LAMPIRAN – LAMPIRAN

LAMPIRAN A

PERANGKAT LUNAK

```
; Program SMS Antrian

#INCLUDE            "8051.H"
DISP_SELECT1        .EQU    P1.0
DISP_SELECT2        .EQU    P1.1
DISP_SELECT3        .EQU    P1.2
DATADISPLAY         .EQU    P0
DATAPRINTER         .EQU    P0
KONTROLPRINTER      .EQU    P1.4
STROBEPRINTER       .EQU    P1.5
BUSYPRINTER         .EQU    P1.6
LEDMERAH            .EQU    P3.4
LEDHIJAU            .EQU    P3.5
SAKLAR1             .EQU    P3.2
SAKLAR2             .EQU    P3.3


ADDRRAM             .EQU    $2000
ADRSTATUS           .EQU    $2000
ADRKIRIM            .EQU    $2010
ADRDATA             .EQU    $2100

                    .ORG    $30
BUFPDU1             .BLOCK  1
BUFPDU2             .BLOCK  1
BUFPDU3             .BLOCK  1
BUFPDU4             .BLOCK  1
BUFPDU5             .BLOCK  1
BUFPDU6             .BLOCK  1
BUFPDU7             .BLOCK  1
BUFPDU8             .BLOCK  1
BUFPDU9             .BLOCK  1
BUFPDU10            .BLOCK  1
BUFISI1             .BLOCK  1
BUFISI2             .BLOCK  1
BUFISI3             .BLOCK  1
BUFISI4             .BLOCK  1
BUFISI5             .BLOCK  1
BUFISI6             .BLOCK  1
BUFISI7             .BLOCK  1
BUFISI8             .BLOCK  1
BUFISI9             .BLOCK  1
BUFISI10            .BLOCK  1
BUFKIRIM            .BLOCK  17

BUFSEG1             .BLOCK  1
BUFSEG2             .BLOCK  1
BUFSEG3             .BLOCK  1
BUFSMSC             .BLOCK  1
DATASMS             .BLOCK  6
```

```
STSMS           .BLOCK  1       ;ADA ATAU TIDAK
STDATA          .BLOCK  1       ;BENAR ATAU TIDAK
STMEM           .BLOCK  1
STLENGTH        .BLOCK  1
NUMCS           .BLOCK  1
NUMPT           .BLOCK  1
BUFNOMOR        .BLOCK  20
BUFPRINT        .BLOCK  3


                .ORG    $0
                LJMP    MULAI

                .ORG    $100
MULAI:          MOV     SP,#$20
                MOV     PSW,#$00
                LCALL   DELAY1DETIK
                SETB    STROBEPRINTER
                CLR     KONTROLPRINTER
                CLR     DISP_SELECT1
                CLR     DISP_SELECT2
                CLR     DISP_SELECT3
                MOV     BUFSEG1,#$C0
                MOV     BUFSEG2,#$3F
                MOV     BUFSEG3,#$C0
                MOV     NUMCS,#00
                MOV     NUMPT,#00
                MOV     STSMS,#0
                MOV     STMEM,#0
                MOV     STDATA,#'S'
                CLR     LEDMERAH
                SETB    LEDHIJAU
                LCALL   INITSERIALHP
                LCALL   TAMPILKEDISPLAY
                LCALL   DELAY1DETIK
                MOV     DPTR,#TESMODEM
                LCALL   PROC_KIRIMDATA


BACALAGI1:      LCALL   READCHR
                CJNE    A,#$'O',BACALAGI1
BACALAGI2:      LCALL   READCHR
                CJNE    A,#$'K',BACALAGI2

                MOV     DPTR,#HAPUSSMS1
                LCALL   PROC_KIRIMDATA
                CLR     LEDHIJAU
                LCALL   DELAY1DETIK
                MOV     DPTR,#HAPUSSMS2
                LCALL   PROC_KIRIMDATA
```

```
                SETB    LEDHIJAU
                LCALL   DELAY1DETIK
                MOV     DPTR,#HAPUSSMS3
                LCALL   PROC_KIRIMDATA
                CLR     LEDHIJAU
                LCALL   DELAY1DETIK
                SETB    LEDHIJAU
                LCALL   DELAY1DETIK
                CLR     LEDHIJAU
                LCALL   DELAY1DETIK
                SETB    LEDHIJAU
                SETB    LEDMERAH


LOOP:           MOV     SP,#$20
                JB      SAKLAR1,CEKSAKLAR2
                LCALL   DELAY500M
                JB      SAKLAR1,CEKSAKLAR2
                LCALL   TAMBAHCS
                LCALL   CETAKKARCIS

CEKSAKLAR2:     JB      SAKLAR2,PEMBACAANSMS1
                LCALL   DELAY500M
                JB      SAKLAR2,PEMBACAANSMS1
                LCALL   TAMBAHPT
                LCALL   CETAKKARCIS
PEMBACAANSMS1:
                MOV     DPTR,#BACASMS1
                LCALL   PROC_KIRIMDATA
                LCALL   PEMBACAANSMS
                MOV     A,STSMS
                CJNE    A,#$0,ADA1
                LJMP    PASS
ADA1:           MOV     A,STDATA
                MOV     STMEM,#1
                CJNE    A,#'A',CEKBTP1
                LCALL   TAMBAHCS
                LCALL   PROSESKIRIMSMS1
                LJMP    PASS
CEKBTP1:        CJNE    A,#'B',CEKSALAH1
                LCALL   TAMBAHPT
                LCALL   PROSESKIRIMSMS1
                LJMP    PASS
CEKSALAH1:
                MOV     DPTR,#HAPUSSMS1
                LCALL   PROC_KIRIMDATA
                LCALL   DELAY1DETIK
                LCALL   DELAY1DETIK
PASS:
                LJMP    LOOP
```

```
PROSESKIRIMSMS1:
                LCALL   PROSESPENYATUAN

                MOV     A,STMEM
                CJNE    A,#1,CEKMEM2
                MOV     DPTR,#HAPUSSMS1
                LCALL   PROC_KIRIMDATA

                LCALL   LEDMERAHKEDIP
                LCALL   LEDMERAHKEDIP
                LCALL   LEDMERAHKEDIP

CEKMEM2:

                LCALL   DELAY1DETIK
                LCALL   DELAY1DETIK
                RET

PROSESPENYATUAN:
                MOV     DPTR,#ADRKIRIM
                MOV     A,#'0'
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#'0'
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#'1'
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#'1'
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#'0'
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#'0'
                MOVX    @DPTR,A
                INC     DPTR

                MOV     R0,#BUFNOMOR
PENGISIANNOMOR:
                MOV     A,@R0
                CJNE    A,#$FF,ISIKEBUFKIRIM
                SJMP    ISISELESAI
ISIKEBUFKIRIM:  MOVX    @DPTR,A
                INC     R0
                INC     DPTR
                LJMP    PENGISIANNOMOR
```

```
ISISELESAI:
                    MOV     A,#'0'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'0'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'0'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'0'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'A'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'A'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'0'
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,#'3'
                    MOVX    @DPTR,A
                    INC     DPTR

                    MOV     R0,#BUFKIRIM
                    MOV     R2,#6
ISIBUFKIRIM:
                    MOV     A,@R0
                    MOVX    @DPTR,A
                    INC     DPTR
                    INC     R0
                    DJNZ    R2,ISIBUFKIRIM

                    MOV     A,#$0D          ;END OF DATA KIRIM
                    MOVX    @DPTR,A


                    MOV     A,STLENGTH
                    CJNE    A,#'C',CEKLD
                    MOV     DPTR,#CSMS1
                    SJMP    CEKKRM
CEKLD:
                    MOV     DPTR,#CSMS2
CEKKRM:
                    LCALL   PROC_KIRIMDATA
                    LCALL   DELAY1DETIK
```

```
COBATAMPIL:
                        MOV     DPTR,#ADRKIRIM
                        MOV     R2,#16
COBATAMPIL1:
                        MOVX    A,@DPTR
                        CJNE    A,#$0D,TAMPIL1
                        MOV     A,#26
                        LCALL   SENDCHR
                        LCALL   DELAY1DETIK
                        LCALL   DELAY1DETIK
                        LCALL   DELAY1DETIK
                        RET

TAMPIL1:                LCALL   SENDCHR
                        INC     DPTR
                        DJNZ    R2,COBATAMPIL1
                        LJMP    COBATAMPIL1

                        RET


LEDMERAHKEDIP:          CLR     LEDMERAH
                        LCALL   DELAY500M
                        SETB    LEDMERAH
                        LCALL   DELAY500M
                        RET

;-------------------------------------------------------------------------------
; RUTIN PROSES SMS MASUK
;-------------------------------------------------------------------------------
PEMBACAANSMS:
BACASTAT0:              LCALL   READCHR
                        CJNE    A,#$0A,BACASTAT0

                        MOV     DPTR,#ADRSTATUS
BACASTAT1:              LCALL   READCHR
                        MOVX    @DPTR,A
                        INC     DPTR
                        CJNE    A,#$0A,BACASTAT1

                        MOV     DPTR,#ADRDATA
BACASTAT2:              LCALL   READCHR
                        MOVX    @DPTR,A
                        INC     DPTR
                        CJNE    A,#$0A,BACASTAT2

                        MOV     DPTR,#ADRSTATUS
CEKSTATUS:
                        MOVX    A,@DPTR
```

```
                        INC     DPTR
                        CJNE    A,#$0D,CEKSTATUS
                        DEC     DPL
                        DEC     DPL
                        MOVX    A,@DPTR
                        CJNE    A,#'0',ADADATA
                        MOV     A,#'N'
                        LCALL   LEDKEDIP
                        MOV     STSMS,#0
                        RET
ADADATA:                MOV     STSMS,#1

                        MOV     DPTR,#ADRDATA
                        INC     DPTR
                        MOVX    A,@DPTR
                        ANL     A,#$0F
                        MOV     BUFSMSC,A
                        ADD     A,BUFSMSC

                        MOV     DPTR,#ADRDATA
                        INC     DPTR
                        INC     DPTR
                        MOV     R2,A
TAMPILSMSC:
                        MOVX    A,@DPTR
                        INC     DPTR
                        DJNZ    R2,TAMPILSMSC

                        INC     DPTR
                        INC     DPTR
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     STLENGTH,A
                        LCALL   ASCIITOHEX
                        ANL     A,#$0F
                        ADD     A,#$04
                        MOV     R2,A
;PENGAMBILAN NO TELP
                        DEC     DPL

                        MOV     R0,#BUFNOMOR
TAMPILNOMOR:
                        MOVX    A,@DPTR
                        MOV     @R0,A
                        INC     DPTR
                        INC     R0
                        DJNZ    R2,TAMPILNOMOR
                        MOV     @R0,#$FF

                        MOV     R2,#18
```

```
KEPOSISIDATA:       INC    DPTR
                    DJNZ   R2,KEPOSISIDATA

                    MOV    R0,#DATASMS
                    MOV    R2,#6
TAMPILDATASMS:      MOVX   A,@DPTR
                    MOV    @R0,A
                    INC    DPTR
                    INC    R0
                    DJNZ   R2,TAMPILDATASMS

                    LCALL  PROSESBANDING  ;UNTUK MENGETAHUI DATA
                    LCALL  DELAY1DETIK
                    RET
;-------------------------------------------------------------------------------

PROSESBANDING:
BANDINGDATA1:
                    MOV    DPTR,#DATASMSB1
                    MOV    R0,#DATASMS
                    MOV    R2,#6
BANDING1:
                    CLR    A
                    MOVC   A,@A+DPTR
                    MOV    B,A
                    MOV    A,@R0
                    CJNE   A,B,BANDINGDATA2
                    INC    DPTR
                    INC    R0
                    DJNZ   R2,BANDING1
                    MOV    STDATA,#'A'
                    RET
BANDINGDATA2:


                    MOV    DPTR,#DATASMSB2
                    MOV    R0,#DATASMS
                    MOV    R2,#6
BANDING2:
                    CLR    A
                    MOVC   A,@A+DPTR
                    MOV    B,A
                    MOV    A,@R0
                    CJNE   A,B,BANDINGDATA3
                    INC    DPTR
                    INC    R0
                    DJNZ   R2,BANDING2
                    MOV    STDATA,#'B'
                    RET
```

```
BANDINGDATA3:
                MOV     DPTR,#DATASMSB3
                MOV     R0,#DATASMS
                MOV     R2,#6
BANDING3:
                CLR     A
                MOVC    A,@A+DPTR
                MOV     B,A
                MOV     A,@R0
                CJNE    A,B,BANDINGDATA4
                INC     DPTR
                INC     R0
                DJNZ    R2,BANDING3
                MOV     STDATA,#'A'
                RET

BANDINGDATA4:
                MOV     DPTR,#DATASMSB4
                MOV     R0,#DATASMS
                MOV     R2,#6
BANDING4:
                CLR     A
                MOVC    A,@A+DPTR
                MOV     B,A
                MOV     A,@R0
                CJNE    A,B,BANDINGDATASALAH
                INC     DPTR
                INC     R0
                DJNZ    R2,BANDING4
                MOV     STDATA,#'B'
                RET

BANDINGDATASALAH:
                MOV     STDATA,#'S'
                RET


LEDKEDIP:       CLR     LEDHIJAU
                LCALL   DELAY500M
                SETB    LEDHIJAU
                RET

TAMBAHCS:
                MOV     A,NUMCS
                ADD     A,#$01
                DA      A
                MOV     NUMCS,A

                MOV     A,NUMCS
                ANL     A,#$0F
```

```
                LCALL   CONV7SEG
                MOV     BUFSEG1,A
                LCALL   TAMPILKEDISPLAY

                MOV     A,NUMCS
                SWAP    A
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFPRINT,A
                MOV     A,NUMCS
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFPRINT+1,A
                MOV     BUFPRINT+2,#'A'

                MOV     A,NUMCS
                SWAP    A
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFISI1,A

                MOV     A,NUMCS
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFISI2,A
                MOV     BUFISI3,#'A'
                LCALL   KONVERSIKEPDU
                LCALL   ISIKEMBALI

                RET

TAMBAHPT:
                MOV     A,NUMPT
                ADD     A,#$01
                DA      A
                MOV     NUMPT,A

                MOV     A,NUMPT
                ANL     A,#$0F
                LCALL   CONV7SEG
                MOV     BUFSEG3,A
                LCALL   TAMPILKEDISPLAY

                MOV     A,NUMPT
                SWAP    A
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFPRINT,A
                MOV     A,NUMPT
                ANL     A,#$0F
```

```
                ADD     A,#$30
                MOV     BUFPRINT+1,A
                MOV     BUFPRINT+2,#'B'

                MOV     A,NUMPT
                SWAP    A
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFISI1,A

                MOV     A,NUMPT
                ANL     A,#$0F
                ADD     A,#$30
                MOV     BUFISI2,A
                MOV     BUFISI3,#'B'
                LCALL   KONVERSIKEPDU
                LCALL   ISIKEMBALI

                RET

ISIKEMBALI:
                MOV     R0,#BUFPDU1
                MOV     R1,#BUFKIRIM

                MOV     R2,#8
ISIKEBUFFKIRIM:
                MOV     A,@R0
                SWAP    A
                ANL     A,#$0F
                LCALL   KONVERSIKEASCII
                MOV     @R1,A
                INC     R1
                MOV     A,@R0
                ANL     A,#$0F
                LCALL   KONVERSIKEASCII
                MOV     @R1,A
                INC     R0
                INC     R1
                DJNZ    R2,ISIKEBUFFKIRIM
                RET
```

```
;-------------------------------------
;ROUTINE KONVERSI KE PDU
;-------------------------------------
KONVERSIKEPDU:
DATAKE1:        MOV     A,BUFISI1
                MOV     R3,#0
                LCALL   GESERKANAN
                MOV     BUFPDU1,A

                MOV     A,BUFISI2
                MOV     R4,#7
                LCALL   GESERKIRI
                ORL     A,BUFPDU1
                MOV     BUFPDU1,A

DATAKE2:        MOV     A,BUFISI2
                MOV     R3,#1
                LCALL   GESERKANAN
                MOV     BUFPDU2,A

                MOV     A,BUFISI3
                MOV     R4,#6
                LCALL   GESERKIRI
                ORL     A,BUFPDU2
                MOV     BUFPDU2,A

DATAKE3:        MOV     A,BUFISI3
                MOV     R3,#2
                LCALL   GESERKANAN
                MOV     BUFPDU3,A

                MOV     A,BUFISI4
                MOV     R4,#5
                LCALL   GESERKIRI
                ORL     A,BUFPDU3
                MOV     BUFPDU3,A

DATAKE4:        MOV     A,BUFISI4
                MOV     R3,#3
                LCALL   GESERKANAN
                MOV     BUFPDU4,A

                MOV     A,BUFISI5
                MOV     R4,#4
                LCALL   GESERKIRI
                ORL     A,BUFPDU4
                MOV     BUFPDU4,A

DATAKE5:        MOV     A,BUFISI5
                MOV     R3,#4
```

```
                    LCALL   GESERKANAN
                    MOV     BUFPDU5,A

                    MOV     A,BUFISI6
                    MOV     R4,#3
                    LCALL   GESERKIRI
                    ORL     A,BUFPDU5
                    MOV     BUFPDU5,A

DATAKE6:            MOV     A,BUFISI6
                    MOV     R3,#5
                    LCALL   GESERKANAN
                    MOV     BUFPDU6,A
                    MOV     A,BUFISI7
                    MOV     R4,#2
                    LCALL   GESERKIRI
                    ORL     A,BUFPDU6
                    MOV     BUFPDU6,A

DATAKE7:            MOV     A,BUFISI7
                    MOV     R3,#6
                    LCALL   GESERKANAN
                    MOV     BUFPDU7,A

                    MOV     A,BUFISI8
                    MOV     R4,#1
                    LCALL   GESERKIRI
                    ORL     A,BUFPDU7
                    MOV     BUFPDU7,A

DATAKE8:            MOV     A,BUFISI8
                    MOV     R3,#7
                    LCALL   GESERKANAN
                    MOV     BUFPDU8,A
                    MOV     A,BUFISI9
                    MOV     R4,#0
                    LCALL   GESERKIRI
                    ORL     A,BUFPDU8
                    MOV     BUFPDU8,A

                    RET

GESERKIRI:
                    CJNE    R4,#0,GESERKIRI1
                    RET
GESERKIRI1:
                    CLR C
                    RLC     A
                    DJNZ    R4,GESERKIRI
                    RET
```

```
GESERKANAN:
                    CJNE    R3,#0,GESERKANAN1
                    RET
GESERKANAN1:
                    CLR C
                    RRC     A
                    DJNZ    R3,GESERKANAN
                    RET

KONVERSIKEASCII:
ASCIIA:             CJNE    A,#$A,ASCIIB
                    MOV     A,#'A'
                    RET
ASCIIB:             CJNE    A,#$B,ASCIIC
                    MOV     A,#'B'
                    RET
ASCIIC:             CJNE    A,#$C,ASCIID
                    MOV     A,#'C'
                    RET
ASCIID:             CJNE    A,#$D,ASCIIE
                    MOV     A,#'D'
                    RET
ASCIIE:             CJNE    A,#$E,ASCIIF
                    MOV     A,#'E'
                    RET
ASCIIF:             CJNE    A,#$F,ASCIIANGKA
                    MOV     A,#'F'
                    RET
ASCIIANGKA:         ADD     A,#$30
                    RET

ISI:
                    MOV     DPTR,#DATATEXT
                    MOV     R0,#BUFISI1
ISIKAN
                    CLR     A
                    MOVC    A,@A+DPTR
                    JZ      SELESAIISI
                    MOV     @R0,A
                    INC     DPTR
                    INC     R0
                    LJMP    ISIKAN
SELESAIISI:         RET


ASCIITOHEX:
TOASCIIA:           CJNE    A,#'A',TOASCIIB
                    MOV     A,#$0A
                    RET
```

```
TOASCIIB:           CJNE    A,#'B',TOASCIIC
                    MOV     A,#$0C
                    RET
TOASCIIC:           CJNE    A,#'C',TOASCIID
                    MOV     A,#$0C
                    RET
TOASCIID:           CJNE    A,#'D',TOASCIIE
                    MOV     A,#$0E
                    RET
TOASCIIE:           CJNE    A,#'E',TOASCIIF
                    MOV     A,#$0E
                    RET
TOASCIIF:           CJNE    A,#'F',TOASCII
                    MOV     A,#$0F
                    RET
TOASCII:            ANL     A,#$0F
                    RET

TAMPILKEDISPLAY:
                    MOV     DATADISPLAY,BUFSEG1
                    SETB    DISP_SELECT1
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    CLR     DISP_SELECT1
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    MOV     DATADISPLAY,#$3F
                    SETB    DISP_SELECT2
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    CLR     DISP_SELECT2
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    MOV     DATADISPLAY,BUFSEG3
                    SETB    DISP_SELECT3
                    NOP
                    NOP
                    NOP
```

```
                NOP
                NOP
                CLR     DISP_SELECT3
                NOP
                NOP
                NOP
                NOP
                NOP
                RET


CETAKKARCIS:
                MOV     DPTR,#DATAPRINT1
                LCALL   PRINTING
                MOV     DPTR,#DATAPRINT2
                LCALL   PRINTING
                MOV     DPTR,#DATAPRINT3
                LCALL   PRINTING
                MOV     A,BUFPRINT
                LCALL   KONTROLSTROBE
                MOV     A,BUFPRINT+1
                LCALL   KONTROLSTROBE
                MOV     A,BUFPRINT+2
                LCALL   KONTROLSTROBE
                MOV     A,#$0A
                LCALL   KONTROLSTROBE
                MOV     A,#$0D
                LCALL   KONTROLSTROBE
                MOV     DPTR,#DATAPRINT1
                LCALL   PRINTING
                MOV     A,#$0A
                LCALL   KONTROLSTROBE
                MOV     A,#$0D
                LCALL   KONTROLSTROBE
                MOV     A,#$0A
                LCALL   KONTROLSTROBE
                MOV     A,#$0D
                LCALL   KONTROLSTROBE
                RET


PRINTING:       CLR     A
                MOVC    A,@A+DPTR
                CJNE    A,#$FF,PRINTING1
                RET
PRINTING1:      LCALL   KONTROLSTROBE
                INC     DPTR
                LJMP    PRINTING
```

```
KONTROLSTROBE:
                        MOV       DATAPRINTER,A
                        NOP
                        NOP
                        NOP
                        SETB      KONTROLPRINTER
                        NOP
                        NOP
                        NOP
                        NOP
                        NOP
                        NOP
                        NOP
                        NOP
                        CLR       KONTROLPRINTER
                        NOP
                        NOP
                        NOP
                        NOP
                        NOP
                        CLR           STROBEPRINTER
                        LCALL         DELAYPRINT
                        SETB          STROBEPRINTER
                        LCALL         DELAYPRINT
                        NOP
                        NOP
                        NOP
                        NOP
                        RET

;---------------------------------------------------------------------
; RUTIN PENGIRIMAN STRING DARI LOOKUP TABLE
;---------------------------------------------------------------------
PROC_KIRIMDATA:
                        CLR     A
                        MOVC    A,@A+DPTR
                        CJNE    A,#0FFH,TULISDATA
                        RET
TULISDATA:              LCALL   SENDCHR
                        INC     DPTR
                        LJMP    PROC_KIRIMDATA

SENDCHR:
                        CLR  TI
                        MOV  SBUF,A
TXLOOP:
                        JNB  TI,TXLOOP
                        RET
```

```
;-------------------------------------------------------------------------------
; SUBROUTINE BACA DATA RS232 DARI HANDPHONE DAN
; DISIMPAN DI AKUMULATOR
;-------------------------------------------------------------------------------
READCHR:
                        JNB    RI,READCHR      ; TUNGGU KARAKTER
                        MOV    A,SBUF          ; AMBIL KARAKTER
                        ANL    A,#$7F          ; MASK OFF BIT KE 8
                        CLR    RI              ; CLEAR SERIAL STATUS BIT
                        RET
;-------------------------------------------------------------------------------------------
; INISIAL KOMUNIKASI SERIAL 19200,8,1,N UNTUK KE HANDPHONE
;-------------------------------------------------------------------------------------------
INITSERIALHP:
                MOV   TMOD, #$21   ;
                MOV   TCON, #$41   ;
                MOV   TH1,  #$fD   ; Set 19200 baud with xtal=11.059mhz
                MOV   SCON, #$50   ; set serial control reg for 8 bit data
                                   ; and mode 1
                ORL   87H,#$80
                RET

;------ ROUTINE KONVERSI ANGKA KE DALAM FORMAT 7 SEGMENT ----
CONV7SEG:
                MOV    DPTR,#SEG
LOAD:           MOVC   A,@A+DPTR
                RET
PROC_4XDELAY:
                LCALL  DELAY
                LCALL  DELAY
                LCALL  DELAY
                LCALL  DELAY
                RET

DELAY1DETIK:
DLY1:           MOV    R5,#100
DLY2:           MOV    R6,#100
DLY3:           MOV    R7,#50
DLY4:           DJNZ   R7,DLY4
                DJNZ   R6,DLY3
                DJNZ   R5,DLY2
                RET
DELAY500M:
DLY11:          MOV    R5,#50
DLY21:          MOV    R6,#100
DLY31:          MOV    R7,#50
DLY41:          DJNZ   R7,DLY41
                DJNZ   R6,DLY31
                DJNZ   R5,DLY21
                RET
```

```
DELAY:
                MOV     R7,#$ff
DLD1:           MOV     R6,#$FF
DLD2:           DJNZ    R6,DLD2
                DJNZ    R7,DLD1
                RET

DELAYPRINT:     MOV     R6,#$0A
DELAYPRINT1:    MOV     R7,#$FF
DELAYPRINT2:    DJNZ    R7,DELAYPRINT2
                DJNZ    R6,DELAYPRINT1
                RET

DELAYSW:
                MOV     R5,#02H
DELAYSW0:       MOV     R6,#0FFH
DELAYSW1:       MOV     R7,#0FFH
DELAYSW2:       DJNZ    R7,DELAYSW2
                DJNZ    R6,DELAYSW1
                DJNZ    R5,DELAYSW0
                RET

TESMODEM:       .BYTE       "AT",$0D,$0A,$FF
BACASMS1:       .BYTE       "AT+CMGR=1",$0D,$0A,$FF
BACASMS2:       .BYTE       "AT+CMGR=2",$0D,$0A,$FF
BACASMS3:       .BYTE       "AT+CMGR=3",$0D,$0A,$FF
BACASMS4:       .BYTE       "AT+CMGR=4",$0D,$0A,$FF
HAPUSSMS1:      .BYTE       "AT+CMGD=1",$0D,$0A,$FF
HAPUSSMS2:      .BYTE       "AT+CMGD=2",$0D,$0A,$FF
HAPUSSMS3:      .BYTE       "AT+CMGD=3",$0D,$0A,$FF
HAPUSSMS4:      .BYTE       "AT+CMGD=4",$0D,$0A,$FF

INITMODEM       .BYTE       "ATH",0DH,0AH,0FFH
                            ;HANGE MODEM
SMS1            .BYTE       "AT+CMGF=0",0DH,0AH,0FFH
                            ;SET PDU FORMAT
CSMS1           .BYTE       "AT+CMGS=17",0DH,0AH,0FFH
                            ;KIRIM PERINTAH SEND SMS PANJANG DATA 23
CSMS2           .BYTE       "AT+CMGS=18",0DH,0AH,0FFH
                            ;KIRIM PERINTAH SEND SMS PANJANG DATA 23
DATASMSB1       .BYTE       "02C329",$FF    ;CS
DATASMSB2       .BYTE       "025021",$FF    ;PB
DATASMSB3       .BYTE       "02E339",$FF    ;cs
DATASMSB4       .BYTE       "027031",$FF    ;pb
DATAPRINT1      .BYTE       "  ===============",$0A,$0D,$FF
DATAPRINT2      .BYTE       "   NOMOR ANTRIAN",$0A,$0D,$FF
DATAPRINT3      .BYTE       "       ",$FF
;                           0   1   2   3   4   5   6   7   8   9
SEG:            .BYTE   $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90,$C0,$BF,$BF
                .END
```

LAMPIRAN B

DATA KOMPONEN MIKROKONTROLER AT89C52

Features

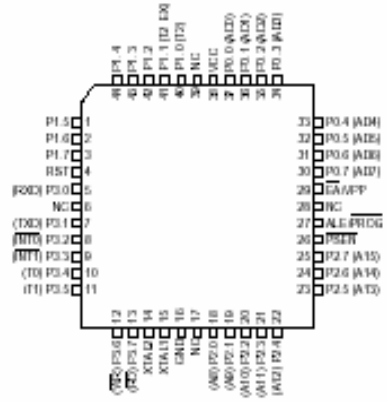
- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

Description

The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 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 Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

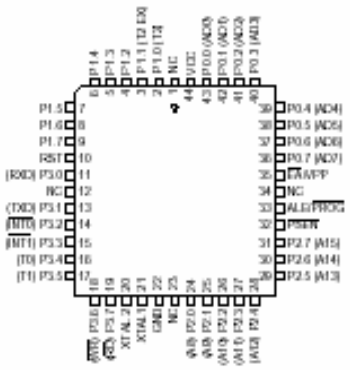
Pin Configurations

PQFP/TQFP



PDIP

PLCC



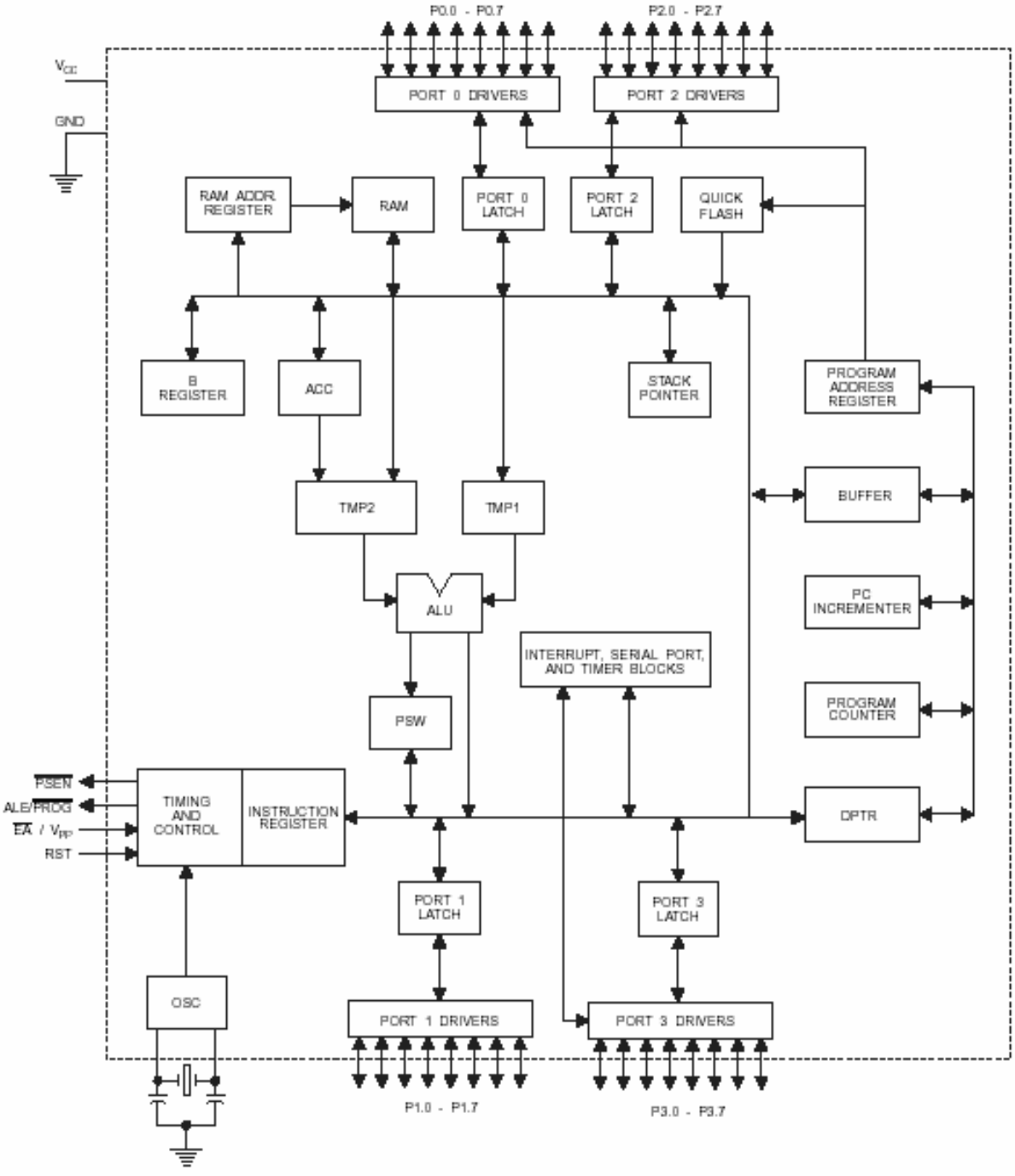
8-bit Microcontroller with 8K Bytes Flash

AT89C52

Not Recommended for New Designs. Use AT89S52.

Rev. 0313H–02/00

Block Diagram



The AT89C52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, 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 AT89C52 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 hardware reset.

Pin Description

VCC
Supply voltage.

GND
Ground.

Port 0

Port 0 is an 8-bit open drain bi-directional 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 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.

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)

Port 2

Port 2 is an 8-bit bi-directional 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 bi-directional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51, as 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.

PSEN

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

When the AT89C52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

External Access Enable. 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, EA will be internally latched on reset.

EA should be strapped to V_{CC} for internal program executions.

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

Table 1. AT89C52 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							0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 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 4) 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.r

Table 2. T2CON – Timer/Counter 2 Control Registe

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.

Data Memory

The AT89C52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means 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 that use direct addressing access 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.

Timer 0 and 1

Timer 0 and Timer 1 in the AT89C52 operate the same way as Timer 0 and Timer 1 in the AT89C51.

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/$\overline{RL2}$	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)

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. 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.

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.

Figure 1. Timer in Capture Mode

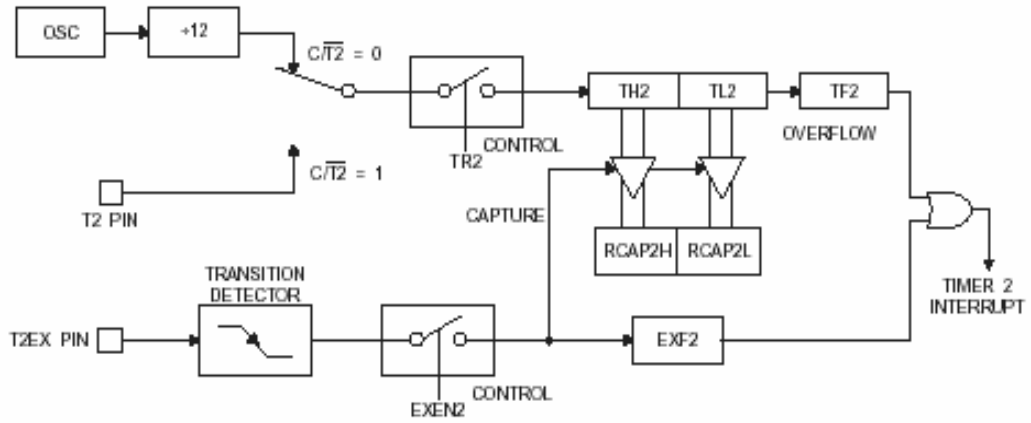


Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H 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 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.

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

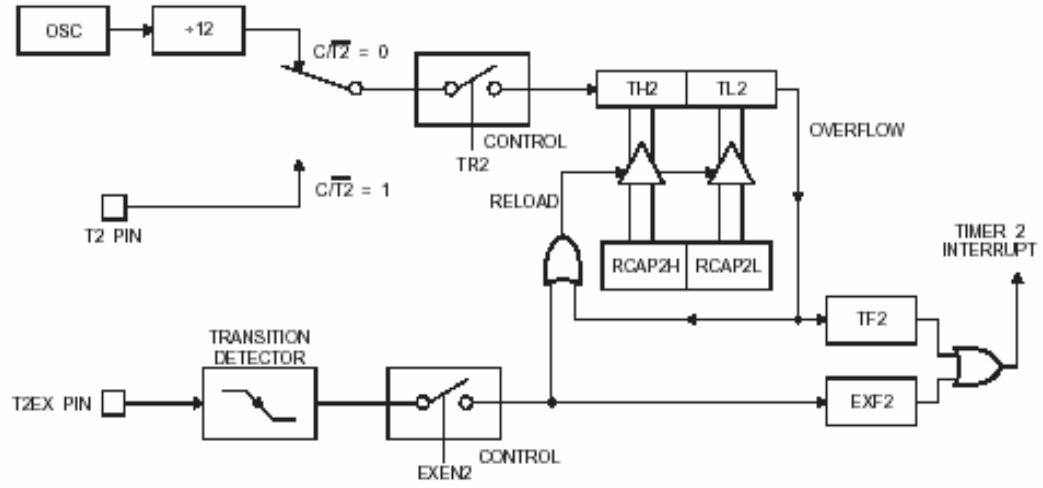


Table 4. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

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

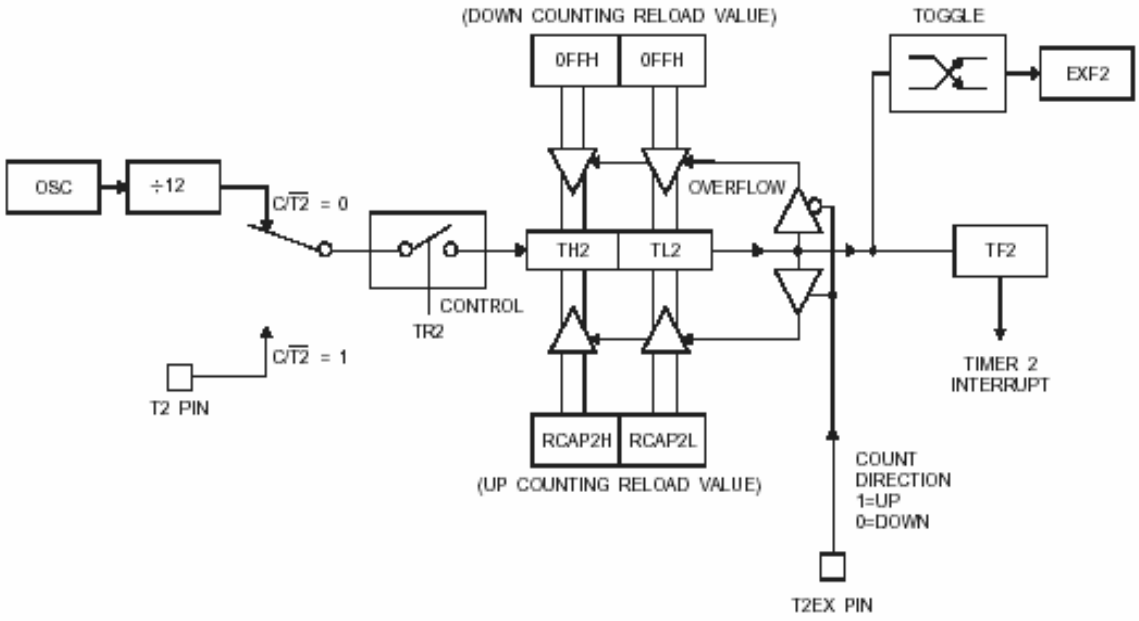
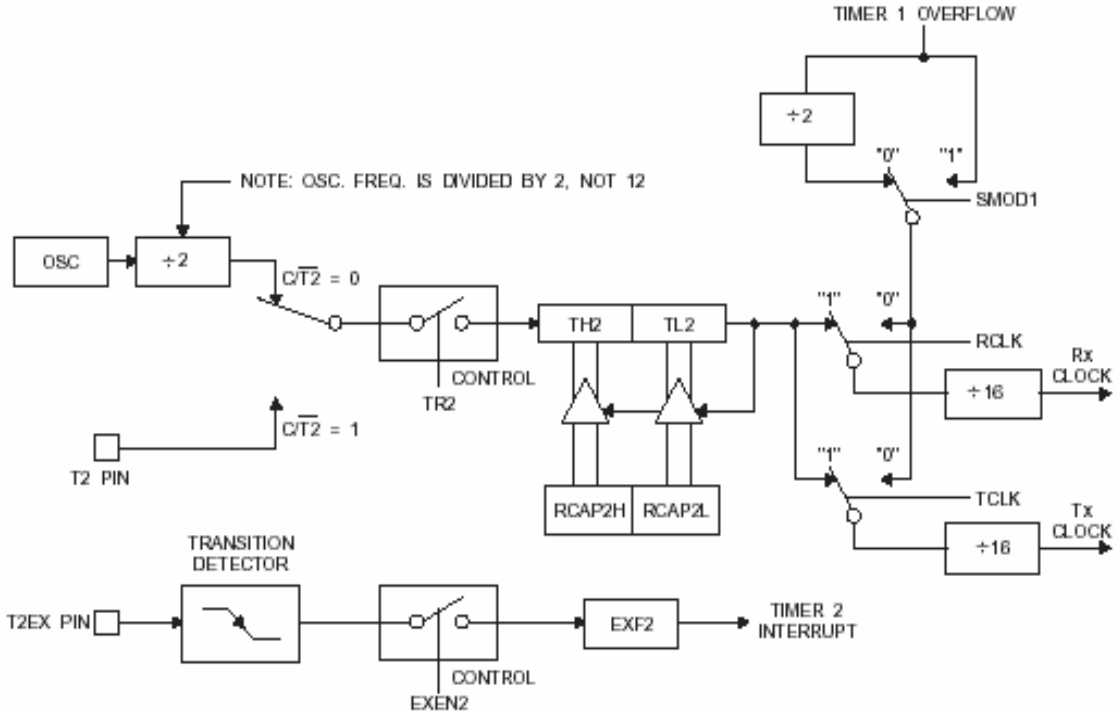


Figure 4. Timer 2 in Baud Rate Generator Mode



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 4.

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.

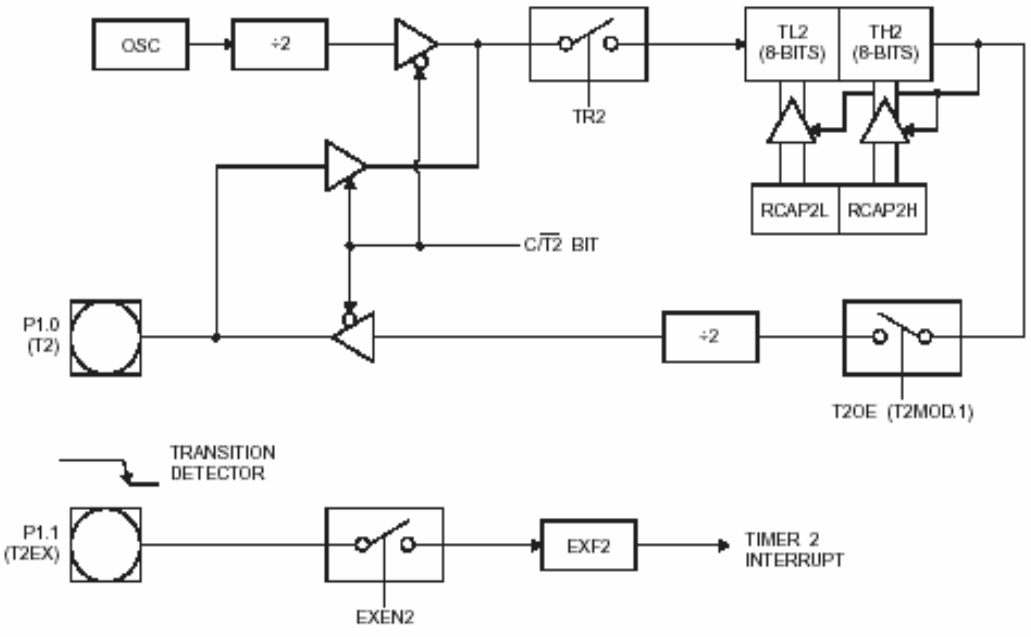
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

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 4. 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 5. 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 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.

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

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.

UART

The UART in the AT89C52 operates the same way as the UART in the AT89C51.

Interrupts

The AT89C52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 6.

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 shows that bit position IE.6 is unimplemented. In the AT89C51, 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

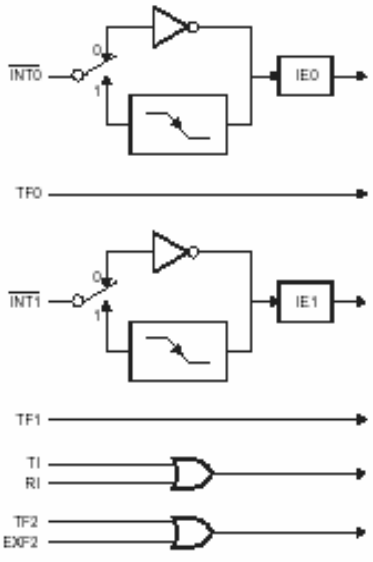
(MSB)							(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0

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 6. 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 7. 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 8. 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

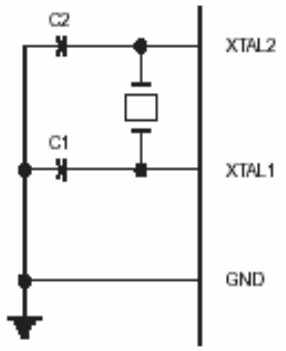
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

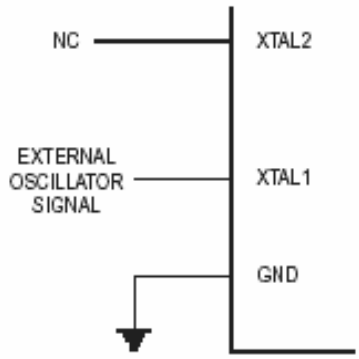
In the power-down mode, the oscillator is stopped, and the instruction that invokes power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power-down mode is terminated. The only exit from power-down is a hardware reset. Reset redefines 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 7. Oscillator Connections



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

Figure 8. External Clock Drive Configuration



Status of External Pins During Idle and Powe-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 AT89C52 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.

Lock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ 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

The AT89C52 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The Low-voltage programming mode provides a convenient way to program the AT89C52 inside the user's system, while the high-voltage programming mode is compatible with conventional third-party Flash or EPROM programmers.

The AT89C52 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP} = 12V	V_{PP} = 5V
Top-side Mark	AT89C52 xxxx yyww	AT89C52 xxxx - 5 yyww
Signature	(030H) = 1EH (031H) = 52H (032H) = FFH	(030H) = 1EH (031H) = 52H (032H) = 05H

The AT89C52 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm Before programming the AT89C52, the address, data and control signals should be set up according to the Flash programming mode table and Figure 9 and Figure 10. To program the AT89C52, 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 for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling The AT89C52 features $\overline{Data}$ Polling to indicate the end of a 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/$\overline{Busy}$ The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. 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 data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all 1s. The chip erase operation must be executed before the code memory can be reprogrammed.

Reading the Signature Bytes The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 52H indicates 89C52
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

Programming Interface

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

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

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	 (1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10 ms $\overline{PROG}$ pulse.

Figure 9. Programming the Flash Memory

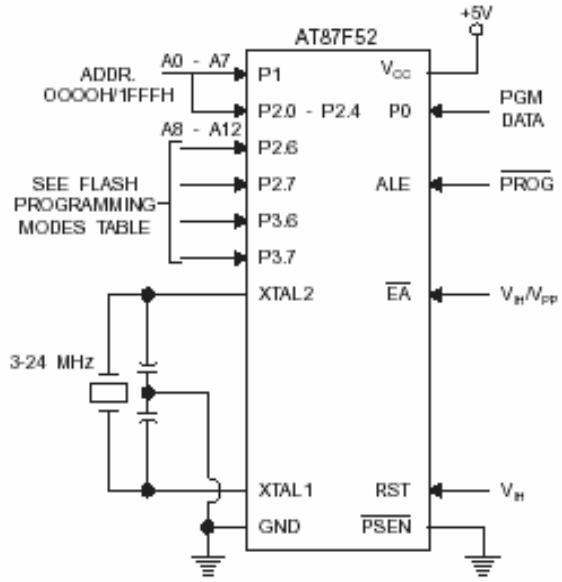
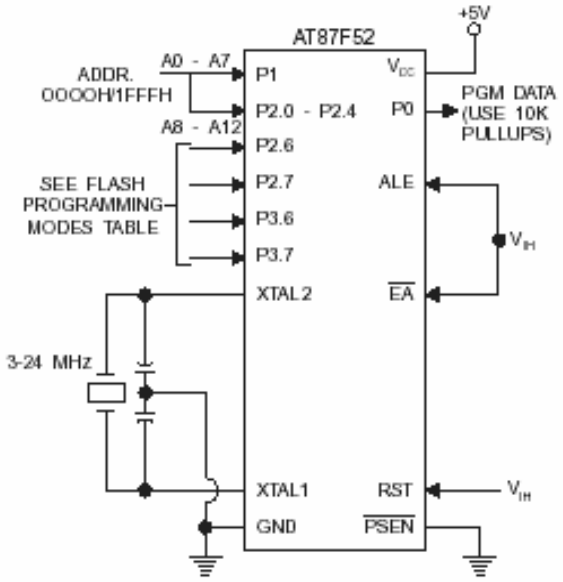


Figure 10. Verifying the Flash Memory



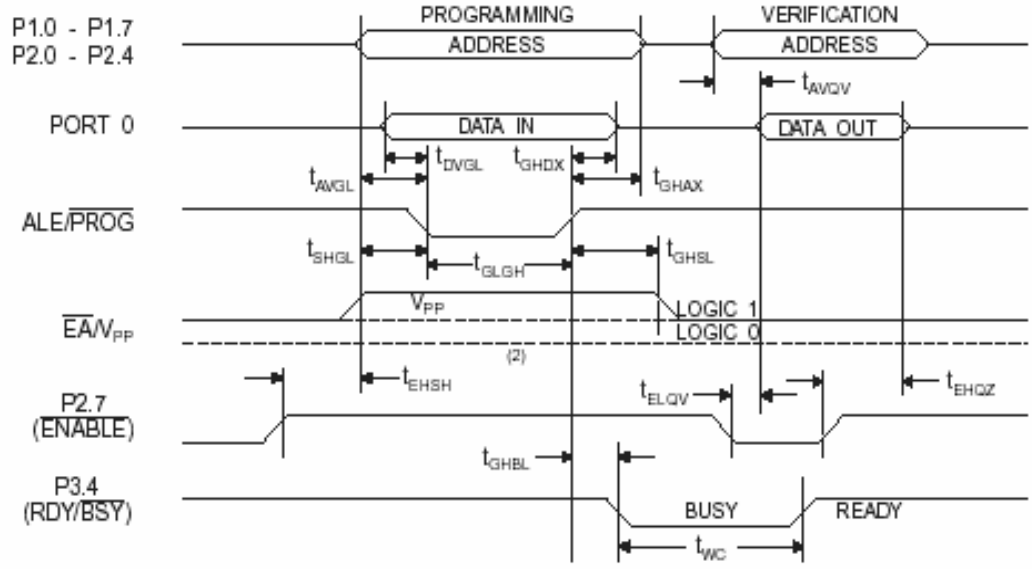
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

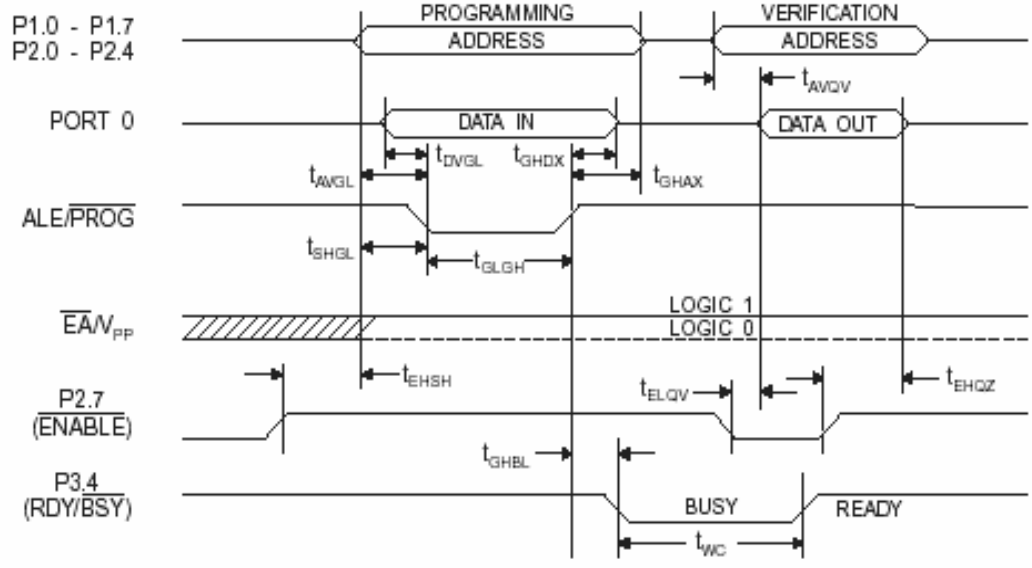
Symbol	Parameter	Min	Max	Units
V_{PP}[1]	Programming Enable Voltage	11.5	12.5	V
I_{PP}[1]	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}[1]	V_{PP} Hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

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

Flash Programming and Verification Waveforms - High-voltage Mode (V_{PP}=12V)



Flash Programming and Verification Waveforms - Low-voltage Mode (V_{PP}=5V)



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°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 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[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low-voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High-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 High-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 Pulldown 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[1]	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

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.

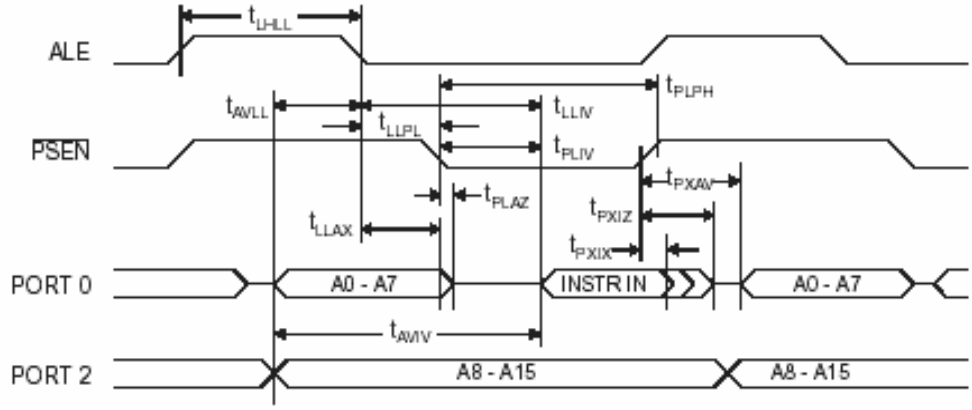
AC Characteristics

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

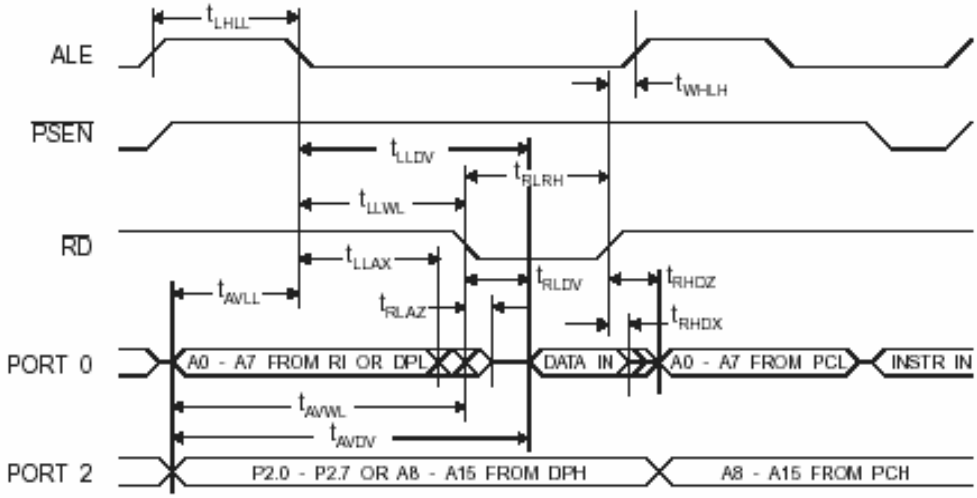
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	24	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
t_{AVLL}	Address Valid to ALE Low	43		t_{CLCL}-13		ns
t_{LLAX}	Address Hold After ALE Low	48		t_{CLCL}-20		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-13		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-20		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-45	ns
t_{PXIX}	Input Instruction Hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float after $\overline{PSEN}$		59		t_{CLCL}-10	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}$-55	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-20		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-120		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		t_{CLCL}-20		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-20	t_{CLCL}+25	ns

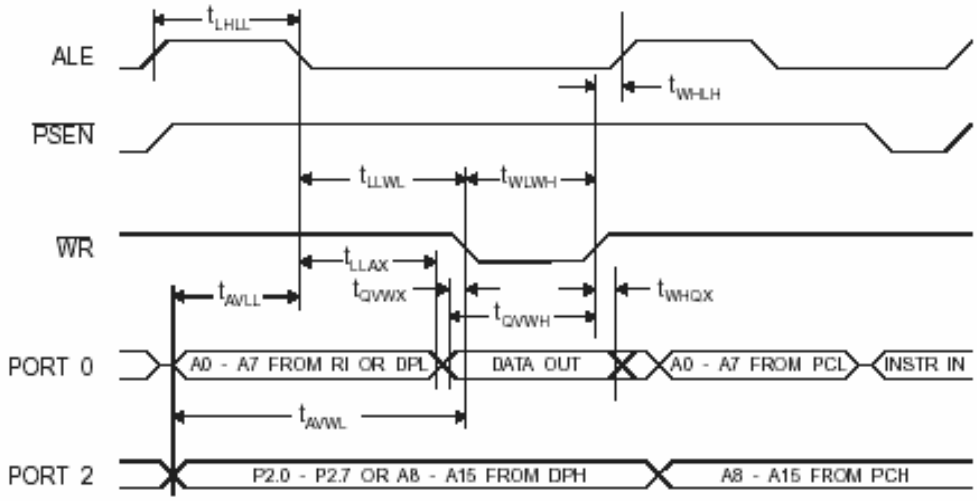
External Program Memory Read Cycle



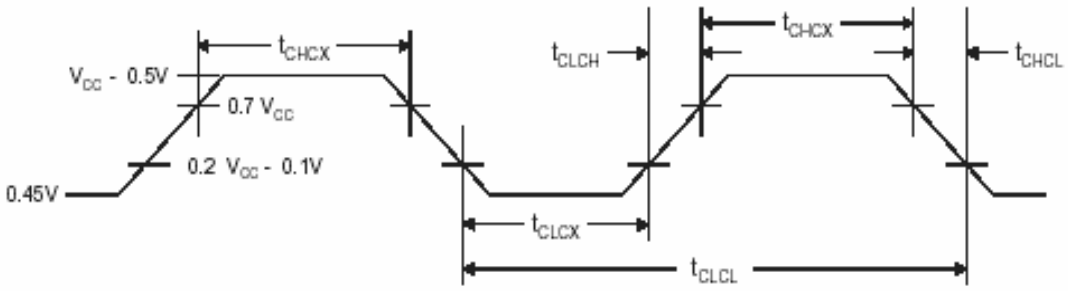
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

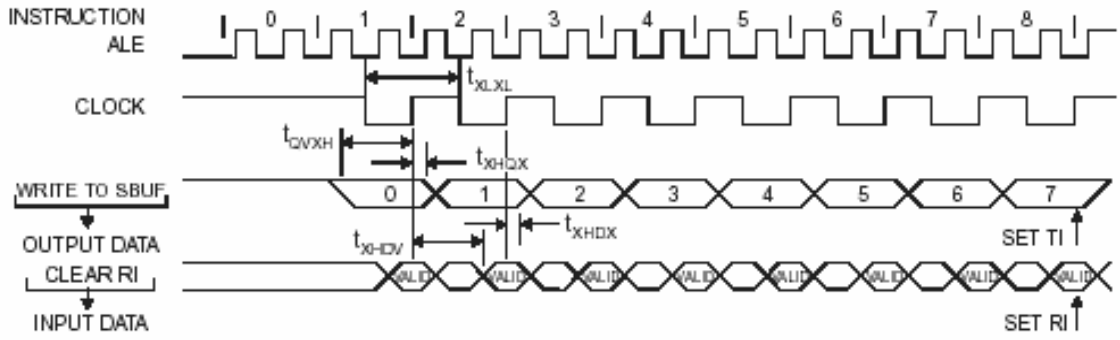
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

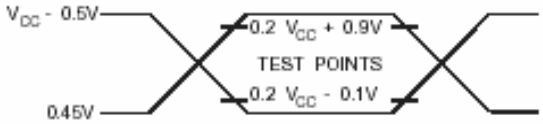
The values in this table are valid for V_{CC} = 5.0V ±20% and Load Capacitance = 80 pF.

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		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

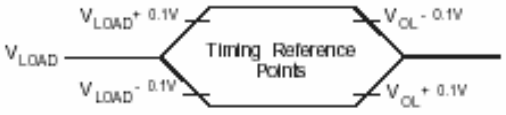


AC Testing Input/Output Waveforms[1]



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[1]



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

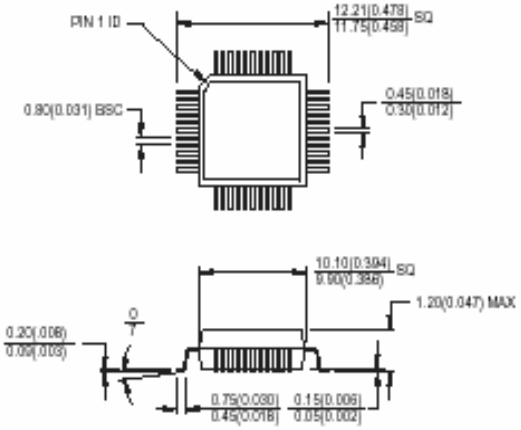
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ±20%	AT89C52-12AC AT89C52-12JC AT89C52-12PC AT89C52-12QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C52-12AI AT89C52-12JI AT89C52-12PI AT89C52-12QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
16	5V ±20%	AT89C52-16AC AT89C52-16JC AT89C52-16PC AT89C52-16QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C52-16AI AT89C52-16JI AT89C52-16PI AT89C52-16QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
20	5V ±20%	AT89C52-20AC AT89C52-20JC AT89C52-20PC AT89C52-20QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C52-20AI AT89C52-20JI AT89C52-20PI AT89C52-20QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
24	5V ±20%	AT89C52-24AC AT89C52-24JC AT89C52-24PC AT89C52-24QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C52-24AI AT89C52-24JI AT89C52-24PI AT89C52-24QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)

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

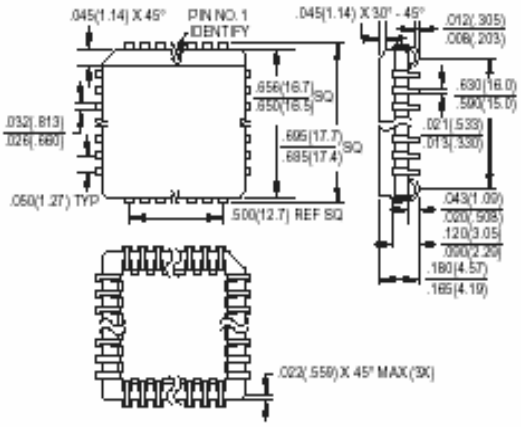
Packaging Information

44A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flatpack (TQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-026 ACB

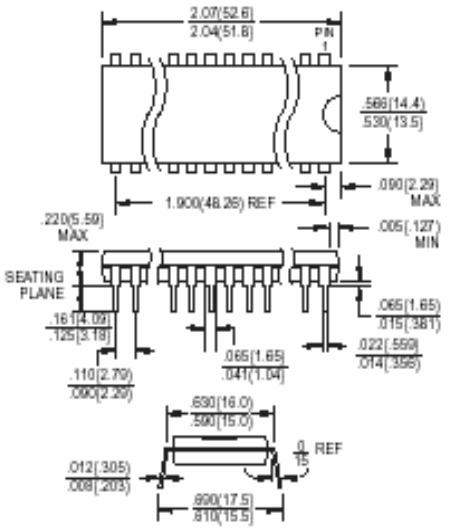


Controlling dimension: millimeters

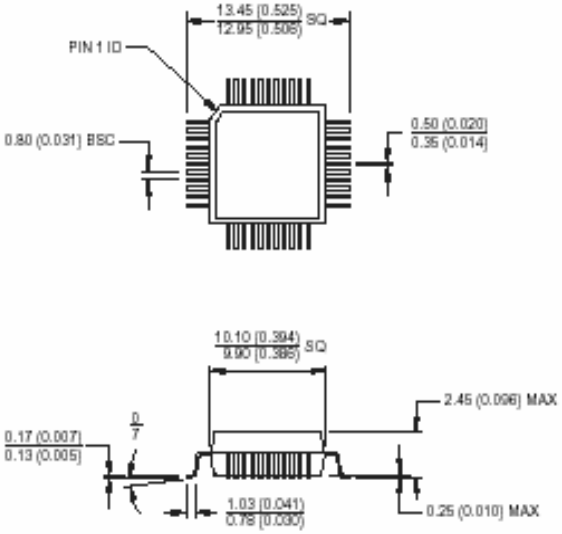
44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC



40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)



44Q, 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB



Controlling dimension: millimeters

Atmel Headquarters

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

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

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 Operations

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

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

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

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

0313H–02/00/xM

LAMPIRAN C
PDU (PROTOCOL DATA UNIT) FOR SIEMENSMOBILE PHONE

Developer's Guide:

SMS with the SMS PDU-mode

1. Introduction

To use the SMS you have to declare the number of the SMSC[1] (Short Message Service Center) in the MS (Mobile Station), provided that the MS support SMS-MO (Short Message Service-Mobile Orginated).
The SIEMENS **S25, SL10, S10**, **S10 active, E10, M1 Module** for example are providing SMS-MO.

card	SMSC-number (Germany)
D1	491710760000
D2	491722270000

At the MOBILE you enter the SMSC-number with the AT+Celular command:

at+csca = "<SMSC-number>"

If the receiver of the SMS possesses a D2 card, the AT command has to be entered in the following way:

at+csca = "+491722270000"

With the command

at+csca?

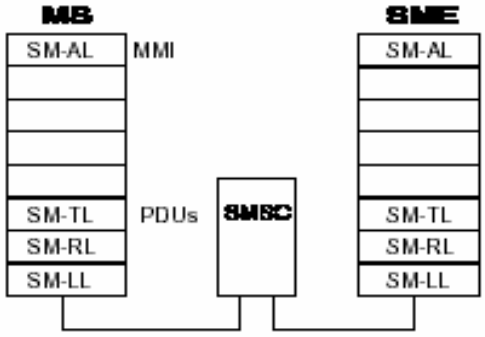
you can question the actual adjusted SMSC-number

Ask your network operator for the right SMSC-number !!

! notice: In addition to the AT+CSCA command it is possible to enter The SMSC-number in front of the Protocol Data Unit (PDU) see chapter 3.1!

[1] sometimes you can see the abbreviation SC (Service Center) that means the same as SMSC

2. Overview:



MS: Mobile Station
SME: Short Message Entity
SMSC: Short Message Service Center
MMI: Man Machine Interface
PDUs: Protocol Data Units
SM-AL: Short Message Aplication Layer
SM-TL: Short Message Transport Layer
SM-RL: Short Message Relay Layer
SM-LL: Short Message Link Layer

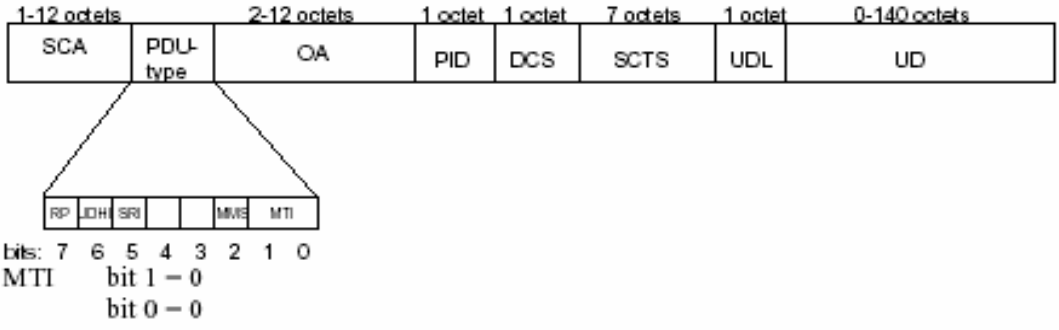
The MMI is based on the command set of AT+Cellular, and could be realized by means of a terminal (for example Triodata, Telix, WIN-Terminal) or the display of a handy.

The SM-TL provides a service to the Short Message Application Layer. This service enables the SM-AL to transfer short messages to its peer entity, receive short messages from its peer entity and receive reports about earlier requests for short messages to be transferred.
The SM-TL communicates with its peer entity with six several PDUs (Protocol Data Units):

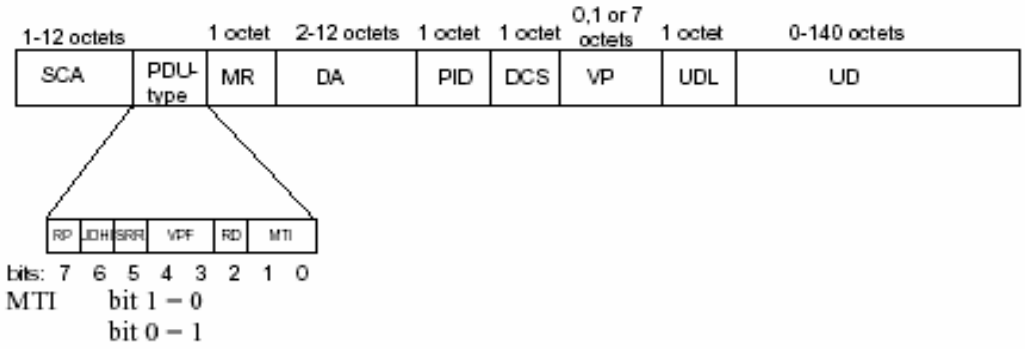
- **SMS-DELIVER, conveying a short message from the SMSC to the MS**
- SMS-DELIVER-REPORT, conveying a failure cause (if necessary)
- **SMS-SUBMIT, conveying a short message from the MS to the SMSC**
- SMS-SUBMIT-REPORT, conveying a failure cause (if necessary)
- SMS-STATUS-REPORT, conveying a status report from the SMSC to the MS
- SMS-COMMAND, conveying a command from the MS to the SMSC

The SMS-DELIVER and SMS-SUBMIT PDUs are described in the following sections.

2.1 SMS-DELIVER (Mobile Terminated)



2.2 SMS-SUBMIT (Mobile Originated)

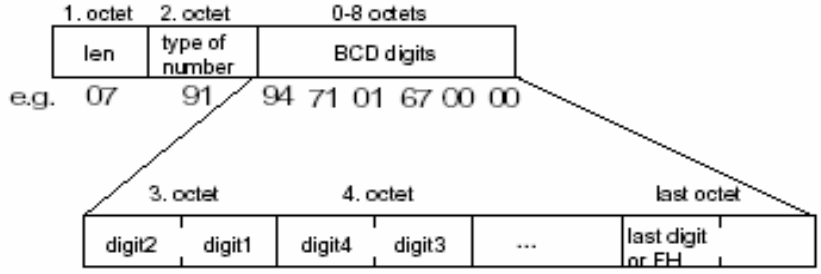


! notice: Any unused bits will be set to zero by the sending entity and will be ignored by the receiving entity !

SCA	Service Center Adress - information element	Telephone number of the Service Center
PDU Type	Protocol Data Unit Type	
MR	Message Reference	sucessive number (0..255) of all SMS-SUBMIT Frames set by the MOBILE
OA	Originator Adress	Adress of the originating SME
DA	Destination Adress	Adress of the destination SME
PID	Protocol Identifier	Parameter showing the SMSC how to process the SM (as FAX, Voice etc)
DCS	Data Coding Scheme	Parameter identifying the coding scheme within the User Data (UD)
SCTS	Service Center Time Stamp	Parameter identifying time when the SMSC received the message
VP	Validity Period	Parameter identifying the time from where the message is no longer valid in the SMSC
UDL	User Data Length	Parameter indicating the length of the UD-field
UD	User Data	Data of the SM
RP	Reply Path	Parameter indicating that Reply Path exists
UDHI	User Data Header Indicator	Parameter indicating that the UD field contains a header
SRI	Status Report Indication	Parameter indicating if the SME has requested a status report
SRR	Status Report Request	Parameter indicating if the MS has requested a status report
VPF	Validity Period Format	Parameter indicating whether or not the VP field is present
MMS	More Messages to Send	Parameter indicating whether or not there are more messages to send
RD	Reject Duplicate	
MTI	Message Type Indicator	Parameter describing the message type 00 means SMS-DELIVER 01 means SMS-SUBMIT

3. Parameter description

3.1 Service Center adress information element (SCA info element)



len:
The octet "len" contains the number of octets reqiured for the number of the Service Center plus the 1 byte „type of number"

type of number:
81H: the following number is national
91H: the following number international
(for further information see GSM 04.08 chapter 10.5.4.6)

octet:
One octet includes two BCD-digit Fields
If the called party BCD number contains an odd number of digits, the last digit shall be filled with an end mark coded as "FH"

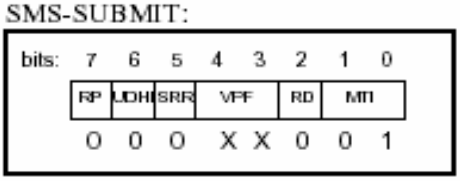
Example:
if you have the SC-number +49 171 0760000 you have to type:

0791947101670000

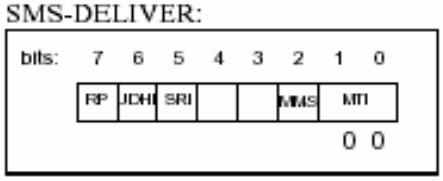
! notice: If the „len" field is set to Zero the MOBILE takes the default value of the Service Center adress set by the AT+CSCA command!

3.2 Protocol Data Unit Type (PDU Type)

SMS-SUBMIT:



SMS-DELIVER:



! notice: you have to write the PDU-type in Hex-Format, a possible example is "11H" !

RP: **0** **Reply Path parameter is not set in this PDU**
1 **Reply Path parameter is set in this PDU**

UDHI: **0** **The UD field contains only the short message**
1 **The beginning of the UD field contains a header in addition of the short message**

SRI: (is only set by the SMSC)
0 A status report will not be returned to the SME
1 A status report will be returned to the SME

SRR: **0** **A status report is not requested**
1 **A status report is requested**

VPF:

	bit4	bit3	
	0	**0**	**VP field is not present**
	0	1	Reserved
	1	**0**	**VP field present an integer represented (relative)**
	1	**1**	**VP field present an semi-octet represented (absolute)**

any reserved values may be rejected by the SMSC

MMS: (is only set by the SMSC)
0 More messages are waiting for the MS in the SMSC
1 No more messages are waiting for the MS in the SMSC

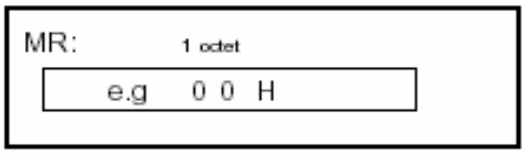
RD: **0** **Instruct the SMSC to accept an SMS-SUBMIT for an short message still held in the SMSC which has the same MR and DA as a previosly submitted short message from the same OA.**
1 **Instruct the SMSC to reject an SMS-SUBMIT for an short message still held in the SMSC which has the same MR and DA as a previosly submitted short message from the same OA.**

MTI:

bit1	bit0	Message type
0	**0**	**SMS-DELIVER (SMSC --> MS)**
0	**0**	**SMS-DELIVER REPORT** (MS --> SMSC, is generated automatically by the MOBILE, after receiving a SMS-DELIVER)
0	**1**	**SMS-SUBMIT (MS --> SMSC)**
0	1	SMS-SUBMIT REPORT (SMSC --> MS)
1	0	SMS-STATUS REPORT (SMSC --> MS)
1	**0**	**SMS-COMMAND (MS --> SMSC)**
1	1	Reserved

(the fat-marked lines represent the features supported by the MOBILE)
! notice: not every PDU Type is supported by the Service Center !

3.3 Message Reference MR

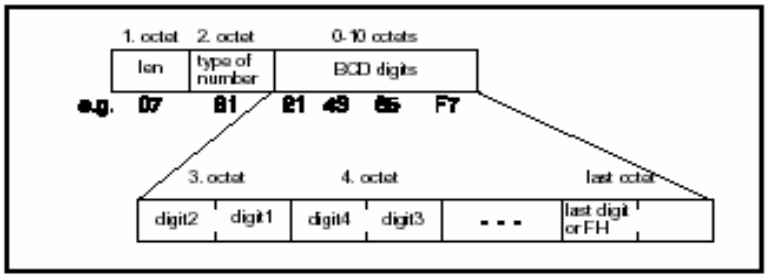


The MR field gives an integer (0..255) representation of a reference number of the SMS-SUBMIT submitted to the SMSC by the MS.

! notice: at the MOBILE the MR is generated automatically, -anyway you have to generate it-
a possible entry is for example "00H" !

3.4 Originator Adress OA Destination Adress DA

OA and DA have the same format explained in the following lines:



len:
The octet "len" contains the number of BCD digits

type of number:
81H: the following number is national
91H: the following number international
(for further information see GSM 04.08 chapter 10.5.4.6)

BCD-digits:
The BCD-digit Field contains the BCD-number of the Destination e.g. of the Originator
If the called party BCD number contains an odd number of digits, the last digit shall be filled with an end mark coded as "FH"

Example:
if you have the national number 1234567 you have to type:

0781214365F7

3.5 Protocol Identifier PID

PID:

0	0	H

The PID is the information element by which the Transport Layer either refers to the higher layer protocol being used, or indicates interworking with a certain type of telematic device. here are some examples of PID codings:

00H: The PDU has to be treat as a short message
41H: Replace Short Message Type1
42H: Replace Short Message Type2
43H: Replace Short Message Type3
.....
47H: Replace Short Message Type7

If „Replace Short Message Type x" is present, then the MS will check the associated SC address and originating address and replace any existing stored message having the same Protocol Identifier code, SC address and originating address with the new short message and other parameter values. If there is no message to be replaced, the MS shall store the message in the normal way.

(for further information see GSM 03.40 chapter 9.2.3.9)

! notice: it is not guaranteed that the SMSC supports every PID codings!

3.6 Data Coding Scheme DCS

bits:	7	6	5	4	3	2	1	0	
	Coding Group				0	x	x	x	
e.g.	0	0	0	0	0	0	0	0	= 00 H
	means: 7- bit data coding default alphabet								
e.g.	1	1	1	1	0	1	1	0	= F6 H
	means: 8-bit data coding Class 2								

The DCS field indicates the data coding scheme of the UD (User Data) field, and may indicate a message class. the octet is used according to a coding group which is indicated in bits 7..4. The octet is then coded as follows:

Coding group: bits 7..4	bits 3..0

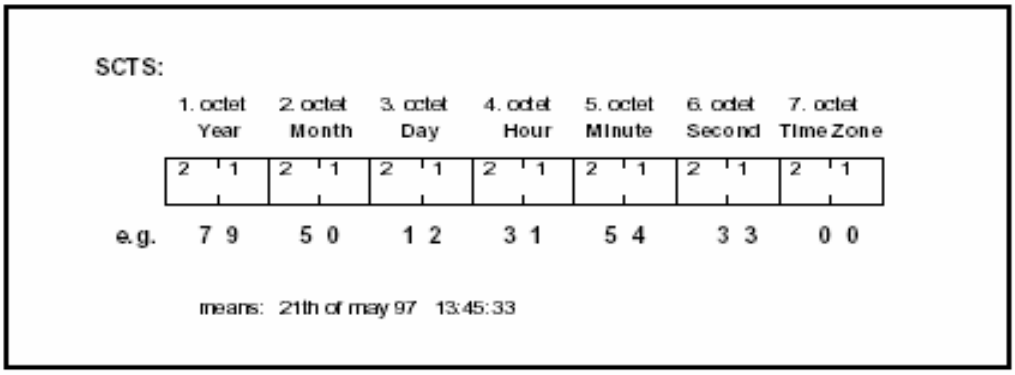
0000	**Alphabet indication** Unspecified message handling at the MS 0000 Default alphabet (7 bit data coding in the User Data) 0001-1111 reserved
0001-1110	Reserved coding groups
1111	**Data Coding/message class** **bit 3** is reserved, set 0 **bit 2** (message coding) **0** **Default alphabet (7 bit data coding in the User Data)** **1** **8-bit data coding in the User Data** **bit 1 bit 0** (message class) 0 0 Class0 immediate display **0 1 Class1 ME (Mobile Equipment)- specific** **1 0 Class2 SIM specific message** **1 1 Class3 TE (Terminate Equipment)- specific**

Default alphabet indicates that the UD (User Data) is coded from the 7-bit alphabet given in the appendix. When this alphabet is used, eight characters of the message are packed in seven octets, and the message can consist of up to 160 characters (instead of 140 characters in 8-bit data coding)
In 8-bit data coding, you can relate to the INTEL ASCII-HEX table.
In Class 0 (immediate display) the short message is written directly in the display, as the M1 has no display the Class 0 message can be realised only in a roundabout way.
In Class 1 to Class 3 the short message is stored in the several equipments ME, SIM-card and TE.
In time the Class 2 is supported, if you choose Class 1 or Class 3 the short message is treated the same way as a Class 2 message.

! note: It is recommended to use the Class2 message, or the coding group "0000 bin" !

3.7 Service Center Time Stamp SCTS

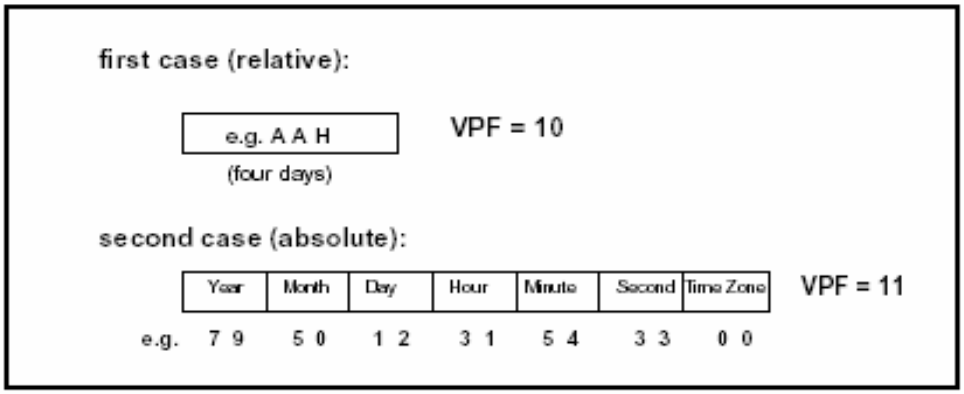
The SCTS is the information element by which the SMSC informs the recipient MS about the time of arrival of the short message at the Transport Layer entity of the SMSC. The time value is included in every SMS-DELIVER being delivered to the SMSC, and represents the local time in the following way:



The Time Zone indicates the difference, expressed in quarters of an hour, between the local time and GMT (Greenwich Main Time).

3.8 Validity Period VP

The Validity-Period is the information element which gives an MS submitting an SMS-SUBMIT to the SMSC the possibility to include a specific time period value in the short message. The Validity Period parameter value indicates the time period for which the short message is valid, i.e. for how long the SMSC shall guarantee its existence in the SMSC memory before delivery to the recipient has been carried out.



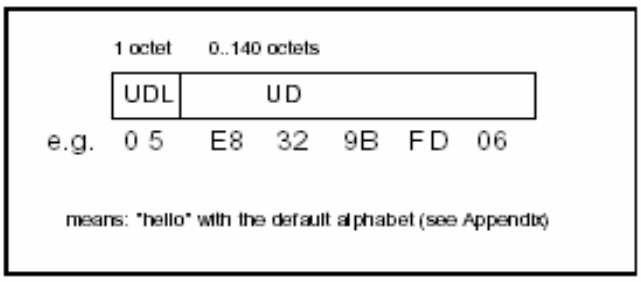
The VP field is given in either integer or semi-octet representation. In the first case, the VP comprises 1 octet, giving the length of the validity period, counted from when the SMS-SUBMIT is received by the SMSC. In the second case, the VP comprises 7 octets, giving the absolute time of the vality period termination.
In the first case, the representation of time is as follows:

VP Value	Validity period value
0-143	(VP + 1) x 5 minutes (i.e 5 minutes intervals up to 12 hours)
144-167	12 hours + ((VP-143) x 30 minutes)
168-196	(VP-166) x 1 day
197-255	(VP - 192) x 1 week

in the second case, the representation of time is identical to the representation or the SCTS (Service Center Time Stamp)

The case of representation is set in the VPF (Validity Period Format) in the PDU-type.

3.9 User Data Length UDL and User Data UD



The UDL field gives an integer representation of the number of characters within the User Data field to follow.

4. PDU Examples

here are two examples how to send a short message with AT+Cellular:

first enter PIN-number and the Service Center Adress:

at+cpin="XXXX"	*enter the PIN-number*
OK	
at+csca="+491722270000"	*enter the Service-Center-Adress (here D2)*
OK	

1st example:

at+cmgs=140	*enter "send message", 140 is the maximum length (in byte) of the following PDU*
> 00110007812143 65F70000AA05E8329BFD06	*type the PDU (SMS-SUBMIT) and finish with "ctrl Z" the thin-typed characters are the Destination Adress e.g. the own tel.-number the Service Center adress is the same as set via at+csca command*
+CMGS: 0	

OK

at+cpms? *are messages stored on the SIM-Card?*
+CPMS: "SM" , 1 , 7 , "SM" , 1 , 7 on this SIM-Card is 1 message stored
OK you can store at most 7 messages

at+cmgr=1 *read stored message in location 1*
+CMGR: 0 ,, 24
00040C9194718215219200006930824161840005E8329BFD06 This is a PDU (SMS-DELIVER) sent by the Service Center
OK

2nd example:

at+cmgw=140 *write message in the memory of the SIM-card*

> ***07919471016700001100078121 4365F700F6AA0568656C6C6F*** *type the PDU (SMS-SUBMIT) and finish with "ctrl Z" the thin-typed characters are the Destination Adress e.g. the own tel.-number. The Service Center Adress is „+491710760000"*
+CMGW: 2
OK

at+cmgr=2 *read stored message in location 2*
+CMGR: 2 ,, 17
0011000781214365F700F6AA0568656C6C6F this is the PDU stored in location 2
OK

at+cmss=2 *send the message stored in location 2*
+CMSS: 3
OK

at+cmss=2,"7654321",129 *send the message stored in location 2 to the national (129 = 81H) destination address „7654321"*

at+cmss=2,"+491717654321",145 *send the message stored in location 2 to the international (145 = 91H) destination address „+491717654321"*

at+cpms? *are messages stored on the SIM-Card?*
+CPMS: "SM" , 3 , 7 , "SM" , 3 , 7 on this SIM-Card are 3 message stored
OK you can store at most 7 messages

at+cmgr=3 *read stored message in location 3*
+CMGR: 0 ,, 24
00040C9194718215219200F6693082519472000568656C6C6F This is a PDU (SMS-DELIVER) sent by the Service Center
OK

5. Appendix

Default alphabet:

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	2	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	@	Δ	SP	0	-	P	¨	p
0	0	0	1	1			!	1	A	Q	a	q
0	0	1	0	2	$	Φ	"	2	B	R	b	r
0	0	1	1	3		Γ	#	3	C	S	c	s
0	1	0	0	4		Λ		4	D	T	d	t
0	1	0	1	5		Ω	%	5	E	U	e	u
0	1	1	0	6		Π	&	6	F	V	f	v
0	1	1	1	7		Ψ	'	7	G	W	g	w
1	0	0	0	8		Σ	(	8	H	X	h	x
1	0	0	1	9		Θ	)	9	I	Y	i	y
1	0	1	0	10	LF	Ξ	*	:	J	Z	j	z
1	0	1	1	11			+	;	K	Ä	k	ä
1	1	0	0	12			,	<	L	Ö	l	ö
1	1	0	1	13	CR		-	=	M		m	
1	1	1	0	14		ß	.	>	N	Ü	n	ü
1	1	1	1	15			/	?	O		o	

abbreviations:

MS	Mobile Station
SME	Short Message Entity
SMSC	Short Message Service Center
MMI	Man Machine Interface
PDUs	Protocol Data Units
SM-AL	Short Message Aplication Layer
SM-TL	Short Message Transport Layer
SM-RL	Short Message Relay Layer
SM-LL	Short Message Link Layer
PDU Type	Protocol Data Unit Type
MR	Message Reference
OA	Originator Adress
DA	Destination Adress
PID	Protocol Identifier
DCS	Data Coding Scheme
SCTS	Service Center Time Stamp
VP	Validity Period
UDL	User Data Length
UD	User Data
RP	Reply Path
UDHI	User Data Header Indicator
SRI	Status Report Indication
SRR	Status Report Request
VPF	Validity Period Format
MMS	More Messages to Send
RD	Reject Duplicate
MTI	Message Type Indicator
ME	Mobile Equipment
TE	Terminal Equipment
SIM	Subscriber Identity Modul

error codes:

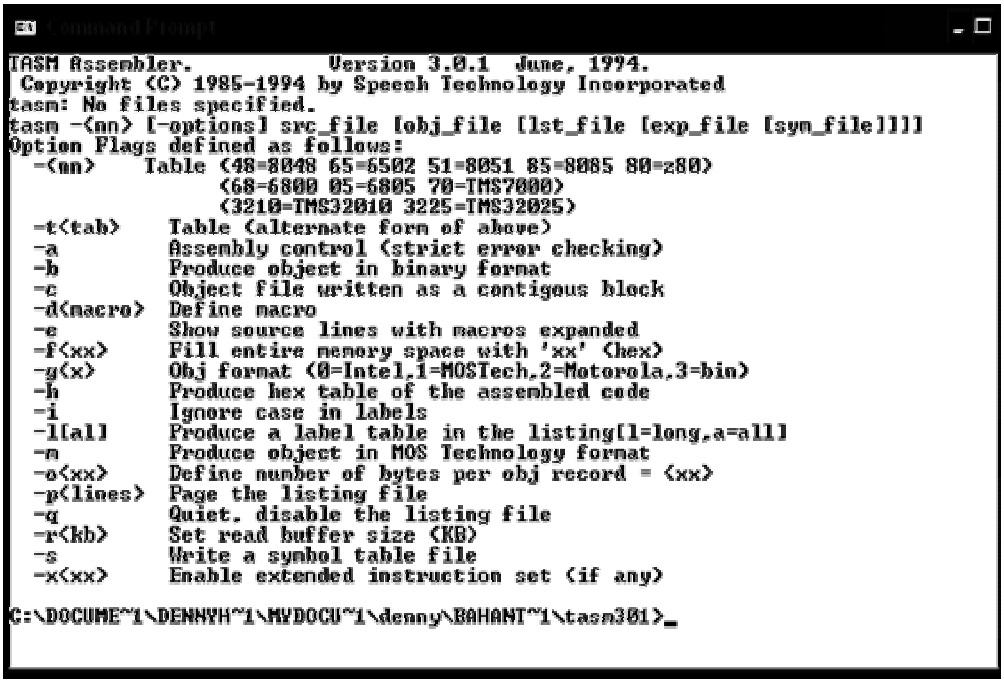
0	phone failure
1	no connection to phone
2	Phone-adaptor link reserved
3	operation not allowed
4	operation not supported
5	PH-SIM PIN necessary
10	SIM not inserted
11	SIM PIN required
12	SIM PUK required
13	SIM failure
14	SIM busy
15	SIM wrong
16	incorrect password
20	memory full
21	invalid index
22	not found
23	memory failure
24	text string too long (+CPBW)
25	invalid characters in text string
26	dial string to long
27	invalid characters in dial string
30	no network service
31	network timeout
100	unknown
265	PUK for theft protection necessary
266	PUK2 for SIM necessary
267	PIN2 for SIM necessary

LAMPIRAN D
PROSES PERANCANGAN PERANGKAT LUNAK DENGAN TASM301 DAN PENGISIAN MIKROKONTROLLER

Pengujian Perangkat lunak

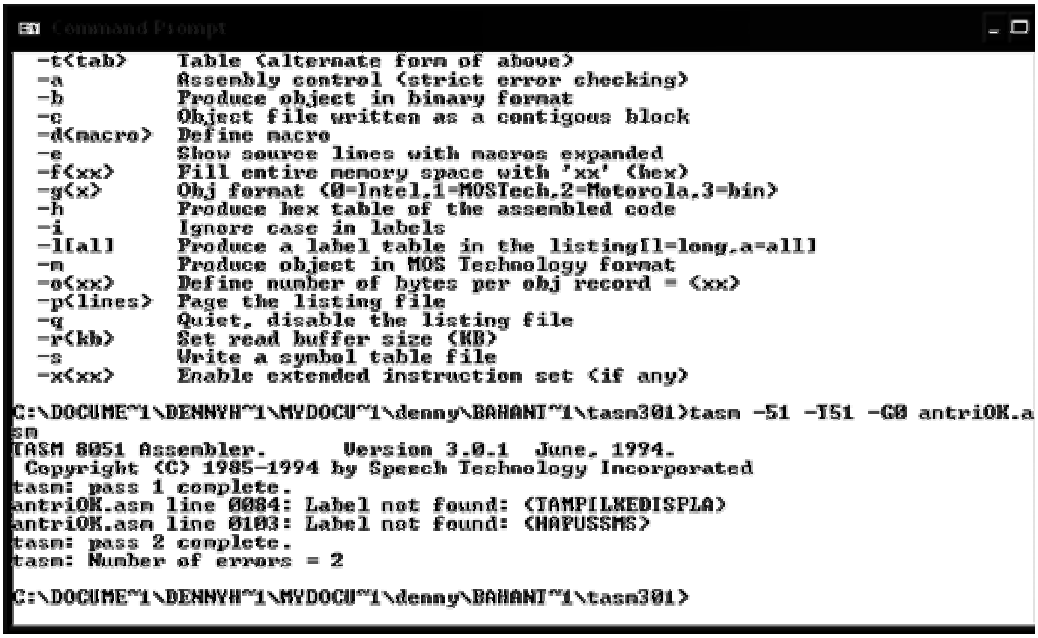
terhadap perangkat lunak yaitu menguji apakah perangkat lunak tersebut sesuai dengan kinerja *hardware* yang diinginkan atau tidak, oleh karena itu langkah selengkapnya pengujian perangkat lunak dapat diuraikan sebagai berikut:

1. Membuka MS-DOS prompt dengan bantuan program TASM301 bisa mengetahui apakah listing program yang kita buat itu benar atau salah yaitu dengan cara memanggil file program yang dibuat dengan catatan file yang dibuat harus satu folder dengan file TASM301 dan tipe file yang digunakan adalah ASM. Langkah ini juga merupakan langkah untuk meng-konversi file ASM kedalam bentuk file HEX. Tampilan program TASM301 diperlihatkan pada gambar dibawah ini:



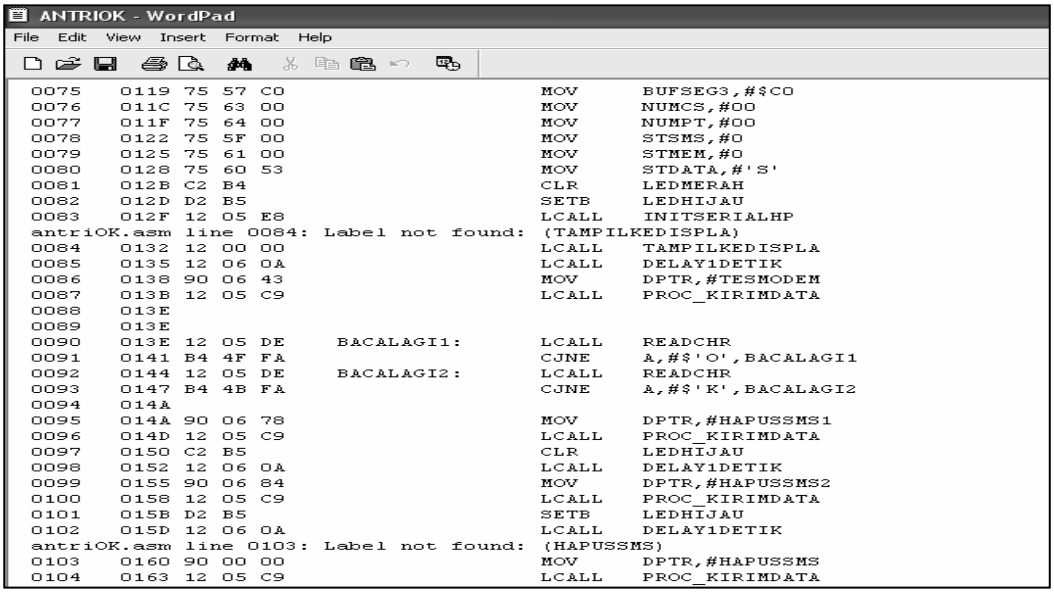
Gambar Tampilan Program TASM301

2. Jika terjadi kesalahan pada listng program yang dibuat maka akan tampil pada layar, seperti gambar dibawah ini:



Gambar Informasi error pada program TASM301

3. Jika terdapat kesalahan seperti gambar 4.6 maka selanjutnya membuka file dengan ekstensi LST, hal ini dilakukan untuk mengetahui letak kesalahan listing program yang buat dapat dilihat pada gambar dibawah ini.



Gambar Informasi letak kesalahan dalam file LST

4. Setelah letak kesalahan diketahui, maka selanjutnya yang harus dilakukan adalah memperbaiki kesalahan tersebut dengan cara mengedit file tersebut

dengan cara membuka kembali nama file dengan ekstensi ASM pada program text editor (notepad).

5. pada langkah nomor 1, dan ini terus menerus dilakukan sampai program menampilkan pesan tidak terdapat kesalahan (error) seperti yang ditunjukkan pada gambar dibawah ini :

Gambar Informasi eksekusi program tanpa error

6. Setelah program yang kita buat tersebut tidak ada kesalahan lagi, maka selanjutnya dilakukan proses pengisian IC mikrokontroler AT89C52 dengan program yang telah dibuat tersebut. Proses pengisian program ke dalam mikrokontroler, membutuhkan perangkat keras dan perangkat lunak yang berfungsi sebagai downloader. Sebagai perangkat keras digunakan downloader yang terhubung dengan serial port pada PC. Sedangkan perangkat lunak yang mendukung programer atau downloader yaitu EZ Uploader V3.0.

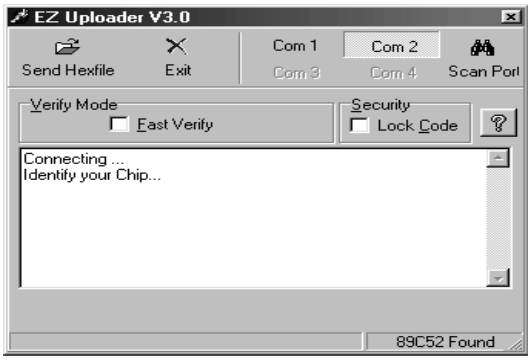
   Adapun langkah-langkah yang ditempuh untuk mengisikan program ke dalam IC mikrokontroler yaitu sebagai berikut :

a. Membuka program EZ Uploader V3.0 (gambar 4.9)

b. Pilih ada di Com berapa Programmer yang kita pasang.

c. Jika telah dipilih maka pada aplikasi ini akan tampil pesan Connecting yang dilanjutkan dengan pencaraian IC mikrokontroler yang terhubung dengan downloader.

d. Tunggu sampai aplikasi ini menampilkan pesan ditemukannya IC mikrokontroler yang akan kita isi program.

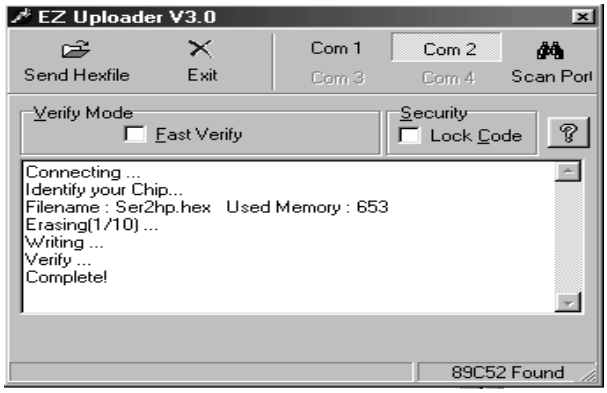


Gambar Program EZ Uploader

e. Panggil file HEX yang telah kita buat tersebut dengan cara pada aplikasi EZ Uploader V3.0 ini klik *send Hex File*, ditunjukan pada gambar di bawah ini.

Gambar Pemanggilan File Hex

f. Tunggu sampai pada aplikasi ini mengeluarkan pesan *Complete.* Ditunjukan pada gambar dibawah ini.



Gambar Pesan Complete pada EZ 3.0

g. Jika telah selesai maka berarti IC mikrokontroler telah terisi oleh program yang telah kita buat tadi dan siap untuk diaplikaskan dengan rangkaian sebenarnya.

h. Masukkan IC yang sudah terprogram pada rangkain *minimum system* modul yang telah kita rancang.

i. Operasikan alat tersebut sesuai dengan rancangan.

j. Bila alat tidak jalan, cek kembali apakah ada kesalahan di program atau perangkat keras, hingga alat dapat bekerja sesuai dengan yang telah kita rancang.

LAMPIRAN E

PRINT OUT NOMOR ANTRIAN

CONTOH PRINTOUT UNTUK PEMESANAN PELAYANAN CUSTOMER SERVICE (CS)

==========

NOMOR ANTRIAN
001A

==========

==========

NOMOR ANTRIAN
002A

==========

==========

NOMOR ANTRIAN
003A

==========

==========

NOMOR ANTRIAN
004A

==========

==========

NOMOR ANTRIAN
005A

==========

==========

NOMOR ANTRIAN
006A

==========

==========

NOMOR ANTRIAN
007A

==========

==========

NOMOR ANTRIAN
008A

==========

==========

NOMOR ANTRIAN
009A

==========

==========

NOMOR ANTRIAN
010A

==========

==========

NOMOR ANTRIAN
011A

==========

CONTOH PRINTOUT UNTUK PEMESANAN PELAYANAN PEMBAYARAN (PB)

==========

NOMOR ANTRIAN
001B

==========

==========

NOMOR ANTRIAN
002B

==========

==========

NOMOR ANTRIAN
003B

==========

==========

NOMOR ANTRIAN
004B

==========

==========

NOMOR ANTRIAN
005B

==========

==========

NOMOR ANTRIAN
006B

==========

==========

NOMOR ANTRIAN
007B

==========

==========

NOMOR ANTRIAN
008B

==========

==========

NOMOR ANTRIAN
009B

==========

==========

NOMOR ANTRIAN
010B

==========

==========

NOMOR ANTRIAN
011B

==========