

## BAB VI

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah alat pemanfaatan mikrokontroler *AT89S52* untuk mencetak karakter pada printer *dot matrik* berjalan dengan baik. Pengetikan dengan menggunakan *keyboard* sangat efektif karena tidak membutuhkan keahlian dan ketrampilan khusus. Penulisan karakter dengan menggunakan *LCD 16 x 2* sangat efektif karena dapat menampilkan karakter tulisan, angka dan simbol dengan baik. Mencetak karakter dengan menggunakan printer *dot matrik* sangat efektif karena hasil cetak yang dihasilkan sangat dapat dibaca oleh setiap orang serta untuk pemilihan *font* cukup menekan tombol yang telah tersedia pada printer.

#### 6.2 Saran

Berdasarkan hasil penelitian pada perancangan alat yang dibuat, maka untuk perkembangan selanjutnya disarankan agar:

1. Menggunakan *LCD* grafik atau *dot matrik moving sign* agar tampilan karakter lebih besar dan sempurna.
2. Menggunakan printer *dot matrik* yang kecil agar dapat diletakkan dimana saja.
3. Penambahan sistem *barcode* pada *port* mikrokontroler yang masih kosong.

## DAFTAR PUSTAKA

Anonimaus, 1987, *Liquid Crystal Display Module*, Seiko Instrument Inc, Jepang.

Anonimaus, 1989, "*User's Guide LX-800*", SEIKO EPSON Corporation, Nagano, Jepang.

Cross, N., 1994, *Engineering Design Methods: Strategies for Product Design* 2<sup>nd</sup> edition, John Wiley and Sons, USA.

Mahendra, Ivan, 2008, *Penampil Karakter Pada LCD 16 X 2 Dengan Menggunakan Keyboard*, Skripsi di Program Studi Teknik Industri, Fakultas Teknologi Industri Universitas Atma Jaya, Yogyakarta.

Nalwan, Paulus A., 2003, *Panduan Praktis Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51*. Jakarta Elex Media Komputindo.

Putra, A.E, 2006, *Belajar Mikrokontroler AT89C51/52/55 Teori dan Aplikasi*, Gava Media, Yogyakarta.

Wasito, S., 1986, *Vademekum Elektronika*, PT. Gramedia Pustaka Utama, Jakarta.

Wichit, Sirichote, 2003, *Experimenting the 2051 with C Programming*, [kswichit@kmitl.ac.th](mailto:kswichit@kmitl.ac.th) .

Lampiran 1 : Program

```
#include <at89x52.h>
//
#define LCD_RS          P0_0
#define LCD_RW          P0_1
#define LCD_E          P0_2
#define LCD_port      P0
//
#define busy            P3_0
#define strobe          P3_1
#define data_print      P2
//
#define CLK            P3_2
#define DAT            P3_3
//
data unsigned char
data_out,alamat,data_keyboard,sementara,kode_shift,kode
_del,kode_print,ngeprint;
data unsigned char
L10,L11,L12,L13,L14,L15,L16,L17,L18,L19,L1a,L1b,L1c,L1d
,L1e,L1f;
data unsigned char
L20,L21,L22,L23,L24,L25,L26,L27,L28,L29,L2a,L2b,L2c,L2d
,L2e,L2f;
//
111111111122222222223333333333344444444444
//
01234567890123456789012345678901234567890123456789
code unsigned char  kosong[16]      ={"
"};
```

```
//
code unsigned char key_besar[132] ={" ~
Q! ZSAW@ CXDE$# VFTR% N"
"BHGY^ MJU&* <KIO)(
>?L:P_ ' {+ } | "
" 1 47 0.2568 +3-*9
"};
//
code unsigned char key_kecil[132] ={" `
q1 zsaW2 cxde43 vftr5 n"
"bhgy6 mju78 ,kio09
./l;p- ' [= ] | "
" 1 47 0.2568 +3-*9
"};
//-----
//program tundaan
//-----
void long_delay(unsigned int nilai)
{
    unsigned char i;
    unsigned int j;
    for(i=0;i<255;i++)
        {for(j=0;j<nilai;j++);}
}

void short_delay(unsigned char nilai)
{
    unsigned char i;
    for(i=0;i<nilai;i++)
    {
        _asm
```

```
        nop
        _endasm;
    }
}
//
void delay_print()
{
    unsigned char i,j;
    for(i=0;i<20;i++)
        {for(j=0;j<2;j++);}
}
//-----
//program inisialisasi LCD
//-----
void reset_LCD(unsigned char kode)
{
    LCD_port=kode;
    LCD_RS=0;LCD_RW=0;
    LCD_E=1;short_delay(100);LCD_E=0;
}
void tulis_inst_LCD(unsigned char kode)
{
    unsigned char nibble_h,nibble_l;

    nibble_h=kode&0xf0;
    LCD_port=nibble_h;
    LCD_RS=0;LCD_RW=0;
    LCD_E=1;short_delay(100);LCD_E=0;

    short_delay(1);
```

```
nible_l=(kode&0x0f)<<4;
LCD_port=nible_l;
LCD_RS=0;LCD_RW=0;
LCD_E=1;short_delay(100);LCD_E=0;
}
void tulis_data_LCD(unsigned char kode)
{
    unsigned char nible_h,nible_l;

    nible_h=kode&0xf0;
    LCD_port=nible_h;
    LCD_RS=1;LCD_RW=0;
    LCD_E=1;short_delay(100);LCD_E=0;

    short_delay(1);

    nible_l=(kode&0x0f)<<4;
    LCD_port=nible_l;
    LCD_RS=1;LCD_RW=0;
    LCD_E=1;short_delay(100);LCD_E=0;
}

void init_LCD()
{
    long_delay(60);           //tunda selama 15.3 msec

    reset_LCD(0x30);long_delay(17); //tunda selama 4.3
msec
    reset_LCD(0x30);short_delay(100); //tunda selama
100 usec
```

```
    reset_LCD(0x30);short_delay(40); //tunda selama 40
usec
    reset_LCD(0x20);short_delay(40); //tunda selama 40
usec

    tulis_inst_LCD(0x28);short_delay(40); //tunda selama
40 usec
    tulis_inst_LCD(0x06);short_delay(40); //tunda selama
40 usec
    tulis_inst_LCD(0x0c);short_delay(40); //tunda selama
40 usec
    tulis_inst_LCD(0x01);long_delay(7); //tunda selama
1.7 msec
}
//-----
//program print
//-----
void print_go(unsigned char ngeprint)
{
    data_print=ngeprint;
    while(busy==1){;}
    strobe=1;strobe=0;
    delay_print();
    strobe=1;
    delay_print();
}
//
void space()
{
    print_go(0x0d);
    delay_print(); //line feed
```

```
    print_go(0x0a);
}
//=====
//progam penyimpan data tulisan ke mikro
//=====
void simpan()
{
    if(alamat==0x80)
    {L10=sementara;}
    else if(alamat==0x81)
    {L11=sementara;}
    else if(alamat==0x82)
    {L12=sementara;}
    else if(alamat==0x83)
    {L13=sementara;}
    else if(alamat==0x84)
    {L14=sementara;}
    else if(alamat==0x85)
    {L15=sementara;}
    else if(alamat==0x86)
    {L16=sementara;}
    else if(alamat==0x87)
    {L17=sementara;}
    else if(alamat==0x88)
    {L18=sementara;}
    else if(alamat==0x89)
    {L19=sementara;}
    else if(alamat==0x8a)
    {L1a=sementara;}
    else if(alamat==0x8b)
    {L1b=sementara;}
}
```



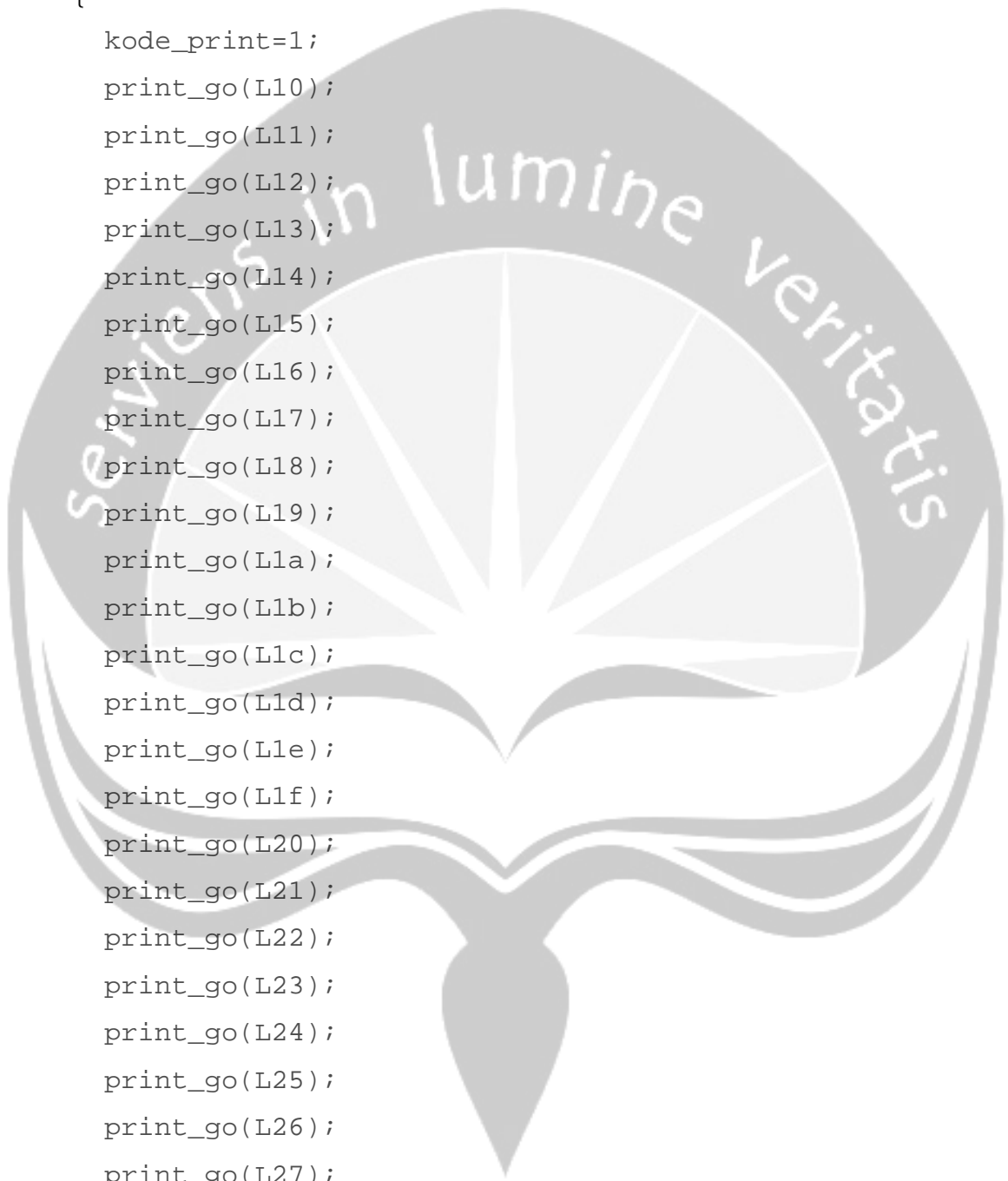
```
else if(alamat==0x8c)
{L1c=sementara;}
else if(alamat==0x8d)
{L1d=sementara;}
else if(alamat==0x8e)
{L1e=sementara;}
else if(alamat==0x8f)
{L1f=sementara;}
else if(alamat==0xc0)
{L20=sementara;}
else if(alamat==0xc1)
{L21=sementara;}
else if(alamat==0xc2)
{L22=sementara;}
else if(alamat==0xc3)
{L23=sementara;}
else if(alamat==0xc4)
{L24=sementara;}
else if(alamat==0xc5)
{L25=sementara;}
else if(alamat==0xc6)
{L26=sementara;}
else if(alamat==0xc7)
{L27=sementara;}
else if(alamat==0xc8)
{L28=sementara;}
else if(alamat==0xc9)
{L29=sementara;}
else if(alamat==0xca)
{L2a=sementara;}
else if(alamat==0xcb)
```



```
{L2b=sementara;}
else if(alamat==0xcc)
{L2c=sementara;}
else if(alamat==0xcd)
{L2d=sementara;}
else if(alamat==0xce)
{L2e=sementara;}
else if(alamat==0xcf)
{L2f=sementara;}
else{;}
}
//-----
//program tulis di lcd
//-----
void nolkan()
{
sementara=0x20;
L10=sementara;
L11=sementara;
L12=sementara;
L13=sementara;
L14=sementara;
L15=sementara;
L16=sementara;
L17=sementara;
L18=sementara;
L19=sementara;
L1a=sementara;
L1b=sementara;
L1c=sementara;
L1d=sementara;
```

```
L1e=sementara;  
L1f=sementara;  
L20=sementara;  
L21=sementara;  
L22=sementara;  
L23=sementara;  
L24=sementara;  
L25=sementara;  
L26=sementara;  
L27=sementara;  
L28=sementara;  
L29=sementara;  
L2a=sementara;  
L2b=sementara;  
L2c=sementara;  
L2d=sementara;  
L2e=sementara;  
L2f=sementara;  
}  
void kosongkan()  
{  
    unsigned char i;  
    tulis_inst_LCD(0x80);  
    for (i=0;i<16;i++)  
        {tulis_data_LCD(kosong[i]);}  
    //  
    tulis_inst_LCD(0xc0);  
    for (i=0;i<16;i++)  
        {tulis_data_LCD(kosong[i]);}  
    alamat=0x80;  
}
```

```
//  
void print()  
{  
    kode_print=1;  
    print_go(L10);  
    print_go(L11);  
    print_go(L12);  
    print_go(L13);  
    print_go(L14);  
    print_go(L15);  
    print_go(L16);  
    print_go(L17);  
    print_go(L18);  
    print_go(L19);  
    print_go(L1a);  
    print_go(L1b);  
    print_go(L1c);  
    print_go(L1d);  
    print_go(L1e);  
    print_go(L1f);  
    print_go(L20);  
    print_go(L21);  
    print_go(L22);  
    print_go(L23);  
    print_go(L24);  
    print_go(L25);  
    print_go(L26);  
    print_go(L27);  
    print_go(L28);  
    print_go(L29);  
    print_go(L2a);  
}
```



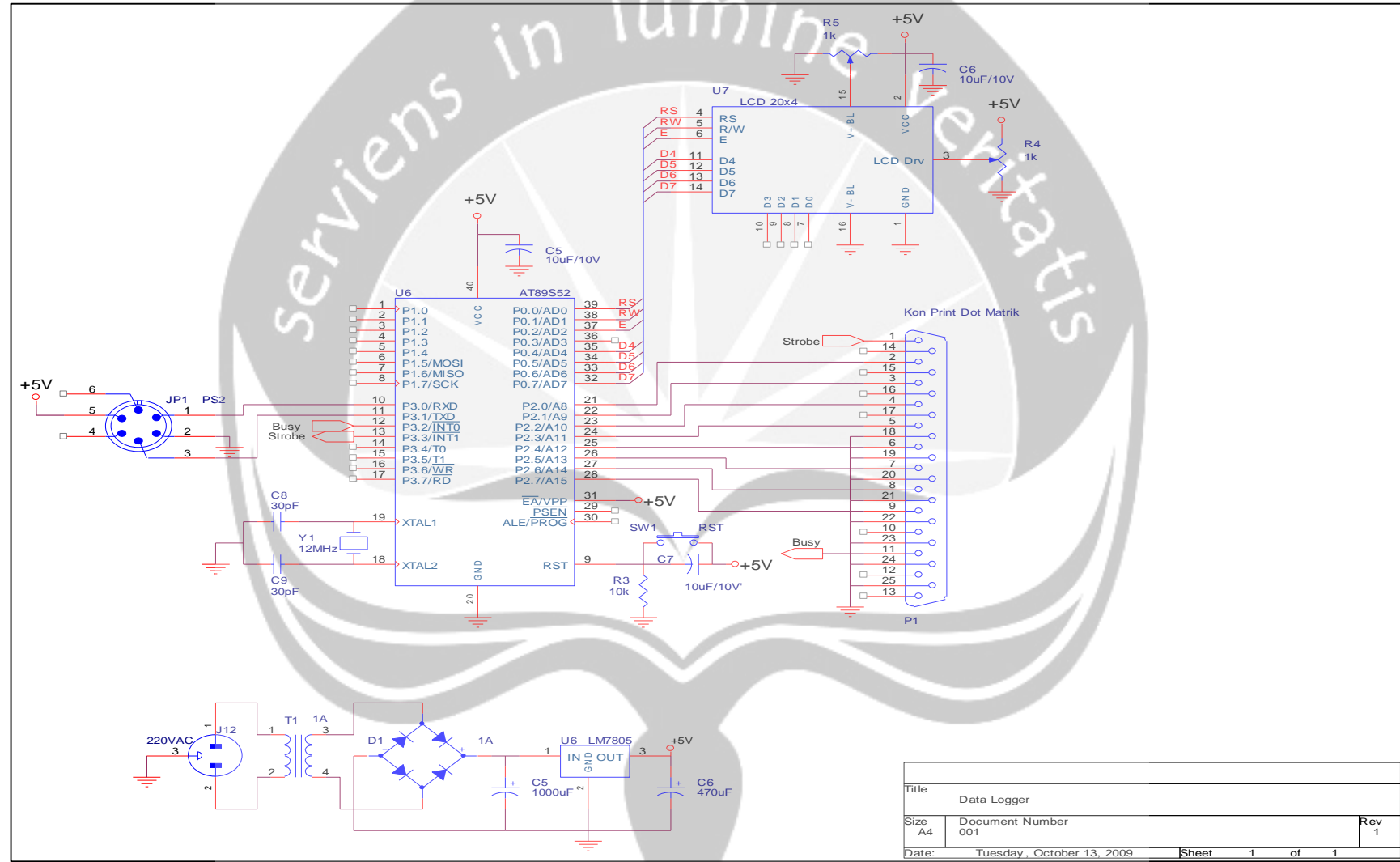
```
print_go(L2b);
print_go(L2c);
print_go(L2d);
print_go(L2e);
print_go(L2f);
space();
}
//
void tampil_dilayar()
{
    if(data_keyboard==0x12)
    {kode_shift=0;}
    else if(data_keyboard==0x59)
    {kode_shift=0;}
    else if(data_keyboard==0x58)
    {kode_shift=!kode_shift;}
    else if(data_keyboard==0x5a)
    {print();alamat=0x80;nolkan();kosongkan();}
    else if(data_keyboard==0x66)
    {
        alamat=alamat-1;
        if(alamat<0x80)
        {alamat=0x80;}
        else if(alamat>0x8f&alamat<0xc0)
        {alamat=0x8f;}
        else{;}
        tulis_inst_LCD(alamat);
        tulis_data_LCD(key_kecil[data_keyboard]);
    }
    else
    {
```

```
tulis_inst_LCD(alamat);
if(kode_shift==0)
{
    tulis_data_LCD(key_kecil[data_keyboard]);
    sementara=key_kecil[data_keyboard];
    simpan();
}
else
{
    tulis_data_LCD(key_besar[data_keyboard]);
    sementara=key_besar[data_keyboard];
    simpan();
}
alamat=alamat+1;
if(alamat>0xd0)
{alamat=0xd0;}
else if(alamat>0x8f&alamat<0xc0)
{alamat=0xc0;}
else{;}
}
}
//-----
//program keyboard
//-----
void baca_keyboard()
{
    unsigned char i,rd_bit;
    data_keyboard=0x00;
    while(CLK==0){;}
    for(i=0;i<8;i++)
    {
```

```
while(CLK==1){;}
rd_bit=(DAT<<7)&0x80;
data_keyboard = data_keyboard >> 1;
data_keyboard = data_keyboard | rd_bit;
while(CLK==0){;}
}
while(CLK==1){;}
while(CLK==0){;}
while(CLK==1){;}
while(DAT==0){;}
while(CLK==0){;}
}
//
void keyboard()
{
while(CLK==1){;}
while(DAT==1){;}
baca_keyboard();
if(data_keyboard==0xe0)
{kode_del=1;}
else if(data_keyboard==0xf0)
{
while(CLK==1){;}
while(DAT==1){;}
baca_keyboard();
tampil_dilayar();
if(data_keyboard==0x75&kode_del==1)
{
print_go(0x0a);
kosongkan();
nolkan();
}
```

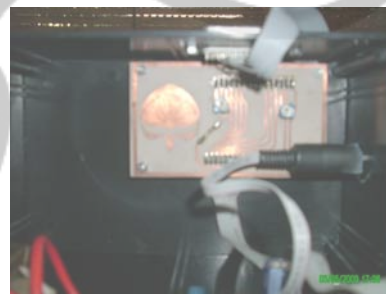
```
        kode_del=0;
    }
    else if(data_keyboard==0x71&kode_del==1)
    {
        kosongkan();
        nolkan();
        kode_del=0;
    }
    else{;}
}
else if(data_keyboard==0x12)
{kode_shift=1;}
else if(data_keyboard==0x59)
{kode_shift=1;}
else{;}
}
//-----
//program utama
//-----
void main()
{
    nolkan();
    data_keyboard=0;alamat=0x80;kode_del=0;
    LCD_port=0xff;kode_shift=0;
    init_LCD();
    while(1)
    {
        keyboard();
    }
}
```





Title		
Data Logger		
Size	Document Number	Rev
A4	001	1
Date:	Tuesday, October 13, 2009	Sheet 1 of 1

Lampiran 4 : Foto Alat



Lampiran 3 : Foto Alat Pendukung



Printer EPSON LX-300



Keyboard PS2

---

## Features

- Compatible with MCS-51® Products
- 8K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag

## Description

The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with in-system programmable Flash on a monolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.



---

## 8-bit Microcontroller with 8K Bytes In-System Programmable Flash

---

**AT89S52**

**Preliminary**

Rev. 1919A-07/01



# Pin Configurations

## PDIP

(T2) P1.0	1	40	VCC
(T2 EX) P1.1	2	39	P0.0 (AD0)
P1.2	3	38	P0.1 (AD1)
P1.3	4	37	P0.2 (AD2)
P1.4	5	36	P0.3 (AD3)
(MOSI) P1.5	6	35	P0.4 (AD4)
(MISO) P1.6	7	34	P0.5 (AD5)
(SCK) P1.7	8	33	P0.6 (AD6)
RST	9	32	P0.7 (AD7)
(RXD) P3.0	10	31	$\overline{EA}/VPP$
(TXD) P3.1	11	30	ALE/PROG
( $\overline{INT0}$ ) P3.2	12	29	PSEN
( $\overline{INT1}$ ) P3.3	13	28	P2.7 (A15)
(T0) P3.4	14	27	P2.6 (A14)
(T1) P3.5	15	26	P2.5 (A13)
( $\overline{WR}$ ) P3.6	16	25	P2.4 (A12)
( $\overline{RD}$ ) P3.7	17	24	P2.3 (A11)
XTAL2	18	23	P2.2 (A10)
XTAL1	19	22	P2.1 (A9)
GND	20	21	P2.0 (A8)

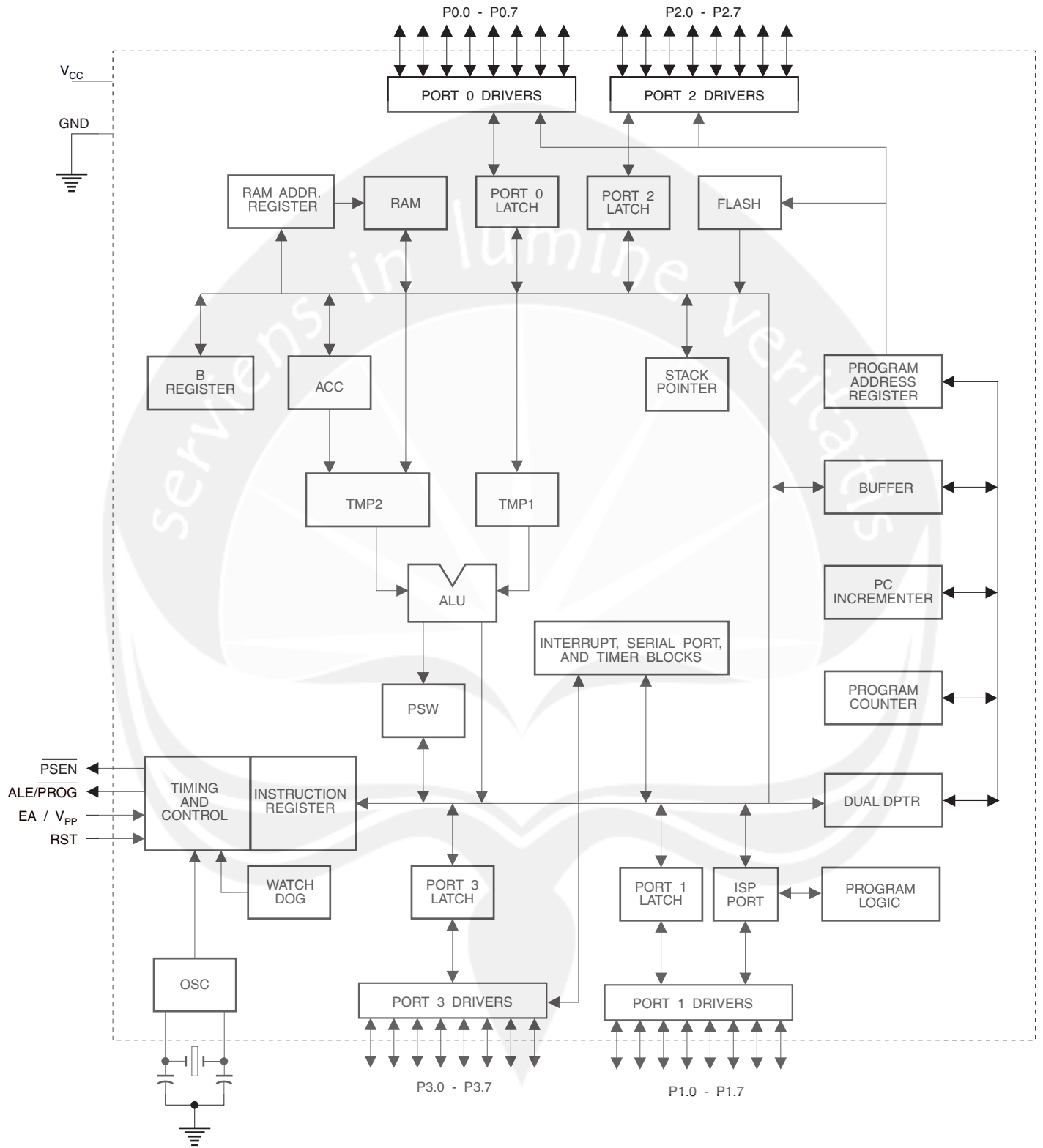
## PLCC

(MOSI) P1.5	7	39	P0.4 (AD4)
(MISO) P1.6	8	38	P0.5 (AD5)
(SCK) P1.7	9	37	P0.6 (AD6)
RST	10	36	P0.7 (AD7)
(RXD) P3.0	11	35	$\overline{EA}/VPP$
NC	12	34	NC
(TXD) P3.1	13	33	ALE/PROG
( $\overline{INT0}$ ) P3.2	14	32	PSEN
( $\overline{INT1}$ ) P3.3	15	31	P2.7 (A15)
(T0) P3.4	16	30	P2.6 (A14)
(T1) P3.5	17	29	P2.5 (A13)
( $\overline{WR}$ ) P3.6	18	28	P2.4 (A12)
( $\overline{RD}$ ) P3.7	19	27	P2.3 (A11)
XTAL2	20	26	P2.2 (A10)
XTAL1	21	25	P2.1 (A9)
GND	22	24	P2.0 (A8)
NC	23	23	NC
(A8) P2.0	24	22	P2.0 (A8)
(A9) P2.1	25	21	P2.1 (A9)
(A10) P2.2	26	20	P2.2 (A10)
(A11) P2.3	27	19	P2.3 (A11)
(A12) P2.4	28	18	P2.4 (A12)
(A13) P2.5	29	17	P2.5 (A13)
(A14) P2.6	30	16	P2.6 (A14)
(A15) P2.7	31	15	P2.7 (A15)
P0.0 (AD0)	32	14	P0.0 (AD0)
P0.1 (AD1)	33	13	P0.1 (AD1)
P0.2 (AD2)	34	12	P0.2 (AD2)
P0.3 (AD3)	35	11	P0.3 (AD3)
P0.4 (AD4)	36	10	P0.4 (AD4)
P0.5 (AD5)	37	9	P0.5 (AD5)
P0.6 (AD6)	38	8	P0.6 (AD6)
P0.7 (AD7)	39	7	P0.7 (AD7)
VCC	40	6	VCC
NC	41	5	NC
(T2 EX) P1.1	42	4	(T2 EX) P1.1
P1.2	43	3	P1.2
P1.3	44	2	P1.3
P1.4	45	1	P1.4

## TQFP

(MOSI) P1.5	1	33	P0.4 (AD4)
(MISO) P1.6	2	32	P0.5 (AD5)
(SCK) P1.7	3	31	P0.6 (AD6)
RST	4	30	P0.7 (AD7)
(RXD) P3.0	5	29	$\overline{EA}/VPP$
NC	6	28	NC
(TXD) P3.1	7	27	ALE/PROG
( $\overline{INT0}$ ) P3.2	8	26	PSEN
( $\overline{INT1}$ ) P3.3	9	25	P2.7 (A15)
(T0) P3.4	10	24	P2.6 (A14)
(T1) P3.5	11	23	P2.5 (A13)
( $\overline{WR}$ ) P3.6	12	22	P2.4 (A12)
( $\overline{RD}$ ) P3.7	13	21	P2.3 (A11)
XTAL2	14	20	P2.2 (A10)
XTAL1	15	19	P2.1 (A9)
GND	16	18	P2.0 (A8)
GND	17	17	GND
(A8) P2.0	18	16	P2.0 (A8)
(A9) P2.1	19	15	P2.1 (A9)
(A10) P2.2	20	14	P2.2 (A10)
(A11) P2.3	21	13	P2.3 (A11)
(A12) P2.4	22	12	P2.4 (A12)
(A13) P2.5	23	11	P2.5 (A13)
(A14) P2.6	24	10	P2.6 (A14)
(A15) P2.7	25	9	P2.7 (A15)
P0.0 (AD0)	26	8	P0.0 (AD0)
P0.1 (AD1)	27	7	P0.1 (AD1)
P0.2 (AD2)	28	6	P0.2 (AD2)
P0.3 (AD3)	29	5	P0.3 (AD3)
P0.4 (AD4)	30	4	P0.4 (AD4)
P0.5 (AD5)	31	3	P0.5 (AD5)
P0.6 (AD6)	32	2	P0.6 (AD6)
P0.7 (AD7)	33	1	P0.7 (AD7)
VCC	34	0	VCC
NC	35	0	NC
(T2 EX) P1.1	36	0	(T2 EX) P1.1
P1.2	37	0	P1.2
P1.3	38	0	P1.3
P1.4	39	0	P1.4

Block Diagram



## Pin Description

### VCC

Supply voltage.

### GND

Ground.

### Port 0

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

Port 0 can also be configured to be the multiplexed low-order 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 bidirectional 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.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

### Port 2

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

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to

external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

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

### Port 3

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

Port 3 also serves the functions of various special features of the AT89S52, 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. This pin drives High for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

### ALE/PROG

Address Latch Enable (ALE) 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 (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.

## PSEN

Program Store Enable ( $\overline{\text{PSEN}}$ ) is the read strobe to external program memory.

When the AT89S52 is executing code from external program memory,  $\overline{\text{PSEN}}$  is activated twice each machine cycle, except that two  $\overline{\text{PSEN}}$  activations are skipped during each access to external data memory.

## $\overline{\text{EA}}/\text{VPP}$

External Access Enable.  $\overline{\text{EA}}$  must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH.

Note, however, that if lock bit 1 is programmed,  $\overline{\text{EA}}$  will be internally latched on reset.

$\overline{\text{EA}}$  should be strapped to  $V_{\text{CC}}$  for internal program executions.

This pin also receives the 12-volt programming enable voltage ( $V_{\text{PP}}$ ) during Flash programming.

## XTAL1

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

## XTAL2

Output from the inverting oscillator amplifier.

**Table 1.** AT89S52 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							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							0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX	0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XX00XX0	8FH
80H	P0 11111111	SP 00001111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	PCON 0XXX0000	87H



## Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User 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 3) 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.

**Interrupt Registers:** The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.

**Table 2.** T2CON – Timer/Counter 2 Control Register

T2CON Address = 0C8H						Reset Value = 0000 0000B		
Bit Addressable								
Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	$C/\overline{T2}$	$CP/\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	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.
EXF2	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).
RCLK	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 overflow to be used for the receive clock.
TCLK	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.
EXEN2	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.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
$C/\overline{T2}$	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).
$CP/\overline{RL2}$	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.

**Table 3a. AUXR: Auxiliary Register**

AUXR	Address = 8EH	Reset Value = XXX0XX0B																
	Not Bit Addressable																	
	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">WDIDLE</td> <td style="width: 20px;">DISRTO</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">DISALE</td> </tr> <tr> <td>Bit</td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> </table>	–	–	–	WDIDLE	DISRTO	–	–	DISALE	Bit	7	6	5	4	3	2	1	0
–	–	–	WDIDLE	DISRTO	–	–	DISALE											
Bit	7	6	5	4	3	2	1	0										
–	Reserved for future expansion																	
DISALE	Disable/Enable ALE																	
	DISALE	Operating Mode																
	0	ALE is emitted at a constant rate of 1/6 the oscillator frequency																
	1	ALE is active only during a MOVX or MOVC instruction																
DISRTO	Disable/Enable Reset out																	
	DISRTO																	
	0	Reset pin is driven High after WDT times out																
	1	Reset pin is input only																
WDIDLE	Disable/Enable WDT in IDLE mode																	
	WDIDLE																	
	0	WDT continues to count in IDLE mode																
	1	WDT halts counting in IDLE mode																

**Dual Data Pointer Registers:** To facilitate accessing both internal 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 AUXR1 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 rest under software control and is not affected by reset.

**Table 3b. AUXR1: Auxiliary Register 1**

AUXR1	Address = A2H	Reset Value = XXXXXXX0B																	
	Not Bit Addressable																		
	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">–</td> <td style="width: 20px;">DPS</td> </tr> <tr> <td>Bit</td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> </table>	–	–	–	–	–	–	–	–	DPS	Bit	7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–	DPS											
Bit	7	6	5	4	3	2	1	0											
–	Reserved for future expansion																		
DPS	Data Pointer Register Select																		
	DPS																		
	0	Selects DPTR Registers DP0L, DP0H																	
	1	Selects DPTR Registers DP1L, DP1H																	

## Memory Organization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

### Program Memory

If the  $\overline{EA}$  pin is connected to GND, all program fetches are directed to external memory.

On the AT89S52, if  $\overline{EA}$  is connected to  $V_{CC}$ , program fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

### Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions which use direct addressing access of the SFR space.

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

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

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

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

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



## Watchdog Timer (One-time Enabled with Reset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 13-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

### Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 13-bit counter overflows when it reaches 8191 (1FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 8191 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is  $96 \times TOSC$ , where  $TOSC = 1/FOSC$ . To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

### WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode.

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

### UART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, refer to the ATMEL Web site (<http://www.atmel.com>). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

### Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers' operation, refer to the ATMEL Web site (<http://www.atmel.com>). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

### 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 3. 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.

**Table 3.** Timer 2 Operating Modes

RCLK +TCLK	CP/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)

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

### 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.

Figure 5. Timer in Capture Mode

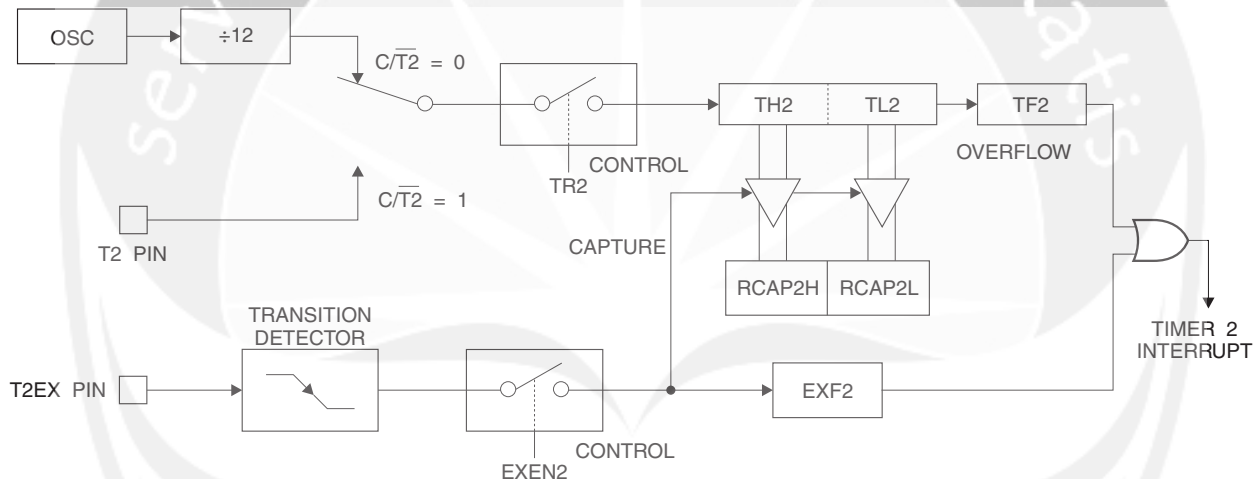


Figure 6 shows Timer 2 automatically counting up when DCEN=0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture Mode RCAP2H and RCAP2L are preset by software. If EXEN2 = 1, 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 6. In this mode, the T2EX pin controls

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 5.

### Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 4). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

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.

Figure 6. Timer 2 Auto Reload Mode (DCEN = 0)

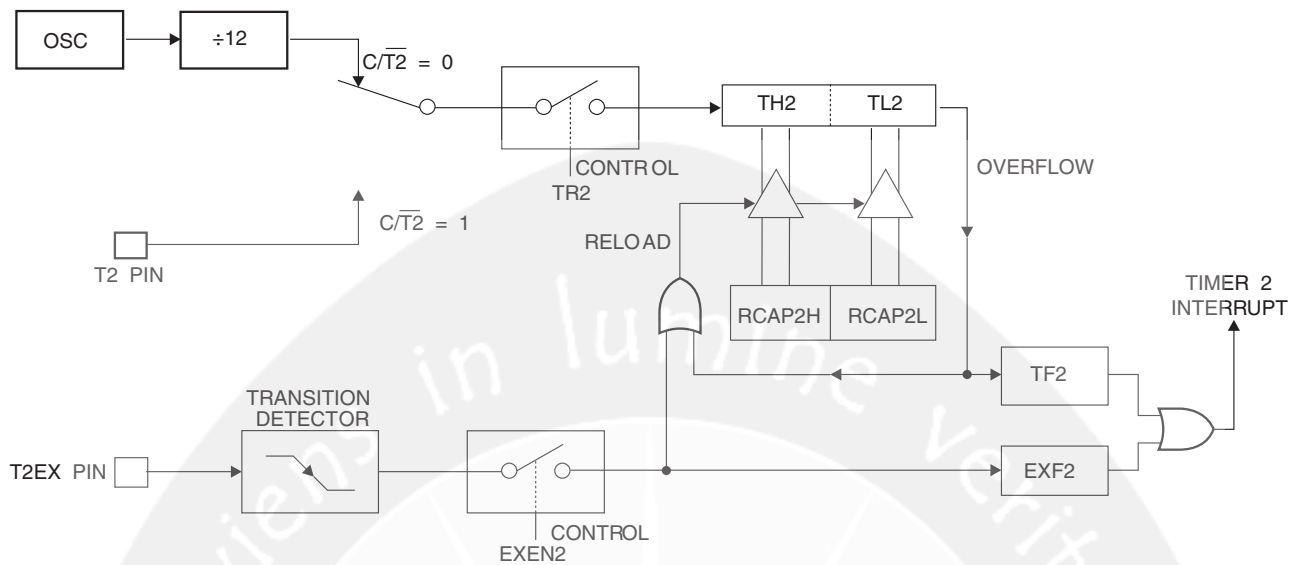
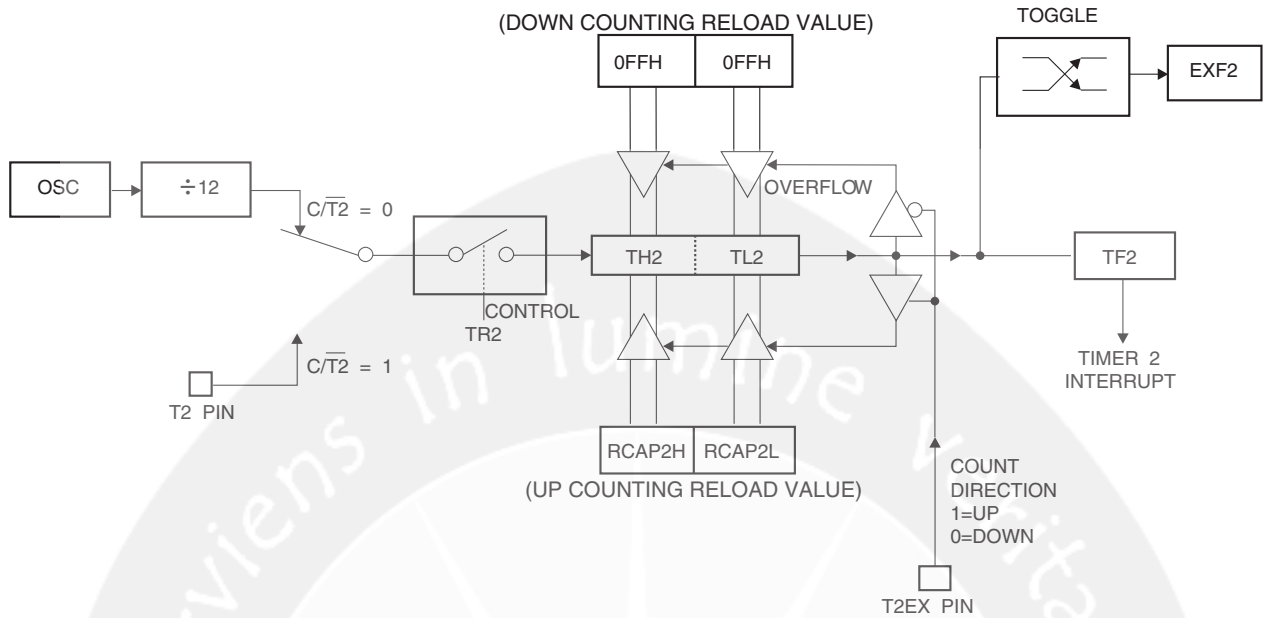


Table 4. T2MOD – Timer 2 Mode Control Register

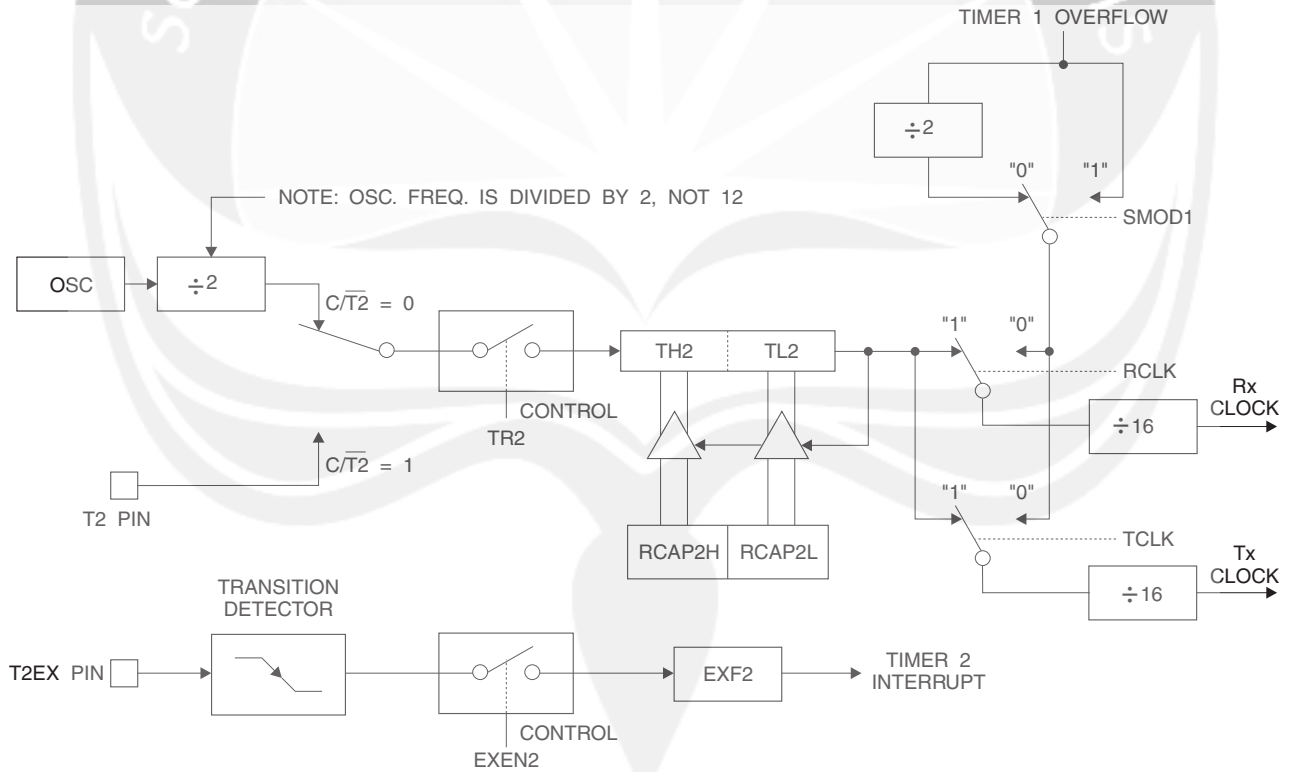
T2MOD Address = 0C9H							Reset Value = XXXX XX0B	
Not Bit Addressable								
Bit	7	6	5	4	3	2	T2OE	DCEN
	-	-	-	-	-	-	1	0

Symbol	Function
-	Not implemented, reserved for future
T2OE	Timer 2 Output Enable bit
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter

**Figure 7. Timer 2 Auto Reload Mode (DCEN = 1)**



**Figure 8. Timer 2 in Baud Rate Generator Mode**



## Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 8.

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

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

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

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation ( $CP/\overline{T2} = 0$ ). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it

increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

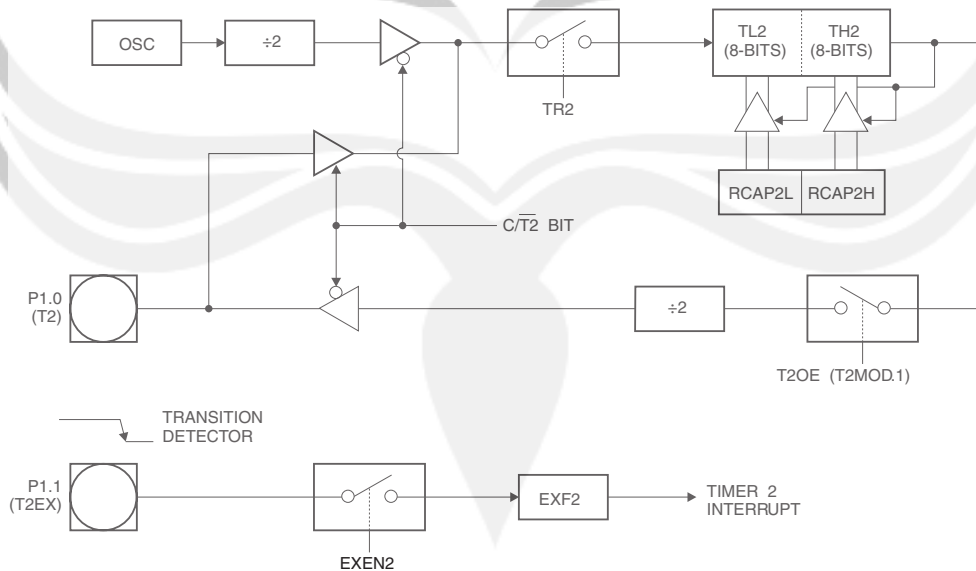
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - \text{RCAP2H}, \text{RCAP2L}]}$$

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

Timer 2 as a baud rate generator is shown in Figure 8. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (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.

Figure 9. Timer 2 in Clock-Out Mode





## Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 9. 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.

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

In the clock-out mode, Timer 2 roll-overs 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.

## Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ( $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$ ), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 10.

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

Note that Table 5 shows that bit position IE.6 is unimplemented. In the AT89S52, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

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

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

**Table 5.** Interrupt Enable (IE) Register

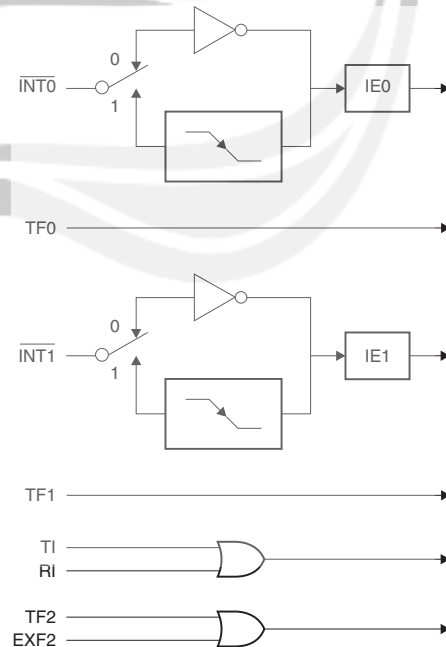
(MSB)								(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0	
Enable Bit = 1 enables the interrupt.								
Enable Bit = 0 disables the interrupt.								

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	Serial Port 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.

**Figure 10.** Interrupt Sources



## Oscillator Characteristics

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

## Idle Mode

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

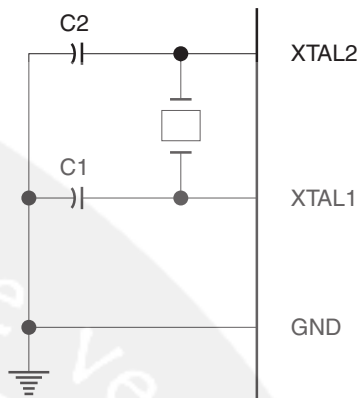
Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is 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.

## Power-down Mode

In the Power-down mode, the oscillator is stopped, and the instruction that invokes Power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power-down mode is terminated. Exit from Power-down mode can be initiated either by a hardware reset or by an enabled external interrupt. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before  $V_{CC}$  is restored to its normal operating level and must be held

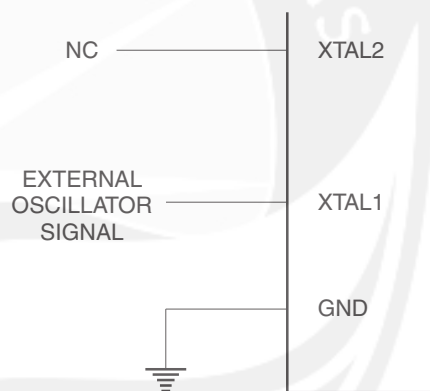
active long enough to allow the oscillator to restart and stabilize.

**Figure 11.** Oscillator Connections



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

**Figure 12.** External Clock Drive Configuration



**Table 6.** Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

## Program Memory Lock Bits

The AT89S52 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.

**Table 7.** Lock Bit Protection Modes

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

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.

## Programming the Flash – Parallel Mode

The AT89S52 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S52 code memory array is programmed byte-by-byte.

**Programming Algorithm:** Before programming the AT89S52, the address, data, and control signals should be set up according to the Flash programming mode table and Figures 13 and 14. To program the AT89S52, take the following steps:

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise  $\overline{EA}/V_{PP}$  to 12V.
5. Pulse  $ALE/\overline{PROG}$  once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 50  $\mu$ s.

Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

**Data Polling:** The AT89S52 features  $\overline{Data}$  Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true data is 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/Busy:** The progress of byte programming can also be monitored by the RDY/ $\overline{BSY}$  output signal. P3.0 is pulled low after ALE goes high during programming to indicate  $\overline{BUSY}$ . P3.0 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 data can be read back via the address and data lines for verification. The status of the individual lock bits can be verified directly by reading them back.

**Reading the Signature Bytes:** The signature bytes are read by the same procedure as a normal verification of locations 000H, 100H, and 200H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

- (000H) = 1EH indicates manufactured by Atmel
- (100H) = 52H indicates 89S52
- (200H) = 06H

**Chip Erase:** In the parallel programming mode, a chip erase operation is initiated by using the proper combination of control signals and by pulsing  $ALE/\overline{PROG}$  low for a duration of 200 ns - 500 ns.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 500 ms.

During chip erase, a serial read from any address location will return 00H at the data output.

## Programming the Flash – Serial Mode

The Code memory array can be programmed using the serial ISP interface while RST is pulled to  $V_{CC}$ . The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before other operations can be executed. Before a reprogramming sequence can occur, a Chip Erase operation is required.

The Chip Erase operation turns the content of every memory location in the Code array into FFH.

Either an external system clock can be 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 1/16 of the crystal frequency. With a 33 MHz oscillator clock, the maximum SCK frequency is 2 MHz.

## Serial Programming Algorithm

To program and verify the AT89S52 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 33 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 16.
3. The Code array is programmed one byte at a time by supplying the address and data together with the

appropriate Write instruction. The write cycle is self-timed and typically takes less than 1 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 device operation.

Power-off sequence (if needed):

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

Set RST to "L".

Turn V<sub>CC</sub> power off.

**Data Polling:** The  $\overline{\text{Data}}$  Polling feature is also available in the serial mode. In this mode, during a write cycle an attempted read of the last byte written will result in the complement of the MSB of the serial output byte on MISO.

## Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 10.

## Programming Interface – Parallel Mode

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

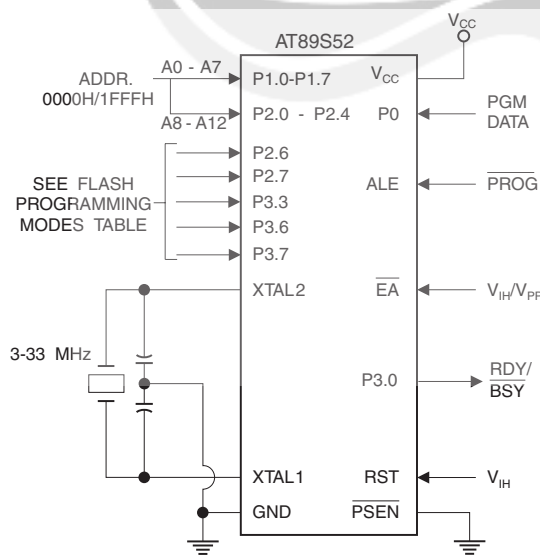
All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

**Table 8.** Flash Programming Modes

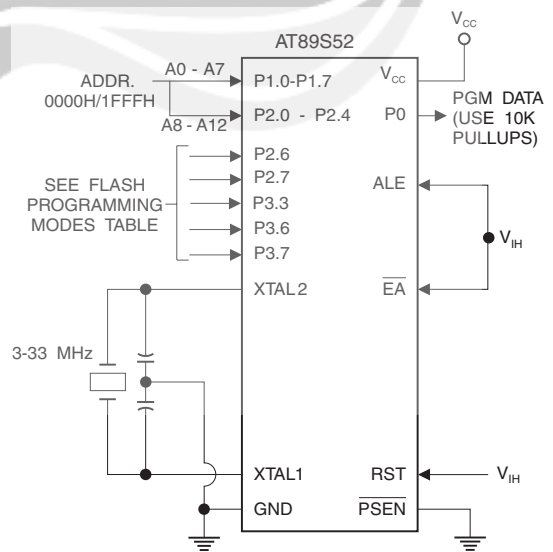
Mode	V <sub>CC</sub>	RST	PSEN	ALE/ PROG	EA/ V <sub>PP</sub>	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.4-0	P1.7-0
												Address	
Write Code Data	5V	H	L		12V	L	H	H	H	D <sub>IN</sub>	A12-8	A7-0	
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D <sub>OUT</sub>	A12-8	A7-0
Write Lock Bit 1	5V	H	L		12V	H	H	H	H	X	X	X	
Write Lock Bit 2	5V	H	L		12V	H	H	H	L	L	X	X	
Write Lock Bit 3	5V	H	L		12V	H	L	H	L	L	X	X	
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L		12V	H	L	H	L	L	X	X	
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	X 0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	52H	X 0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	06H	X 0010	00H

- Notes:
1. Each PROG pulse is 200 ns - 500 ns for Chip Erase.
  2. Each PROG pulse is 200 ns - 500 ns for Write Code Data.
  3. Each PROG pulse is 200 ns - 500 ns for Write Lock Bits.
  4. RDY/BSY signal is output on P3.0 during programming.
  5. X = don't care.

**Figure 13.** Programming the Flash Memory (Parallel Mode)



**Figure 14.** Verifying the Flash Memory (Parallel Mode)

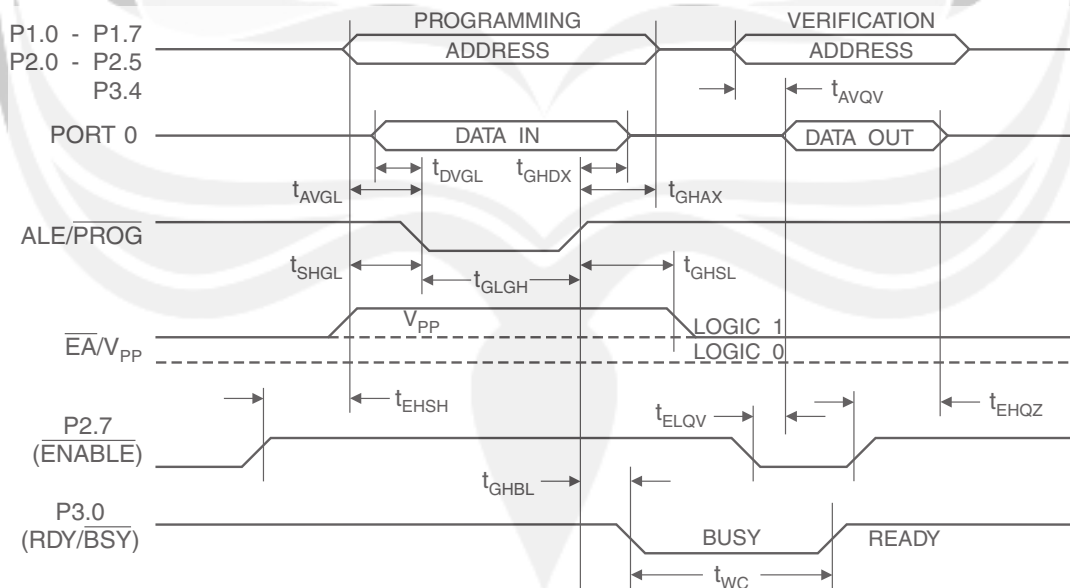


### Flash Programming and Verification Characteristics (Parallel Mode)

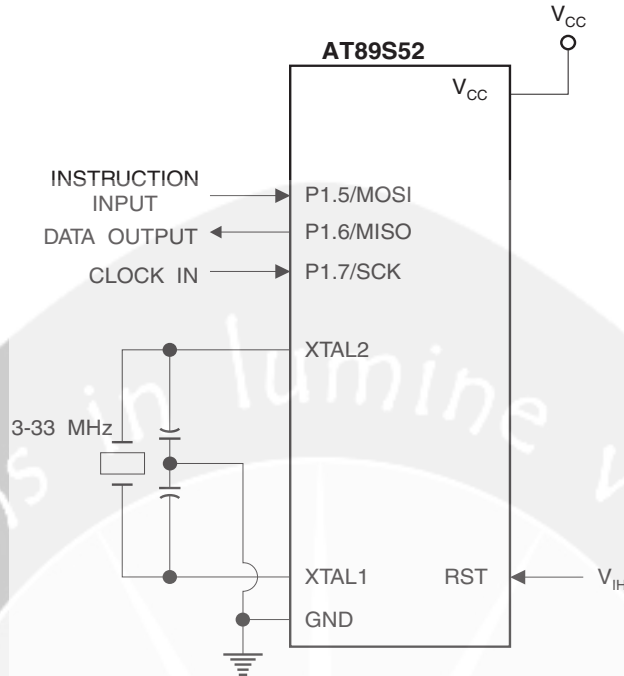
$T_A = 20^\circ\text{C}$  to  $30^\circ\text{C}$ ,  $V_{CC} = 4.5$  to  $5.5\text{V}$

Symbol	Parameter	Min	Max	Units
$V_{PP}$	Programming Supply Voltage	11.5	12.5	V
$I_{PP}$	Programming Supply Current		10	mA
$I_{CC}$	$V_{CC}$ Supply Current		30	mA
$1/t_{CLCL}$	Oscillator Frequency	3	33	MHz
$t_{AVGL}$	Address Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
$t_{GHAX}$	Address Hold After $\overline{\text{PROG}}$	$48t_{CLCL}$		
$t_{DVGL}$	Data Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
$t_{GHDX}$	Data Hold After $\overline{\text{PROG}}$	$48t_{CLCL}$		
$t_{EHSH}$	P2.7 ( $\overline{\text{ENABLE}}$ ) High to $V_{PP}$	$48t_{CLCL}$		
$t_{SHGL}$	$V_{PP}$ Setup to $\overline{\text{PROG}}$ Low	10		$\mu\text{s}$
$t_{GHSL}$	$V_{PP}$ Hold After $\overline{\text{PROG}}$	10		$\mu\text{s}$
$t_{GLGH}$	$\overline{\text{PROG}}$ Width	0.2	1	$\mu\text{s}$
$t_{AVQV}$	Address to Data Valid		$48t_{CLCL}$	
$t_{ELQV}$	$\overline{\text{ENABLE}}$ Low to Data Valid		$48t_{CLCL}$	
$t_{EHQZ}$	Data Float After $\overline{\text{ENABLE}}$	0	$48t_{CLCL}$	
$t_{GHBL}$	$\overline{\text{PROG}}$ High to $\overline{\text{BUSY}}$ Low		1.0	$\mu\text{s}$
$t_{WC}$	Byte Write Cycle Time		50	$\mu\text{s}$

Figure 15. Flash Programming and Verification Waveforms – Parallel Mode

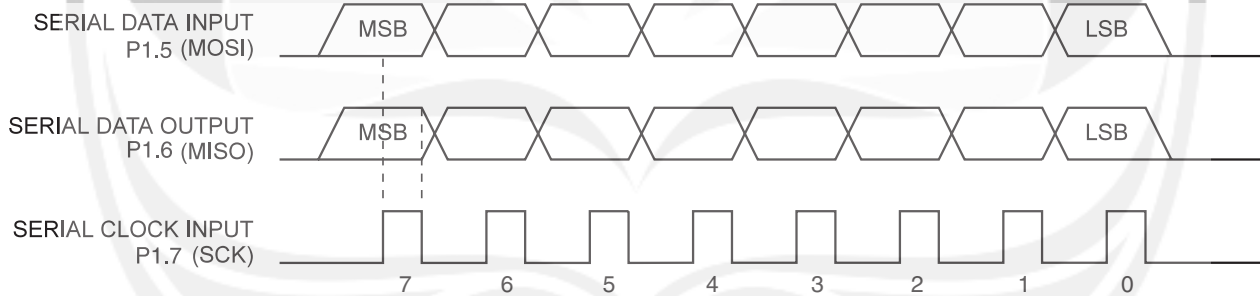


**Figure 16.** Flash Memory Serial Downloading



## Flash Programming and Verification Waveforms – Serial Mode

**Figure 17.** Serial Programming Waveforms





**Table 9. Serial Programming Instruction Set**

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output)	Enable Serial Programming while RST is high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
Read Program Memory (Byte Mode)	0010 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Read data from Program memory in the byte mode
Write Program Memory (Byte Mode)	0100 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Write data to Program memory in the byte mode
Write Lock Bits <sup>(2)</sup>	1010 1100	1110 00 B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (2).
Read Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a '1')
Read Signature Bytes <sup>(1)</sup>	0010 1000	xxx A5 A4 A3 A2 A1 A0	xxx xxxx	Signature Byte	Read Signature Byte
Read Program Memory (Page Mode)	0011 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Write Program Memory (Page Mode)	0101 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

Notes: 1. The signature bytes are not readable in Lock Bit Modes 3 and 4.

2. B1 = 0, B2 = 0 ---> Mode 1, no lock protection  
 B1 = 0, B2 = 1 ---> Mode 2, lock bit 1 activated  
 B1 = 1, B2 = 0 ---> Mode 3, lock bit 2 activated  
 B1 = 1, B2 = 1 ---> Mode 4, lock bit 3 activated

Each of the lock bits needs to be activated sequentially before Mode 4 can be executed.

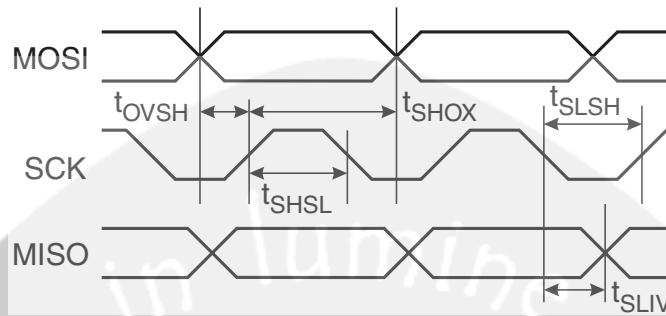
After Reset signal is high, SCK should be low for at least 64 system clocks before it goes high to clock in the enable data bytes. No pulsing of Reset signal is necessary. SCK should be no faster than 1/16 of the system clock at XTAL1.

For Page Read/Write, the data always starts from byte 0 to 255. After the command byte and upper address byte are latched, each byte thereafter is treated as data until all 256 bytes are shifted in/out. Then the next instruction will be ready to be decoded.



## Serial Programming Characteristics

**Figure 18.** Serial Programming Timing



**Table 10.** Serial Programming Characteristics,  $T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ ,  $V_{CC} = 4.0 - 5.5\text{V}$  (Unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{\text{CLCL}}$	Oscillator Frequency	0		33	MHz
$t_{\text{CLCL}}$	Oscillator Period	30			ns
$t_{\text{SHSL}}$	SCK Pulse Width High	$2 t_{\text{CLCL}}$			ns
$t_{\text{SLSH}}$	SCK Pulse Width Low	$2 t_{\text{CLCL}}$			ns
$t_{\text{OVSH}}$	MOSI Setup to SCK High	$t_{\text{CLCL}}$			ns
$t_{\text{SHOX}}$	MOSI Hold after SCK High	$2 t_{\text{CLCL}}$			ns
$t_{\text{SLIV}}$	SCK Low to MISO Valid	10	16	32	ns
$t_{\text{ERASE}}$	Chip Erase Instruction Cycle Time			500	ms
$t_{\text{SWC}}$	Serial Byte Write Cycle Time			$64 t_{\text{CLCL}} + 400$	$\mu\text{s}$

## Absolute Maximum Ratings\*

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

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

## DC Characteristics

The values shown in this table are valid for  $T_A = -40^\circ\text{C}$  to  $85^\circ\text{C}$  and  $V_{CC} = 4.0\text{V}$  to  $5.5\text{V}$ , 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 High Voltage	(Except XTAL1, RST)	$0.2 V_{CC} + 0.9$	$V_{CC} + 0.5$	V
$V_{IH1}$	Input High Voltage	(XTAL1, RST)	$0.7 V_{CC}$	$V_{CC} + 0.5$	V
$V_{OL}$	Output Low Voltage <sup>(1)</sup> (Ports 1,2,3)	$I_{OL} = 1.6 \text{ mA}$		0.45	V
$V_{OL1}$	Output Low Voltage <sup>(1)</sup> (Port 0, ALE, $\overline{PSEN}$ )	$I_{OL} = 3.2 \text{ mA}$		0.45	V
$V_{OH}$	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$ )	$I_{OH} = -60 \mu\text{A}, V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -25 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -10 \mu\text{A}$	$0.9 V_{CC}$		V
$V_{OH1}$	Output High Voltage (Port 0 in External Bus Mode)	$I_{OH} = -800 \mu\text{A}, V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -300 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -80 \mu\text{A}$	$0.9 V_{CC}$		V
$I_{IL}$	Logical 0 Input Current (Ports 1,2,3)	$V_{IN} = 0.45\text{V}$		-50	$\mu\text{A}$
$I_{TL}$	Logical 1 to 0 Transition Current (Ports 1,2,3)	$V_{IN} = 2\text{V}, V_{CC} = 5\text{V} \pm 10\%$		-650	$\mu\text{A}$
$I_{LI}$	Input Leakage Current (Port 0, $\overline{EA}$ )	$0.45 < V_{IN} < V_{CC}$		$\pm 10$	$\mu\text{A}$
RRST	Reset Pulldown Resistor		10	30	$\text{K}\Omega$
$C_{IO}$	Pin Capacitance	Test Freq. = 1 MHz, $T_A = 25^\circ\text{C}$		10	pF
$I_{CC}$	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode <sup>(1)</sup>	$V_{CC} = 5.5\text{V}$		50	$\mu\text{A}$

Notes: 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.

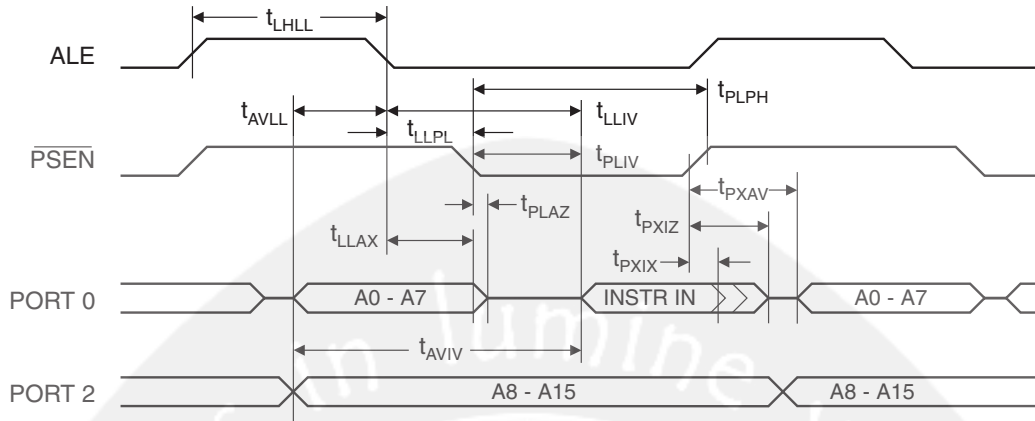
## AC Characteristics

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

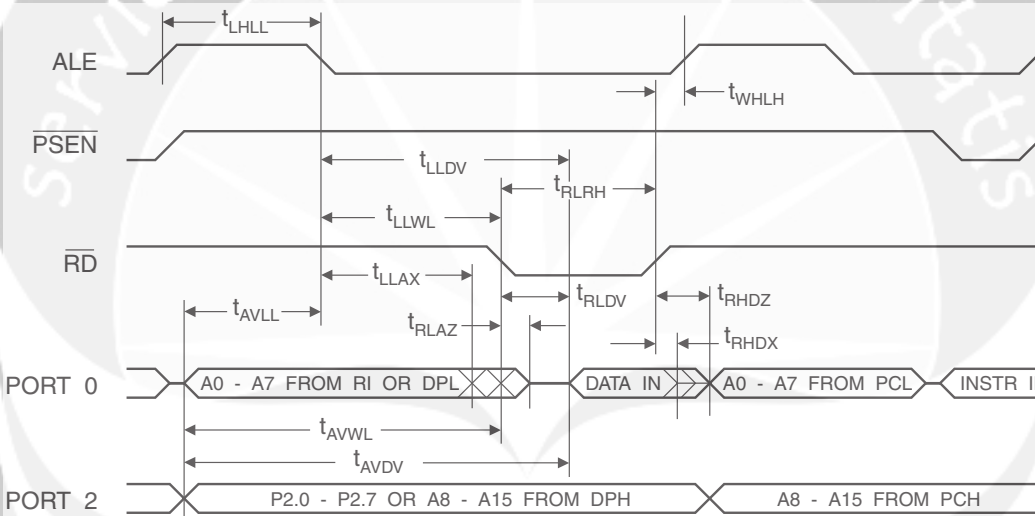
### External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{\text{CLCL}}$	Oscillator Frequency			0	33	MHz
$t_{\text{LHLL}}$	ALE Pulse Width	127		$2t_{\text{CLCL}}-40$		ns
$t_{\text{AVLL}}$	Address Valid to ALE Low	43		$t_{\text{CLCL}}-25$		ns
$t_{\text{LLAX}}$	Address Hold After ALE Low	48		$t_{\text{CLCL}}-25$		ns
$t_{\text{LLIV}}$	ALE Low to Valid Instruction In		233		$4t_{\text{CLCL}}-65$	ns
$t_{\text{LLPL}}$	ALE Low to $\overline{\text{PSEN}}$ Low	43		$t_{\text{CLCL}}-25$		ns
$t_{\text{PLPH}}$	$\overline{\text{PSEN}}$ Pulse Width	205		$3t_{\text{CLCL}}-45$		ns
$t_{\text{PLIV}}$	$\overline{\text{PSEN}}$ Low to Valid Instruction In		145		$3t_{\text{CLCL}}-60$	ns
$t_{\text{PXIX}}$	Input Instruction Hold After $\overline{\text{PSEN}}$	0		0		ns
$t_{\text{PXIZ}}$	Input Instruction Float After $\overline{\text{PSEN}}$		59		$t_{\text{CLCL}}-25$	ns
$t_{\text{PXAV}}$	$\overline{\text{PSEN}}$ to Address Valid	75		$t_{\text{CLCL}}-8$		ns
$t_{\text{AVIV}}$	Address to Valid Instruction In		312		$5t_{\text{CLCL}}-80$	ns
$t_{\text{PLAZ}}$	$\overline{\text{PSEN}}$ Low to Address Float		10		10	ns
$t_{\text{RLRH}}$	$\overline{\text{RD}}$ Pulse Width	400		$6t_{\text{CLCL}}-100$		ns
$t_{\text{WLWH}}$	$\overline{\text{WR}}$ Pulse Width	400		$6t_{\text{CLCL}}-100$		ns
$t_{\text{RLDV}}$	$\overline{\text{RD}}$ Low to Valid Data In		252		$5t_{\text{CLCL}}-90$	ns
$t_{\text{RHDX}}$	Data Hold After $\overline{\text{RD}}$	0		0		ns
$t_{\text{RHDZ}}$	Data Float After $\overline{\text{RD}}$		97		$2t_{\text{CLCL}}-28$	ns
$t_{\text{LLDV}}$	ALE Low to Valid Data In		517		$8t_{\text{CLCL}}-150$	ns
$t_{\text{AVDV}}$	Address to Valid Data In		585		$9t_{\text{CLCL}}-165$	ns
$t_{\text{LLWL}}$	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	200	300	$3t_{\text{CLCL}}-50$	$3t_{\text{CLCL}}+50$	ns
$t_{\text{AVWL}}$	Address to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	203		$4t_{\text{CLCL}}-75$		ns
$t_{\text{QVWX}}$	Data Valid to $\overline{\text{WR}}$ Transition	23		$t_{\text{CLCL}}-30$		ns
$t_{\text{QVWH}}$	Data Valid to $\overline{\text{WR}}$ High	433		$7t_{\text{CLCL}}-130$		ns
$t_{\text{WHQX}}$	Data Hold After $\overline{\text{WR}}$	33		$t_{\text{CLCL}}-25$		ns
$t_{\text{RLAZ}}$	$\overline{\text{RD}}$ Low to Address Float		0		0	ns
$t_{\text{WHLH}}$	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	43	123	$t_{\text{CLCL}}-25$	$t_{\text{CLCL}}+25$	ns

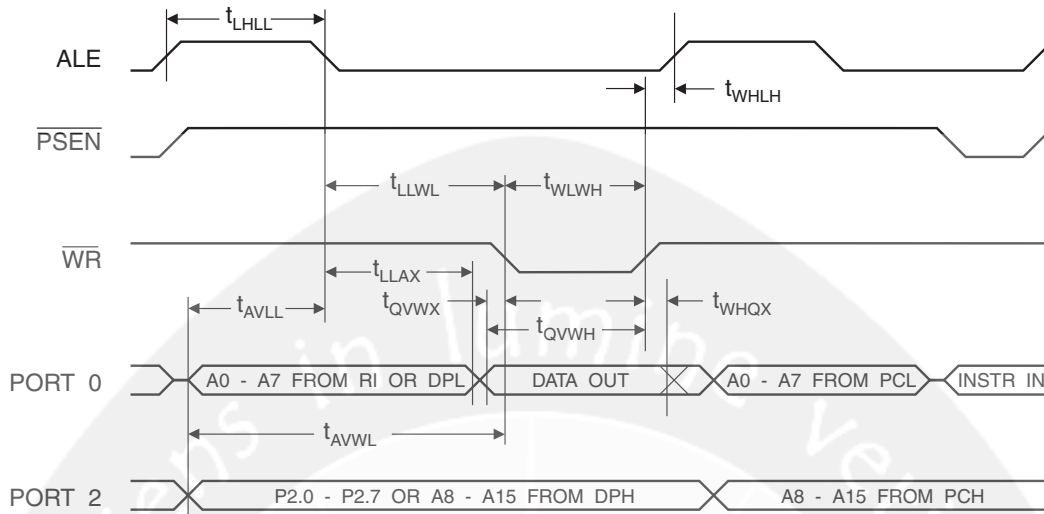
### External Program Memory Read Cycle



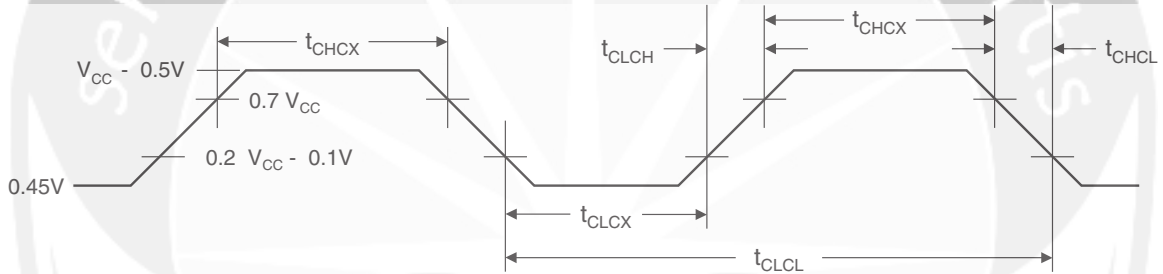
### External Data Memory Read Cycle



## External Data Memory Write Cycle



## External Clock Drive Waveforms



## External Clock Drive

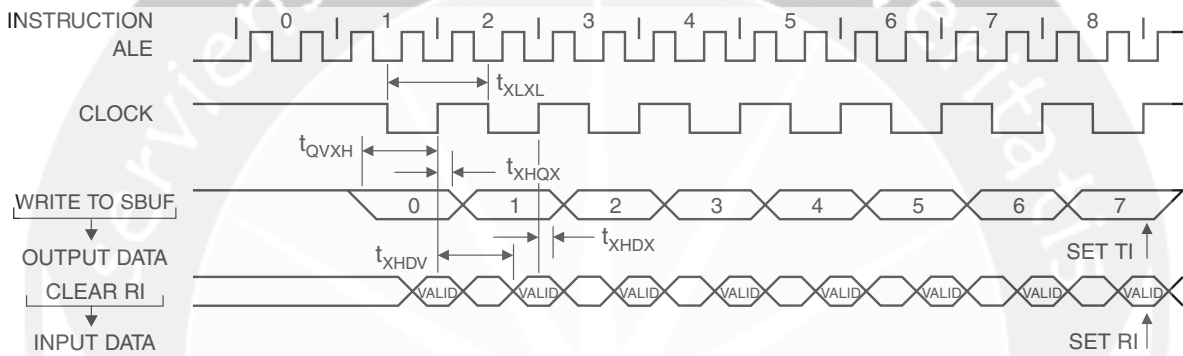
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	33	MHz
$t_{CLCL}$	Clock Period	30		ns
$t_{CHCX}$	High Time	12		ns
$t_{CLCX}$	Low Time	12		ns
$t_{CLCH}$	Rise Time		5	ns
$t_{CHCL}$	Fall Time		5	ns

### Serial Port Timing: Shift Register Mode Test Conditions

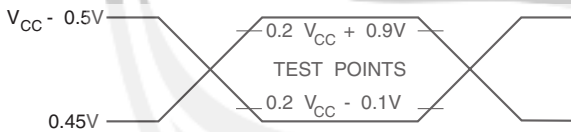
The values in this table are valid for  $V_{CC} = 4.0V$  to  $5.5V$  and Load Capacitance =  $80\text{ pF}$ .

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
$t_{XLXL}$	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		$\mu\text{s}$
$t_{QVXH}$	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}-133$		ns
$t_{XHGX}$	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}-80$		ns
$t_{XHDX}$	Input Data Hold After Clock Rising Edge	0		0		ns
$t_{XHDV}$	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}-133$	ns

### Shift Register Mode Timing Waveforms



### AC Testing Input/Output Waveforms<sup>(1)</sup>



Note: 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<sup>(1)</sup>



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

## Ordering Information

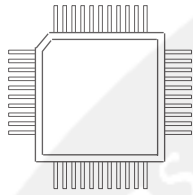
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S52-24AC	44A	Commercial (0°C to 70°C)
		AT89S52-24JC	44J	
		AT89S52-24PC	40P6	
		AT89S52-24AI	44A	Industrial (-40°C to 85°C)
		AT89S52-24JI	44J	
		AT89S52-24PI	40P6	
33	4.5V to 5.5V	AT89S52-33AC	44A	Commercial (0°C to 70°C)
		AT89S52-33JC	44J	
		AT89S52-33PC	40P6	

 = Preliminary Availability



Package Type	
<b>44A</b>	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
<b>44J</b>	44-lead, Plastic J-leaded Chip Carrier (PLCC)
<b>40P6</b>	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

## Packaging Information



\*Controlling dimension: millimeters





## Atmel Headquarters

### *Corporate Headquarters*

2325 Orchard Parkway  
San Jose, CA 95131  
TEL (408) 441-0311  
FAX (408) 487-2600

### *Europe*

Atmel SarL  
Route des Arsenaux 41  
Casa Postale 80  
CH-1705 Fribourg  
Switzerland  
TEL (41) 26-426-5555  
FAX (41) 26-426-5500

### *Asia*

Atmel Asia, Ltd.  
Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimhatsui  
East Kowloon  
Hong Kong  
TEL (852) 2721-9778  
FAX (852) 2722-1369

### *Japan*

Atmel Japan K.K.  
9F, Tonetsu Shinkawa Bldg.  
1-24-8 Shinkawa  
Chuo-ku, Tokyo 104-0033  
Japan  
TEL (81) 3-3523-3551  
FAX (81) 3-3523-7581

## Atmel Product Operations

### *Atmel Colorado Springs*

1150 E. Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
TEL (719) 576-3300  
FAX (719) 540-1759

### *Atmel Grenoble*

Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
TEL (33) 4-7658-3000  
FAX (33) 4-7658-3480

### *Atmel Heilbronn*

Theresienstrasse 2  
POB 3535  
D-74025 Heilbronn, Germany  
TEL (49) 71 31 67 25 94  
FAX (49) 71 31 67 24 23

### *Atmel Nantes*

La Chantrerie  
BP 70602  
44306 Nantes Cedex 3, France  
TEL (33) 0 2 40 18 18 18  
FAX (33) 0 2 40 18 19 60

### *Atmel Rousset*

Zone Industrielle  
13106 Rousset Cedex, France  
TEL (33) 4-4253-6000  
FAX (33) 4-4253-6001

### *Atmel Smart Card ICs*

Scottish Enterprise Technology Park  
East Kilbride, Scotland G75 0QR  
TEL (44) 1355-357-000  
FAX (44) 1355-242-743

---

### *Fax-on-Demand*

North America:  
1-(800) 292-8635  
International:  
1-(408) 441-0732

### *e-mail*

[literature@atmel.com](mailto:literature@atmel.com)

### *Web Site*

<http://www.atmel.com>

### *BBS*

1-(408) 436-4309

### © Atmel Corporation 2001.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL® is the registered trademark of Atmel.

MCS-51® is the registered trademark of Intel Corporation. Terms and product names in this document may be trademarks of others.



Printed on recycled paper.

Rev.1919A-07/01/xM