
                 DJNZ    A,DEL_PROG_89C5X
                 SETB    P3.7            ;PULSE PROG HIGH
                 MOV     A,#100          ;DELAY WAIT PROG
                 DJNZ    A,$
                 INC     ADDR_LSB
                 MOV     A,ADDR_LSB
                 JNZ     PASS_89C5X_WR
                 INC     ADDR_MSB
PASS_89C5X_WR    SETB    P3.0            ;MODE RECEIVE SERIAL
                 MOV     A,#100
                 DJNZ    A,$
                 MOV     SBUF,#0FFH
                 JNB     TI,$
                 CLR     TI
                 DJNZ    TEMP2,WR_89C5X_AGAIN
                 DJNZ    TEMP3,WR_89C5X_AGAIN
                 MOV     A,#0
                 MOV     DPTR,#READ_89C5X_CHR
                 MOVC    A,@A+DPTR
                 ACALL   CONTROL_BYTE
                 SETB    P2.5
                 SETB    P3.0
```

```
                RET

;---------------------------
;       LOCK BIT 89C5X
;---------------------------
PULSE_LOCK89C5X MOV     A,#0
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                ACALL   DELAY_RISE
                CLR     P3.7                    ;PULSE LOW
                MOV     A,#100
                DJNZ    A,$
                SETB    P3.7
                MOV     A,#200
                DJNZ    A,$
                MOV     A,#0
                MOV     DPTR,#READ_89C5X_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                RET

;---------------------------
;       RESET 89CX051
;---------------------------
RESET           MOV     A,#0
                MOV     DPTR,#RESET_X051
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                MOV     A,#10
                DJNZ    A,$
                MOV     A,#0
                MOV     DPTR,#READ_X051_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                RET

;---------------------------
;       BIT INVERSE
;---------------------------
BIT_INVERSE     mov     c,acc.7         ;bit inverse
                mov     temp,c
                mov     c,acc.0
                mov     acc.7,c
                mov     c,temp
                mov     acc.0,c

                mov     c,acc.6         ;bit inverse
                mov     temp,c
                mov     c,acc.1
                mov     acc.6,c
```

```
                mov     c,temp
                mov     acc.1,c

                mov     c,acc.5         ;bit inverse
                mov     temp,c
                mov     c,acc.2
                mov     acc.5,c
                mov     c,temp
                mov     acc.2,c

                mov     c,acc.4         ;bit inverse
                mov     temp,c
                mov     c,acc.3
                mov     acc.4,c
                mov     c,temp
                mov     acc.3,c
                RET

;---------------------------
;       READ 89CX051
;---------------------------
READ_89CX051    MOV     P1,#0FFH
                ACALL   RESET
READ_X051_AGAIN JNB     RI,$
                CLR     RI
                mov     a,p1
                ACALL   BIT_INVERSE
                MOV     SBUF,a
                JNB     TI,$
                CLR     TI
                SETB    P3.6
                CLR     P3.6
                DJNZ    ADDR_LSB,READ_X051_AGAIN
                DJNZ    ADDR_MSB,READ_X051_AGAIN
                RET

;---------------------------
;       ERASE 89CX051
;---------------------------
ERASE_89CX051   MOV     A,#0
                MOV     DPTR,#ERASE_X051_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                ACALL   DELAY_RISE
                CLR     P3.7                    ;PULSE PROG L
OW 10 mS
                MOV     A,#15
DEL_ERASE_X051  MOV     R7,#250
                DJNZ    R7,$
                DJNZ    A,DEL_ERASE_X051
```

```
                SETB    P3.7
                MOV     A,#0
                MOV     DPTR,#READ_X051_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                RET

;---------------------------
;       WRITE 89CX051
;---------------------------
WRITE_89CX051   ACALL   ERASE_89CX051
                ACALL   RESET
                MOV     A,TEMP2
                ACALL   CONTROL_BYTE
                ACALL   DELAY_RISE
WR_X051_AGAIN   JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                ACALL   BIT_INVERSE
                MOV     P1,A
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#100
                DJNZ    A,$
                SETB    P3.7
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.6                    ;PULSE INC
                CLR     P3.6
                MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                DJNZ    ADDR_LSB,WR_X051_AGAIN
                DJNZ    ADDR_MSB,WR_X051_AGAIN
                MOV     A,#0
                MOV     DPTR,#READ_X051_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                RET

;-----------------------
;    PULSE LOCK BIT
;-----------------------
PULSE_LOCKCX051 MOV     A,#0
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                ACALL   DELAY_RISE
                CLR     P3.7                    ;PULSE LOW
```

```
                MOV     A,#200
                DJNZ    A,$
                SETB    P3.7
                MOV     A,#200
                DJNZ    A,$
                MOV     A,#0
                MOV     DPTR,#READ_X051_CHR
                MOVC    A,@A+DPTR
                ACALL   CONTROL_BYTE
                RET

;-----------------------
;       READ EEPROM
;-----------------------
READ_EEPROM     MOV     P1,#0FFH
                ORL     P3,#0F8H
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
RD_EEPROM_AGAIN JNB     RI,$
                CLR     RI
                MOV     P0,ADDR_LSB
                MOV     P2,ADDR_MSB
                MOV     A,#10
                DJNZ    A,$
                CLR     P3.6
                MOV     TEMP1,P1
                SETB    P3.6
                MOV     SBUF,TEMP1
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PAS_EEPROM_RD
                INC     ADDR_MSB
PAS_EEPROM_RD   DJNZ    TEMP2,RD_EEPROM_AGAIN
                DJNZ    TEMP3,RD_EEPROM_AGAIN
                SETB    P3.6
                RET

;-----------------------
;       WRITE EEPROM
;-----------------------
WRITE_EEPROM    MOV     P1,#0FFH
                ORL     P3,#0F0H
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
```

```
WR_EEPROM_AGAIN MOV     P0,ADDR_LSB
                MOV     P2,ADDR_MSB
                MOV     A,#10
                DJNZ    A,$
                JNB     RI,$
                CLR     RI
                MOV     P1,SBUF
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.7
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PAS_EEPROM_WR
                INC     ADDR_MSB
PAS_EEPROM_WR   MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                DJNZ    TEMP2,WR_EEPROM_AGAIN
                DJNZ    TEMP3,WR_EEPROM_AGAIN
                SETB    P3.7
                RET

;------------------------
;   WRITE EEPROM LOCK
;------------------------
WRITE_EE_LOCK   ORL     P3,#0F0H
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
WR_EE_LK_AGAIN  JNB     RI,$
                CLR     RI
                ACALL   CLOSE_LOCK
                MOV     P0,ADDR_LSB
                MOV     P2,ADDR_MSB
                MOV     P1,SBUF
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.7
                ACALL   DEL_PROG_EE             ;tWC
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PAS_EE_LK_WR
                INC     ADDR_MSB
PAS_EE_LK_WR    MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                DJNZ    TEMP2,WR_EE_LK_AGAIN
```

```
                DJNZ    TEMP3,WR_EE_LK_AGAIN
                SETB    P3.7
                RET

;-------------------------
;       READ EPROM
;-------------------------
READ_EPROM      MOV     P1,#0FFH
                ORL     P3,#0F8H
                CLR     P3.4
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
RD_EPROM_AGAIN  JNB     RI,$
                CLR     RI
                MOV     P0,ADDR_LSB
                MOV     P2,ADDR_MSB
                MOV     A,#10
                DJNZ    A,$
                CLR     P3.6
                MOV     TEMP1,P1
                SETB    P3.6
                MOV     SBUF,TEMP1
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PAS_EPROM_RD
                INC     ADDR_MSB
PAS_EPROM_RD    DJNZ    TEMP2,RD_EPROM_AGAIN
                DJNZ    TEMP3,RD_EPROM_AGAIN
                SETB    P3.6
                ORL     P3,#0F8H
                RET

;-------------------------
;       DEL_PROG_EE
;-------------------------
DEL_PROG_EE     MOV     R2,#3
DEL_PROG        MOV     A,#250
                DJNZ    A,$
                DJNZ    R2,DEL_PROG
                RET

;-------------------------
;  LOCK LOAD AAH TO 1555H
;-------------------------
LOAD_AA@1555    MOV     P0,#55H                 ;LOAD 0AAH @1
555H
```

```
                MOV     P2,#15H
                MOV     P1,#0AAH
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.7
                RET

;--------------------------
;  LOCK LOAD 55H TO 0AAAH
;--------------------------
LOAD_55@AAA     MOV     P0,#0AAH                        ;LOAD
 0AAH @1555H
                MOV     P2,#0AH
                MOV     P1,#55H
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.7
                RET

;--------------------------
;  LOCK LOAD X TO 1555H
;--------------------------
LOAD_X@1555     MOV     P0,#55H                 ;LOAD 0AAH @1
555H
                MOV     P2,#15H
                MOV     P1,A
                CLR     P3.7                    ;PULSE PROG
                MOV     A,#250
                DJNZ    A,$
                SETB    P3.7
                RET

;--------------------------
;   OPEN LOCK ALGORITHM
;--------------------------
OPEN_LOCK       ACALL   LOAD_AA@1555
                ACALL   LOAD_55@AAA
                MOV     A,#80H
                ACALL   LOAD_X@1555
                ACALL   LOAD_AA@1555
                ACALL   LOAD_55@AAA
                MOV     A,#20H
                ACALL   LOAD_X@1555
                ACALL   DEL_PROG_EE
                RET

;------------------------
;  CLOSE LOCK ALGORITHM
```

```
;-------------------------
CLOSE_LOCK      ACALL   LOAD_AA@1555
                ACALL   LOAD_55@AAA
                MOV     A,#0A0H
                ACALL   LOAD_X@1555
                ACALL   DEL_PROG_EE
                RET

;-------------------------
;       WRITE EPROM
;-------------------------
WRITE_EPROM     MOV     P1,#0FFH
                ORL     P3,#0F0H
                CLR     P3.4
                CLR     P3.5
                ACALL   DELAY_RISE
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
WR_EPROM_AGAIN  MOV     P0,ADDR_LSB
                MOV     P2,ADDR_MSB
                MOV     A,#10
                DJNZ    A,$
                JNB     RI,$
                CLR     RI
                MOV     P1,SBUF

                CLR     P3.7                    ;PULSE PROG
                MOV     A,#50                   ;DELAY 50mS

DEL_PROG_EPROM  MOV     R7,#250
                DJNZ    R7,$
                DJNZ    A,DEL_PROG_EPROM
                SETB    P3.7

                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     PAS_EPROM_WR
                INC     ADDR_MSB
PAS_EPROM_WR    MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                DJNZ    TEMP2,WR_EPROM_AGAIN
                DJNZ    TEMP3,WR_EPROM_AGAIN
                SETB    P3.7
                RET

;-----------------------
;       START
```

```
;           READ
;-----------------------
READ_24          ACALL   START
                 MOV     A,ADDR_MSB
                 RL      A
                 ORL     A,#0A0H          ;CODE WRITE UTK MENEN
TUKAN ALAMAT
                 ACALL   PROTOCOL
                 MOV     A,ADDR_LSB
                 ACALL   PROTOCOL
                 ACALL   START
                 MOV     A,ADDR_MSB
                 RL      A
                 ORL     A,#0A1H          ;CODE BACA MEMORY
                 ACALL   PROTOCOL

                 MOV     R2,#8
                 SETB    P1.4             ;SDA --> INPUT
BACA1            SETB    P1.5
                 MOV     C,P1.4
                 RLC     A
                 CLR     P1.5
                 DJNZ    R2,BACA1
                 MOV     TEMP1,A
                 SETB    P1.4             ;CEK "NO ACK"
                 SETB    P1.5
                 CLR     P1.5
                 CLR     P1.4
                 ACALL   STOP
                 RET

;---------------------------
;       READ 24CXX
;---------------------------
READ_24CXX       MOV     P1,#0FFH
                 MOV     P1,#30H
                 ACALL   DEL_SERMEM
                 MOV     TEMP2,ADDR_MSB
                 MOV     TEMP3,ADDR_LSB
                 MOV     ADDR_LSB,#0
                 MOV     ADDR_MSB,#0
RD_24CXX_AGAIN   JNB     RI,$
                 CLR     RI
                 ACALL   READ_24
                 MOV     SBUF,TEMP1
                 JNB     TI,$
                 CLR     TI
                 INC     ADDR_LSB
                 MOV     A,ADDR_LSB
                 JNZ     PAS_24_RD
```

```
                RET

;----------------------------
;     CLOCK SERIAL 93CXX
;----------------------------
SK              CLR     P1.1
                SETB    P1.1
                CLR     P1.1
                RET

;----------------------------
;     START SERIAL 93CXX
;----------------------------
START_EE        SETB    P1.3            ;DO AS HIGH Z
                CLR     P1.0            ;CS DISABLE
                SETB    P1.2            ;DIN (START)
                SETB    P1.0            ;CS ENABLE
                ACALL   SK
                RET

;----------------------------
;          WRITE ENABLE
;----------------------------
EWEN            ACALL   START_EE
                CLR     P1.2            ;DIN (0)
                ACALL   SK
                CLR     P1.2            ;DIN (0)
                ACALL   SK
                SETB    P1.2            ;DIN (1)
                ACALL   SK
                SETB    P1.2            ;DIN (1)
                ACALL   SK
                MOV     A,#5
EWEN_SK         ACALL   SK
                DJNZ    A,EWEN_SK
                CLR     P1.0            ;CS DISABLE
                MOV     A,#200
                DJNZ    A,$
                RET

;----------------------------
;         WRITE 93C66
;----------------------------
WR_93C66        ACALL   EWEN
                ACALL   START_EE
                CLR     P1.2            ;DIN (0)
                ACALL   SK
                SETB    P1.2            ;DIN (1)
                ACALL   SK
```

```
                MOV     A,ADDR_MSB
                MOV     C,ACC.0
                MOV     P1.2,C
                ACALL   SK
                MOV     A,ADDR_LSB
                MOV     R7,#8
WR_ADDR_93C66   RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,WR_ADDR_93C66

                MOV     A,TEMP1
                MOV     R7,#8
                SETB    P1.3
WR_DATA_93C66   RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,WR_DATA_93C66
                CLR     P1.0            ;CS DISABLE
                RET

;----------------------------
;       READ 93C66
;----------------------------
RD_93C66        ACALL   START_EE
                SETB    P1.2            ;DIN(1)
                ACALL   SK
                CLR     P1.2            ;DIN (0)
                ACALL   SK

                MOV     A,ADDR_MSB
                MOV     C,ACC.0
                MOV     P1.2,C
                ACALL   SK
                MOV     A,ADDR_LSB

                MOV     R7,#8
RD_ADDR         RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,RD_ADDR

                MOV     R7,#8
                SETB    P1.3
RD_DATA         ACALL   SK
                MOV     C,P1.3
                RLC     A
                DJNZ    R7,RD_DATA
                CLR     P1.0            ;DISABLE CS
                MOV     TEMP1,A
```

```
                RET

;-----------------------------
;        WRITE 93C46
;-----------------------------
WR_93C46        SETB    P1.3
                ACALL   EWEN
                ACALL   START_EE
                CLR     P1.2            ;DIN (0)
                ACALL   SK
                SETB    P1.2            ;DIN (1)
                ACALL   SK

                MOV     A,ADDR_LSB
                RL      A
                MOV     R7,#7
WR_ADDR_46      RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,WR_ADDR_46

                MOV     A,TEMP1
                MOV     R7,#8
WR_DATA_93C46   RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,WR_DATA_93C46
                CLR     P1.0            ;CS DISABLE
                RET

;-----------------------------
;        READ 93C46
;-----------------------------
RD_93C46        SETB    P1.3
                ACALL   START_EE
                SETB    P1.2            ;DIN(1)
                ACALL   SK
                CLR     P1.2            ;DIN (0)
                ACALL   SK

                MOV     A,ADDR_LSB
                RL      A
                MOV     R7,#7
RD_ADDR_46      RLC     A
                MOV     P1.2,C
                ACALL   SK
                DJNZ    R7,RD_ADDR_46

                MOV     R7,#8
RD_DATA_46      ACALL   SK
```

```
                MOV     C,P1.3
                RLC     A
                DJNZ    R7,RD_DATA_46
                CLR     P1.0            ;DISABLE CS
                MOV     TEMP1,A
                RET

;---------------------------
;       READ 93C46
;---------------------------
READ_93C46      MOV     P1,#0FH
                ACALL   DEL_SERMEM
                MOV     TEMP2,ADDR_LSB
                MOV     ADDR_LSB,#0
RD_93C46_AGAIN  JNB     RI,$
                CLR     RI
                ACALL   RD_93C46
                MOV     SBUF,TEMP1
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                DJNZ    TEMP2,RD_93C46_AGAIN

                ACALL   DELAY93CXX
                MOV     P1,#0FFH
                RET

;---------------------------
;       WRITE 93C46
;---------------------------
WRITE_93C46     MOV     P1,#0FH
                ACALL   DEL_SERMEM
                MOV     TEMP2,ADDR_LSB
                MOV     ADDR_LSB,#0
WR_93C46_AGAIN  JNB     RI,$
                CLR     RI
                MOV     TEMP1,SBUF
                ACALL   WR_93C46
                MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                DJNZ    TEMP2,WR_93C46_AGAIN
                ACALL   DELAY93CXX
                MOV     P1,#0FFH
                RET

;---------------------------
;       READ 93C66
;---------------------------
```

```
READ_93C66      MOV     P1,#0FH
                ACALL   DEL_SERMEM
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
RD_93C66_AGAIN  JNB     RI,$
                CLR     RI
                ACALL   RD_93C66
                MOV     SBUF,TEMP1
                JNB     TI,$
                CLR     TI
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     RD_93C66_PAS
                INC     ADDR_MSB
RD_93C66_PAS    DJNZ    TEMP2,RD_93C66_AGAIN
                DJNZ    TEMP3,RD_93C66_AGAIN
                ACALL   DELAY93CXX
                MOV     P1,#0FFH
                RET

;---------------------------
;       WRITE 93C66
;---------------------------
WRITE_93C66     MOV     P1,#0FH
                ACALL   DEL_SERMEM
                MOV     TEMP2,ADDR_LSB
                MOV     TEMP3,ADDR_MSB
                MOV     ADDR_LSB,#0
                MOV     ADDR_MSB,#0
WR_93C66_AGAIN  JNB     RI,$
                CLR     RI
                MOV     TEMP1,SBUF
                ACALL   WR_93C66
                INC     ADDR_LSB
                MOV     A,ADDR_LSB
                JNZ     WR_93C66_PAS
                INC     ADDR_MSB
WR_93C66_PAS    MOV     SBUF,#0FFH
                JNB     TI,$
                CLR     TI
                DJNZ    TEMP2,WR_93C66_AGAIN
                DJNZ    TEMP3,WR_93C66_AGAIN
                ACALL   DELAY93CXX
                MOV     P1,#0FFH
                RET

;---------------------------
;   ROUTINE 89C51 FAMILY
```

```
;---------------------------
IC_89C51        MOV     A,COMMAND
                CJNE    A,#1,IC_89C51_PASS1
                MOV     DPTR,#READ_89C5X_CHR
                ACALL   READ_89C5X              ;READ
                AJMP    ESC_89C51
IC_89C51_PASS1  CJNE    A,#2,IC_89C51_PASS2
                ACALL   ERASE_89C5X
                AJMP    ESC_89C51
IC_89C51_PASS2  CJNE    A,#3,IC_89C51_PASS3
                MOV     A,#0
                MOV     DPTR,#WRITE_89C5X_CHR
                MOVC    A,@A+DPTR
                MOV     TEMP2,A
                ACALL   WRITE_89C5X
                AJMP    ESC_89C51
IC_89C51_PASS3  CJNE    A,#4,IC_89C51_PASS4
                MOV     DPTR,#LOCK1_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                AJMP    ESC_89C51
IC_89C51_PASS4  CJNE    A,#5,IC_89C51_PASS5
                MOV     DPTR,#LOCK1_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                MOV     DPTR,#LOCK2_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                AJMP    ESC_89C51
IC_89C51_PASS5  CJNE    A,#6,ESC_89C51
                MOV     DPTR,#LOCK1_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                MOV     DPTR,#LOCK2_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
ESC_89C51       LJMP    BACK

;---------------------------
;       89S8252 FAMILY
;---------------------------
IC_89S8252      MOV     A,COMMAND
                CJNE    A,#1,IC_89S82_PASS1
                MOV     DPTR,#READ_89S82_CHR
                ACALL   READ_89C5X              ;READ
                AJMP    ESC_89S82
IC_89S82_PASS1  CJNE    A,#2,IC_89S82_PASS2
                ACALL   ERASE_89C5X
                AJMP    ESC_89S82
IC_89S82_PASS2  CJNE    A,#3,IC_89S82_PASS3
                MOV     A,#0
                MOV     DPTR,#WRITE_89C5X_CHR
                MOVC    A,@A+DPTR
```

```
                MOV     TEMP2,A
                ACALL   WRITE_89C5X
                AJMP    ESC_89S82
IC_89S82_PASS3  CJNE    A,#4,IC_89S82_PASS4
                CLR     P1.7
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                AJMP    ESC_89S82
IC_89S82_PASS4  CJNE    A,#5,IC_89S82_PASS5
                CLR     P1.7
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                CLR     P1.6
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                AJMP    ESC_89S82
IC_89S82_PASS5  CJNE    A,#6,ESC_89S82
                CLR     P1.7
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                CLR     P1.6
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
                CLR     P1.5
                MOV     DPTR,#LOCK3_89C5X_CHR
                ACALL   PULSE_LOCK89C5X
ESC_89S82       LJMP    BACK

;--------------------------
;         89CX051
;--------------------------
IC_89CX051      MOV     A,COMMAND
                CJNE    A,#1,IC_89CX051_PAS1
                ACALL   READ_89CX051            ;READ
                AJMP    ESC_89CX051
IC_89CX051_PAS1 CJNE    A,#2,IC_89CX051_PAS2
                ACALL   ERASE_89CX051
                AJMP    ESC_89CX051
IC_89CX051_PAS2 CJNE    A,#3,IC_89CX051_PAS3
                MOV     A,#0
                MOV     DPTR,#WRITE_X051_CHR
                MOVC    A,@A+DPTR
                MOV     TEMP2,A
                ACALL   WRITE_89CX051
                AJMP    ESC_89CX051
IC_89CX051_PAS3 CJNE    A,#4,IC_89CX051_PAS4
                MOV     DPTR,#LOCK1_X051_CHR
                ACALL   PULSE_LOCKCX051
                AJMP    ESC_89CX051
IC_89CX051_PAS4 CJNE    A,#5,ESC_89CX051
```

```
                    MOV     DPTR,#LOCK1_X051_CHR
                    ACALL   PULSE_LOCKCX051
                    MOV     DPTR,#LOCK2_X051_CHR
                    ACALL   PULSE_LOCKCX051
ESC_89CX051         LJMP    BACK

;---------------------------
;          EEPROM
;---------------------------
EEPROM              MOV     A,COMMAND
                    CJNE    A,#1,EEPROM_PAS1
                    ACALL   READ_EEPROM
                    AJMP    ESC_EEPROM
EEPROM_PAS1         CJNE    A,#3,ESC_EEPROM
                    ACALL   WRITE_EEPROM
ESC_EEPROM          LJMP    BACK

;-----------------------------
;        EEPROM LOCK
;-----------------------------
EEPROM_LOCK         MOV     A,COMMAND
                    CJNE    A,#1,EE_LOCK_PAS1
                    ACALL   READ_EEPROM
                    AJMP    ESC_EEPROM
EE_LOCK_PAS1        CJNE    A,#3,ESC_EE_LOCK
                    ACALL   WRITE_EE_LOCK
                    ;ACALL  OPEN_LOCK
ESC_EE_LOCK         LJMP    BACK


;---------------------------
;           EPROM
;---------------------------
EPROM               MOV     A,COMMAND
                    CJNE    A,#1,EPROM_PAS1
                    LCALL   READ_EPROM
                    AJMP    ESC_EPROM
EPROM_PAS1          CJNE    A,#3,ESC_EPROM
                    LCALL   WRITE_EPROM
ESC_EPROM           LJMP    BACK

;---------------------------
;            24CXX
;---------------------------
IC_24CX             MOV     A,COMMAND
                    CJNE    A,#1,IC_24CXX_PAS1
                    ACALL   READ_24CXX
                    LJMP    ESC_24CXX
IC_24CXX_PAS1       CJNE    A,#2,ESC_24CXX
                    LCALL   WRITE_24CXX
```

```
ESC_24CXX       LJMP    BACK

;---------------------------
;          93C46
;---------------------------
IC_93C46        MOV     A,COMMAND
                CJNE    A,#1,IC_93C46_PAS1
                LCALL   READ_93C46
                SJMP    ESC_93C46
IC_93C46_PAS1   CJNE    A,#2,ESC_93C46
                LCALL   WRITE_93C46
ESC_93C46       LJMP    BACK

;---------------------------
;          93C66
;---------------------------
IC_93C66        MOV     A,COMMAND
                CJNE    A,#1,IC_93C66_PAS1
                LCALL   READ_93C66
                SJMP    ESC_93C66
IC_93C66_PAS1   CJNE    A,#2,ESC_93C66
                LCALL   WRITE_93C66
ESC_93C66       LJMP    BACK

;---------------------------
;   MAIN PROGRAM LOCATION
;---------------------------
MULAI           MOV     SP,#40H
                LCALL   INIT
ULANG           MOV     P3,#0FFH
                LCALL   CEK_PASSWORD
                MOV     R0,#CODE_IC
                MOV     R7,#4
GET_COMMAND     SETB    P3.2
                JNB     RI,$
                CLR     RI
                MOV     @R0,SBUF
                INC     R0
                DJNZ    R7,GET_COMMAND
                JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                CJNE    A,#0FFH,ULANG   ;CONTINUE ?
BACK            LJMP    ULANG

JUMP_CODE_IC    LJMP    BACK            ;0
                LJMP    IC_89C51        ;1      AT89C51
                LJMP    BACK            ;2
                LJMP    IC_89C51        ;3      AT89C52
                LJMP    IC_89S8252      ;4      AT89S53
```

```
LJMP    BACK            ;5
LJMP    IC_89S8252      ;6      AT89S8252
LJMP    BACK            ;7
LJMP    BACK            ;8
LJMP    BACK            ;9

LJMP    IC_89CX051      ;10     AT89C1051
LJMP    IC_89CX051      ;11     AT89C2051
LJMP    IC_89CX051      ;12     AT89C4051
LJMP    BACK            ;13
LJMP    BACK            ;14
LJMP    BACK            ;15
LJMP    BACK            ;16
LJMP    BACK            ;17
LJMP    BACK            ;18
LJMP    BACK            ;19

LJMP    EEPROM          ;20     28C64
LJMP    EEPROM          ;21     28C128
LJMP    EEPROM_LOCK     ;22     28C64B
LJMP    BACK            ;23
LJMP    BACK            ;24
LJMP    BACK            ;25
LJMP    BACK            ;26
LJMP    BACK            ;27
LJMP    BACK            ;28
LJMP    BACK            ;29

LJMP    EPROM           ;30     27C64
LJMP    EPROM           ;31     27C128
LJMP    BACK            ;32
LJMP    BACK            ;33
LJMP    BACK            ;34
LJMP    BACK            ;35
LJMP    BACK            ;36
LJMP    BACK            ;37
LJMP    BACK            ;38
LJMP    BACK            ;39

LJMP    IC_24CX         ;40     24C02
LJMP    IC_24CX         ;41     24C04
LJMP    IC_24CX         ;42     24C08
LJMP    BACK            ;43
LJMP    BACK            ;44
LJMP    BACK            ;45
LJMP    BACK            ;46
LJMP    BACK            ;47
LJMP    BACK            ;48
LJMP    BACK            ;49
```

```
LJMP    IC_93C46        ;50     93C46
LJMP    IC_93C66        ;51     93C56
LJMP    IC_93C66        ;52     93C66
LJMP    BACK            ;53
LJMP    BACK            ;54
LJMP    BACK            ;55
LJMP    BACK            ;56
LJMP    BACK            ;57
LJMP    BACK            ;58
LJMP    BACK            ;59
```

Foto perencanaan dan pembuatan alat pemrogram universal dengan menggunakan komunikasi serial.

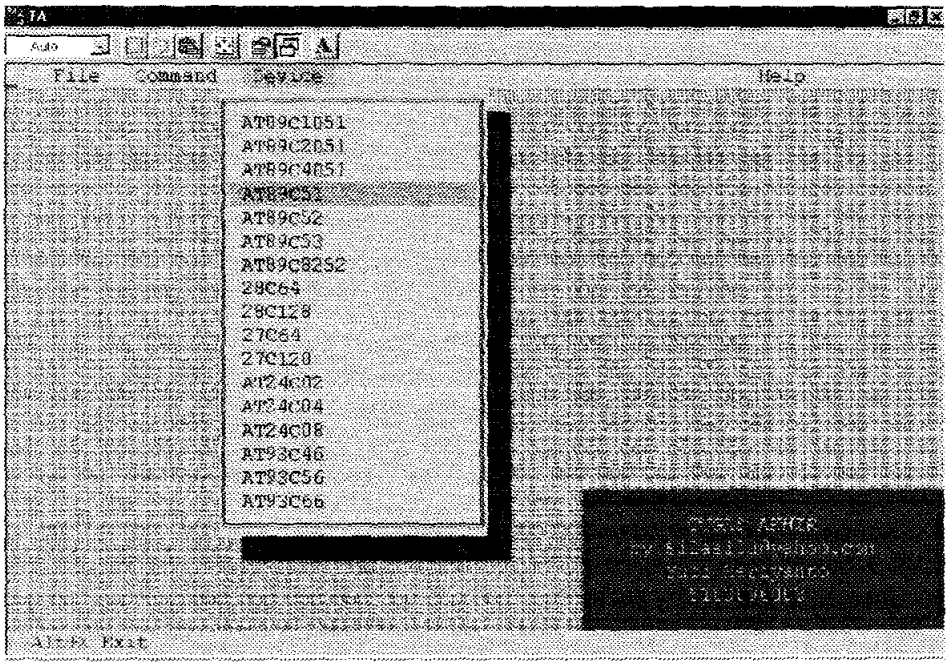


Foto Tampilan Pada Layar Monitor

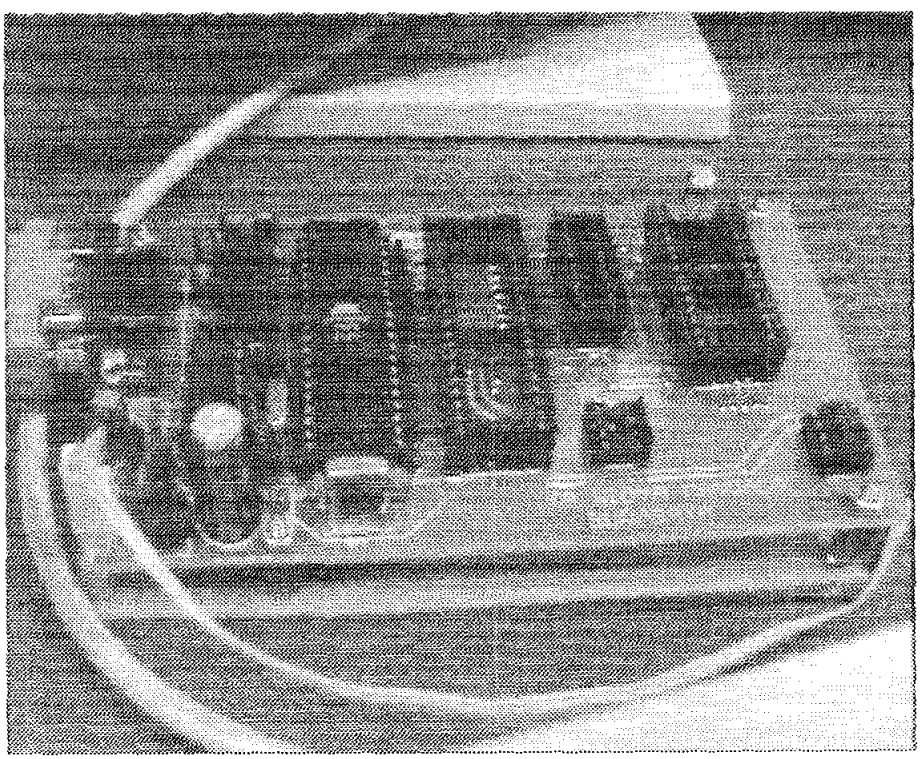


Foto Rangkaian Alat

eatures

- Compatible with MCS-51™ Products
- 4 Kbytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

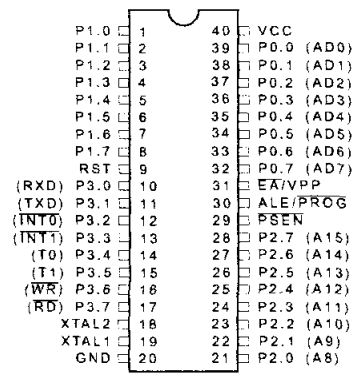
escription

e AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4 ytes of Flash Programmable and Erasable Read Only Memory (PEROM). The vice is manufactured using Atmel's high density nonvolatile memory technology d is compatible with the industry standard MCS-51™ instruction set and pinout. e on-chip Flash allows the program memory to be reprogrammed in-system or by conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU h Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer ich provides a highly flexible and cost effective solution to many embedded control plications.

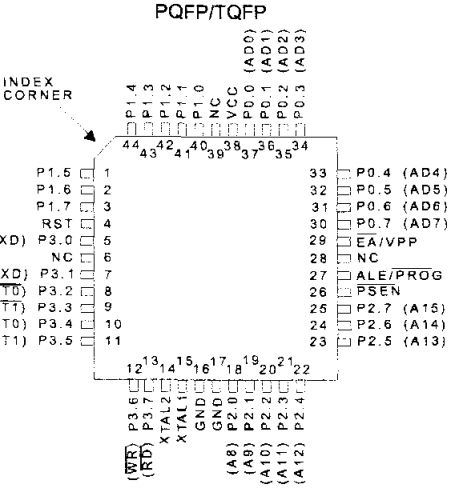
(continued)

n Configurations

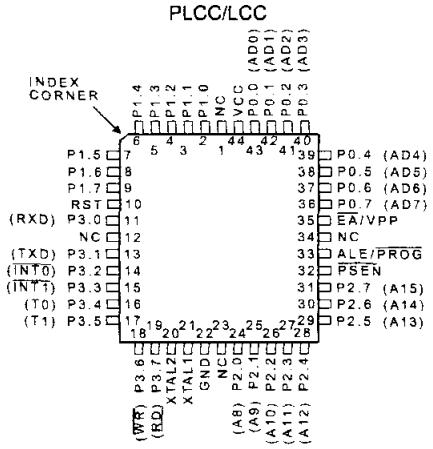
PDIP/Cerdip



PQFP/TQFP



PLCC/LCC

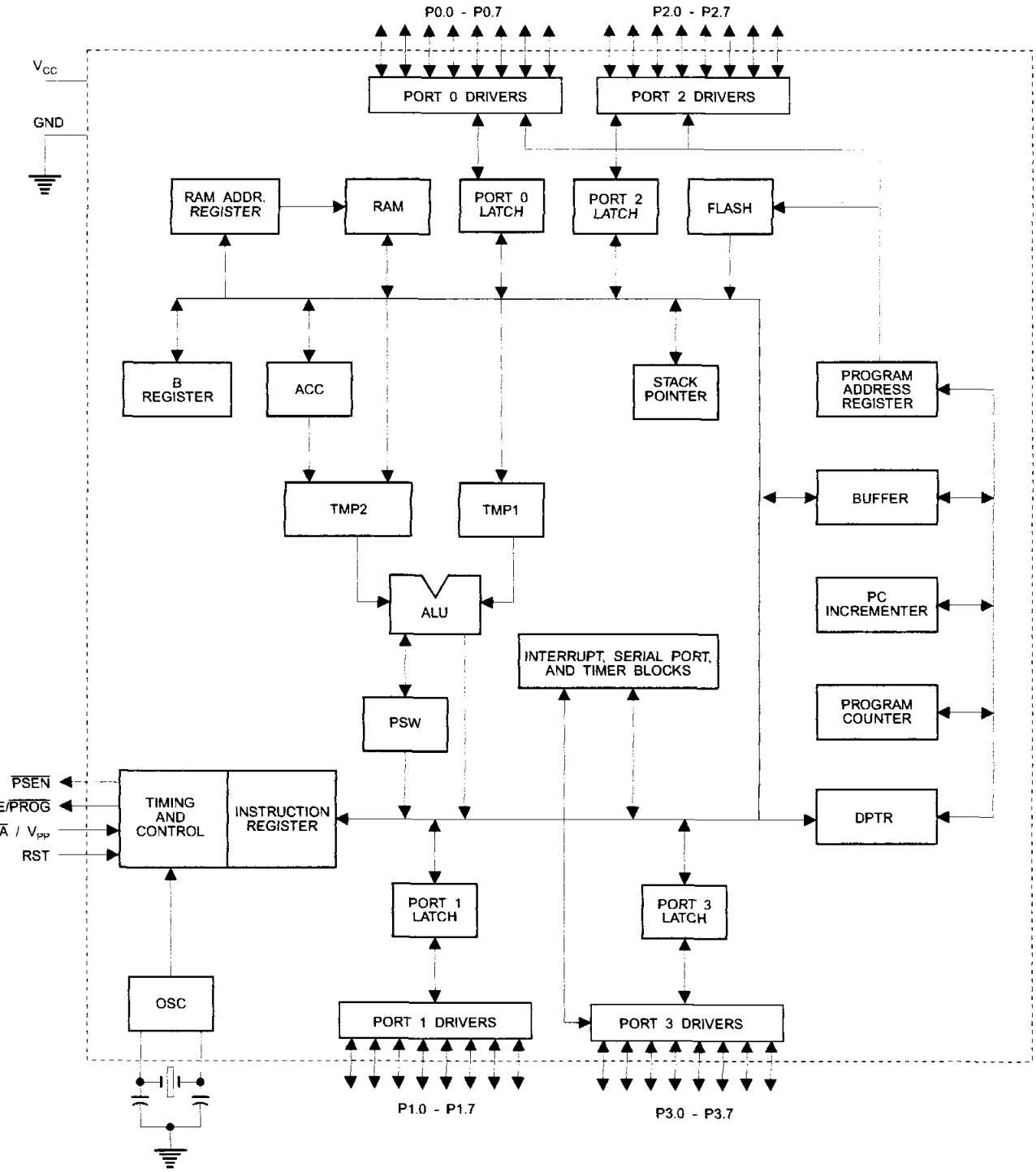


8-Bit Microcontroller with 4 Kbytes Flash

AT89C51

0265E

lock Diagram



escription (Continued)

e AT89C51 provides the following standard features: 4 ytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit er/counters, a five vector two-level interrupt architec-e, a full duplex serial port, on-chip oscillator and clock cuitry. In addition, the AT89C51 is designed with static ic for operation down to zero frequency and supports o software selectable power saving modes. The Idle de stops the CPU while allowing the RAM, timer/count-, serial port and interrupt system to continue function-. The Power Down Mode saves the RAM contents but ezes the oscillator disabling all other chip functions until next hardware reset.

n Description

c

pply voltage.

D

ound.

t 0

rt 0 is an 8-bit open drain bidirectional I/O port. As an put port each pin can sink eight TTL inputs. When 1s written to port 0 pins, the pins can be used as high-im-dance inputs.

rt 0 may also be configured to be the multiplexed low-er address/data bus during accesses to external pro-m and data memory. In this mode P0 has internal pul-s.

rt 0 also receives the code bytes during Flash program-ng, and outputs the code bytes during program verifica-n. External pullups are required during program verifica-n.

t 1

rt 1 is an 8-bit bidirectional I/O port with internal pullups. e Port 1 output buffers can sink/source four TTL inputs. en 1s are written to Port 1 pins they are pulled high by internal pullups and can be used as inputs. As inputs, rt 1 pins that are externally being pulled low will source rent (I_{IL}) because of the internal pullups.

rt 1 also receives the low-order address bytes during sh programming and program verification.

t 2

rt 2 is an 8-bit bidirectional I/O port with internal pullups. e Port 2 output buffers can sink/source four TTL inputs. en 1s are written to Port 2 pins they are pulled high by internal pullups and can be used as inputs. As inputs, rt 2 pins that are externally being pulled low will source rent (I_{IL}) because of the internal pullups.

rt 2 emits the high-order address byte during fetches m external program memory and during accesses to ernal data memory that use 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (extenal interrupt 0)
P3.3	$\overline{INT1}$ (extenal interrupt 1)
P3.4	T0 (timer 0 extenal input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (extenal data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and programming verification.

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcrontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

(continued)

n Description (Continued)

en the AT89C51 is executing code from external pro-m memory, PSEN is activated twice each machine cy-, except that two PSEN activations are skipped during ch access to external data memory.

/PP

ernal Access Enable. EA must be strapped to GND in er to enable the device to fetch code from external pro-m memory locations starting at 0000H up to FFFFH. te, however, that if lock bit 1 is programmed, EA will be rnally latched on reset.

should be strapped to V_{CC} for internal program execu-ns.

s pin also receives the 12-volt programming enable tage (V_{PP}) during Flash programming, for parts that re-re 12-volt V_{PP}.

AL1

ut to the inverting oscillator amplifier and input to the rnal clock operating circuit.

AL2

tput from the inverting oscillator amplifier.

scillator Characteristics

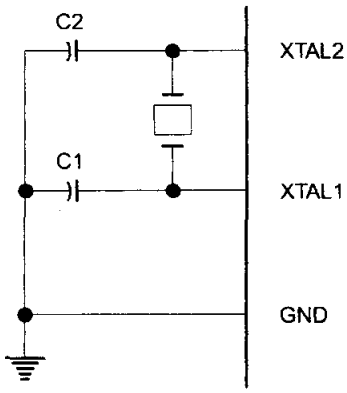
AL1 and XTAL2 are the input and output, respectively, an inverting amplifier which can be configured for use an on-chip oscillator, as shown in Figure 1. Either a artz crystal or ceramic resonator may be used. To drive device from an external clock source, XTAL2 should left unconnected while XTAL1 is driven as shown in ure 2. There are no requirements on the duty cycle of external clock signal, since the input to the internal cking circuitry is through a divide-by-two flip-flop, but nimum and maximum voltage high and low time specifi-ions must be observed.

le Mode

dle mode, the CPU puts itself to sleep while all the on-p peripherals remain active. The mode is invoked by tware. The content of the on-chip RAM and all the spe-l functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

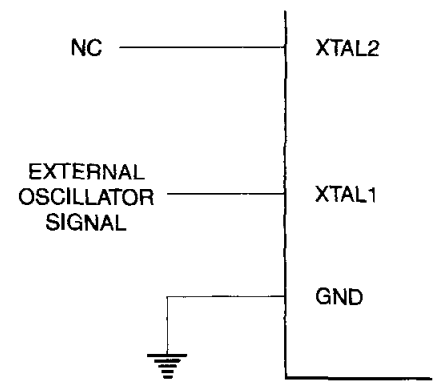
It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hard-

Figure 1. Oscillator Connections



Notes: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



atus of External Pins During Idle and Power Down

lode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
dle	Internal	1	1	Data	Data	Data	Data
dle	External	1	1	Float	Data	Address	Data
ower Down	Internal	0	0	Data	Data	Data	Data
ower Down	External	0	0	Float	Data	Data	Data

re inhibits access to internal RAM in this event, but ac-ss to the port pins is not inhibited. To eliminate the pos-ility of an unexpected write to a port pin when Idle is minated by reset, the instruction following the one that okes Idle should not be one that writes to a port pin or external memory.

ower Down Mode

he power down mode the oscillator is stopped, and the truction that invokes power down is the last instruction ecuted. The on-chip RAM and Special Function Regis-s retain their values until the power down mode is termi-ed. The only exit from power down is a hardware reset. set redefines the SFRs but does not change the on-p RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the Flash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

rogramming the Flash

e AT89C51 is normally shipped with the on-chip Flash mory array in the erased state (that is, contents = FFH) d ready to be programmed. The programming interface cepts either a high-voltage (12-volt) or a low-voltage cc) program enable signal. The low voltage program-ng mode provides a convenient way to program the 89C51 inside the user's system, while the high-voltage gramming mode is compatible with conventional third ty Flash or EPROM programmers.

e AT89C51 is shipped with either the high-voltage or -voltage programming mode enabled. The respective -side marking and device signature codes are listed in following table.

	V_{PP} = 12 V	V_{PP} = 5 V
op-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
ignature	(030H)=1EH (031H)=51H (032H)=FFH	(030H)=1EH (031H)=51H (032H)=05H

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12 V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

Data Polling: The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an at-

(continued)

rogramming the Flash (Continued)

npted read of the last byte written will result in the com-ement of the written datum on PO.7. Once the write cy- has been completed, true data are valid on all outputs, d the next cycle may begin. Data Polling may begin any e after a write cycle has been initiated.

ady/Busy: The progress of byte programming can o be monitored by the RDY/BSY output signal. P3.4 is lled low after ALE goes high during programming to in-cate BUSY. P3.4 is pulled high again when program-ng is done to indicate READY.

ogram Verify: If lock bits LB1 and LB2 have not been ogrammed, the programmed code data can be read ck via the address and data lines for verification. The k bits cannot be verified directly. Verification of the lock s is achieved by observing that their features are en-led.

ip Erase: The entire Flash array is erased electrically using the proper combination of control signals and by lding ALE/PROG low for 10 ms. The code array is writ-n with all "1"s. The chip erase operation must be exe-ted before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 51H indicates 89C51
(032H) = FFH indicates 12 V programming
(032H) = 05H indicates 5 V programming

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

ash Programming Modes

Mode		RST	PSEN	ALE/PROG	EA/VPP	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L	⎍	H/12V(1)	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L	⎍	H/12V	H	H	H	H
	Bit - 2	H	L	⎍ (2)	H/12V	H	H	L	L
	Bit - 3	H	L	⎍	H/12V	H	L	H	L
Chip Erase		H	L	⎍	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Notes: 1. The signature byte at location 032H designates whether V_{PP} = 12 V or V_{PP} = 5 V should be used to enable programming.

2. Chip Erase requires a 10 ms $\overline{PROG}$ pulse.

ure 3. Programming the Flash

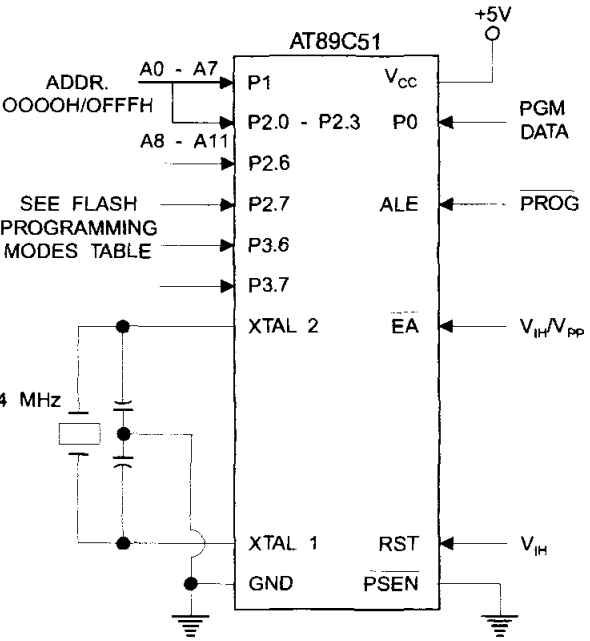
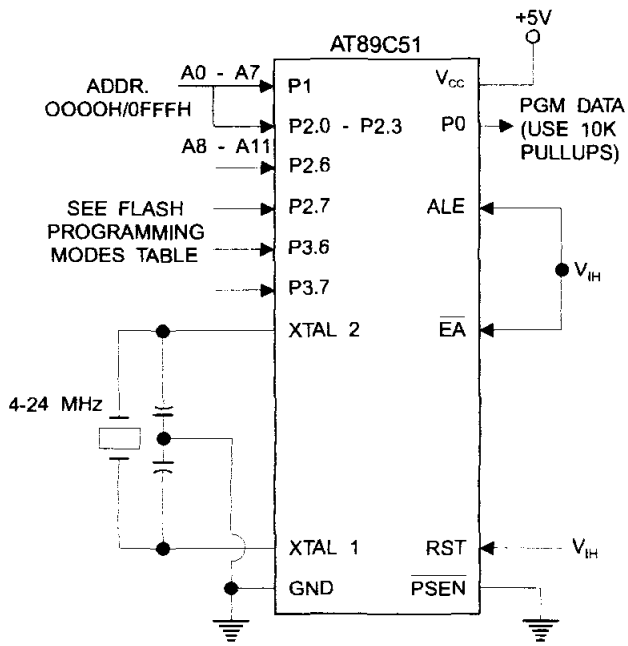


Figure 4. Verifying the Flash



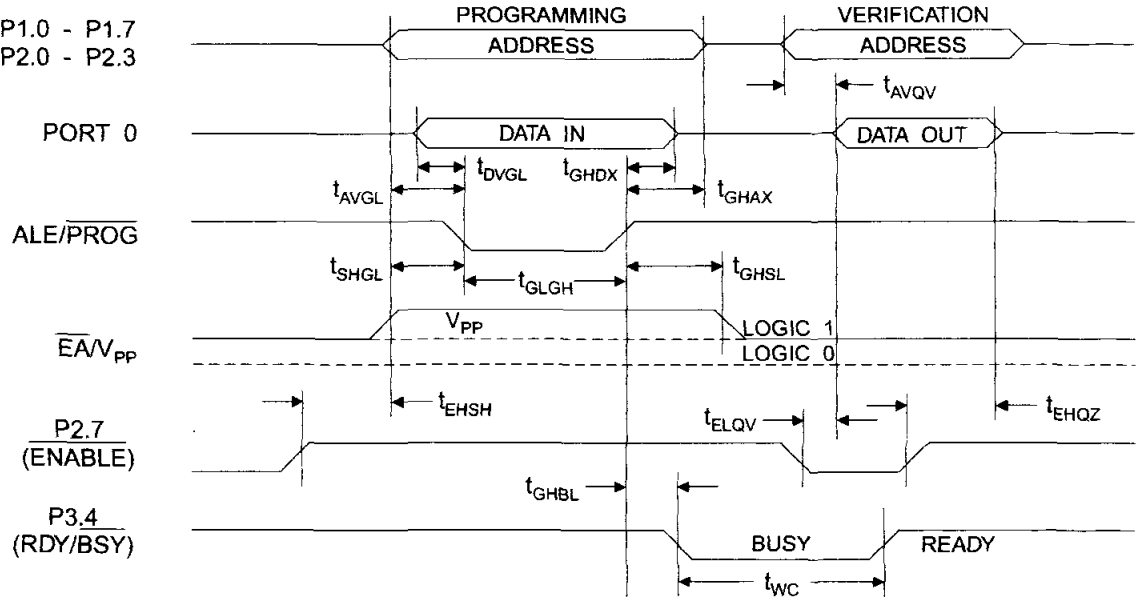
ash Programming and Verification Characteristics

= 21°C to 27°C, V_{CC} = 5.0 ± 10%

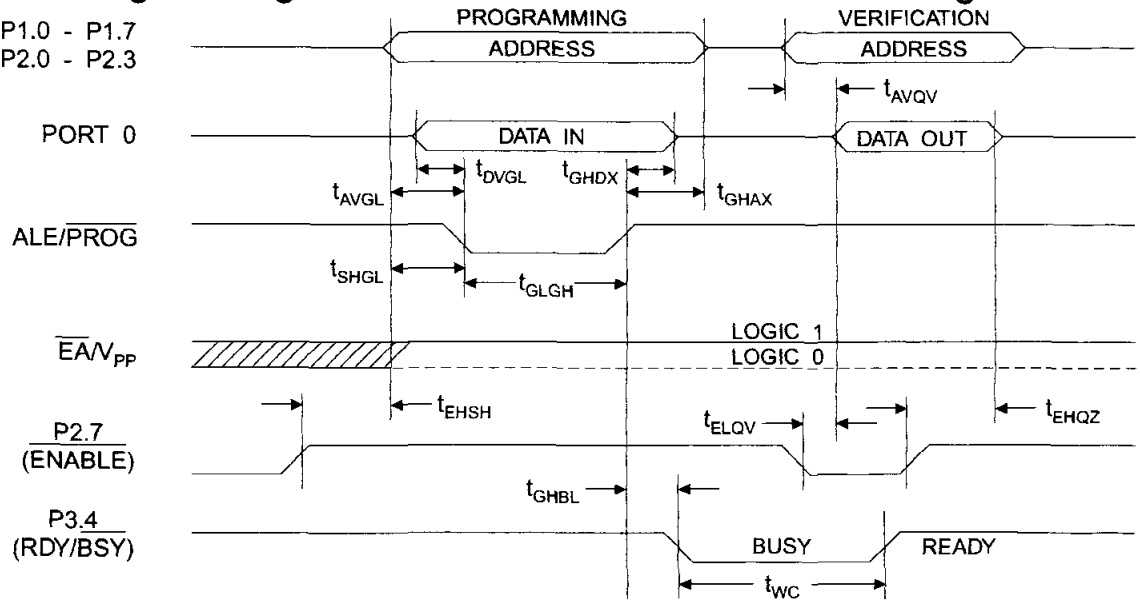
ymbol	Parameter	Min	Max	Units
PP(1)	Programming Enable Voltage	11.5	12.5	V
P(1)	Programming Enable Current		1.0	mA
tCLCL	Oscillator Frequency	4	24	MHz
VGL	Address Setup to $\overline{PROG}$ Low	48tCLCL		
HAX	Address Hold After $\overline{PROG}$	48tCLCL		
VGL	Data Setup to $\overline{PROG}$ Low	48tCLCL		
HDX	Data Hold After $\overline{PROG}$	48tCLCL		
HSH	P2.7 ($\overline{ENABLE}$) High to VPP	48tCLCL		
HGL	VPP Setup to $\overline{PROG}$ Low	10		µs
HSL(1)	VPP Hold After $\overline{PROG}$	10		µs
LGH	$\overline{PROG}$ Width	1	110	µs
VQV	Address to Data Valid		48tCLCL	
LQV	$\overline{ENABLE}$ Low to Data Valid		48tCLCL	
HQV	Data Float After $\overline{ENABLE}$	0	48tCLCL	
HBL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
/C	Byte Write Cycle Time		2.0	ms

e: 1. Only used in 12-volt programming mode.

ash Programming and Verification Waveforms - High Voltage Mode



ash Programming and Verification Waveforms - Low Voltage Mode



bsolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0 V to +7.0 V
Maximum Operating Voltage	6.6 V
DC Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

C. Characteristics

= -40°C to 85°C, V_{CC} = 5.0 V ± 20% (unless otherwise noted)

ymbol	Parameter	Condition	Min	Max	Units
L	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
L1	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
H	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
H1	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
OL	Output Low Voltage(1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
OL1	Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
OH	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
OH1	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45 V		-50	µA
L	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2 V		-650	µA
I	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RST	Reset Pulldown Resistor		50	300	KΩ
IO	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
C	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power Down Mode(2)	V_{CC} = 6 V		100	µA
		V_{CC} = 3 V		40	µA

es: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin:10 mA
Maximum I_{OL} per 8-bit port:
Port 0:26 mA
Ports 1,2, 3:15 mA
Maximum total IOL for all output pins:71 mA
If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum VCC for Power Down is 2 V.

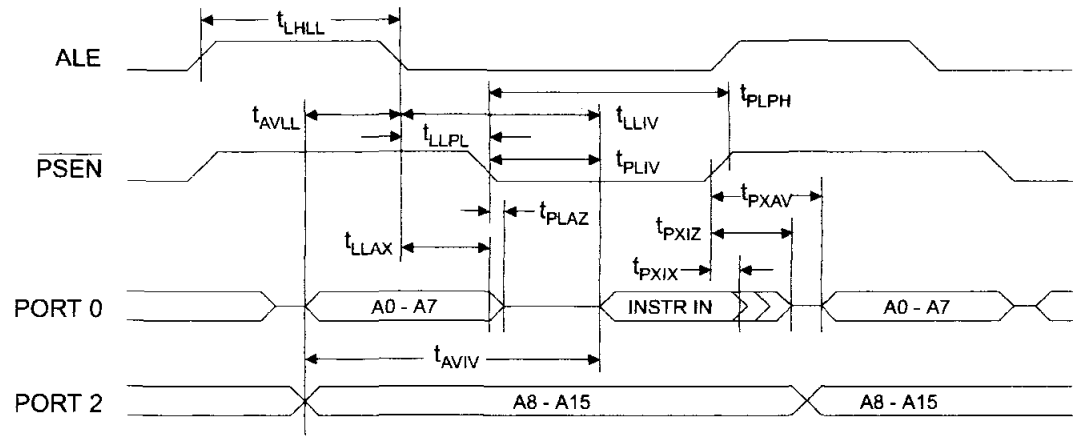
C. Characteristics

ıder Operating Conditions; Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all er outputs = 80 pF)

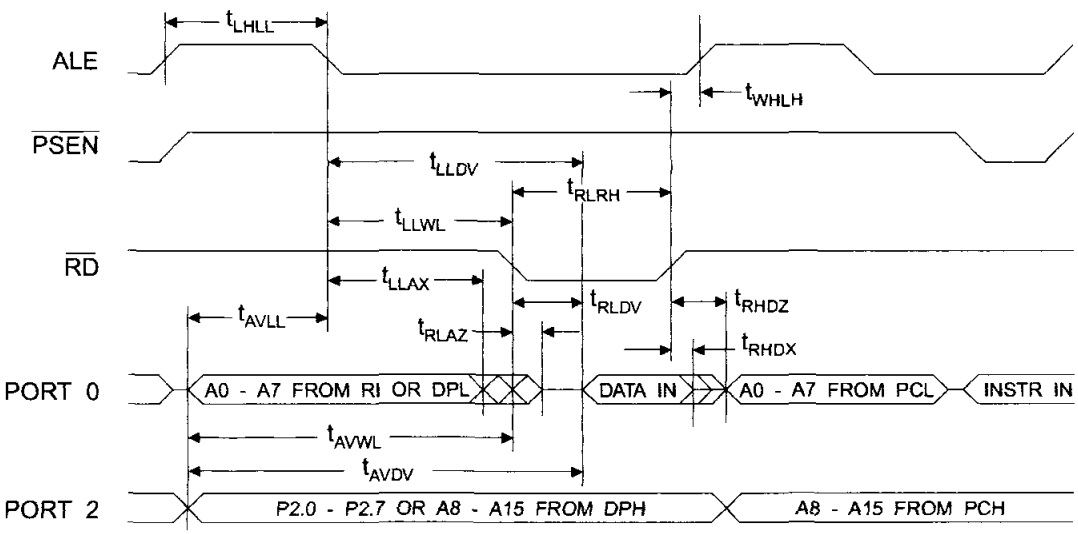
ıternal Program and Data Memory Characteristics

ymbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
/tCLCL	Oscillator Frequency			0	24	MHz
HLL	ALE Pulse Width	127		2tCLCL-40		ns
AVLL	Address Valid to ALE Low	28		tCLCL-13		ns
LAX	Address Hold After ALE Low	48		tCLCL-20		ns
LIV	ALE Low to Valid Instruction In		233		4tCLCL-65	ns
LPL	ALE Low to $\overline{PSEN}$ Low	43		tCLCL-13		ns
PLPH	$\overline{PSEN}$ Pulse Width	205		3tCLCL-20		ns
PLIV	$\overline{PSEN}$ Low to Valid Instruction In		145		3tCLCL-45	ns
PXIX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
PXIZ	Input Instruction Float After $\overline{PSEN}$		59		tCLCL-10	ns
PXAV	$\overline{PSEN}$ to Address Valid	75		tCLCL-8		ns
AVIV	Address to Valid Instruction In		312		5tCLCL-55	ns
PLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
RLRH	$\overline{RD}$ Pulse Width	400		6tCLCL-100		ns
WLWH	$\overline{WR}$ Pulse Width	400		6tCLCL-100		ns
RLDV	$\overline{RD}$ Low to Valid Data In		252		5tCLCL-90	ns
RHDX	Data Hold After $\overline{RD}$	0		0		ns
RHDZ	Data Float After $\overline{RD}$		97		2tCLCL-28	ns
LDV	ALE Low to Valid Data In		517		8tCLCL-150	ns
AVDV	Address to Valid Data In		585		9tCLCL-165	ns
LWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3tCLCL-50	3tCLCL+50	ns
AVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4tCLCL-75		ns
QVWX	Data Valid to $\overline{WR}$ Transition	23		tCLCL-20		ns
QVWH	Data Valid to $\overline{WR}$ High	433		7tCLCL-120		ns
WHQX	Data Hold After $\overline{WR}$	33		tCLCL-20		ns
RLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
WHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	tCLCL-20	tCLCL+25	ns

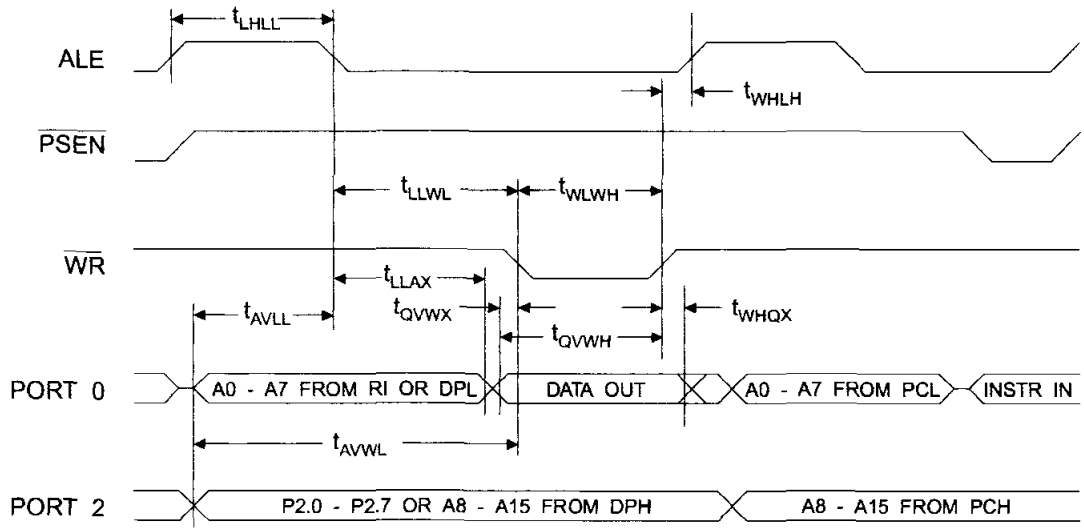
xternal Program Memory Read Cycle



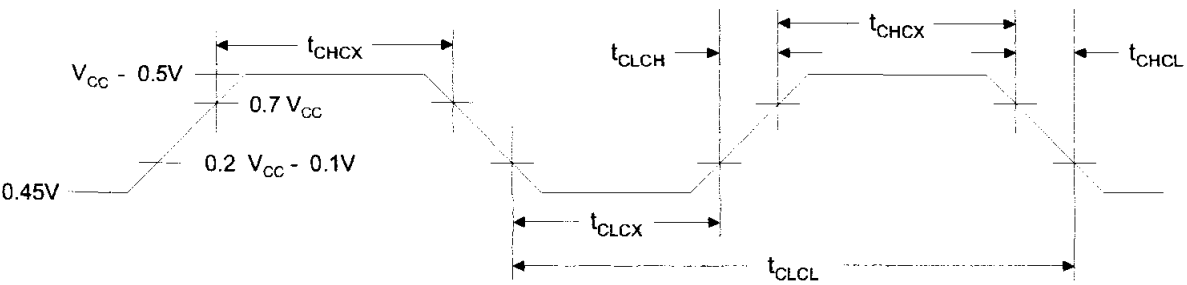
xternal Data Memory Read Cycle



xternal Data Memory Cycle



xternal Clock Drive Waveforms



xternal Clock Drive

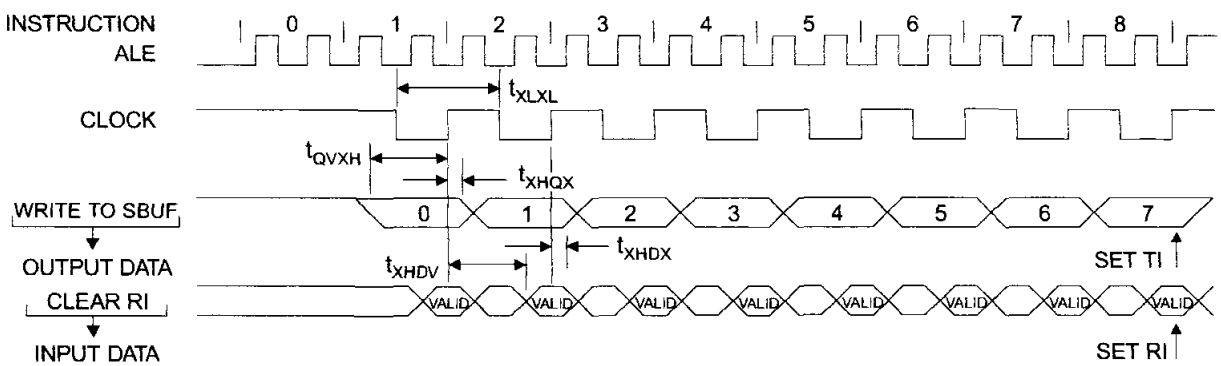
Symbol	Parameter	Min	Max	Units
1/tCLCL	Oscillator Frequency	0	24	MHz
CLCL	Clock Period	41.6		ns
CHCX	High Time	15		ns
CLCX	Low Time	15		ns
CLCH	Rise Time		20	ns
CHCL	Fall Time		20	ns

erial Port Timing: Shift Register Mode Test Conditions

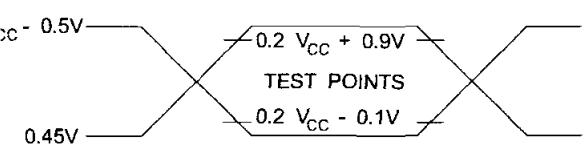
CC = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
XLXL	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
QVXH	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
XHQX	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-33		ns
XHDX	Input Data Hold After Clock Rising Edge	0		0		ns
XHDV	Clock Rising Edge to Input Data Valid		700		10tCLCL-133	ns

hift Register Mode Timing Waveforms

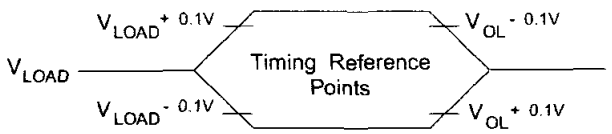


C Testing Input/Output Waveforms[1]



ote: 1. AC Inputs during testing are driven at V_{CC} - 0.5 V for a logic 1 and 0.45 V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms[1]



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

rdering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5 V ± 20%	AT89C51-12AC AT89C51-12JC AT89C51-12PC AT89C51-12QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-12AI AT89C51-12JI AT89C51-12PI AT89C51-12QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
		AT89C51-12AA AT89C51-12JA AT89C51-12PA AT89C51-12QA	44A 44J 40P6 44Q	Automotive (-40°C to 125°C)
	5 V ± 10%	AT89C51-12DM AT89C51-12LM	40D6 44L	Military (-55°C to 125°C)
		AT89C51-12DM/883 AT89C51-12LM/883	40D6 44L	Military/883C Class B, Fully Compliant (-55°C to 125°C)
16	5 V ± 20%	AT89C51-16AC AT89C51-16JC AT89C51-16PC AT89C51-16QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-16AI AT89C51-16JI AT89C51-16PI AT89C51-16QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
		AT89C51-16AA AT89C51-16JA AT89C51-16PA AT89C51-16QA	44A 44J 40P6 44Q	Automotive (-40°C to 125°C)
20	5 V ± 20%	AT89C51-20AC AT89C51-20JC AT89C51-20PC AT89C51-20QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-20AI AT89C51-20JI AT89C51-20PI AT89C51-20QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	5 V ± 20%	AT89C51-24AC AT89C51-24JC AT89C51-24PC AT89C51-24QC	44A 44J 44P6 44Q	Commercial (0°C to 70°C)
		AT89C51-24AI AT89C51-24JI AT89C51-24PI AT89C51-24QI	44A 44J 44P6 44Q	Industrial (-40°C to 85°C)

Package Type	
44A	44 Lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
40D6	40 Lead, 0.600" Wide, Non-Windowed, Ceramic Dual Inline Package (Cerdip)
44J	44 Lead, Plastic J-Leaded Chip Carrier (PLCC)
44L	44 Pad, Non-Windowed, Ceramic Leadless Chip Carrier (LCC)
40P6	40 Lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44Q	44 Lead, Plastic Gull Wing Quad Flatpack (PQFP)

BIODATA

Nama	**: Yudi Hariyanto**
NRP	**: 5103098008**
Tempat / Tanggal lahir	**: Probolinggo / 12 Juli 1981**
Alamat	**: JL. Kembang Kuning 24A Surabaya**
Agama	**: Kristen**

PENDIDIKAN :

- **SDK ST MGR. SOEGIJAPRANATA TANGGUL – JEMBER**
- **SMPK BHARA WIDYA LUMAJANG**
- **SMUK ST. LOUIS I SURABAYA**
- **UNIKA WIDYA MANDALA SURABAYA**