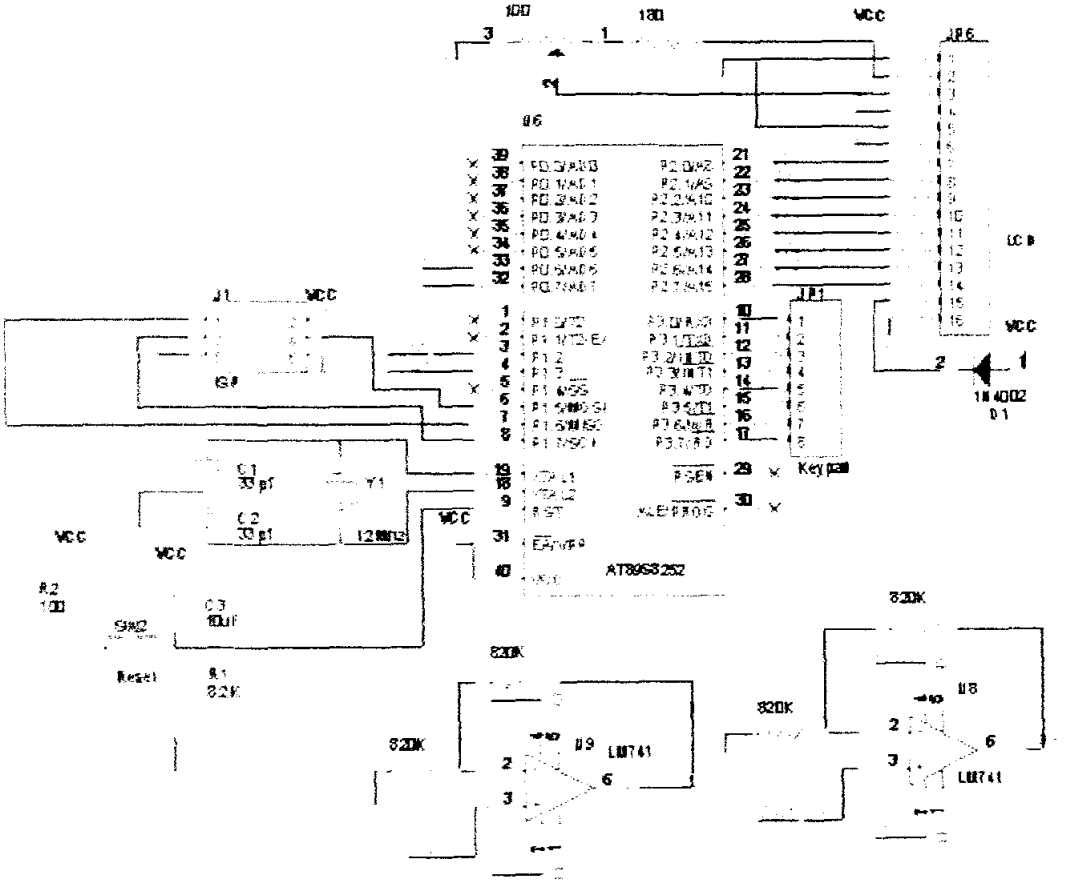
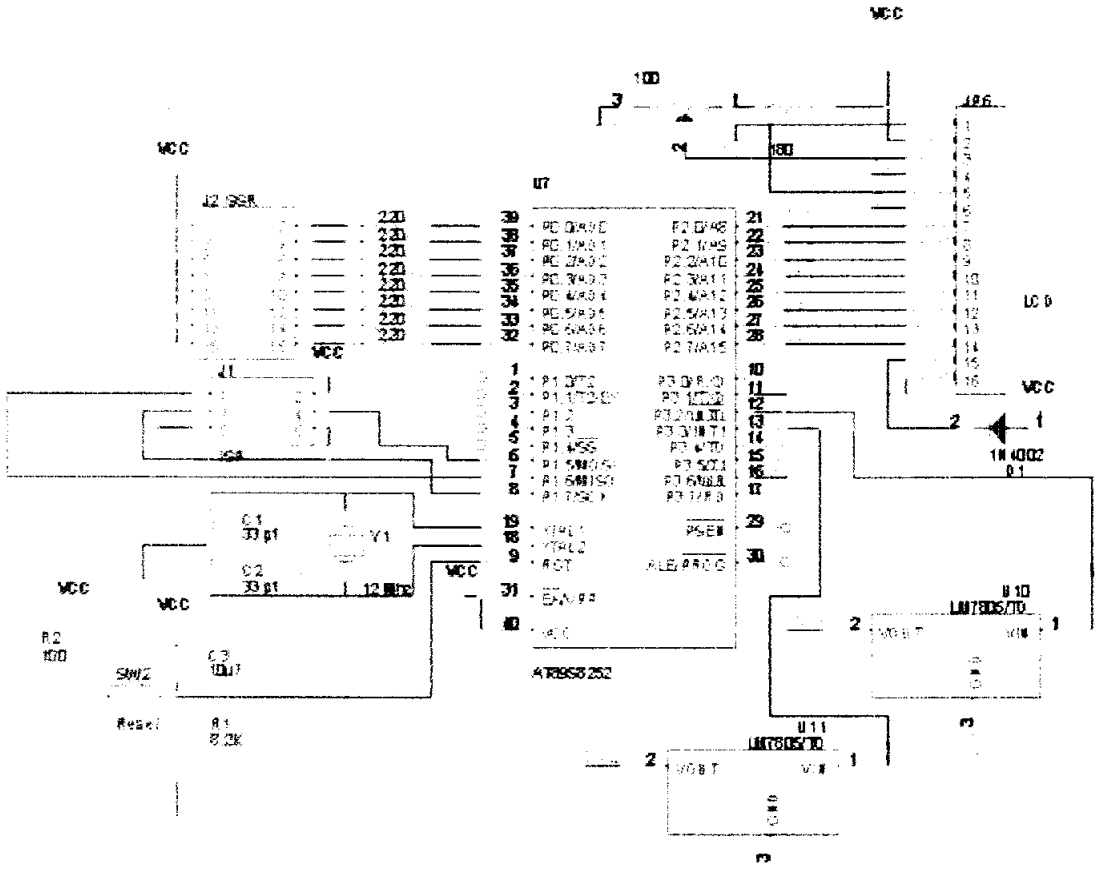
LAMPIRAN A

- **Skema Pengirim**



- **Skema Penerima**



LAMPIRAN B

```
#include<stdio.h>
#include<reg8252.h>

#define en P0_7   //en P0_7
#define rs P0_6   //rs P0_6
#define dataku P0
#define lcd_mem P2
#define clock P1_3
#define datanya P1_2                                    //P1_2 itu data
at 0x0 xdata unsigned char eeprom[56];
void tunda(unsigned char);
void init_lcd();
void clrscr();
void tulis(unsigned char,unsigned char,unsigned char,char);
unsigned char keypad(void);
void inisialisasi_serial();
void kirim_serial(unsigned char);
void tulis_eeprom(unsigned char,unsigned char);
void menu2(unsigned char);
void clock_serial(unsigned char);
void semua_penerima(unsigned char,unsigned char);

char code menu[67][20]={{"1.Masukkan inputan"},{"2.Status Alat"},{"3.Help"},{" <<By
Deathscythe>>"}, {"<<-Masukkan angka->>"},{"Pnr(0-7),alat(0-8),"},{"Status(0/1)"}, {"Pnr
terlalu besar"},{"Alat terlalu besar"},{"Stat terlalu besar"},{"                "}, {"Pnr  :"},
{"Tekan tombol Can utk"},{"keluar,panah atas"},{"utk balik,panah ba-"},{"wah utk
lanjut."}, {"Digit pertama pada"},{"menu pertama adalah"},{"nomor penerima."},{"Inputnya
antara 0-7"}, {"Digit kedua pada "},{"menu pertama adalah"},{"nomor alat."},{"Inputnya
antara 0-8"}, {"Digit ketiga pada "},{"menu pertama adalah"},{"status alat."},{"Inputnya
hanya 0&1"}, {"Untuk mengontrol se-"},{"mua penerima maka"},{"digit pertama
diisi"},{"dengan 0"},  . {"Untuk mengontrol se"},{"mua alat maka digit"},{"kedua diisi
0"},{"Berikut contohnya :"}, {"340=alat ke-4 pada"},{"penerima ke-3 off"},{"621=alat ke-2
pada"},{"penerima ke-6 on"}, {"010=alat ke-1 pada"},{"semua penerima off"},{"051=alat
ke-5 pada"},{"semua penerima on"}, {"200=semua alat pada"},{"penerima ke-2
off"},{"701=semua alat pada"},{"penerima ke-7 on"},  {"000=semua alat pada"},{"setiap
penerima off"},{"001=semua alat pada"},{"setiap penerima on"}, {"Status alat dapat
di"},{"lihat pada menu no-"},{"2. Tombol yang di  "},{"pakai pada menu no"}, {"2 : panah
atas utk "},{"balik,panah bawah "},{"utk lanjut,tombol"},{"CAN utk keluar."},  {"<
Create&Design By >"},{"Chornelius Ariesta"},{"NRP : 5103000066  "},{" <= Terima kasih
=>"}, {"Penerima kosong !!"},{"Alat kosong !!"},{"status kosong !!"}};

/*=============================================================================
 |                                  Program Utama                             |
 |                                                                            |
=============================================================================*/
void main(void)
{
        unsigned char posisi_y;
        init_lcd();                     //Inisialisasi LCD
```

```
tulis(0,1,2,0);                 //Tampilkan Menu Pertama
        tulis(1,2,2,0);                 //Tampilkan Menu Kedua
        tulis(2,3,2,0);                 //Tampilkan Menu Ketiga
        tulis(3,4,0,0);
        posisi_y=1;
        tulis(0,posisi_y,0,'=');        //Tampilkan Posisi Panah
        tulis(1,posisi_y,1,'>');    //Tampilkan Posisi Panah =>
        while(1)
        {
                switch(keypad())        //Cek Keypad apakah ada tom bol yg ditekan
                {
                case 'u':       //Tombol Panah Atas ditekan
                                tulis(0,posisi_y,0,' ');
                                tulis(1,posisi_y,1,' ');
                                posisi_y--;
                                if (posisi_y==0)posisi_y=1;
                                tulis(0,posisi_y,0,'=');
                                tulis(1,posisi_y,1,'>');
                                tunda(4);
                                break;
                case 'd':       //Tombol Panah bawah ditekan
                                tulis(0,posisi_y,0,' ');
                                tulis(1,posisi_y,1,' ');
                                posisi_y++;
                                if (posisi_y==4)posisi_y=3;
                                tulis(0,posisi_y,0,'=');
                                tulis(1,posisi_y,1,'>');
                                tunda(4);
                                break;

                case 'e':tunda(4);      //Tombol ENT ditekan
                                tunda(4);
                                menu2(posisi_y);
                                clrscr();
                                tulis(0,1,2,0);
                                tulis(1,2,2,0);
                                tulis(2,3,2,0);
                                tulis(3,4,0,0);
                                tulis(0,posisi_y,0,'=');
                                tulis(1,posisi_y,1,'>');
                                break;

                }
        }

}
```

```
/*=============================================================
|                Prosedur untuk menulis kalimat pada LCD      |
|                                                             |
=============================================================*/

void tulis(unsigned char no_kal, unsigned char pos_Y, unsigned char pos_X, char karakter)
{       unsigned char letak_mem;
        unsigned char y;
        switch(pos_Y)
        {case 1:letak_mem=0x80;          //Untuk meletakkan kursor pada baris 1
                        break;
         case 2:letak_mem=0xC0;          //Untuk meletakkan kursor pada baris 2
                        break;
         case 3:letak_mem=0x94;          //Untuk meletakkan kursor pada baris 3
                        break;
         case 4:letak_mem=0xD4;          //Untuk meletakkan kursor pada baris 4
                        break;
         }

         letak_mem=letak_mem+pos_X;
        rs=0;
         lcd_mem=letak_mem;
         en=1;
         en=0;
         tunda(1);
         rs=1;
         if (karakter!=0) {lcd_mem=karakter; en=1; en=0; tunda(2);}
         else
         {
         for (y=0;menu[no_kal][y];y++){ lcd_mem=menu[no_kal][y];    //untuk menulis
kalimat/karakter pada LCD
         en=1;
         en=0;
         tunda(2);}
         }
}

/*=============================================================
|                Prosedur untuk Inisialisasi LCD              |
|                                                             |
=============================================================*/

void init_lcd()
{
        rs=0;
        lcd_mem=0x38;
        en=1;
        en=0;
```

```
        tunda(1);

        rs=0;
        lcd_mem=0x38;
        en=1;
        en=0;
        tunda(1);

        rs=0;
        lcd_mem=0x38;
        en=1;
        en=0;
        tunda(1);

        rs=0;
        lcd_mem=0x38;
        en=1;
        en=0;
        tunda(1);

        lcd_mem=0x06;
        en=1;
        en=0;
        tunda(1);
        lcd_mem=0x0C;
        en=1;
        en=0;
        tunda(1);
        lcd_mem=0x01;
        en=1;
        en=0;
        tunda(1);
}

/*=============================================================================
|                Prosedur untuk menghapus tampilan LCD                        |
|                                                                             |
=============================================================================*/
void clrscr()
{  rs=0;
        lcd_mem=0x01;
        en=1;
        en=0;
        tunda(1);
}
```

```
/*=============================================================
|                Prosedur Delay untuk LCD, Clock Serial & Keypad |
|                                                                |
=============================================================*/
void tunda(unsigned char nilai)
{       unsigned char X;
        switch(nilai)
        {
                case 1:  TMOD=0x21;
                         TH0=0xC5;                  //DELAY 20 ms untuk Init LCD
                         TL0=0x67;
                         TR0=1;
                         TF0=0;
                         while (TF0==0);
                         break;
                case 2:  TMOD=0x21;
                         TH0=0xff;          //delay  30 us untuk jeda tulisan
                         TL0=0xCD;
                         TR0=1;
                         TF0=0;
                         while (TF0==0);
                         break;
                case 3:  TH2=0xF8;          //delay 2 ms untuk clock serialku
                         TL2=0x30;
                         TF2=0;
                         TR2=1;
                         while (TF2==0);
                         break;
                case 4:  X=0;
                         do
                         {
                         TH0=0x3C;          //delay 150 ms untuk debounce
                         TL0=0xAF;
                         TMOD=0x21;
                         TR0=1;
                         TF0=0;
                         while (TF0==0);
                         X++;
                         }
                         while(X!=3);
                         break;
  }
  }
```

```
/*=============================================================================
|                                                                             |
|                        Fungsi untuk menerima masukan dari keypad            |
|                                                                             |
=============================================================================*/

unsigned char keypad(void)

{unsigned char perulangan,tombol,cek_baris[4]={0x7F,0xBF,0xDF,0xEF};

perulangan=0;
do {
P3=cek_baris[perulangan];
        switch(P3)
        {       case 0x77:      tombol='1';     //Angka 1
                                break;
                case 0x7B:      tombol='2';     //Angka 2
                                break;
                case 0x7D:      tombol='3';     //Angka 3
                                break;
                case 0x7E:      tombol='c';     //tombol Cor
                                break;
                case 0xB7:      tombol='4';     //Angka 4
                                break;
                case 0xBB:      tombol='5';     //Angka 5
                                break;
                case 0xBD:      tombol='6';     //Angka 6
                                break;
                case 0xBE:      tombol='m';     //tombol Men0
                                break;
                case 0xD7:      tombol='7';     //Angka 7
                                break;
                case 0xDB:      tombol='8';     //Angka 8
                                break;
                case 0xDD:      tombol='9';     //Angka 9
                                break;
                case 0xDE:      tombol='u';     //tombol panah atas
                                break;
                case 0xE7:      tombol='C';     //tombol Can
                                break;
                case 0xEB:      tombol='0';     //Angka 0
                                break;
                case 0xED:      tombol='e';     //tombol Ent
                                break;
                case 0xEE:      tombol='d';     //tombol panah bawah
                                break;
                default:        tombol=0;
                        }
 perulangan++;
 if (perulangan==4)perulangan=0;
```

```
  }
 while (tombol==0);
        return(tombol);
}

void inisialisasi_serial()
{
SCON=0x40;
TMOD=0x21;
TH1=0xF4;  //Baudrate 2400
TF1=0;
TR1=1;
}

/*=============================================================================
|                        Prosedur untuk menulis data pada EEPROM internal     |
|                                                                             |
=============================================================================*/
void tulis_eeprom(unsigned char no_data, unsigned char data_eeprom)
{  unsigned char bandingkan;
        WMCON=0x1A;
        eeprom[no_data]=data_eeprom;
        do
        {
        bandingkan=WMCON & 0x02;}
        while    (bandingkan==0);
        WMCON=0x0A;
}

/*=============================================================================
|                                Prosedur Menu                                |
|                                                                             |
=============================================================================*/

void menu2 (unsigned char no_menu)
{       struct kirim_data
                        {
                                unsigned alat0      : 1;
                                unsigned alat1      : 1;
                                unsigned alat2      : 1;
                                unsigned alat3      : 1;
                                unsigned status_bit  : 1;
                                unsigned bit_penerima0: 1;
                                unsigned bit_penerima1: 1;
                                unsigned bit_penerima2: 1;
                        };
        union
                {
```

```
                unsigned char byte_data;
                struct kirim_data byte;
        }bit_data;


unsigned char tomb_pos,alamat_eeprom,nel,check,baris_tulisan,counter;
unsigned char quit,awal_help;
unsigned char dataku[5];
switch(no_menu)
        {
                case 1: dataku[1]=0;
                        dataku[2]=0;
                        dataku[3]=0;
                        clrscr();
                        tulis(4,1,0,0);
                        tulis(5,3,0,0);
                        tulis(6,4,0,0);
                        tomb_pos=5;
                        do
                        { quit='0';
                          switch(keypad())
                          {
                                case 'u':break;
                                case 'd':break;
                                case 'm':break;
                                case 'c':
                                tulis(0,2,tomb_pos,' ');
                                tomb_pos--;
                                if (tomb_pos==4) tomb_pos=5;
                                break;
                        case 'C':quit='1';
                                break;
                        case 'e':dataku[1]=dataku[1]-48; //untuk merubah ASCII nomor
penerima menjadi bilangan desimal
                                dataku[2]=dataku[2]-48;//untuk merubah ASCII nomor alat
menjadi bilangan desimal
                                dataku[3]=dataku[3]-48;//untuk merubah ASCII status alat
menjadi bilangan desimal
                                if (dataku[1]==0xD0) //apabila penerima tidak diisi sama
sekali maka dataku[1] akan bernilai 256-48= 208
                                tulis(64,2,0,0);  //tampilkan pesan bahwa penerima belum
diisi, tunda selama beberapa detik
                                nel=0;
                                do
                                {
                                tunda(4);
                                nel++;
                                }
                                while(nel!=10);
```

```
                    tulis(10,2,0,0);    //untuk menghapus baris 2
                    tomb_pos=5;
                    tulis(0,2,tomb_pos,' ');
                    dataku[1]=0;
                    dataku[2]=0;
                    dataku[3]=0;
                    break;
                    }
                    if (dataku[2]==0xD0)          //apabila alat tidak diisi
sama sekali maka dataku[1] akan bernilai 256-48= 208
                    {       tulis(65,2,0,0);      //tampilkan pesan bahwa
penerima belum diisi, tunda selama beberapa detik
                    nel=0;
                    do
                    {
                    tunda(4);
                    nel++;
                    }
                    while(nel!=10);
                    tulis(10,2,0,0);    //untuk menghapus baris 2
                    tomb_pos=5;
                    tulis(0,2,tomb_pos,' ');
                    dataku[1]=0;
                    dataku[2]=0;
                    dataku[3]=0;
                    break;
                    }
                    if (dataku[3]==0xD0)          //apabila status tidak diisi
sama sekali maka dataku[1] akan bernilai 256-48= 208
                    tulis(66,2,0,0);      //tampilkan pesan bahwa penerima
belum diisi, tunda selama beberapa detik
                    nel=0;
                    do
                    {
                    tunda(4);
                    nel++;
                    }
                    while(nel!=10);
                    tulis(10,2,0,0);    //untuk menghapus baris 2
                    tomb_pos=5;
                    tulis(0,2,tomb_pos,' ');
                    dataku[1]=0;
                    dataku[2]=0;
                    dataku[3]=0;
                    break;
                    }
                    if (dataku[1]>7)      //cek apakah penerima lebih dari 7
                    {       tulis(7,2,0,0);     //tampilkan pesan kesalahan
                    nel=0;
```

```
do
{
tunda(4);
nel++;
}
while(nel!=10);
tulis(10,2,0,0);      //untuk menghapus baris 2
tomb_pos=5;
tulis(0,2,tomb_pos,' ');
dataku[1]=0;
dataku[2]=0;
dataku[3]=0;
break;
}
if (dataku[2]>8)                                    //cek
```

apakah alat lebih dari 8

```
{       tulis(8,2,0,0);                    //tampilkan pesan
```

kesalahan

```
nel=0;
do
{
tunda(4);
nel++;
}
while(nel!=10);
tulis(10,2,0,0);
tomb_pos=5;
tulis(0,2,tomb_pos,' ');
dataku[1]=0;
dataku[2]=0;
dataku[3]=0;
break;
}
if (dataku[3]>1)                                    //cek
```

apakah status lebih dari 1

```
{       tulis(9,2,0,0);                    //tampilkan pesan
```

kesalahan

```
nel=0;
do
{
tunda(4);
nel++;
}
while(nel!=10);
tulis(10,2,0,0);
tomb_pos=5;
tulis(0,2,tomb_pos,' ');
dataku[1]=0;
dataku[2]=0;
```

```
                    dataku[3]=0;
                    break;
                    }
                    dataku[4]=(dataku[1]<<5)+(dataku[3]<<4)+(dataku[2]);
                    bit_data.byte_data=dataku[4];
                    if (dataku[1]==0) semua_penerima(dataku[2],dataku[3]+48);
                //cek apakah penerima=0
                    else if (dataku[2]==0)   //cek apakah alat=0
                    {       nel=0;
                    do
                    {       nel++;
                    alamat_eeprom=((dataku[1]*8)+nel);
                    tulis_eeprom(alamat_eeprom,(dataku[3]+48));
                    }
                    while (nel!=8);
                    }
                    else
                    {
                    alamat_eeprom=((dataku[1]*8)+dataku[2]);
                    tulis_eeprom(alamat_eeprom,(dataku[3]+48));
                    }
                    quit='1';
                    break;
                    default: tomb_pos++;
                    if (tomb_pos==9)tomb_pos=8;
                    dataku[tomb_pos-5]=keypad();
                    tulis(0,2,tomb_pos,dataku[tomb_pos-5]);
                    quit='0';
                    break;
                    }
                    tunda(4);
                    }
                    while (quit!='1');
                    break;
          case 2:  clrscr();
                    baris_tulisan=1;
                    nel=8;
                    check=1;
                    counter=0;
                    do
                    {  tulis(11,baris_tulisan,0,0);          //tampilkan kata "Pnr  : "
pada baris pertama pada LCD
                    tulis(11,baris_tulisan,4,'0'+baris_tulisan);          //tampilkan angka
"1"  pada baris pertama kolom ke empat, sehingga pada LCD tampil tulisan "Pnr 1 : "
                    do
                    {  nel++;
                    counter++;
                    WMCON=0x0A;
                    tulis(0,baris_tulisan,nel-check,eeprom[nel]);
```

```
                    }
                    while (counter!=8);
                    baris_tulisan+=;
                    check=check+8;
                    counter=0;
                    }
                    while (baris_tulisan!=5);
                    do
                    {
                    switch(keypad())
                    {
                    case 'u':
                            baris_tulisan=1;
                            nel=8;
                            check=1;
                            counter=0;
                            do
                            { tulis(11,baris_tulisan,0,0);          //tampilkan kata
"Pnr : " pada baris pertama pada LCD
                            tulis(11,baris_tulisan,4,'0'+baris_tulisan);          //tampilkan
angka "1" pada baris pertama kolom ke empat, sehingga pada LCD tampil tulisan "Pnr 1 : "
                            do
                            { nel++;
                            counter++;
                            WMCON=0x0A;
                            tulis(0,baris_tulisan,nel-check,eeprom[nel]);
                            }
                            while (counter!=8);
                            baris_tulisan++;
                            check=check+8;
                            counter=0;
                            }
                            while (baris_tulisan!=5);
                            break;
                            case 'd':
                            clrscr();
                            baris_tulisan=1;
                            nel=40;
                            check=33;
                            counter=0;
                            do
                            { tulis(11,baris_tulisan,0,0);          //tampilkan kata
"Pnr : " pada baris pertama pada LCD
                            tulis(11,baris_tulisan,4,'4'+baris_tulisan);          //tampilkan
angka "1" pada baris pertama kolom ke empat, sehingga pada LCD tampil tulisan "Pnr 1 : "
                            do
                            { nel++;
                            counter++;
                            WMCON=0x0A;
```

```
}
/*=============================================================================
|                                Prosedur serial                              |
|                                                                             |
=============================================================================*/
void clock_serial(unsigned char xx)
{  datanya=xx;
        clock=0;
        tunda(3);
        clock=1;
        tunda(3);
}

void semua_penerima(unsigned char alamat_alat, unsigned char isi_data)
{       unsigned char perulangan_1,alamat_penerima,perulangan_2;
  perulangan_1=0;
        do
        {       perulangan_1++;
                if (alamat_alat==0)
                {  perulangan_2=0;
                do
                {       perulangan_2++;
                        alamat_penerima=(perulangan_1*8)+perulangan_2;
                        tulis_eeprom(alamat_penerima,isi_data);
                }
                while(perulangan_2!=8);
                }
                else
                {
                        alamat_penerima=(perulangan_1*8)+alamat_alat;
                        tulis_eeprom(alamat_penerima,isi_data);
                }
        }
        while(perulangan_1!=7);
}

void kirim_serial(unsigned char ser)
{  SBUF=ser;
  while(TI==0);
  TI=0;
}
```

```
}
/*=============================================================================
|                                Prosedur serial                              |
|                                                                             |
=============================================================================*/
void clock_serial(unsigned char xx)
{  datanya=xx;
        clock=0;
        tunda(3);
        clock=1;
        tunda(3);
}

void semua_penerima(unsigned char alamat_alat, unsigned char isi_data)
{       unsigned char perulangan_1,alamat_penerima,perulangan_2;
  perulangan_1=0;
        do
        {       perulangan_1++;
                if (alamat_alat==0)
                {  perulangan_2=0;
                do
                {        perulangan_2++;
                        alamat_penerima=(perulangan_1*8)+perulangan_2;
                        tulis_eeprom(alamat_penerima,isi_data);
                }
                while(perulangan_2!=8);
                }
                else
                {
                        alamat_penerima=(perulangan_1*8)+alamat_alat;
                        tulis_eeprom(alamat_penerima,isi_data);
                }
        }
        while(perulangan_1!=7);
}

void kirim_serial(unsigned char ser)
{  SBUF=ser;
  while(TI==0);
  TI=0;
}
```

LAMPIRAN C

```
$include(reg8252.inc)

rs                  bit p1.0
en                  bit p1.1
status              bit 04h
buffer_serial       equ 20h
valid               equ 40h
buffer              equ 57h
dataku              equ 40h
temp                equ 59h
jumlah_lcd          equ 50h
jumlah              equ 51h
hasil               equ 62h

org 0h
sjmp start

org 03h
ljmp terima_data

org 2Bh
ljmp cek_valid

org 30h
start:      mov r2,#0
            mov ie,#0a1h            ;set enable interupt (int external 0 dan timer 2 int)
            setb pt2        ;set supaya int timer 1 menjadi high.
            setb it0 ;set supaya interupt xternal 0 aktifnya negatif edge trigering
            acall init_lcd
            mov dptr,#penerima
            acall tulis_kalimat


start1:     acall cek_isi_eeprom    ;Panggil prosedur utk cek isi eeprom
            ;acall detik
            sjmp start1             ;Loncat ke program utama

terima_data: push acc               ;simpan accumulator ke stack memori
            mov a,buffer_serial
            cjne r2,#0,data_ke_n    ;cek cek jumlah bit data yang diterima
            acall waktu_kirim

data_ke_n:  jnb p3.3,data_nol
            setb acc.7
            sjmp selesai_terima
data_nol:   clr acc.7
selesai_terima: rl a
            mov buffer_serial,a
            inc r2
```

```
sel:            pop acc                 ;ambil isi accumulator dari stack memori
                reti

cek_valid:      push acc
                cjne r2,#8,tidak_valid  ;cek apakah jumlah data yang diterima=8bit
                acall decoder
tidak_valid:    mov r2,#0
                mov buffer_serial,#0    ;kosongkan isi buffr serial
                clr tr2
                clr tf2
                pop acc                 ;ambil isi accumulator dari stack memori
                reti

decoder:        mov a,buffer_serial
                anl a,#11100000b
                cjne a,#20h,decoder_selesai     ;cek apakah data yang diterima untuk
penerima pertama?
                jnb status,status_off           ;cek apakah status alat on/off
                mov dataku,#10h                 ;alat diset on
                sjmp cek_alamat
status_off:     mov dataku,#0                   ;alat diset off
cek_alamat:     mov a,dataku
                anl a,#10h                      ;ambil bit status alat
                mov r5,a                        ;bit status disimpan di r5
                mov a,buffer_serial             ;isi accumulator dengan buffer serial
                anl a,#0fh                      ;ambil bit nomor alat
                mov hasil,a                     ;simpan bit nomor alat pada variable hasil
                mov r4,a
                acall switch_on_off
                mov dptr,#0
                mov dpl,hasil
                acall tulis_eeprom              ;panggil prosedur untuk menulis eeprom
                mov r0,#06h
decoder_selesai:ret

switch_on_off:  cjne r4,#1,dua                  ;prosedur untuk menyalakan
                cjne r5,#10h,satu_off
                setb p0.0
                sjmp dua
satu_off:       clr p0.0

dua:            cjne r4,#2,tiga
                cjne r5,#10h,dua_off
                setb p0.1
                sjmp tiga
dua_off:        clr p0.1
tiga:           cjne r4,#3,empat
                cjne r5,#10h,tiga_off
                setb p0.2
```

```
                sjmp empat
tiga_off:       clr p0.2
empat:          cjne r4,#4,lima
                cjne r5,#10h,empat_off
                setb p0.3
                sjmp lima
empat_off:      clr p0.3
lima:           cjne r4,#5,enam
                cjne r5,#10h,lima_off
                setb p0.4
                sjmp enam
lima_off:       clr p0.4
enam:           cjne r4,#6,tujuh
                cjne r5,#10h,enam_off
                setb p0.5
                sjmp tujuh
enam_off:       clr p0.5
tujuh:          cjne r4,#7,delapan
                cjne r5,#10h,tujuh_off
                setb p0.6
                sjmp delapan
tujuh_off:      clr p0.6

delapan:        cjne r4,#8,switch_selesai
                cjne r5,#10h,delapan_off
                setb p0.7
                sjmp switch_selesai
delapan_off:    clr p0.7
switch_selesai: nop
                ret

waktu_kirim:    clr C_T2                ;prosedur timer untuk menerima data
                setb CP_RL2
                ;mov t2con,#01h
                mov th2,#83h
                mov tl2,#00h
                setb tr2
                clr tf2
                ret

cek_isi_EEPROM:     mov jumlah_lcd,#'1'     ;prosedur untuk memeriksa isi EEPROM
                                            dan menyalakan alat.
                    mov jumlah,#1
                    mov buffer,#0c0h
                    acall fungsi_init
                    mov dptr,#alat_nomer
                    acall tulis_kalimat

cek_isi_eeprom2:    mov buffer,#0c5h
```

```
                    acall fungsi_init
                    mov buffer,jumlah_lcd
                    acall fungsi_tulis
                    mov dptr,#0h
                    mov dpl,jumlah
                    acall baca_eeprom
                    jb acc.4,switch_on
                    mov buffer,#0c9h
                    acall fungsi_init
                    mov dptr,#mati
                    acall tulis_kalimat
;                   acall detik
                    sjmp belum_selesai
switch_on:          mov buffer,#0c9h
                    acall fungsi_init
                    mov dptr,#hidup
                    acall tulis_kalimat
belum_selesai:      mov dptr,#0h
                    mov dpl,jumlah
                    acall baca_eeprom
                    mov a,dataku
                    anl a,#10h
                    mov r5,a
                    mov r4,jumlah
                    acall switch_on_off
                    acall detik
                    inc jumlah_lcd
                    inc jumlah
                    mov r4,jumlah
                    cjne r4,#9,cek_isi_eeprom2
cek_isi_selesai:    ret

tulis_EEPROM:       mov wmcon,#1Ah      ;prosedur untuk mengisi EEPROM
                    mov a,dataku
                    movx @dptr,a
cek_busy:           mov a,wmcon
                    anl a,#02h
                    jz cek_busy
                    mov wmcon,#0ah
                    ret

baca_EEPROM:        mov wmcon,#0aH      ;prosedur untuk membaca isi EEPROM
                    movx a,@dptr
                    mov dataku,a
                    ret

tulis_kalimat:      clr a               ;prosedur untuk menulis kata pada LCD
                    movc a,@a+dptr
                    cjne a,#0ffh,ngga_sama
```

```
                    sjmp selesai
ngga_sama:          acall write_text
                    inc dptr
                    sjmp tulis_kalimat
selesai:            ret

detik:              tunda11 50000,20,11h
                    ret

;--------------------
; Clear LCD Screen
;--------------------
clrscr:             mov buffer,#01h
                    acall fungsi_init
                    ret
;--------------------
; Write Text
;--------------------
write_text:         mov     buffer,a
                    acall   fungsi_tulis
                    ret
;--------------------
; Initialisasi LCD
;--------------------
init_lcd:           acall   wait_lcd
                    mov     buffer,#038H
                    acall   fungsi_init

                    mov     buffer,#06H
                    acall   fungsi_init

                    mov     buffer,#0CH
                    acall   fungsi_init

                    mov     buffer,#01H
                    acall   fungsi_init
                    ret
;--------------------
; Fungsi Init
;--------------------
fungsi_init:    mov p2,buffer
                clr     rs
                setb    en
                clr  en
                acall wait_lcd
                ret
```

```
;----------------------
; Fungsi Tulis
;----------------------
fungsi_tulis:   mov p2,buffer
                setb    rs
                setb    en
                clr     en
                acall wait_tulis
                ret
;-------------------------------
; Wait LCD
;-------------------------------
wait_lcd:       mov     tmod,#01H
                mov     th0,#0C8H
                mov     tl0,#08H
                clr     tf0
                setb    tr0
                jnb     tf0,$
                ret

wait_tulis:     mov r6,#100
                djnz r6,$
                ret

status_alat:    db 'Status :      ',0Fh
hidup:          db 'Hidup',0Fh
mati:           db 'Mati ',0Fh
Penerima:       db 'Penerima 1    ',0Fh
alat_nomer:     db 'Alat  :    ',0Fh

end
```

LAMPIRAN D

atures

ompatible with MCS-51™ Products
K Bytes of In-System Reprogrammable Downloadable Flash Memory
– SPI Serial Interface for Program Downloading
– Endurance: 1,000 Write/Erase Cycles
K Bytes EEPROM
– Endurance: 100,000 Write/Erase Cycles
V to 6V Operating Range
ully Static Operation: 0 Hz to 24 MHz
hree-level Program Memory Lock
56 x 8-bit Internal RAM
2 Programmable I/O Lines
hree 16-bit Timer/Counters
ine Interrupt Sources
rogrammable UART Serial Channel
PI Serial Interface
ow-power Idle and Power-down Modes
nterrupt Recovery From Power-down
rogrammable Watchdog Timer
ual Data Pointer
ower-off Flag

escription

e AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with bytes of downloadable Flash programmable and erasable read only memory and bytes of EEPROM. The device is manufactured using Atmel's high-density nonvole memory technology and is compatible with the industry-standard 80C51 ruction set and pinout. The on-chip downloadable Flash allows the program mem- to be reprogrammed in-system through an SPI serial interface or by a nventional nonvolatile memory programmer. By combining a versatile 8-bit CPU n downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful rocomputer which provides a highly-flexible and cost-effective solution to many bedded control applications.

e AT89S8252 provides the following standard features: 8K bytes of downloadable sh, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchg timer, two data pointers, three 16-bit timer/counters, a six-vector two-level rrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In lition, the AT89S8252 is designed with static logic for operation down to zero frency and supports two software selectable power saving modes. The Idle Mode os the CPU while allowing the RAM, timer/counters, serial port, and interrupt sys- to continue functioning. The Power-down mode saves the RAM contents but ezes the oscillator, disabling all other chip functions until the next interrupt or hardre reset.

e downloadable Flash can be changed a single byte at a time and is accessible ough the SPI serial interface. Holding RESET active forces the SPI bus into a serial gramming interface and allows the program memory to be written to or read from ess Lock Bit 2 has been activated.

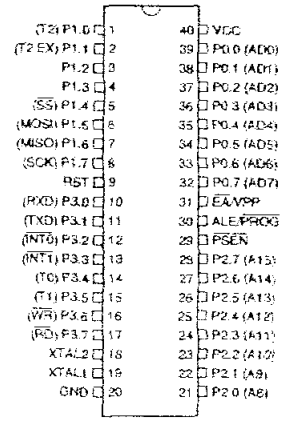
8-bit Microcontroller with 8K Bytes Flash

AT89S8252

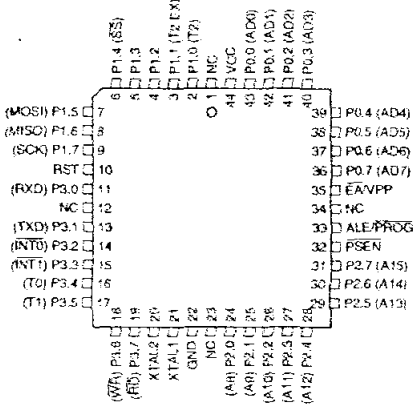
Rev. 0401E–02/00

n Configurations

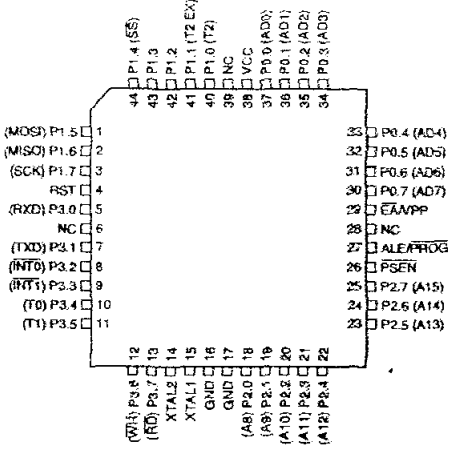
PDIP



PLCC



PQFP/TQFP



in Description

cc

pply voltage.

ND

ound.

rt 0

rt 0 is an 8-bit open drain bbi-didirectional I/O port. As an tput port, each pin can sink eight TTL inputs. When 1s e written to port 0 pins, the pins can be used as high-pedance inputs.

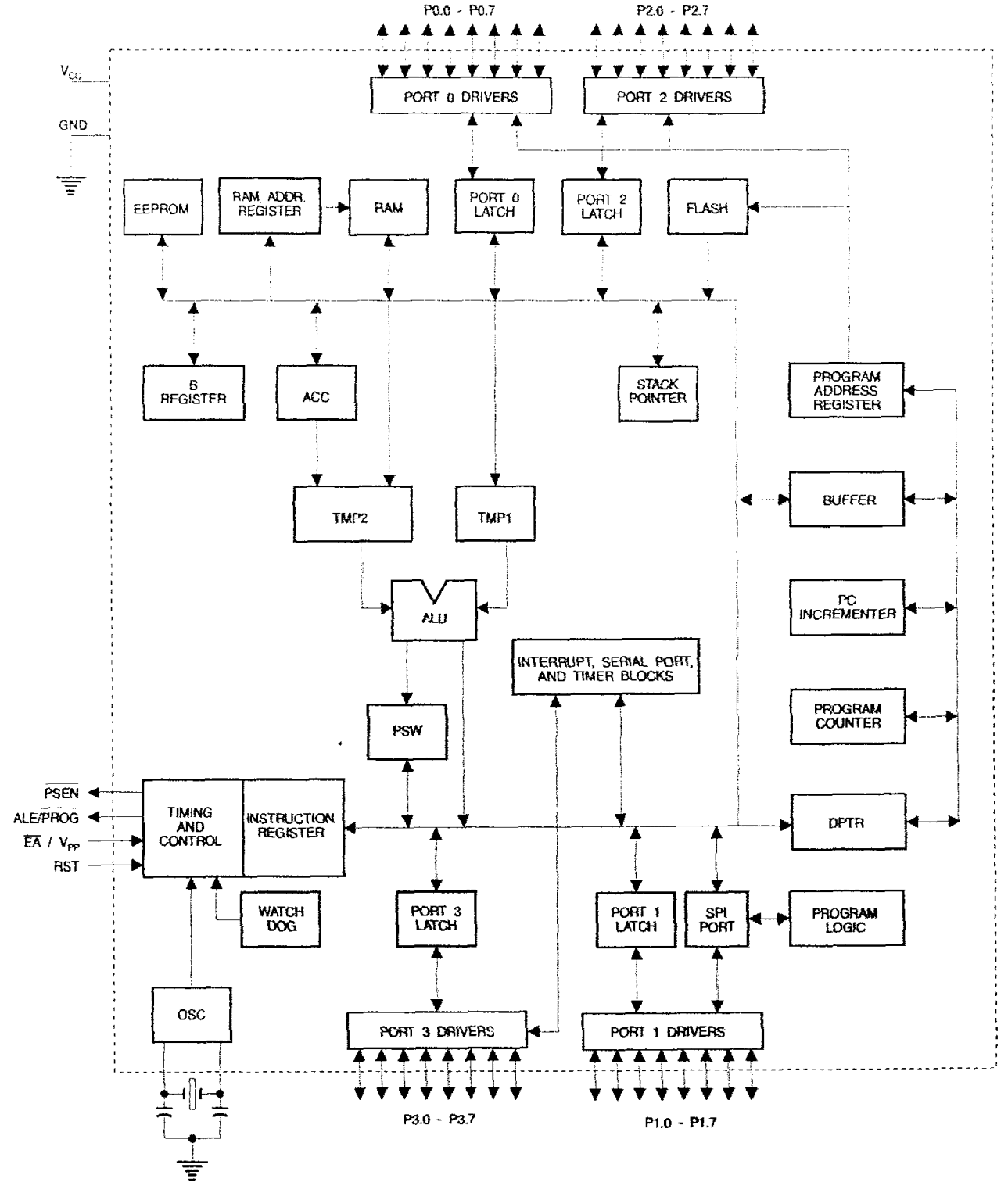
rt 0 can also be configured to be the multiplexed low-der address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

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

ock Diagram


P0.0 - P0.7
P2.0 - P2.7
V_{CC}
GND
PORT 0 DRIVERS
PORT 2 DRIVERS
EEPROM
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
FLASH
B REGISTER
ACC
STACK POINTER
PROGRAM ADDRESS REGISTER
BUFFER
TMP2
TMP1
ALU
PC INCREMENTER
INTERRUPT, SERIAL PORT, AND TIMER BLOCKS
PSW
PROGRAM COUNTER
PSEN
ALE/PROG
EA / V_{PP}
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DPTR
WATCH DOG
PORT 3 LATCH
PORT 1 LATCH
SPI PORT
PROGRAM LOGIC
OSC
PORT 3 DRIVERS
PORT 1 DRIVERS
P3.0 - P3.7
P1.0 - P1.7

me Port 1 pins provide additional functions. P1.0 and .1 can be configured to be the timer/counter 2 external unt input (P1.0/T2) and the timer/counter 2 trigger input .1/T2EX), respectively.

n Description

rthermore, P1.4, P1.5, P1.6, and P1.7 can be configured the SPI slave port select, data input/output and shift ck input/output pins as shown in the following table.

ort Pin	Alternate Functions
1.0	T2 (external count input to Timer/Counter 2), clock-out
1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
1.4	$\overline{SS}$ (Slave port select input)
1.5	MOSI (Master data output, slave data input pin for SPI channel)
1.6	MISO (Master data input, slave data output pin for SPI channel)
1.7	SCK (Master clock output, slave clock input pin for SPI channel)

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

rt 2

rt 2 is an 8-bit bi-directional I/O port with internal pullups. e Port 2 output buffers can sink/source four TTL inputs. hen 1s are written to Port 2 pins, they are pulled high by e internal pullups and can be used as inputs. As inputs, rt 2 pins that are externally being pulled low will source rrent (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 ternal data memory that use 16-bit addresses (MOVX @ PTR). In this application, Port 2 uses strong internal pul- os when emitting 1s. During accesses to external data emory that use 8-bit addresses (MOVX @ RI), Port 2 its the contents of the P2 Special Function Register.

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

rt 3

rt 3 is an 8 bit bi-directional I/O port with internal pullups. e Port 3 output buffers can sink/source four TTL inputs. hen 1s are written to Port 3 pins, they are pulled high by e 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 AT89S8252, as shown in the following table.

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

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

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 is an 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 microcontroller is in external execution mode.

$\overline{PSEN}$

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

When the AT89S8252 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external pro-

am memory locations starting at 0000H up to FFFFH. ote, however, that if lock bit 1 is programmed, $\overline{EA}$ will be ernally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program execu- ns. This pin also receives the 12-volt programming able voltage (V_{PP}) during Flash programming when 12- lt programming is selected.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

ble 1. AT89S8252 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001XX			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

ɔecial Function Registers

nap of the on-chip memory area called the Special Func- ı Register (SFR) space is shown in Table 1.

te that not all of the addresses are occupied, and unoc- ɔied addresses may not be implemented on the chip. ad accesses to these addresses will in general return ıdom data, and write accesses will have an indeterminate ɔct.

er software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 9) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16 bit capture mode or 16-bit auto-reload mode.

ɔle 2. T2CON—Timer/Counter 2 Control Register

2CON Address = 0C8H — Reset Value = 0000 0000B

it Addressable

	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
it	7	6	5	4	3	2	1	0

ymbol	Function
F2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
XF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
CLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflows to be used for the receive clock.
CLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
XEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
R2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
P/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

ıtchdog and Memory Control Register The WMCON gister contains control bits for the Watchdog Timer ıown in Table 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The DPS bit selects one of two DPTR registers available.

ble 3. WMCON—Watchdog and Memory Control Register

/MCON Address = 96H — Reset Value = 0000 0010B

	PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN
it	7	6	5	4	3	2	1	0

ymbol	Function
S2 S1 S0	Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms.
EMWE	EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed.
EEMEN	Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory.
PS	Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1
VDTRST RDY/$\overline{\text{BSY}}$	Watchdog Timer Reset and EEPROM Ready/$\overline{\text{Busy}}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{\text{BSY}}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{\text{BSY}}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{\text{BSY}}$ bit equals "0" and is automatically reset to "1" when programming is completed.
VDTEN	Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer.

PI Registers Control and status bits for the Serial Periphal Interface are contained in registers SPCR (shown in ıble 4) and SPSR (shown in Table 5). The SPI data bits e contained in the SPDR register. Writing the SPI data gister during serial data transfer sets the Write Collision , WCOL, in the SPSR register. The SPDR is double buffed for writing and the values in SPDR are not changed by eset.

terrupt Registers The global interrupt enable bit and the dividual interrupt enable bits are in the IE register. In ldition, the individual interrupt enable bit for the SPI is in e SPCR register. Two priorities can be set for each of the x interrupt sources in the IP register.

Dual Data Pointer Registers To facilitate accessing both internal EEPROM and external data memory, two banks of 16 bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR WMCON selects DP0 and DPS = 1 selects DP1. The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag The Power Off Flag (POF) is located at bit_4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and reset under software control and is not affected by RESET.

le 4. SPCR—SPI Control Register

PCR Address = D5H — Reset Value = 0000 01XXB

	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
it	7	6	5	4	3	2	1	0

mbol	Function
IE	SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts.
E	SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel.
RD	Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission.
TR	Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode.
OL	Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control.
HA	Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control.
R0 R1	SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 1 1 128

le 5. SPSR – SPI Status Register

PSR Address = AAH — Reset Value = 00XX XXXXB

	SPIF	WCOL	–	–	–	–	–	–
it	7	6	5	4	3	2	1	0

mbol	Function
IF	SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then accessing the SPI data register.
COL	Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register.

le 6. SPDR – SPI Data Register

PDR Address = 86H — Reset Value = unchanged

	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
Bit	7	6	5	4	3	2	1	0

ata Memory – EEPROM and RAM

e AT89S8252 implements 2K bytes of on-chip EEPROM data storage and 256 bytes of RAM. The upper 128 tes of RAM occupy a parallel space to the Special Function Registers. That means the upper 128 bytes have the me addresses as the SFR space but are physically separate from SFR space.

nen an instruction accesses an internal location above dress 7FH, the address mode used in the instruction ecifies whether the CPU accesses the upper 128 bytes RAM or the SFR space. Instructions that use direct dressing access SFR space.

r example, the following direct addressing instruction cesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

structions that use indirect addressing access the upper 8 bytes of RAM. For example, the following indirect dressing instruction, where R0 contains 0A0H, accesses e data byte at address 0A0H, rather than P2 (whose dress is 0A0H).

```
MOV @R0, #data
```

ote that stack operations are examples of indirect dressing, so the upper 128 bytes of data RAM are available as stack space.

e on-chip EEPROM data memory is selected by setting e EEMEN bit in the WMCON register at SFR address cation 96H. The EEPROM address range is from 000H to FFH. The MOVX instructions are used to access the EPROM. To access off-chip data memory with the MOVX structions, the EEMEN bit needs to be set to "0".

e EEMWE bit in the WMCON register needs to be set to ' before any byte location in the EEPROM can be written. ser software should reset EEMWE bit to "0" if no further EPROM write is required. EEPROM write cycles in the rial programming mode are self-timed and typically take 5 ms. The progress of EEPROM write can be monitored reading the RDY/$\overline{BSY}$ bit (read-only) in SFR WMCON. DY/$\overline{BSY}$ = 0 means programming is still in progress and DY/$\overline{BSY}$ = 1 means EEPROM write cycle is completed d another write cycle can be initiated.

addition, during EEPROM programming, an attempted ad from the EEPROM will fetch the byte being written th the MSB complemented. Once the write cycle is completed, true data are valid at all bit locations.

rogrammable Watchdog Timer

e programmable Watchdog Timer (WDT) operates from independent oscillator. The prescaler bits, PS0, PS1 d PS2 in SFR WMCON are used to set the period of the atchdog Timer from 16 ms to 2048 ms. The available ner periods are shown in the following table and the actual timer periods (at V_{CC} = 5V) are within ±30% of the nominal.

The WDT is disabled by Power-on Reset and during Power-down. It is enabled by setting the WDTEN bit in SFR WMCON (address = 96H). The WDT is reset by setting the WDTRST bit in WMCON. When the WDT times out without being reset or disabled, an internal RST pulse is generated to reset the CPU.

Table 7. Watchdog Timer Period Selection

WDT Prescaler Bits			
PS2	PS1	PS0	Period (nominal)
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S8252 operate the same way as Timer 0 and Timer 1 in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-45, section titled, "Timer/Counters."

Timer 2

Timer 2 is a 16 bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 8.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which

e transition was detected. Since two machine cycles (24 scillator periods) are required to recognize a 1-to-0 transi-on, the maximum count rate is 1/24 of the oscillator equency. To ensure that a given level is sampled at least ice before it changes, the level should be held for at least e full machine cycle.

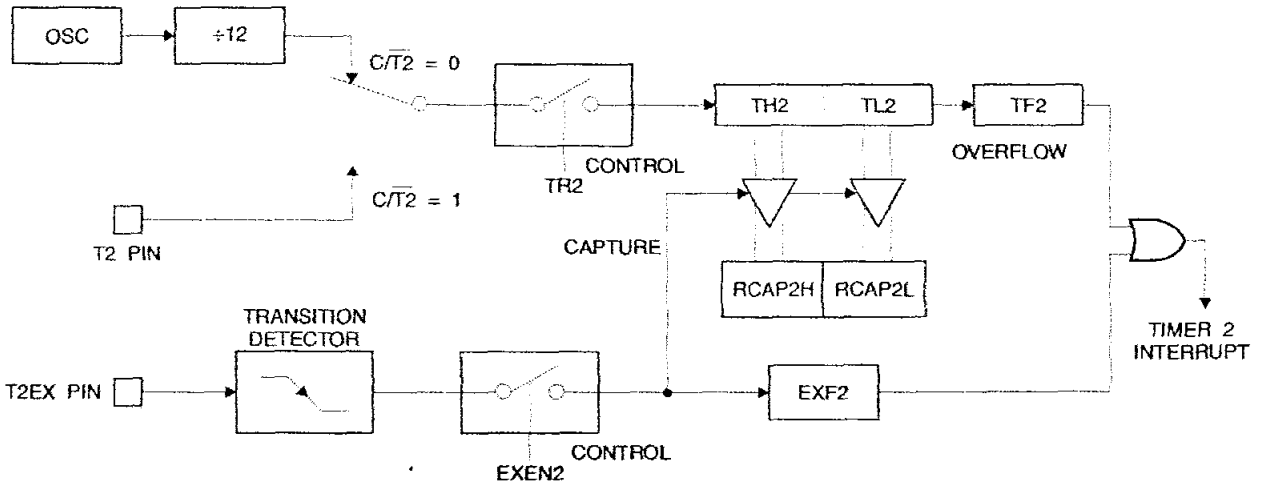
ble 8. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16 bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 1.

gure 1. Timer 2 in Capture Mode



ıto-reload (Up or Down Counter)

ıer 2 can be programmed to count up or down when nfigured in its 16 bit auto-reload mode. This feature is oked by the DCEN (Down Counter Enable) bit located in SFR T2MOD (see Table 9). Upon reset, the DCEN bit set to 0 so that timer 2 will default to count up. When EN is set, Timer 2 can count up or down, depending on value of the T2EX pin.

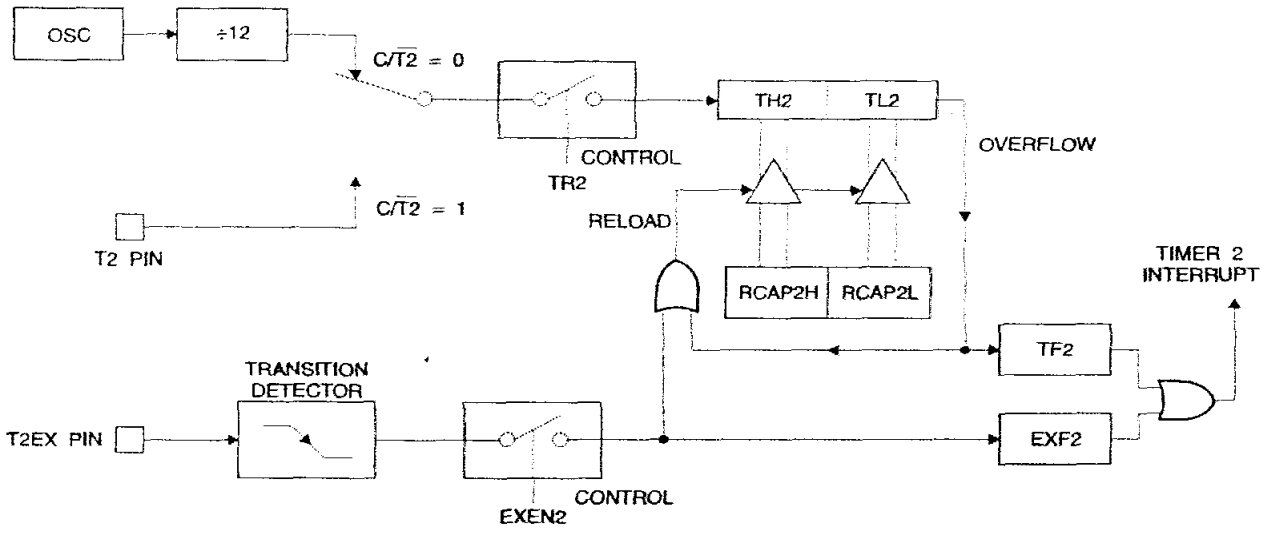
ure 2 shows Timer 2 automatically counting up when EN = 0. In this mode, two options are selected by bit EN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to FFFH and then sets the TF2 bit upon overflow. The erflow also causes the timer registers to be reloaded with 16 bit value in RCAP2H and RCAP2L. The values in AP2H and RCAP2L are preset by software. If EXEN2 = a 16 bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16 bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

ure 2. Timer 2 in Auto Reload Mode (DCEN = 0)



ble 9. T2MOD – Timer 2 Mode Control Register

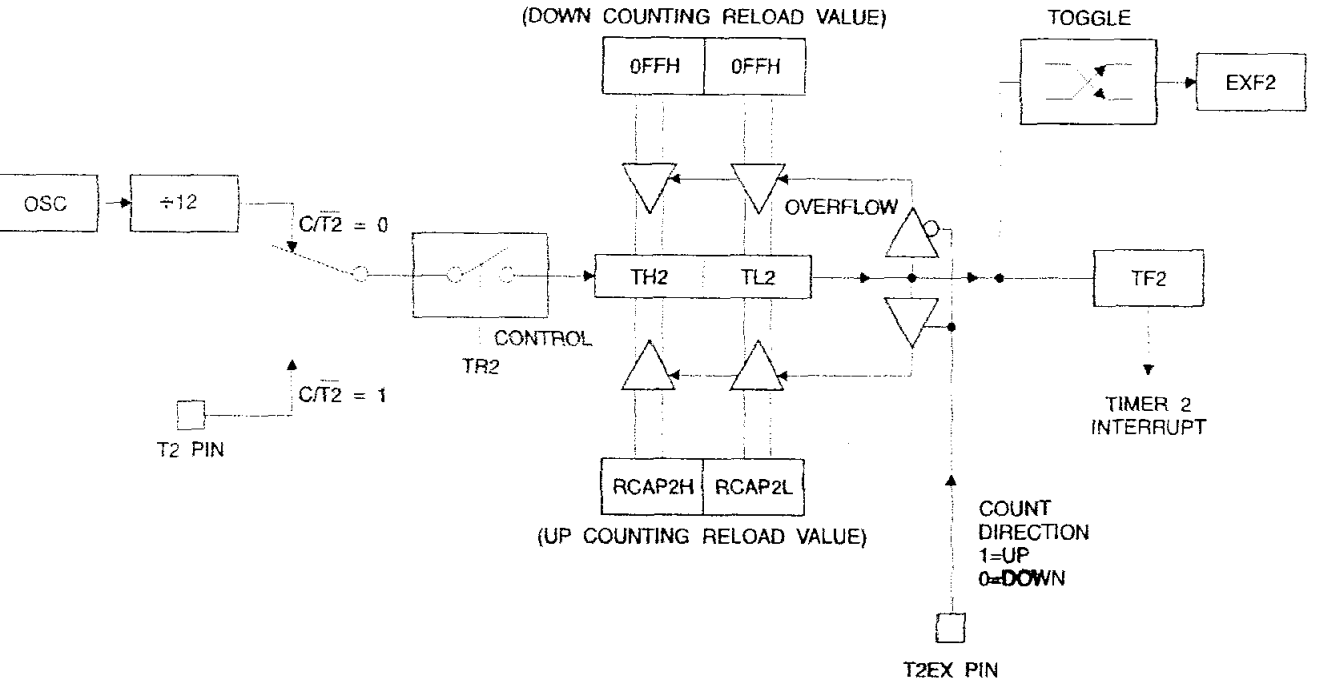
2MOD Address = 0C9H — Reset Value = XXXX XX00B

ot Bit Addressable

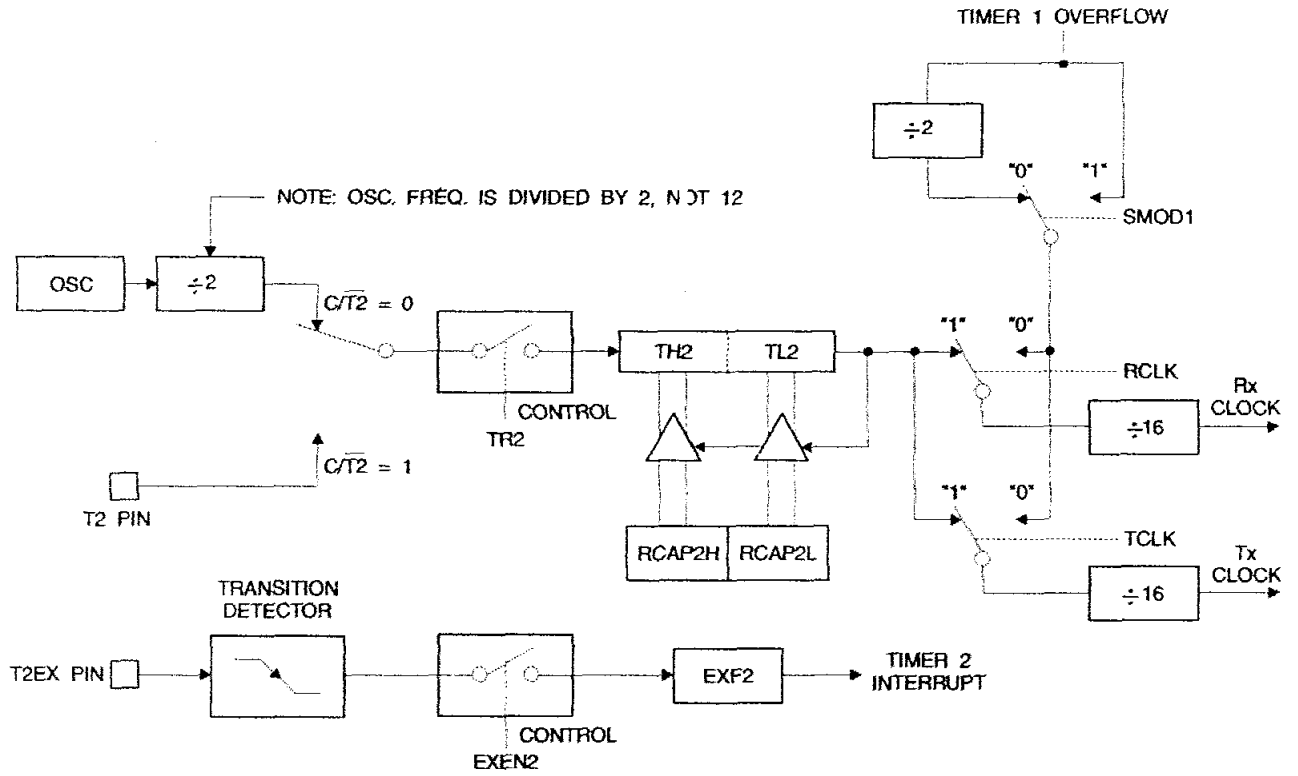
	–	–	–	–	–	–	T2OE	DCEN
Bit	7	6	5	4	3	2	1	0

ymbol	Function
	Not implemented, reserved for future use.
2OE	Timer 2 Output Enable bit.
CEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

gure 3. Timer 2 Auto Reload Mode (DCEN = 1)



gure 4. Timer 2 in Baud Rate Generator Mode



aud Rate Generator

mer 2 is selected as the baud rate generator by setting CLK and/or RCLK in T2CON (Table 2). Note that the ıud rates for transmit and receive can be different if Timer is used for the receiver or transmitter and Timer 1 is used r the other function. Setting RCLK and/or TCLK puts mer 2 into its baud rate generator mode, as shown in Fige 4.

ıe baud rate generator mode is similar to the auto-reload ode, in that a rollover in TH2 causes the Timer 2 registers be reloaded with the 16 bit value in registers RCAP2H d RCAP2L, which are preset by software.

ıe baud rates in Modes 1 and 3 are determined by Timer s overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

ıe Timer can be configured for either timer or counter eration. In most applications, it is configured for timer eration ($C/\overline{T2} = 0$). The timer operation is different for mer 2 when it is used as a baud rate generator. Normally, a timer, it increments every machine cycle (at 1/12 the cillator frequency). As a baud rate generator, however, it crements every state time (at 1/2 the oscillator freıency). The baud rate formula is given below.

$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

ıere (RCAP2H, RCAP2L) is the content of RCAP2H and CAP2L taken as a 16 bit unsigned integer.

mer 2 as a baud rate generator is shown in Figure 4. This ıure is valid only if RCLK or TCLK = 1 in T2CON. Note at a rollover in TH2 does not set TF2 and will not gener- e an interrupt. Note too, that if EXEN2 is set, a 1-to-0 ansition in T2EX will set EXF2 but will not cause a reload om (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

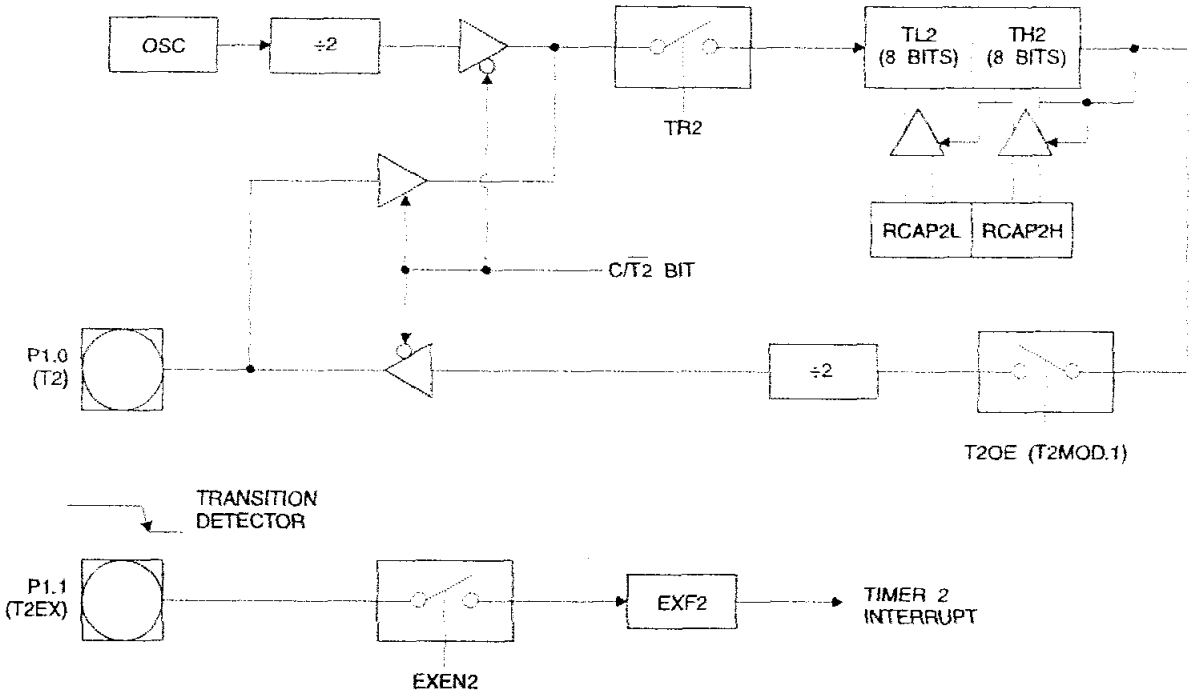
To configure the Timer/Counter 2 as a clock generator, bit $C/\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

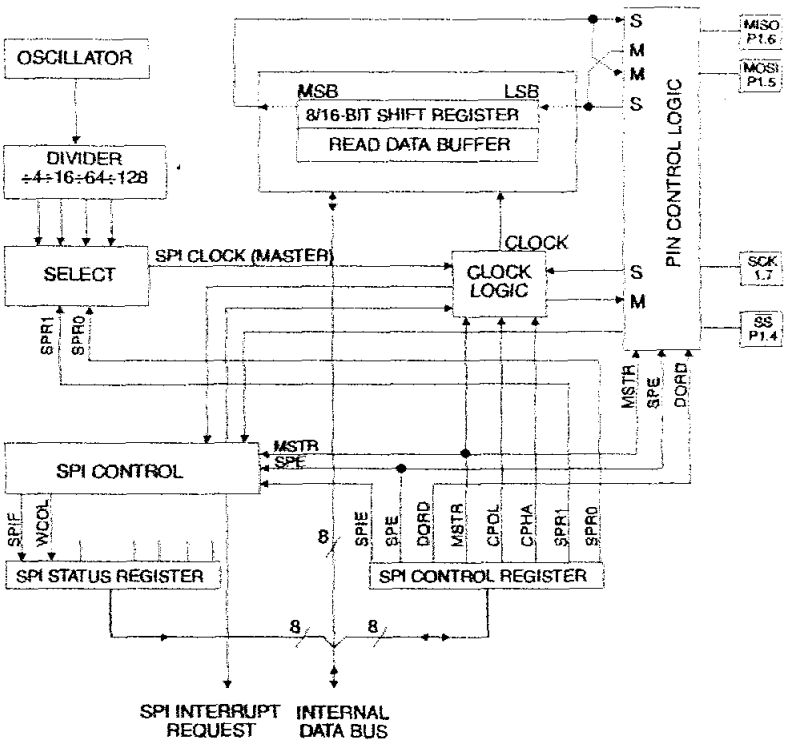
$$\text{Clock Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 rollovers will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

gure 5. Timer 2 in Clock-out Mode



gure 6. SPI Block Diagram



ART

e UART in the AT89S8252 operates the same way as e UART in the AT89C51, AT89C52 and AT89C55. For ther information, see the October 1995 Microcontroller ıta Book, page 2-49, section titled, "Serial Interface."

erial Peripheral Interface

e serial peripheral interface (SPI) allows high-speed synronous data transfer between the AT89S8252 and ripheral devices or between several AT89S8252 vices. The AT89S8252 SPI features include the lowing:

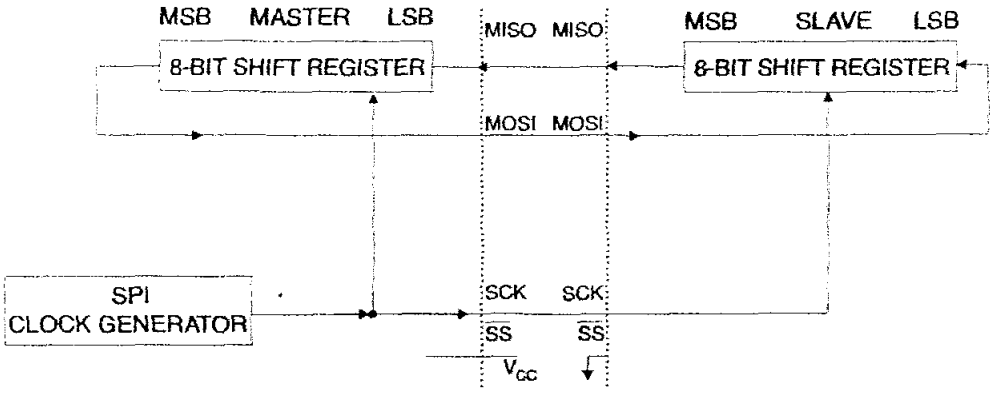
- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 1.5 MHz Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

The interconnection between master and slave CPUs with SPI is shown in the following figure. The SCK pin is the clock output in the master mode but is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the MOSI pin and into the MOSI pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If both the SPI interrupt enable bit (SPIE) and the serial port interrupt enable bit (ES) are set, an interrupt is requested.

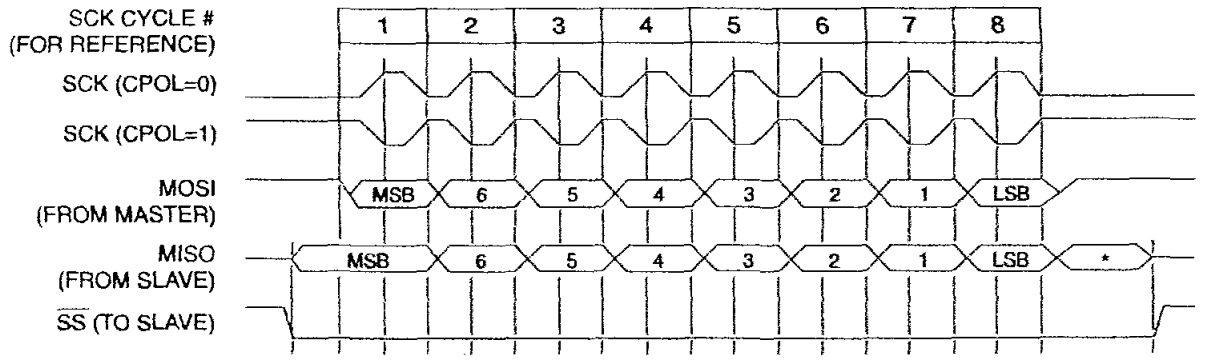
The Slave Select input, $\overline{SS}$/P1.4, is set low to select an individual SPI device as a slave. When $\overline{SS}$/P1.4 is set high, the SPI port is deactivated and the MOSI/P1.5 pin can be used as an input.

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 8 and Figure 9.

gure 7. SPI Master-slave Interconnection

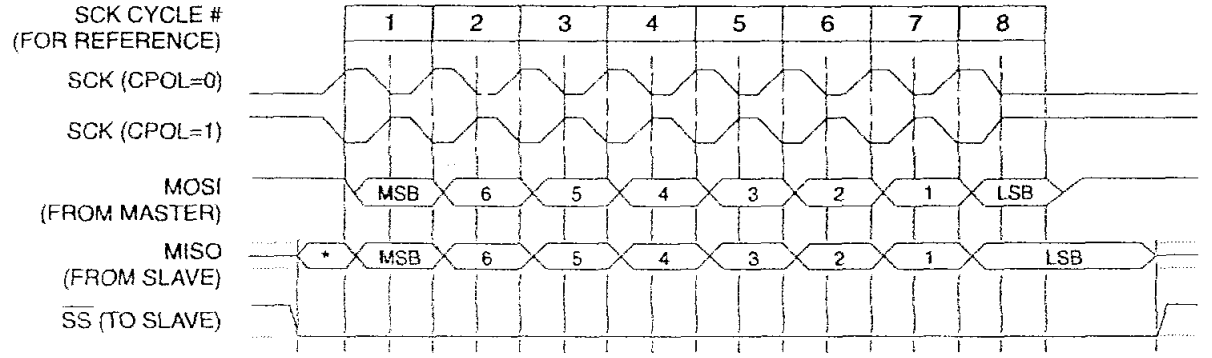


gure 8. SPI transfer Format with CPHA = 0



lot defined but normally MSB of character just received

gure 9. SPI Transfer Format with CPHA = 1



ot defined but normally LSB of previously transmitted character

terrupts

ne AT89S8252 has a total of six interrupt vectors: two ternal interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts imers 0, 1, and 2), and the serial port interrupt. These errupts are all shown in Figure 10.

ach of these interrupt sources can be individually enabled disabled by setting or clearing a bit in Special Function egister IE. IE also contains a global disable bit, EA, which sables all interrupts at once.

ote that Table 10 shows that bit position IE.6 is unimple-ented. In the AT89C51, bit position IE.5 is also implemented. User software should not write 1s to these t positions, since they may be used in future AT89 oducts.

mer 2 interrupt is generated by the logical OR of bits TF2 nd EXF2 in register T2CON. Neither of these flags is eared by hardware when the service routine is vectored . In fact, the service routine may have to determine nether it was TF2 or EXF2 that generated the interrupt, nd that bit will have to be cleared in software.

ne Timer 0 and Timer 1 flags, TF0 and TF1, are set at 5P2 of the cycle in which the timers overflow. The values e then polled by the circuitry in the next cycle. However, e Timer 2 flag, TF2, is set at S2P2 and is polled in the me cycle in which the timer overflows.

Table 10. Interrupt Enable (IE) Register

(MSB)(LSB)

EA	–	ET2	ES	ET1	EX1	ET0	EX0

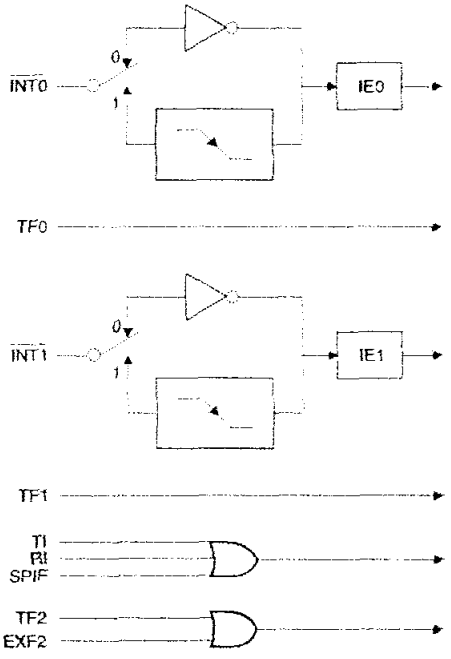
Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	SPI and UART interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.

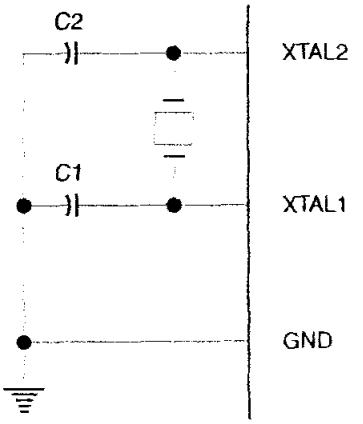
ure 10. Interrupt Sources



scillator Characteristics

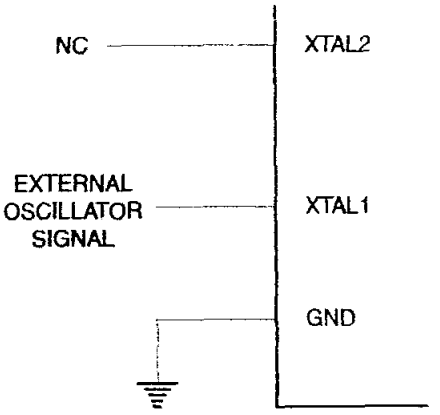
AL1 and XTAL2 are the input and output, respectively, an inverting amplifier that can be configured for use as on-chip oscillator, as shown in Figure 11. Either a quartz stal or ceramic resonator may be used. To drive the vice from an external clock source, XTAL2 should be left connected while XTAL1 is driven, as shown in Figure 12. ere are no requirements on the duty cycle of the external ck signal, since the input to the internal clocking circuitry hrough a divide-by-two flip-flop, but minimum and maxi- m voltage high and low time specifications must be served.

Figure 11. Oscillator Connections



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

Figure 12. External Clock Drive Configuration



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 de. The idle mode can be terminated by any enabled rrupt or by a hardware reset.

te that when idle mode is terminated by a hardware set, 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 hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

atus of External Pins During Idle and Power-down Modes

ode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
le	Internal	1	1	Data	Data	Data	Data
le	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

ower-down Mode

the 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 minated. Exit from power-down can be initiated either by ardware reset or by an enabled external interrupt. Reset defines the SFRs but does not change the on-chip RAM. e reset should not be activated before V_{CC} is restored to normal operating level and must be held active long ough to allow the oscillator to restart and stabilize.

exit power-down via an interrupt, the external interrupt st be enabled as level sensitive before entering power-wn. The interrupt service routine starts at 16 ms (nomi-l) after the enabled interrupt pin is activated.

Program Memory Lock Bits

The AT89S8252 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

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. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

Once programmed, the lock bits can only be unprogrammed with the Chip Erase operations in either the parallel or serial modes.

ock Bit Protection Modes[1][2]

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No internal memory lock feature.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory. $\overline{EA}$ is sampled and latched on reset and further programming of the Flash memory (parallel or serial mode) is disabled.
3	P	P	U	Same as Mode 2, but parallel or serial verify are also disabled.
4	P	P	P	Same as Mode 3, but external execution is also disabled.

tes: 1. U = Unprogrammed
2. P = Programmed

rogramming the Flash and EEPROM

nel's AT89S8252 Flash Microcontroller offers 8K bytes n-system reprogrammable Flash Code memory and 2K es of EEPROM Data memory.

e AT89S8252 is normally shipped with the on-chip Flash de and EEPROM Data memory arrays in the erased te (i.e. contents = FFH) and ready to be programmed. s device supports a High-voltage (12V) Parallel pro- amming mode and a Low-voltage (5V) Serial gramming mode. The serial programming mode pro- es a convenient way to download the AT89S8252 inside user's system. The parallel programming mode is com- tible with conventional third party Flash or EPROM grammers.

e Code and Data memory arrays are mapped via sepa- e address spaces in the serial programming mode. In parallel programming mode, the two arrays occupy one ntiguous address space: 0000H to 1FFFH for the Code ay and 2000H to 27FFH for the Data array.

e Code and Data memory arrays on the AT89S8252 are grammed byte-by-byte in either programming mode. An o-erase cycle is provided with the self-timed program- ng operation in the serial programming mode. There is need to perform the Chip Erase operation to reprogram y memory location in the serial programming mode ess any of the lock bits have been programmed.

the parallel programming mode, there is no auto-erase cle. To reprogram any non-blank byte, the user needs to e the Chip Erase operation first to erase both arrays.

rallel Programming Algorithm: To program and verify AT89S8252 in the parallel programming mode, the fol- ving sequence is recommended:

Power-up sequence:

Apply power between V_{CC} and GND pins.

Set RST pin to "H".

Apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.

Set $\overline{PSEN}$ pin to "L"

ALE pin to "H"

$\overline{EA}$ pin to "H" and all other pins to "H".

Apply the appropriate combination of "H" or "L" logic levels to pins P2.6, P2.7, P3.6, P3.7 to select one of the programming operations shown in the Flash Programming Modes table.

Apply the desired byte address to pins P1.0 to P1.7 and P2.0 to P2.5.

Apply data to pins P0.0 to P0.7 for Write Code operation.

5. Raise $\overline{EA}/V_{PP}$ to 12V to enable Flash programming, erase or verification.
6. Pulse ALE/$\overline{PROG}$ once to program a byte in the Code memory array, the Data memory array or the lock bits. The byte-write cycle is self-timed and typically takes 1.5 ms.
7. To verify the byte just programmed, bring pin P2.7 to "L" and read the programmed data at pins P0.0 to P0.7.
8. Repeat steps 3 through 7 changing the address and data for the entire 2K or 8K bytes array or until the end of the object file is reached.
9. Power-off sequence:

   Set XTAL1 to "L".

   Set RST and $\overline{EA}$ pins to "L".

   Turn V_{CC} power off.

In the parallel programming mode, there is no auto-erase cycle and to reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

Data Polling: The AT89S8252 features $\overline{DATA}$ Polling to indicate the end of a write cycle. During a write cycle in the parallel or serial programming mode, an attempted read of the last byte written will result in the complement of the written datum on P0.7 (parallel mode), and on the MSB of the serial output byte on MISO (serial mode). Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{DATA}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming in the parallel programming mode can also be monitored by the RDY/$\overline{BSY}$ output signal. Pin P3.4 is pulled Low after ALE goes High during programming to indicate $\overline{BUSY}$. P3.4 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed Code or Data byte can be read back via the address and data lines for verification. The state of the lock bits can also be verified directly in the parallel programming mode. In the serial programming mode, the state of the lock bits can only be verified indirectly by observing that the lock bit features are enabled.

Chip Erase: Both Flash and EEPROM arrays are erased electrically at the same time. In the parallel programming mode, chip erase is initiated by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The Code and Data arrays are written with all "1"s in the Chip Erase operation.

the serial programming mode, a chip erase operation is iated by issuing the Chip Erase instruction. In this mode, p erase is self-timed and takes about 16 ms.

ring chip erase, a serial read from any address location return 00H at the data outputs.

rial Programming Fuse: A programmable fuse is avail-le to disable Serial Programming if the user needs ximum system security. The Serial Programming Fuse n only be programmed or erased in the Parallel Program-ng Mode.

e AT89S8252 is shipped with the Serial Programming de enabled.

ading the Signature Bytes: The signature bytes are ad by the same procedure as a normal verification of ations 030H and 031H, except that P3.6 and P3.7 must pulled to a logic low. The values returned are as follows:

(030H) = 1EH indicates manufactured by Atmel
(031H) = 72H indicates 89S8252

rogramming Interface

ery code byte in the Flash and EEPROM arrays can be tten, and the entire array can be erased, by using the propriate combination of control signals. The write oper-on cycle is self-timed and once initiated, will tomatically time itself to completion.

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

erial Downloading

th the Code and Data memory arrays can be pro-ammed using the serial SPI bus while RST is pulled to . The serial interface consists of pins SCK, MOSI (input) d MISO (output). After RST is set high, the Programming able instruction needs to be executed first before pro-am/erase operations can be executed.

auto-erase cycle is built into the self-timed programming eration (in the serial mode ONLY) and there is no need first execute the Chip Erase instruction unless any of the k bits have been programmed. The Chip Erase opera-n turns the content of every memory location in both the de and Data arrays into FFH.

e Code and Data memory arrays have separate address aces:

0000H to 1FFFH for Code memory and 000H to 7FFH for Data memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/40 of the crystal frequency. With a 24 MHz oscillator clock, the maximum SCK frequency is 600 kHz.

Serial Programming Algorithm

To program and verify the AT89S8252 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

   If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
3. The Code or Data array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. The selected memory location is first automatically erased before new data is written. The write cycle is self-timed and typically takes less than 2.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal operation.

Power-off sequence (if needed):

Set XTAL1 to "L" (if a crystal is not used).

Set RST to "L".

Turn V_{CC} power off.

Serial Programming Instruction

The Instruction Set for Serial Programming follows a 3-byte protocol and is shown in the following table:

ıstruction Set

ıstruction	Input Format			Operation
	Byte 1	Byte 2	Byte 3	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	Enable serial programming interface after RST goes high.
Chip Erase	1010 1100	xxxx x100	xxxx xxxx	Chip erase both 8K & 2K memory arrays.
Read Code Memory	aaaa a001	low addr	xxxx xxxx	Read data from Code memory array at the selected address. The 5 MSBs of the first byte are the high order address bits. The low order address bits are in the second byte. Data are available at pin MISO during the third byte.
Write Code Memory	aaaa a010	low addr	data in	Write data to Code memory location at selected address. The address bits are the 5 MSBs of the first byte together with the second byte.
Read Data Memory	00aa a101	low addr	xxxx xxxx	Read data from Data memory array at selected address. Data are available at pin MISO during the third byte.
Write Data Memory	00aa a110	low addr	data in	Write data to Data memory location at selected address.
Write Lock Bits	1010 1100	x x111	xxxx xxxx	Write lock bits. Set LB1, LB2 or LB3 = "0" to program lock bits.

ote: 1. DATA polling is used to indicate the end of a write cycle which typically takes less than 2.5 ms at 5V.
2. "aaaaa" = high order address.
3. "x" = don't care.

ash and EEPROM Parallel Programming Modes

ode	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7	Data I/O P0.7:0	Address P2.5:0 P1.7:0
erial Prog. Modes	H	h(1)	h(1)	x						
hip Erase	H	L	(2)	12V	H	L	L	L	X	X
rite (10K bytes) Memory	H	L		12V	L	H	H	H	DIN	ADDR
ead (10K bytes) Memory	H	L	H	12V	L	L	H	H	DOUT	ADDR
rite Lock Bits:	H	L		12V	H	L	H	L	DIN	X
Bit - 1									P0.7 = 0	X
Bit - 2									P0.6 = 0	X
Bit - 3									P0.5 = 0	X
ead Lock Bits:	H	L	H	12V	H	H	L	L	DOUT	X
Bit - 1									@P0.2	X
Bit - 2									@P0.1	X
Bit - 3									@P0.0	X
ead Atmel Code	H	L	H	12V	L	L	L	L	DOUT	30H
ead Device Code	H	L	H	12V	L	L	L	L	DOUT	31H
erial Prog. Enable	H	L	(2)	12V	L	H	L	H	P0.0 = 0	X
erial Prog. Disable	H	L	(2)	12V	L	H	L	H	P0.0 = 1	X
ead Serial Prog. Fuse	H	L	H	12V	H	H	L	H	@P0.0	X

tes:
1. "h" = weakly pulled "High" internally.
2. Chip Erase and Serial Programming Fuse require a 10 ms $\overline{PROG}$ pulse. Chip Erase needs to be performed first before reprogramming any byte with a content other than FFH.
3. P3.4 is pulled Low during programming to indicate $\overline{RDY/BSY}$.
4. "X" = don't care

ıure 13. Programming the Flash/EEPROM Memory

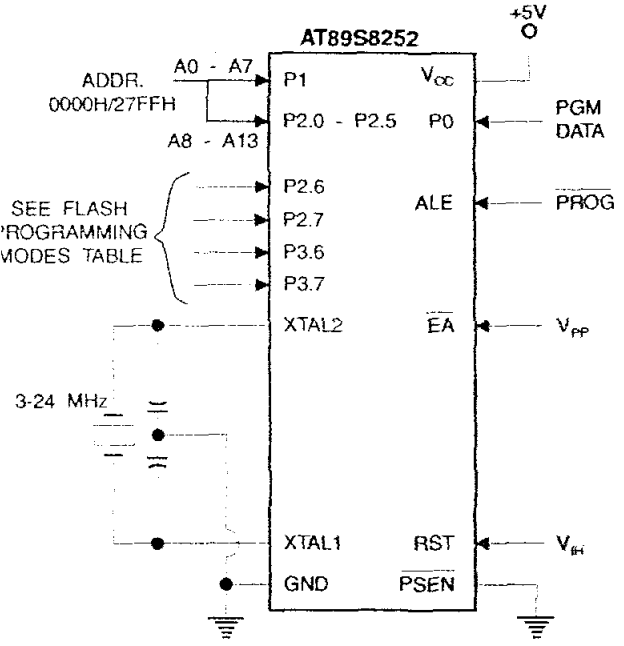
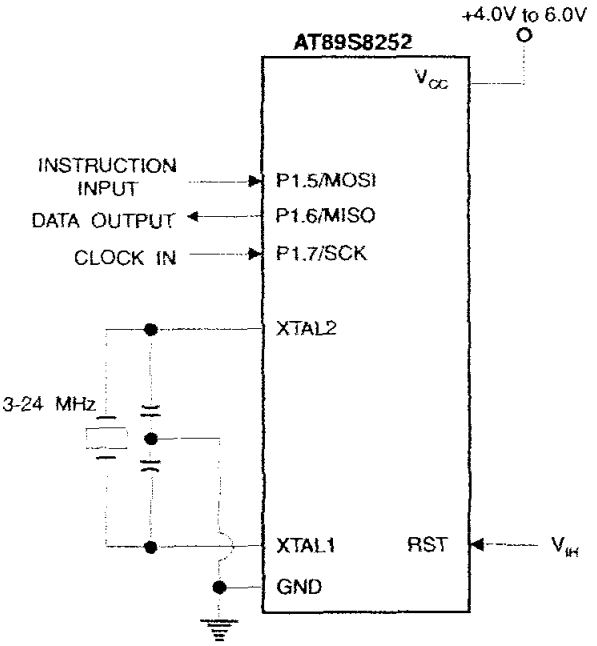
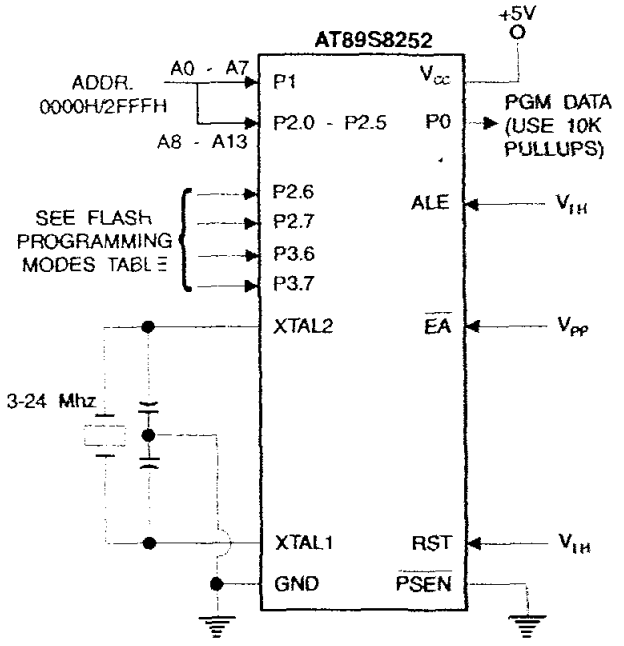


Figure 15. Flash/EEPROM Serial Downloading



gure 14. Verifying the Flash/EEPROM Memory

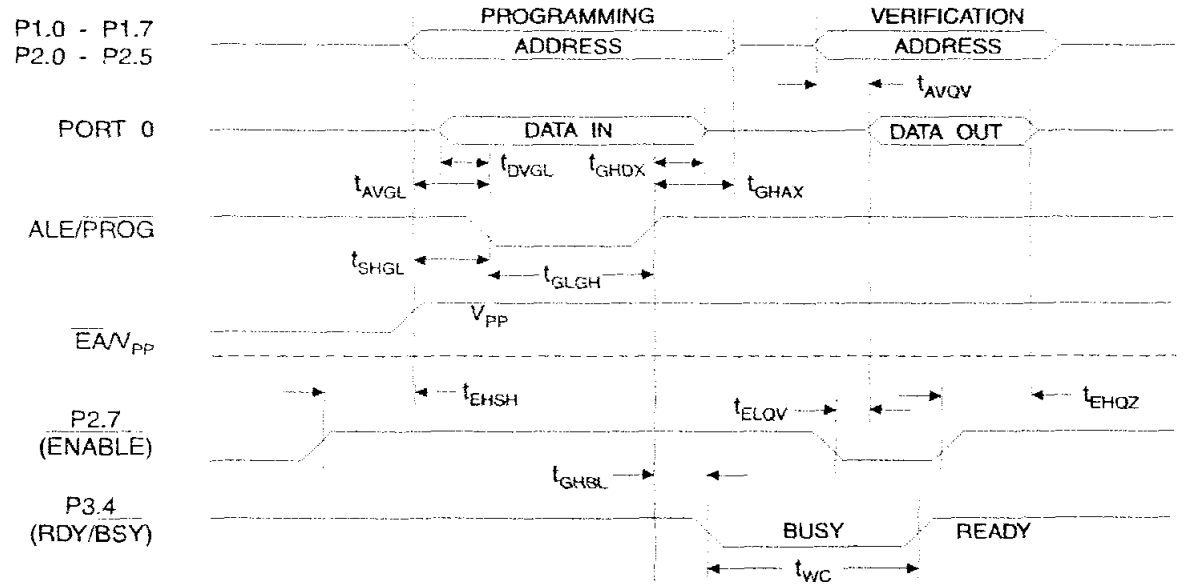


ash Programming and Verification Characteristics – Parallel Mode

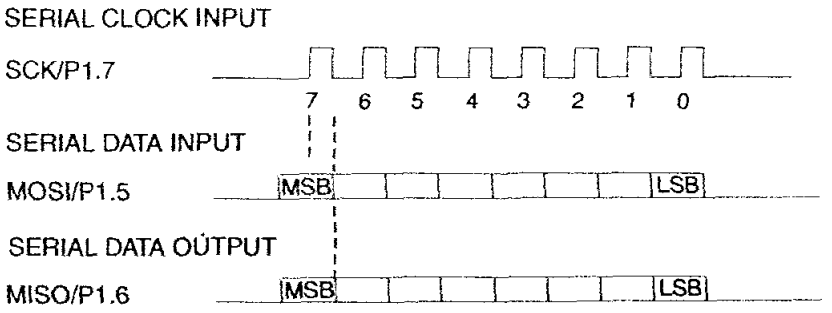
= 0°C to 70°C, V_{CC} = 5.0V ± 10%

mbol	Parameter	Min	Max	Units
P	Programming Enable Voltage	11.5	12.5	V
	Programming Enable Current		1.0	mA
CLCL	Oscillator Frequency	3	24	MHz
GL	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
IAX	Address Hold after $\overline{PROG}$	$48t_{CLCL}$		
GL	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
DX	Data Hold after $\overline{PROG}$	$48t_{CLCL}$		
SH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
GL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
GH	$\overline{PROG}$ Width	1	110	µs
QV	Address to Data Valid		$48t_{CLCL}$	
QV	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
QZ	Data Float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
BL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
C	Byte Write Cycle Time		2.0	ms

ash/EEPROM Programming and Verification Waveforms – Parallel Mode



erial Downloading Waveforms



bsolute Maximum Ratings*

)perating Temperature	-55°C to +125°C
;torage Temperature	-65°C to +150°C
/oltage on Any Pin ;ith Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
)C 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

ıe values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ± 20%, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
V_{IL1}	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
V_{IH}	Input Hifh-voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input Hifh-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low-voltage [1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.5	V
V_{OL1}	Output Low-voltage [1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.5	V
V_{OH}	Output Hifh-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output Hifh-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pull-down Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode [2]	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

otes: 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 I_{OL} for all output pins: 71 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V

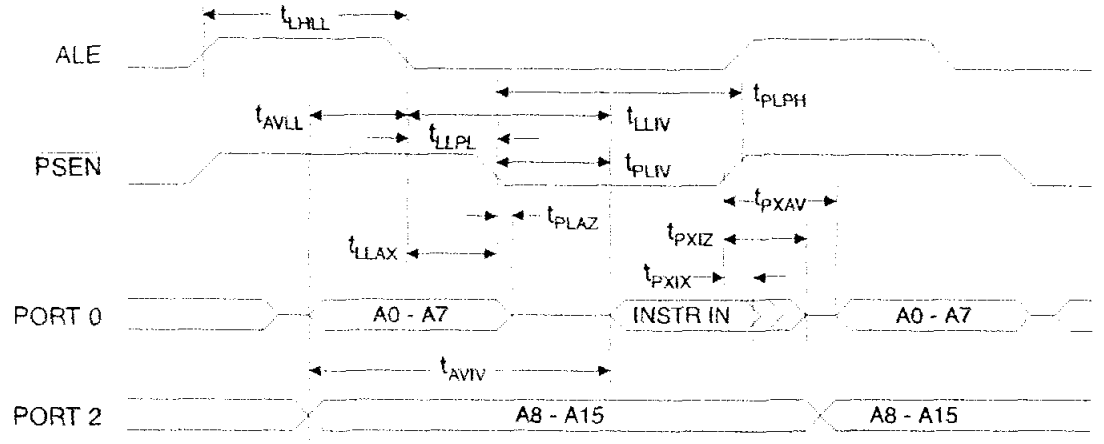
C Characteristics

der operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other tputs = 80 pF.

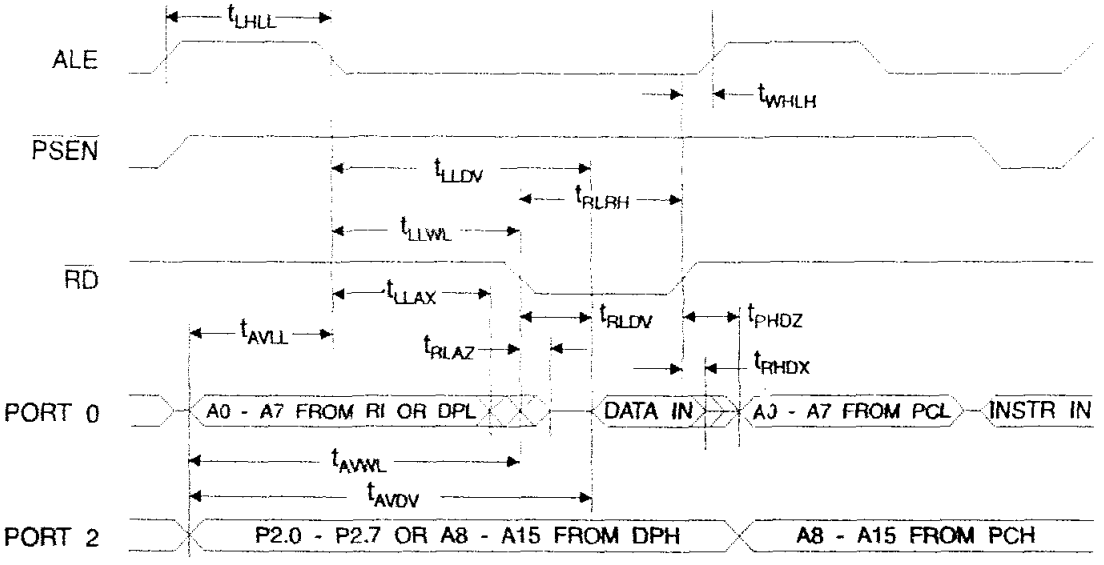
xternal Program and Data Memory Characteristics

ymbol	Parameter	Variable Oscillator		Units
		Min	Max	
$^1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
HLL	ALE Pulse Width	$2t_{CLCL}$ - 40		ns
VLL	Address Valid to ALE Low	t_{CLCL} - 13		ns
LAX	Address Hold after ALE Low	t_{CLCL} - 20		ns
LIV	ALE Low to Valid Instruction In		$4t_{CLCL}$ - 65	ns
LPL	ALE Low to $\overline{PSEN}$ Low	t_{CLCL} - 13		ns
PLPH	$\overline{PSEN}$ Pulse Width	$3t_{CLCL}$ - 20		ns
PLIV	$\overline{PSEN}$ Low to Valid Instruction In		$3t_{CLCL}$ - 45	ns
PXIX	Input Instruction Hold after $\overline{PSEN}$	0		ns
PXIZ	Input Instruction Float after $\overline{PSEN}$		t_{CLCL} - 10	ns
PXAV	$\overline{PSEN}$ to Address Valid	t_{CLCL} - 8		ns
AVIV	Address to Valid Instruction In		$5t_{CLCL}$ - 55	ns
PLAZ	$\overline{PSEN}$ Low to Address Float		10	ns
RLRH	$\overline{RD}$ Pulse Width	$6t_{CLCL}$ - 100		ns
WLWH	$\overline{WR}$ Pulse Width	$6t_{CLCL}$ - 100		ns
RLDV	$\overline{RD}$ Low to Valid Data In		$5t_{CLCL}$ - 90	ns
RHDX	Data Hold after $\overline{RD}$	0		ns
RHDZ	Data Float after $\overline{RD}$		$2t_{CLCL}$ - 28	ns
LDV	ALE Low to Valid Data In		$8t_{CLCL}$ - 150	ns
AVDV	Address to Valid Data In		$9t_{CLCL}$ - 165	ns
LWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$3t_{CLCL}$ - 50	$3t_{CLCL}$ + 50	ns
AVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	$4t_{CLCL}$ - 75		ns
QVWX	Data Valid to $\overline{WR}$ Transition	t_{CLCL} - 20		ns
QVWH	Data Valid to $\overline{WR}$ High	$7t_{CLCL}$ - 120		ns
WHQX	Data Hold after $\overline{WR}$	t_{CLCL} - 20		ns
RLAZ	$\overline{RD}$ Low to Address Float		0	ns
WHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{CLCL} - 20	t_{CLCL} + 25	ns

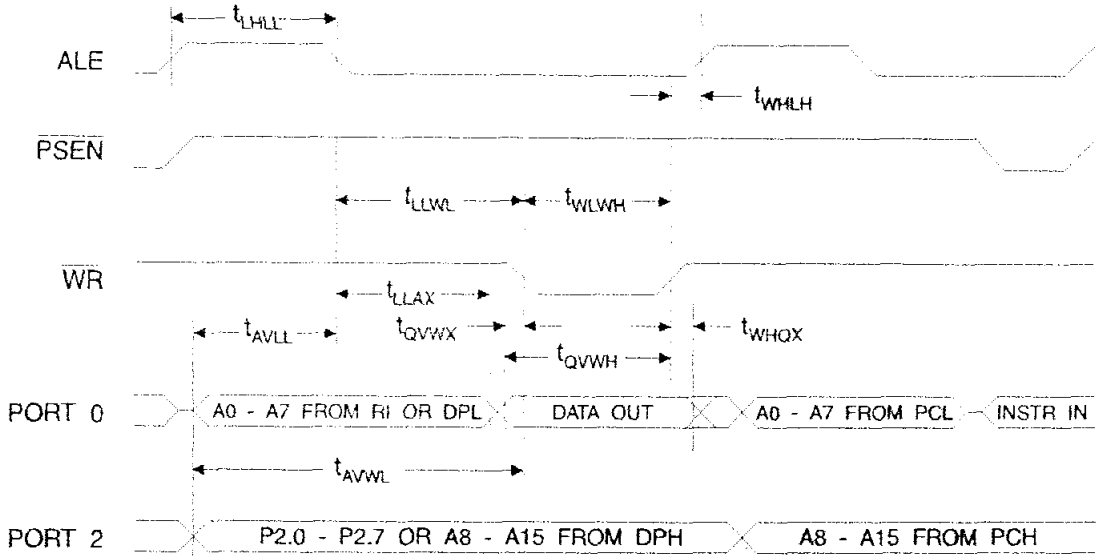
xternal Program Memory Read Cycle



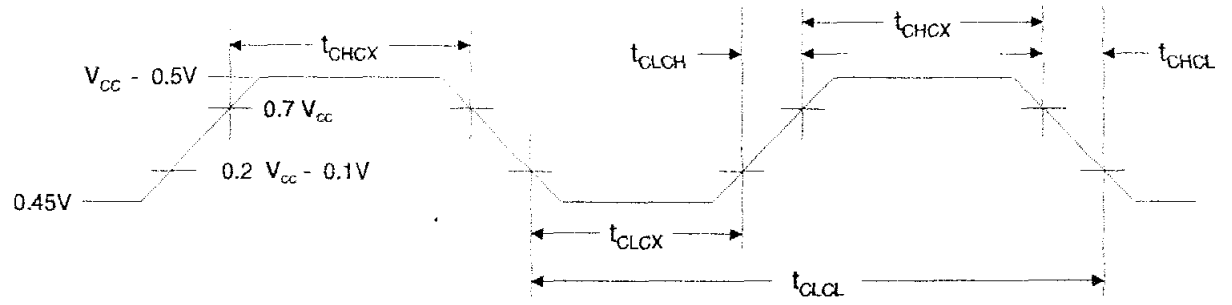
xternal Data Memory Read Cycle



xternal Data Memory Write Cycle



xternal Clock Drive Waveforms



xternal Clock Drive

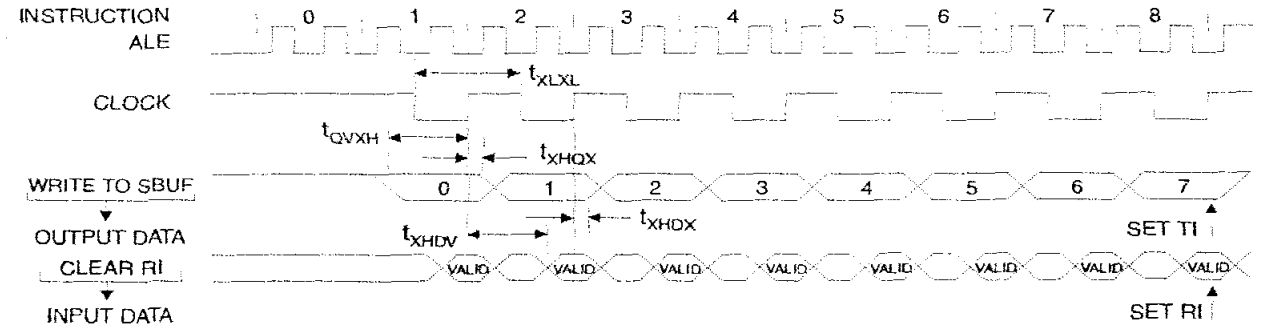
ymbol	Parameter	V_{CC} = 4.0V to 6.0V		Units
		Min	Max	
/t_{CLCL}	Oscillator Frequency	0	24	MHz
LCL	Clock Period	41.6		ns
HCX	High Time	15		ns
LCX	Low Time	15		ns
LCH	Rise Time		20	ns
HCL	Fall Time		20	ns

erial Port Timing: Shift Register Mode Test Conditions

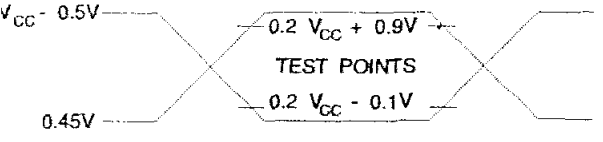
e values in this table are valid for V_{CC} = 4.0V to 6V and Load Capacitance = 80 pF.

ymbol	Parameter	Variable Oscillator		Units
		Min	Max	
LXL	Serial Port Clock Cycle Time	$12t_{CLCL}$		µs
VXH	Output Data Setup to Clock Rising Edge	$10t_{CLCL}$ - 133		ns
HQX	Output Data Hold after Clock Rising Edge	$2t_{CLCL}$ - 117		ns
HDX	Input Data Hold after Clock Rising Edge	0		ns
HDV	Clock Rising Edge to Input Data Valid		$10t_{CLCL}$ - 133	ns

hift Register Mode Timing Waveforms

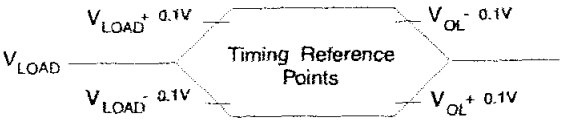


C Testing Input/Output Waveforms[1]



tes: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V 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]



Notes: 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.

LAMPIRAN E

National Semiconductor

August 2000

LM741
Operational Amplifier

General Description

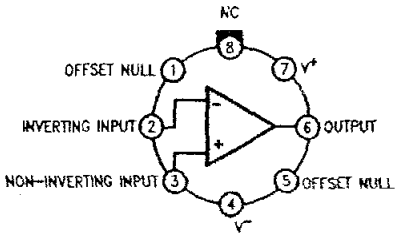
The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications.

The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

The LM741C is identical to the LM741/LM741A except that the LM741C has their performance guaranteed over a 0°C to +70°C temperature range, instead of −55°C to +125°C.

Connection Diagrams

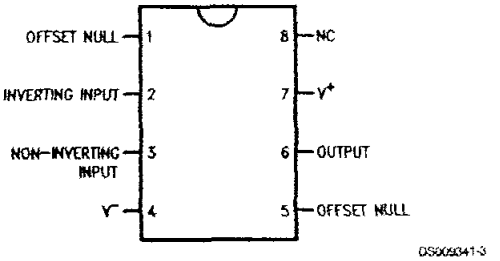
Metal Can Package



Note 1: LM741H is available per JM38510/10101

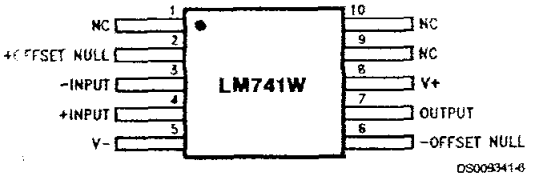
Order Number LM741H, LM741H/883 (Note 1), LM741AH/883 or LM741CH
See NS Package Number H08C

Dual-In-Line or S.O. Package



Order Number LM741J, LM741J/883, LM741CN
See NS Package Number J08A, M08A or N08E

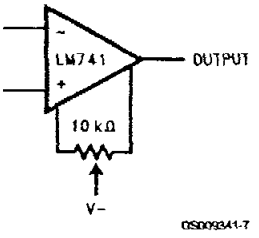
Ceramic Flatpak



Order Number LM741W/883
See NS Package Number W10A

Typical Application

Offset Nulling Circuit



Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

(Note 7)

	LM741A	LM741	LM741C
Supply Voltage	±22V	±22V	±18V
Power Dissipation (Note 3)	500 mW	500 mW	500 mW
Differential Input Voltage	±30V	±30V	±30V
Input Voltage (Note 4)	±15V	±15V	±15V
Output Short Circuit Duration	Continuous	Continuous	Continuous
Operating Temperature Range	−55°C to +125°C	−55°C to +125°C	0°C to +70°C
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C	−65°C to +150°C
Junction Temperature	150°C	150°C	100°C
Soldering Information			
N-Package (10 seconds)	260°C	260°C	260°C
J- or H-Package (10 seconds)	300°C	300°C	300°C
M-Package			
Vapor Phase (60 seconds)	215°C	215°C	215°C
Infrared (15 seconds)	215°C	215°C	215°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.			
ESD Tolerance (Note 8)	400V	400V	400V

Electrical Characteristics (Note 5)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	$T_A = 25°C$										
	$R_S \le 10\ k\Omega$					1.0	5.0		2.0	6.0	mV
	$R_S \le 50\Omega$		0.8	3.0							mV
	$T_{AMIN} \le T_A \le T_{AMAX}$										
	$R_S \le 50\Omega$			4.0							mV
	$R_S \le 10\ k\Omega$						6.0			7.5	mV
Average Input Offset Voltage Drift				15							µV/°C
Input Offset Voltage Adjustment Range	$T_A = 25°C$, $V_S = \pm 20V$	±10				±15			±15		mV
Input Offset Current	$T_A = 25°C$		3.0	30		20	200		20	200	nA
	$T_{AMIN} \le T_A \le T_{AMAX}$			70		85	500			300	nA
Average Input Offset Current Drift				0.5							nA/°C
Input Bias Current	$T_A = 25°C$		30	80		80	500		80	500	nA
	$T_{AMIN} \le T_A \le T_{AMAX}$			0.210			1.5			0.8	µA
Input Resistance	$T_A = 25°C$, $V_S = \pm 20V$	1.0	6.0		0.3	2.0		0.3	2.0		MΩ
	$T_{AMIN} \le T_A \le T_{AMAX}$, $V_S = \pm 20V$	0.5									MΩ
Input Voltage Range	$T_A = 25°C$							±12	±13		V
	$T_{AMIN} \le T_A \le T_{AMAX}$				±12	±13					V

Electrical Characteristics (Note 5) (Continued)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
rge Signal Voltage Gain	T_A = 25˚C, $R_L \geq 2$ kΩ										
	V_S = ±20V, V_O = ±15V	50									V/mV
	V_S = ±15V, V_O = ±10V				50	200		20	200		V/mV
	$T_{AMIN} \leq T_A \leq T_{AMAX}$, $R_L \geq 2$ kΩ,										
	V_S = ±20V, V_O = ±15V	32									V/mV
	V_S = ±15V, V_O = ±10V				25			15			V/mV
	V_S = ±5V, V_O = ±2V	10									V/mV
utput Voltage Swing	V_S = ±20V										
	$R_L \geq 10$ kΩ	±16									V
	$R_L \geq 2$ kΩ	±15									V
	V_S = ±15V										
	$R_L \geq 10$ kΩ				±12	±14		±12	±14		V
	$R_L \geq 2$ kΩ				±10	±13		±10	±13		V
utput Short Circuit	T_A = 25˚C	10	25	35		25			25		mA
urrent	$T_{AMIN} \leq T_A \leq T_{AMAX}$	10		40							mA
ommon-Mode	$T_{AMIN} \leq T_A \leq T_{AMAX}$										
ejection Ratio	$R_S \leq 10$ kΩ, V_{CM} = ±12V				70	90		70	90		dB
	$R_S \leq 50$Ω, V_{CM} = ±12V	80	95								dB
upply Voltage Rejection	$T_{AMIN} \leq T_A \leq T_{AMAX}$,										
atio	V_S = ±20V to V_S = ±5V										
	$R_S \leq 50$Ω	86	96								dB
	$R_S \leq 10$ kΩ				77	96		77	96		dB
ansient Response	T_A = 25˚C, Unity Gain										
Rise Time			0.25	0.8		0.3			0.3		µs
Overshoot			6.0	20		5			5		%
andwidth (Note 6)	T_A = 25˚C	0.437	1.5								MHz
ew Rate	T_A = 25˚C, Unity Gain	0.3	0.7			0.5			0.5		V/µs
upply Current	T_A = 25˚C					1.7	2.8		1.7	2.8	mA
ower Consumption	T_A = 25˚C										
	V_S = ±20V		80	150							mW
	V_S = ±15V					50	85		50	85	mW
LM741A	V_S = ±20V										
	$T_A = T_{AMIN}$			165							mW
	$T_A = T_{AMAX}$			135							mW
LM741	V_S = ±15V										
	$T_A = T_{AMIN}$					60	100				mW
	$T_A = T_{AMAX}$					45	75				mW

Note 2: "Absolute Maximum Ratings" indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits.

Electrical Characteristics (Note 5) (Continued)

Note 3: For operation at elevated temperatures, these devices must be derated based on thermal resistance, and T_j max. (listed under "Absolute Maximum Ratings"). $T_j = T_A + (\theta_{jA} P_D)$.

Thermal Resistance	Cerdip (J)	DIP (N)	HO8 (H)	SO-8 (M)
θ_{jA} (Junction to Ambient)	100°C/W	100°C/W	170°C/W	195°C/W
θ_{jC} (Junction to Case)	N/A	N/A	25°C/W	N/A

Note 4: For supply voltages less than ±15V, the absolute maximum input voltage is equal to the supply voltage.

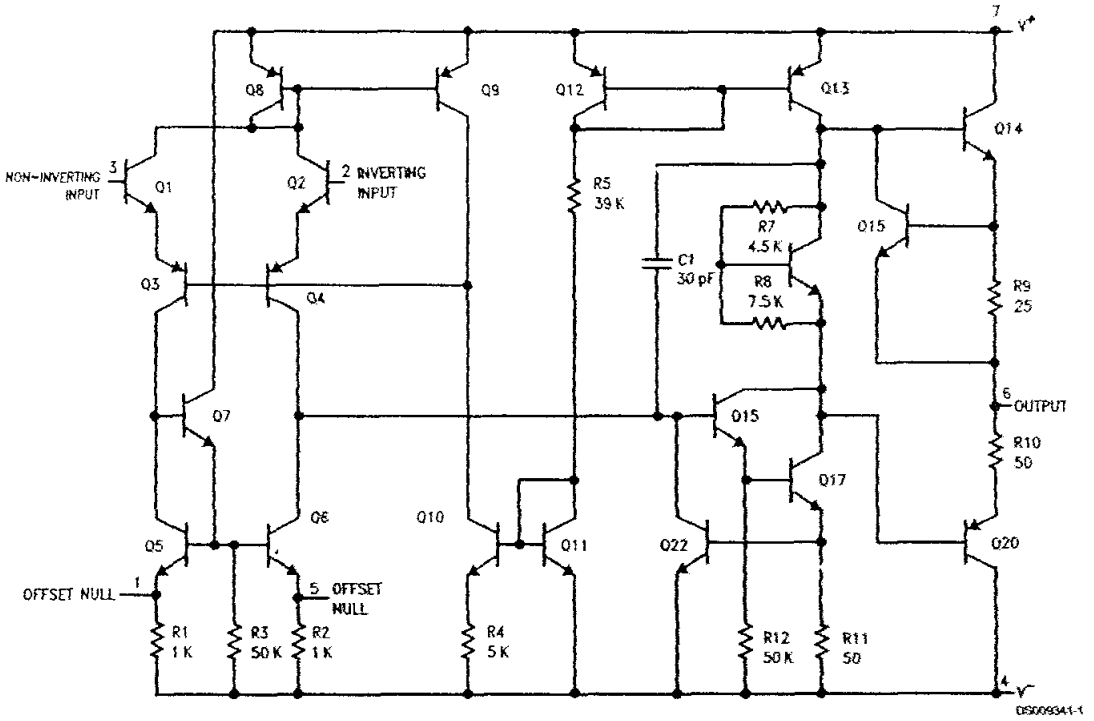
Note 5: *Unless otherwise specified, these specifications apply for* V_S = ±15V, −55°C ≤ T_A ≤ +125°C (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to 0°C ≤ T_A ≤ +70°C.

Note 6: *Calculated value from: BW (MHz) = 0.35/Rise Time(μs).*

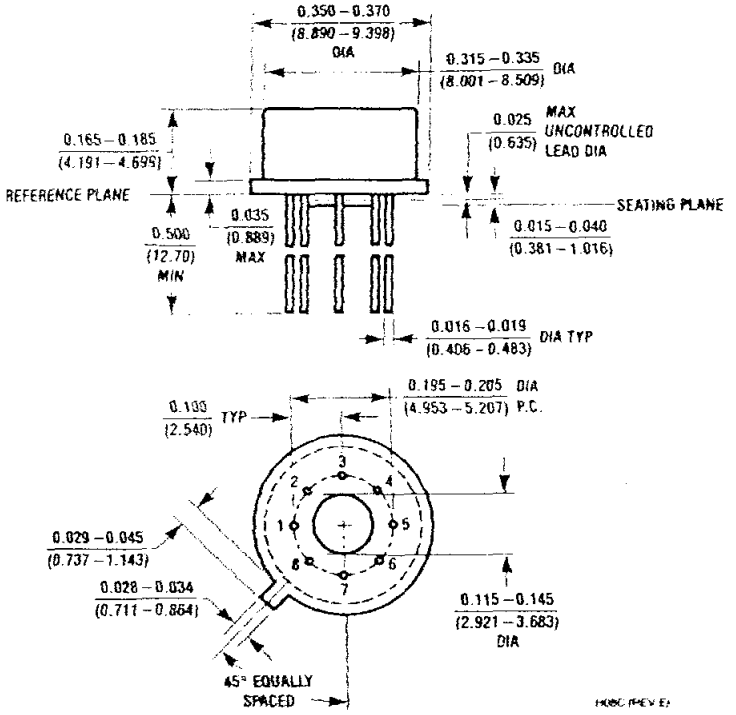
Note 7: For military specifications see RETS741X for LM741 and RETS741AX for LM741A.

Note 8: Human body model, 1.5 kΩ in series with 100 pF.

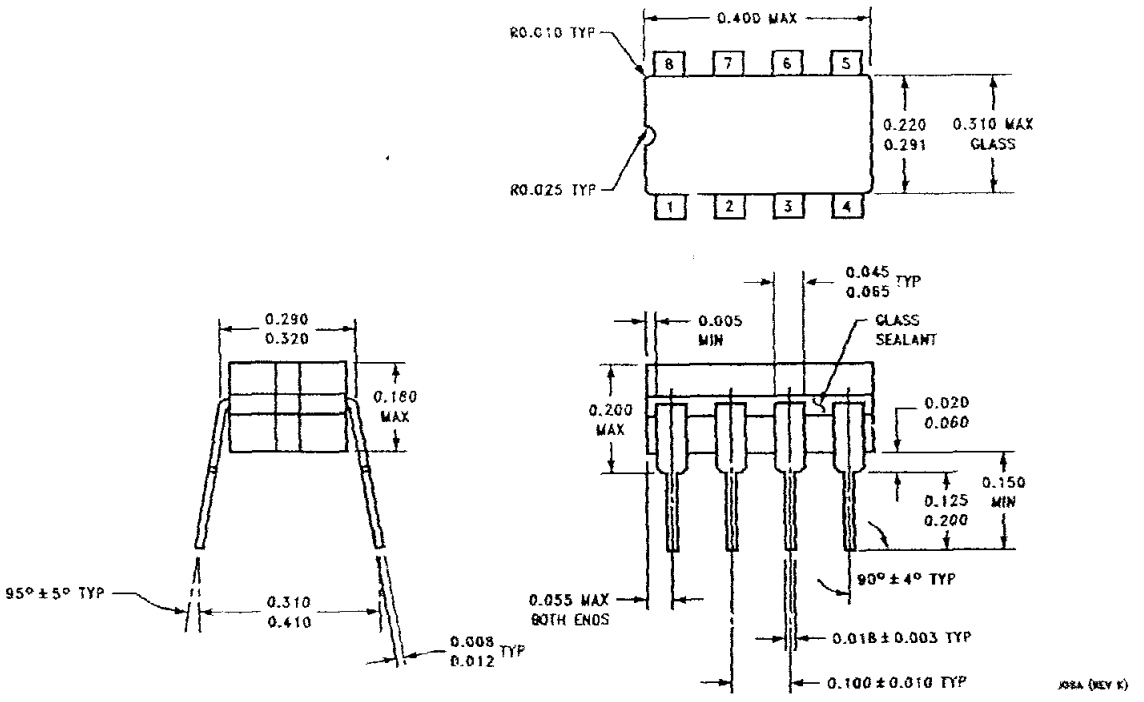
Schematic Diagram



Physical Dimensions inches (millimeters) unless otherwise noted

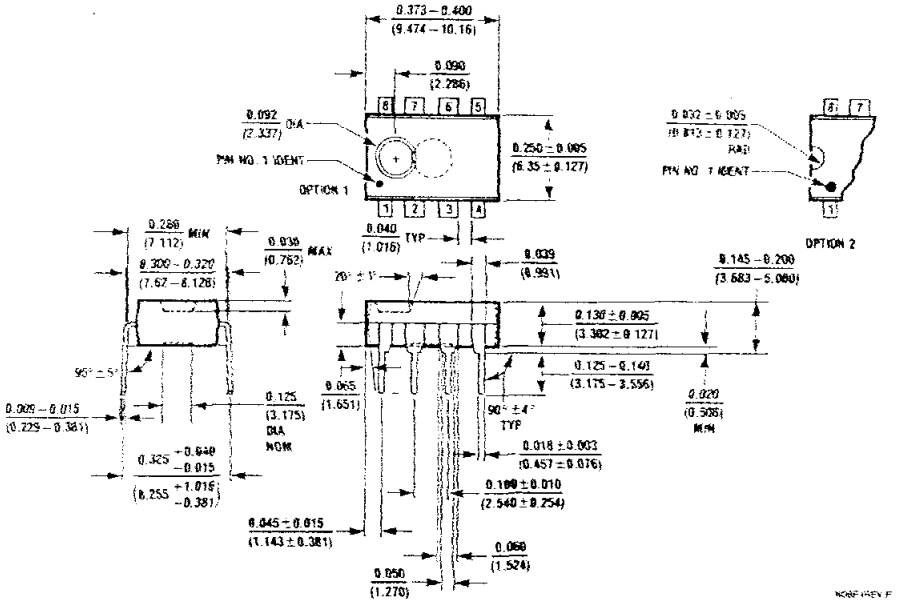


Metal Can Package (H)
Order Number LM741H, LM741H/883, LM741AH/883, LM741AH-MIL or LM741CH
NS Package Number H08C

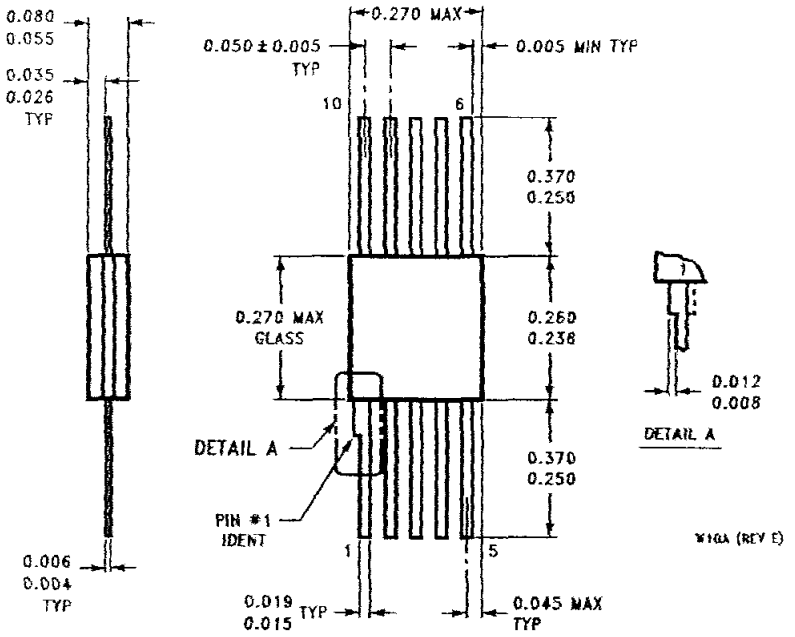


Ceramic Dual-In-Line Package (J)
Order Number LM741J/883
NS Package Number J08A

Physical Dimensions *inches (millimeters) unless otherwise noted (Continued)*



Dual-In-Line Package (N)
Order Number LM741CN
NS Package Number N08E



10-Lead Ceramic Flatpak (W)
Order Number LM741W/883, LM741WG-MPR or LM741WG/883
NS Package Number W10A

LAMPIRAN F

National Semiconductor

February 1995

LM78XX Series Voltage Regulators

General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Considerable effort was expanded to make the LM78XX series of regulators easy to use and mininize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

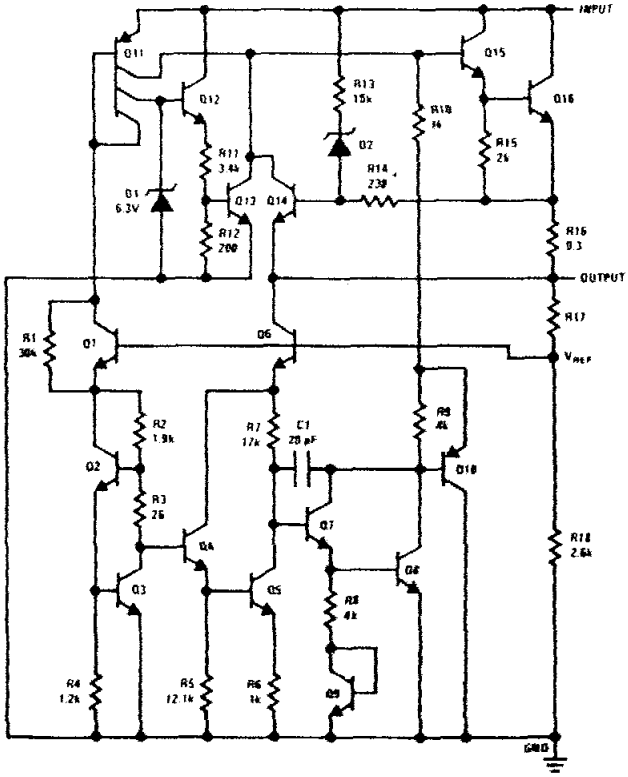
Features

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

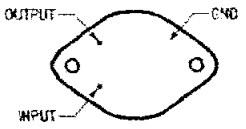
LM7805C	5V
LM7812C	12V
LM7815C	15V

Schematic and Connection Diagrams



TL/H/7746–1

Metal Can Package
TO-3 (K)
Aluminum

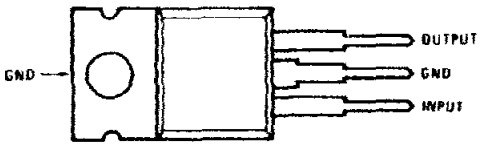


TL/H/7746–2

Bottom View

Order Number LM7805CK, LM7812CK or LM7815CK
See NS Package Number KC02A

Plastic Package
TO-220 (T)



TL/H/7746–3

Top View

Order Number LM7805CT, LM7812CT or LM7815CT
See NS Package Number T03B

RRD-B30M115/Printed in U. S. A.

solute Maximum Ratings

litary/Aerospace specified devices are required, se contact the National Semiconductor Sales e/Distributors for availability and specifications.

Voltage (V_O = 5V, 12V and 15V)	35V
al Power Dissipation (Note 1)	Internally Limited
ating Temperature Range (T_A)	0°C to +70°C
Maximum Junction Temperature	
(K Package)	150°C
(T Package)	150°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	
TO-3 Package K	300°C
TO-220 Package T	230°C

ctrical Characteristics LM78XXC (Note 2) 0°C ≤ Tj ≤ 125°C unless otherwise noted.

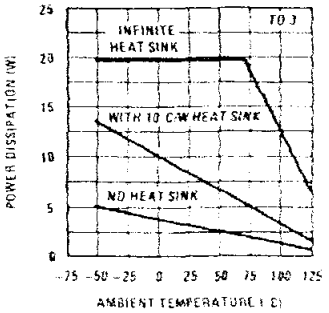
Output Voltage			5V			12V			15V			Units
Input Voltage (unless otherwise noted)			10V			19V			23V			
Parameter	Conditions		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output Voltage	Tj = 25°C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
	P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A		4.75		5.25	11.4		12.6	14.25		15.75	V
	V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.5 ≤ V_{IN} ≤ 27)			(17.5 ≤ V_{IN} ≤ 30)			V
Line Regulation	I_O = 500 mA	Tj = 25°C		3	50		4	120		4	150	mV
		ΔV_{IN}	(7 ≤ V_{IN} ≤ 25)			14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
		0°C ≤ Tj ≤ +125°C			50			120			150	mV
		ΔV_{IN}	(8 ≤ V_{IN} ≤ 20)			(15 ≤ V_{IN} ≤ 27)			(18.5 ≤ V_{IN} ≤ 30)			V
	I_O ≤ 1A	Tj = 25°C			50			120			150	mV
		ΔV_{IN}	(7.5 ≤ V_{IN} ≤ 20)			(14.6 ≤ V_{IN} ≤ 27)			(17.7 ≤ V_{IN} ≤ 30)			V
		0°C ≤ Tj ≤ +125°C			25			60			75	mV
		ΔV_{IN}	(8 ≤ V_{IN} ≤ 12)			(16 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)			V
Load Regulation	Tj = 25°C	5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
		250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
	5 mA ≤ I_O ≤ 1A, 0°C ≤ Tj ≤ +125°C				50			120			150	mV
Quiescent Current	I_O ≤ 1A	Tj = 25°C			8			8			8	mA
		0°C ≤ Tj ≤ +125°C			8.5			8.5			8.5	mA
Quiescent Current Change	5 mA ≤ I_O ≤ 1A				0.5			0.5			0.5	mA
	Tj = 25°C, I_O ≤ 1A				1.0			1.0			1.0	mA
	V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
	I_O ≤ 500 mA, 0°C ≤ Tj ≤ +125°C				1.0			1.0			1.0	mA
	V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
Output Noise Voltage	T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		μV
Ripple Rejection	f = 120 Hz	I_O ≤ 1A, Tj = 25°C or	62	80		55	72		54	70		dB
		I_O ≤ 500 mA	62			55			54			dB
		0°C ≤ Tj ≤ +125°C										
	V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)			V
Dropout Voltage	Tj = 25°C, I_{OUT} = 1A			2.0			2.0			2.0		V
Output Resistance	f = 1 kHz			8			18			19		mΩ
Short-Circuit Current	Tj = 25°C			2.1			1.5			1.2		A
Peak Output Current	Tj = 25°C			2.4			2.4			2.4		A
Average TC of V_{OUT}	0°C ≤ Tj ≤ +125°C, I_O = 5 mA			0.6			1.5			1.8		mV/°C
Input Voltage Required to Maintain Line Regulation	Tj = 25°C, I_O ≤ 1A			7.5		14.6			17.7			V

I: Thermal resistance of the TO-3 package (K, KC) is typically 4°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 ge (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

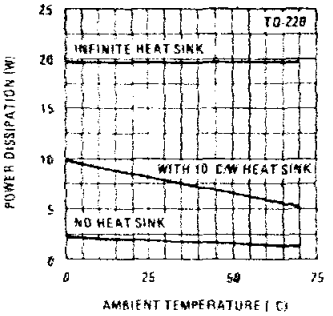
2: All characteristics are measured with capacitor across the input of 0.22 μF, and a capacitor across the output of 0.1 μF. All characteristics except noise and ripple rejection ratio are measured using pulse techniques (t_w ≤ 10 ms, duty cycle ≤ 5%). Output voltage changes due to changes in internal rature must be taken into account separately.

Typical Performance Characteristics

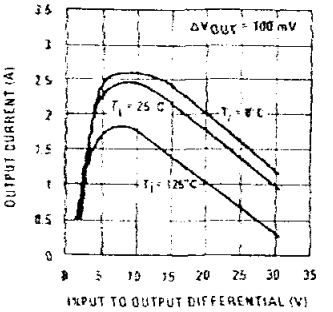
Maximum Average Power Dissipation



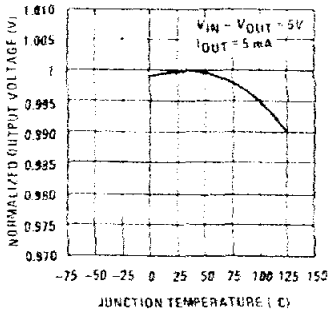
Maximum Average Power Dissipation



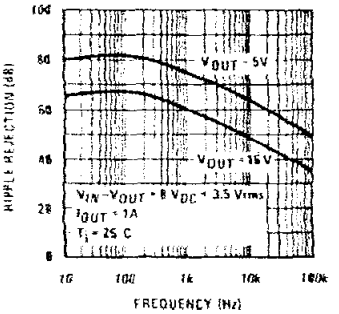
Peak Output Current



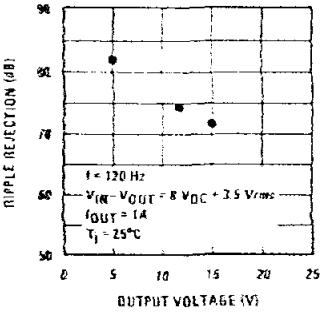
Output Voltage (Normalized to 1V at Tj = 25°C)



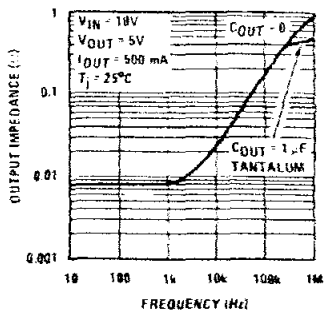
Ripple Rejection



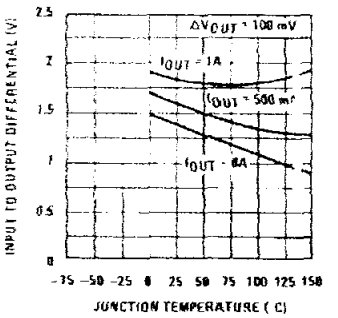
Ripple Rejection



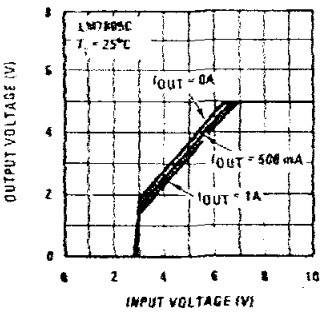
Output Impedance



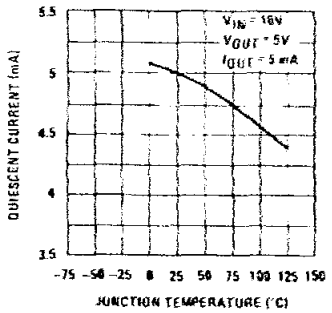
Dropout Voltage



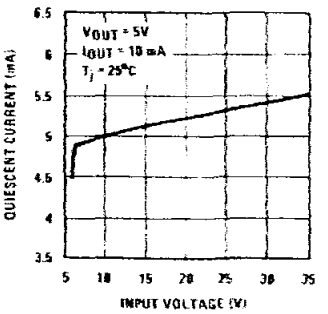
Dropout Characteristics



Quiescent Current

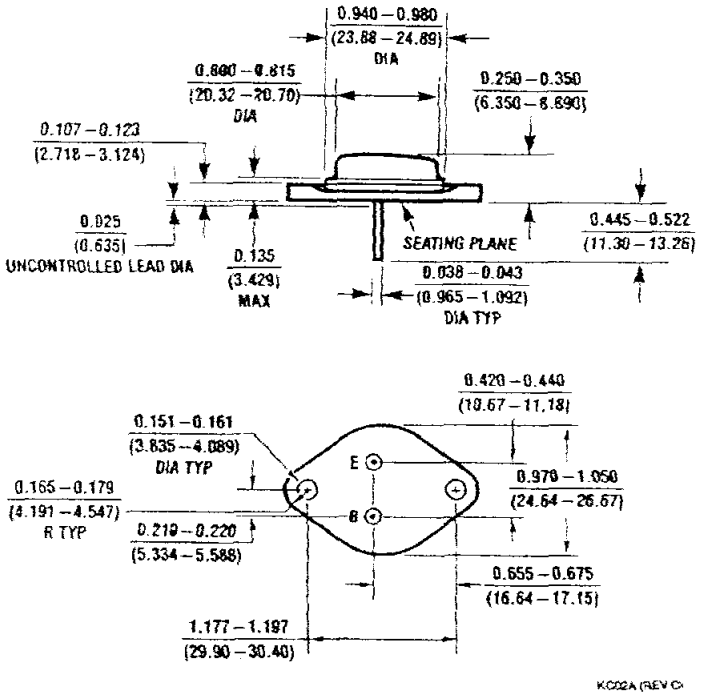


Quiescent Current



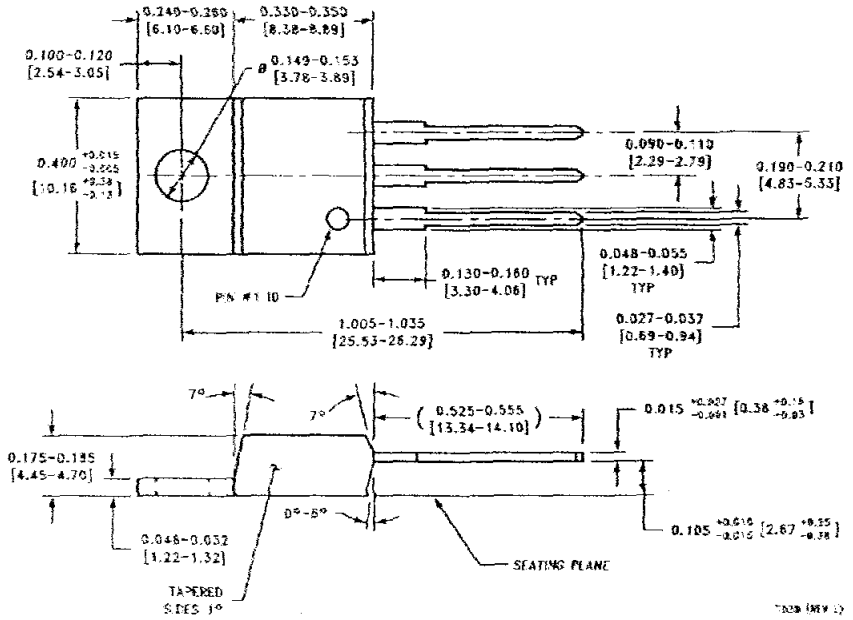
TL/H/7746–4

Physical Dimensions inches (millimeters)



Aluminum Metal Can Package (KC)
Order Number LM7805CK, LM7812CK or LM7815CK
NS Package Number KC02A

Physical Dimensions inches (millimeters) (Continued)



TO-220 Package (T)
Order Number LM7805CT, LM7812CT or LM7815CT
NS Package Number T03B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

BIODATA

Nama : Chornelius Ariesta Abdi

NRP : 5103000066

Tempat, Tgl. Lahir : Surabaya, 07-04-1982

Agama : Kristen

Alamat Rumah : Jl. Bendul Merisi Permai E/4

Surabaya.

Riwayat Pendidikan :

- Tahun 1994 Lulus SDK. Kartini, Surabaya.
- Tahun 1997 Lulus SMPK Santo Carolus, Surabaya.
- Tahun 2000 Lulus SMK Kristen Petra, Surabaya.