LAMPIRAN

LAMPIRAN I

```
program Kirim;
uses crt,dos;
const
      Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
      dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
      lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
      msr=$3FE;irq4=$C;ocw=$20;


var
 cx:byte;
 i, brpkl, datake,wktakt,sik:integer;
 vektor:pointer;cocok:boolean;
 jmldata:array[0..2] of integer;

procedure cetak;
interrupt;
   begin
        jmldata[datake]:=port[Rxb];
        inc(datake,1);
        port[ocw]:=eoi;
   end;


Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$0B;
     port[Txb]:=data;
end;


procedure siapbaca;
```

```
begin
  port[ier]:=1;
  port[mcr]:=8;
  port[imr]:=port[imr] and $EF;
end;

procedure initserial;
begin
  getintvec(irg4,vektor);
  setintvec(irg4,@cetak);
  siapbaca;
  prepare;
end;

 begin
  textbackground(1);
  cocok:=false;
  clrscr;
  prepare;
  GOToXY(25,8);  write(´waktu aktivasi (dalam menit) =   ´);readln(wktakt)
  GOToXY(25,10);  write(´siklik ?       Ya=1   /tidak=0 =   ´);readln(sik);
  while sik=1 do
  begin
  gotoxy(25,12); write(´berapa kali cyclik ?   ´);readln(sik);
  end;
  while cocok=false do
  begin
  gotoxy(25,14);write(´kirim protokol´);ngirim(prot1);delay(11);
  gotoxy(25,16);write(´kirim wktakt´);ngirim(wktakt);delay(11);
  gotoxy(25,18);write(´akan kirim sik´);ngirim(sik);delay(11);
  datake:=0;
  clrscr;
  initserial;
  while datake <=2 do
    begin
         gotoxy(25,8);write(´tunggu interupsi´);
    end;
 clrscr;
 gotoxy(25,12); writeln(´ data telah diterima´);
  setintvec(irg4,vektor);
  port[imr]:=port[imr] or $10;
   writeln(´ ´);
   writeln(´                    protokol:´,jmldata[0]);
   writeln(´                    waktu aktivasi:´,jmldata[1]);
       if jmldata[2]<>0 then
       begin
   writeln(´                    siklik:´,jmldata[2]);
       end;
          if jmldata[2]=0 then
          begin
   writeln(´                    no siklik´);
          end;
if jmldata[0]=prot1 then
begin
   if jmldata[1]=wktakt then
    begin
      if jmldata[2]=sik then
         begin
         cocok:=true;
         ngirim(01);delay(11);
         end
         else
         begin
         ngirim(0);delay(11);
         end;
    end
    Else
    begin
    ngirim(0);delay(11);
    end
end
else
begin
ngirim(0);delay(11);
end;
end;
writeln(´ ´);
   writeln(´                    cocok=    ´,cocok);
end.
```

LAMPIRAN II

```
Wed Jun  3 1998 11:20
                                          Page    1



                   2500 A.D. 8051 Macro Assembler  -  Version 4.01d
                   ------------------------------------------------

                           Input  Filename : c:\user\yusak\inout.asm
                           Output Filename : c:\user\yusak\inout.obj


  1    0000                                       org 0000h
  2    0000    02 01 00                           ljmp prog
  3
  4    0023                                       org 0023h
  5    0023    02 01 AA                           ljmp servis
  6
  7    0100                                       org 0100h
  8    0100    75 87 80          prog             mov pcon,#80h
  9    0103    75 89 20                           mov tmod,#20h
 10    0106    75 98 50                           mov scon,#50h
 11    0109    75 88 50                           mov tcon,#50h
 12    010C    75 8D F3                           mov th1,#f3h
 13
 14    010F    D2 AF                              setb ea
 15    0111    D2 BC                              setb ps
 16    0113    D2 AC             ulang            setb es
 17
 18    0115    C2 90                              clr p1.0
 19    0117    C2 91                              clr p1.1
 20    0119    C2 92                              clr p1.2
 21    011B    C2 93                              clr p1.3
 22
 23
 24
 25    011D    7A 00                              mov r2,#0      ;tanda prot=
 26    011F    78 72                              mov r0,#72h   ;alamat data
```

```

0121   79 70                    mov r1,#70h ;tanda telah di
0123   77 00                    mov @r1,#00

0125   79 69                    mov r1,#69h ;tanda cocok
0127   77 00                    mov @r1,#00

0129   79 69      tunggu        mov r1,#69h
012B   E7                       mov a,@r1
012C   60 FB                    jz tunggu

012E   BA 02 FD   start         cjne r2,#02h,start ;akan mu
bekerja
0131   7A 00                    mov r2,#00h

0133   C2 93      siklik        clr p1.3
0135   C2 91                    clr p1.1
0137   C2 90                    clr p1.0
0139   C2 AC                    clr es
013B   E4                       clr A

013C   D2 93                    setb p1.3 ;katup penghenti
```

```
013E   78 6E .                  mov r0,#6eH
0140   D2 90      aktivasi      setb p1.0 ;katup pengirim
0142   12 02 2E                 call delay_1ms
0145   04                       inc A
0146   30 95 F7                 jnb p1.5,aktivasi
0149   F6                       mov @r0,A
014A   C2 90                    clr p1.0
014C   79 72                    mov r1,#72H
014E   E7                       mov A,@r1
014F   D2 92      putar         setb p1.2 ;katup pemutar
0151   12 02 17                 call delay_1dt
0154   14                       dec a
0155   70 F8                    jnz putar
0157   C2 92                    clr p1.2
```

```
0159    E4                                  clr A
015A    C2 93                               clr p1.3 ;tutup penghenti
015C    78 6F                               mov r0,#6fH
015E    D2 91               analisa         setb p1.1  ;katup pengemb
0160    12 02 2E                            call delay_1ms
0163    04                                  inc A
0164    30 96 F7                            jnb p1.6,analisa
0167    F6                                  mov @r0,A
0168    74 03                               mov a,#03H
016A    12 02 24                            call kirim
016D    D2 AC                               setb es
016F    BA 03 FD            tdksama         cjne r2,#03h,tdksama
0172    7A 00                               mov r2,#00h
0174    C2 AC                               clr es
0176    C2 91                               clr p1.1
0178    78 71                               mov r0,#71h
017A    E6                                  mov a,@r0
017B    60 0D                               jz simpandata
017D    14                                  dec a
017E    F6                                  mov @r0,a
017F    70 B2                               jnz siklik

0181    D2 AC                               setb es
0183    BA 04 FD            kirimdata       cjne r2,#04h,kirimdata
0186    7A 00                               mov r2,#00
0188    C2 AC                               clr es
018A    78 6E               simpandata      mov r0,#6eH
018C    E6                                  mov a,@r0
018D    12 02 24                            call kirim
0190    78 6F                               mov r0,#6fH
0192    E6                                  mov a,@r0
0193    12 02 24                            call kirim
0196    78 6D                               mov r0,#6dh
0198    E6                                  mov a,@r0
0199    12 02 24                            call kirim
019C    D2 93                               setb p1.3  ;katup pengher
019E    D2 91               selesai         setb p1.1  ;katup pengemb
01A0    30 94 FB                            jnb p1.4,selesai
01A3    C2 91                               clr p1.1
01A5    C2 90                               clr p1.0
01A7    02 01 13                            ljmp ulang
```

```
01AA   C0 E0         servis        push a
01AC   C0 D0                       push psw
01AE   C2 98                       clr ri
01B0   E5 99                       mov a,sbuf
01B2   BA 00 06                    cjne r2,#0,label1

01B7   FA                          mov r2,a
01B8   02 02 09                    ljmp balik

01BB   BA 01 4B      label1        cjne r2,#01,balik

01C0   C0 E0                       push a
01C2   79 70                       mov r1,#70h
01C4   E7                          mov a,@r1
01C5   60 16                       jz isidata
01C7   C2 92                       clr p1.2
01C9   D0 E0                       pop a
01CB   79 69                       mov r1,#69h
01CD   F7                          mov @r1,a
01CE   7A 00                       mov r2,#00  ;tanda prot=0
01D0   79 70                       mov r1,#70h
01D2   77 00                       mov @r1,#00 ;tanda dikirii
01D4   79 71                       mov r1,#71h
01D6   E7                          mov a,@r1
01D7   79 6D                       mov r1,#6dh
01D9   F7                          mov @r1,a
01DA   02 02 09                    ljmp balik


01DF   D0 E0       isi data       pop a
01E1   F6                         mov @r0,a
01E2   18                         dec r0
01E3   B8 70 23                   cjne r0,#70h,balik

01E8   C2 AC                      clr es
```

```
01EA    EA                                  mov a,r2
01EB    12 02 24                            call kirim

01F0    78 72                               mov r0,#72h
01F2    E6                  lagi            mov a,@r0
01F3    12 02 24                            call kirim
01F6    18                                  dec r0
01F7    B8 70 F8                            cjne r0,#70h,lagi

01FC    79 70                               mov r1,#70h
01FE    77 01                               mov @r1,#01
0200    79 69                               mov r1,#69h
0202    78 72                               mov r0,#72h

0204    D2 AC                               setb es

0206    02 02 09                            ljmp balik


0209    D0 D0               balik           pop psw
020B    D0 E0                               pop a
020D    32                                  reti


Wed Jun  3 1998 11:20                   Page     4



020E    7B FF               delay           mov r3,#ffh
0210    7C F8               dua             mov r4,#f8h
0212    DC FE               satu            djnz r4,satu
0214    DB FA                               djnz r3,dua
0216    22                                  ret

0217    7D 08               delay_1dt     mov r5,#08H
0219    7C F5               lima          mov r4,#f5H
021B    7B FE               empat          mov r3,#feH
021D    DB FE               tiga          djnz r3,tiga
021F    DC FA                             djnz r4,empat
0221    DD F6                             djnz r5,lima
0223    22                                 RET
```

```
176
177   0224   F5 99          kirim         mov sbuf,a
178   0226   30 99 FD                     jnb ti,$
179   0229   C2 99                         clr ti
180   022B   51 0E                         call delay
181   022D   22                            ret
182
183   022E   7B 02          delay_1ms       mov r3,#02h
184   0230   7C F8          tujuh           mov r4,#f8h
185   0232   DC FE          delapan         djnz r4,delapan
186   0234   DB FA                          djnz r3,tujuh
187   0236   22                             ret
188
189   0237   7B 02          delay_1mnt      mov r3,#02h
190   0239   7C E8          enam            mov r4,#E8H
191   023B   7D FC          ttiga           mov r5,#fch
192   023D   7E FF          ddua            mov r6,#ffh
193   023F   DE FE          ssatu           djnz r6,ssatu
194   0241   DD FA                          djnz r5,ddua
195   0243   DC F6                          djnz r4,ttiga
196   0245   7E 2A                          mov r6,#2ah
197   0247   7A 55          llima           mov r2,#55h
198   0249   DD FE          eempat           djnz r5,eempat
199   024B   DE FA                          djnz r6,llima
200   024D   DB EA                          djnz r3,enam
201   024F   22                             ret
202


        Lines Assembled :  202              Assembly Errors :  0
```

LAMPIRAN III

```
program MULAI_MENGERJAKAN_PROSES;
uses crt,dos;
const
     Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
     dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
     lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
     msr=$3FE;irq4=$C;ocw=$20;


var
 data:byte;


Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$08;
     port[Txb]:=data;
end;

 begin
  textbackground(1);
  clrscr;
prepare;
  gotoxy(25,14);writeln('akan mulai mengerjakan proses');
  gotoxy(25,16);write('tekan ENTER untuk memulai');readln;
  ngirim(02);
end.
```

LAMPIRAN IV

```
program TERIMA_03;
uses crt,dos;
const
     Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
     dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
     lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
     msr=$3FE;irg4=$C;ocw=$20;


var
 cx:byte;
 i, brpkl, datake,wktakt,sik:integer;
 vektor:pointer;cocok:boolean;
 jmldata:array[0..2] of integer;

procedure cetak;
interrupt;
   begin
       jmldata[datake]:=port[Rxb];
       inc(datake,1);
       port[ocw]:=eoi;
   end;


Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$08;
     port[Txb]:=data;
end;


procedure siapbaca;
```

```
begin
  port[ier]:=1;
  port[mcr]:=8;
  port[imr]:=port[imr] and $EF;
end;

procedure initserial;
begin
  getintvec(irq4,vektor);
  setintvec(irq4,@cetak);
  siapbaca;
  prepare;
end;

 begin
  textbackground(1);
  clrscr;
  datake:=0;
  initserial;
  while datake <=0 do
    begin
         gotoxy(25,4);write('tunggu interupsi');
    end;
  clrscr;

  setintvec(irq4,vektor);
  port[imr]:=port[imr] or $10;


  gotoxy(25,20);writeln('data   ',jmldata[0],'     diterima');
END.
```

LAMPIRAN V

```
program KIRIM_03_TERIMA_WAKTU_TRANSFER;
uses crt,dos;
const
     Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
     dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
     lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
     msr=$3FE;irq4=$C;ocw=$20;


var
 cx:byte;
 i, brpkl, datake,wktakt,sik:integer;
 vektor:pointer;cocok:boolean;
 jmldata:array[0..2] of integer;

procedure cetak;
interrupt;
   begin
       jmldata[datake]:=port[Rxb];
       inc(datake,1);
       port[ocw]:=eoi;
   end;


Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$0B;
     port[Txb]:=data;
end;


procedure siapbaca;
```

```
begin
  port[ier]:=1;
  port[mcr]:=8;
  port[imr]:=port[imr] and $EF;
end;

procedure initserial;
begin
  getintvec(irq4,vektor);
  setintvec(irq4,@cetak);
  siapbaca;
  prepare;
end;

 begin
  textbackground(1);
  clrscr;
  prepare;
  ngirim(03);delay(11);ngirim(04);
  gotoxy(25,14);write('03 diterima');
  datake:=0;
   initserial;
  while datake <=1 do
    begin
         gotoxy(25,16);write('tunggu interupsi');
    end;
    setintvec(irq4,vektor);
  port[imr]:=port[imr] or $10;
 CLRSCR;
 gotoxy(25,10);writeln('DATA WAKTU TRANSFER 1 DAN 2');
 gotoxy(25,12);writeln('waktu transfer1 :  ',jmldata[0],'          detik')
 gotoxy(25,13);writeln('waktu transfer2 :  ',jmldata[1],'          detik')
 gotoxy(25,14);write('operasi ini tanpa siklik');
 gotoxy(25,17);wrkte('operasi telah selesai');
 gotoxy(25,18);write('tekan ENTER untuk keluar !');readln;
end.
```

LAMPIRAN VI

```
program KIRIM_03;
uses crt,dos;
const
     Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
     dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
     lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
     msr=$3FE;irq4=$C;ocw=$20;


var
 data:byte;

Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$0B;
     port[Txb]:=data;
end;


procedure siapbaca;
begin
  port[ier]:=1;
  port[mcr]:=8;
  port[imr]:=port[imr] and $EF;
end;

 begin
  textbackground(1);
  clrscr;
  prepare;
  ngirim(03);
 END.
```

LAMPIRAN VII

```
program KIRIM_04_TERIMA_WAKTU_TRANSFER;
uses crt,dos;
const
     Txb=$3F8;lcr=$3FB;mcr=$3fC;dll=$3f8;
     dlm=$3F9;prot1=01;prot2=2;prot3=3;Rxb=$3F8;
     lsr=$3FD;iir=$3FA;ier=$3F9;eoi=$20;imr=$21;
     msr=$3FE;irg4=$C;ocw=$20;


var
 cx:byte;
 i, brpkl, datake,wktakt,sik:integer;
 vektor:pointer;cocok:boolean;
 jmldata:array[0..2] of integer;

procedure cetak;
interrupt;
   begin
       jmldata[datake]:=port[Rxb];
       inc(datake,1);
       port[ocw]:=eoi;
   end;


Procedure prepare;
begin
  textcolor(yellow);
  port[lcr]:=$83;
  port[dlm]:=0;
  port[dll]:=$18;
  port[lcr]:=port[lcr] and $7F;
end;


procedure ngirim(data:byte);
begin
     port[mcr]:=$0B;
     port[Txb]:=data;
end;


procedure siapbaca;
```

```
begin
  port[ier]:=1;
  port[mcr]:=8;
  port[imr]:=port[imr] and $EF;
end;

procedure initserial;
begin
  getintvec(irq4,vektor);
  setintvec(irq4,@cetak);
  siapbaca;
  prepare;
end;

 begin
  textbackground(1);
  clrscr;
  prepare;
  ngirim(04);
  gotoxy(25,14);write('04 diterima');
  datake:=0;
   initserial;
  while datake <=2 do
    begin
          gotoxy(25,16);write('tunggu interupsi');
    end;
    setintvec(irq4,vektor);
  port[imr]:=port[imr] or $10;
 CLRSCR;
 gotoxy(25,10);writeln('DATA WAKTU TRANSFER 1 DAN 2');
 gotoxy(25,12);writeln('waktu transfer1 :   ',jmldata[0],'    detik');
 gotoxy(25,13);writeln('waktu transfer2 :   ',jmldata[1],'    detik');
 gotoxy(25,14);write('jumlah siklik sebanyak    ',jmldata[2],'    kali ');
 gotoxy(25,17);write('operasi telah selesai');
 gotoxy(25,18);write('tekan ENTER untuk keluar !');readln;
end.
```

LAMPIRAN VIII

MCS – 51 INSTRUCTION SET

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag C	Flag OV	Flag AC	Instruction	Flag C	Flag OV	Flag AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry*	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)			
INC DPTR	Increment Data Pointer	1	24
MUL AB	Multiply A & B	1	48
DIV AB	Divide A by B	1	48
DA A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS			
ANL A,Rn	AND Register to Accumulator	1	12
ANL A,direct	AND direct byte to Accumulator	2	12
ANL A,@Ri	AND indirect RAM to Accumulator	1	12
ANL A,#data	AND immediate data to Accumulator	2	12
ANL direct,A	AND Accumulator to direct byte	2	12
ANL direct,#data	AND immediate data to direct byte	3	24
ORL A,Rn	OR register to Accumulator	1	12
ORL A,direct	OR direct byte to Accumulator	2	12
ORL A,@Ri	OR indirect RAM to Accumulator	1	12
ORL A,#data	OR immediate data to Accumulator	2	12
ORL direct,A	OR Accumulator to direct byte	2	12
ORL direct,#data	OR immediate data to direct byte	3	24
XRL A,Rn	Exclusive-OR register to Accumulator	1	12
XRL A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR A	Clear Accumulator	1	12
CPL A	Complement Accumulator	1	12

Mnemonic	Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)			
RL A	Rotate Accumulator Left	1	12
RLC A	Rotate Accumulator Left through the Carry	1	12
RR A	Rotate Accumulator Right	1	12
RRC A	Rotate Accumulator Right through the Carry	1	12
SWAP A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER			
MOV A,Rn	Move register to Accumulator	1	12
MOV A,direct	Move direct byte to Accumulator	2	12
MOV A,@Ri	Move indirect RAM to Accumulator	1	12
MOV A,#data	Move immediate data to Accumulator	2	12
MOV Rn,A	Move Accumulator to register	1	12
MOV Rn,direct	Move direct byte to register	2	24
MOV Rn,#data	Move immediate data to register	2	12
MOV direct,A	Move Accumulator to direct byte	2	12
MOV direct,Rn	Move register to direct byte	2	24
MOV direct,direct	Move direct byte to direct	3	24
MOV direct,@Ri	Move indirect RAM to direct byte	2	24
MOV direct,#data	Move immediate data to direct byte	3	24
MOV @Ri,A	Move Accumulator to indirect RAM	1	12

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange Indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	

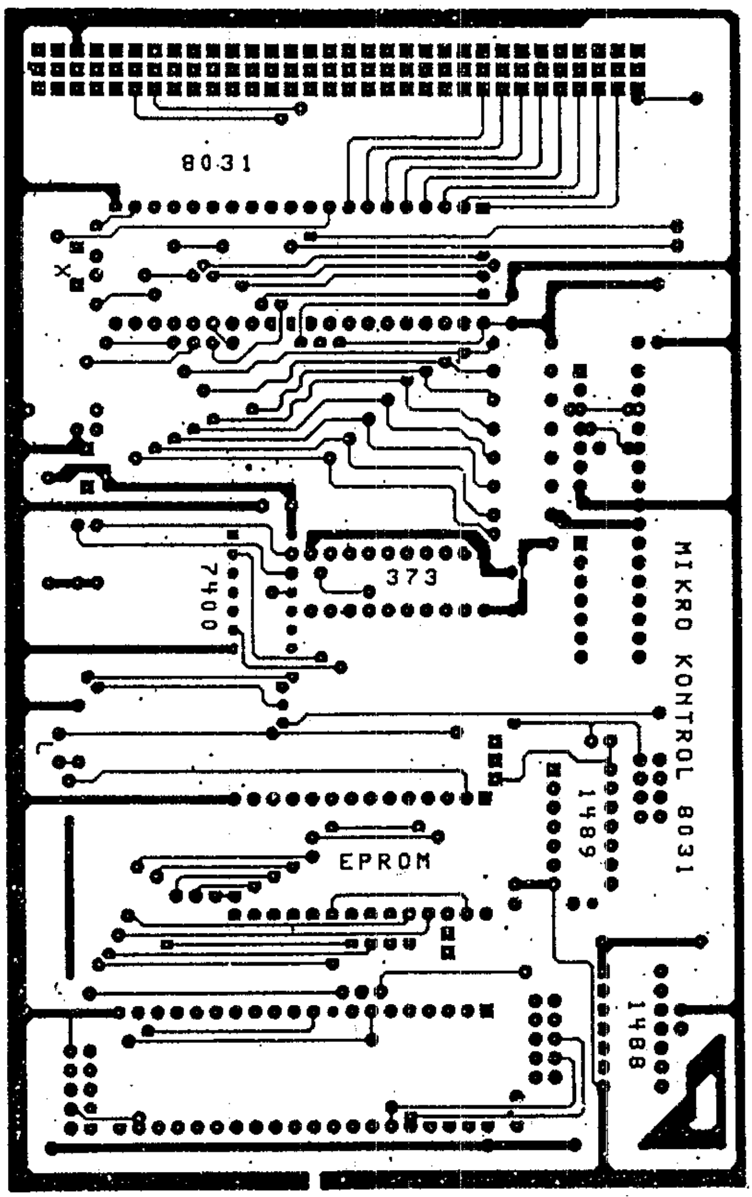
Hex Code	Number of Bytes	Mnemonic	Operands
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0

Hex Code	Number of Bytes	Mnemonic	Operands
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr

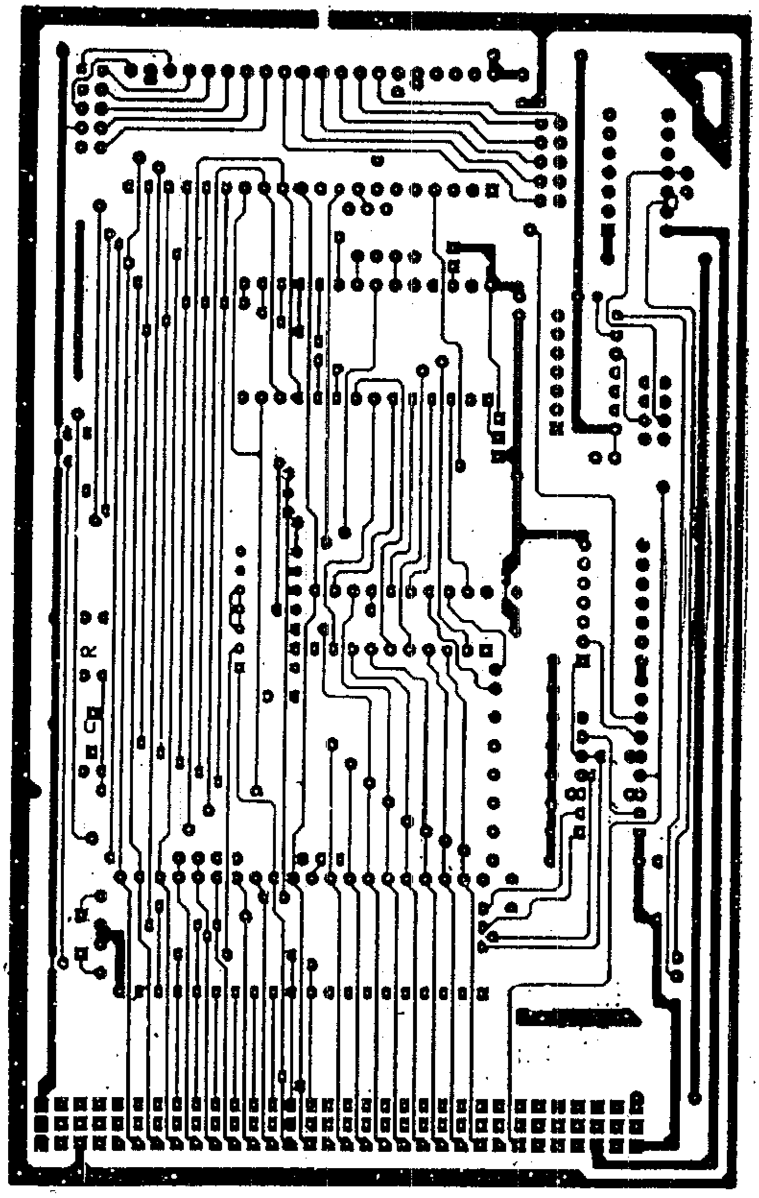
Hex Code	Number of Bytes	Mnemonic	Operands
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

LAMPIRAN IX



Gambar alur PCB sisi komponen board pengontrol

LAMPIRAN X



Gambar: alur PCB sisi solder board pengontrol