8051 Serial Port Routines
by Kevin W. Clark

These files show a number of ways to use the serial port.

CHARNAME.A51	Equates for control characters
SERIO1A.A51	Auto baud rate using Timer 1, slow mode
SERIO1F.A51	Auto Baud using timer 1, fast mode
SERIO1M.A51	Manual Baud using Timer 1
SERIO110.A51	110 Baud using Timer 1
SERIO1_I.A51	Simple Interrupt driven sample.
SERIO2A.A51	Auto Baud using Timer 2
SERIO2M.A51	Manual Baud using Timer 2
SERIO_SP.A51	Split baud rate using both timer 1 and 2
SERTEST.A51	Master module for serial test program
MODE0.A51	A simple program to test MODE 0
BUILD.BAT	A simple batch file to assemble and link a SERIO file

the SERIO*.A51 files contain a number of routines:

S_INIT - Initializes Serial port.
C_IN - waits for character from serial port. Returns it in A.
C_OUT - sends character in A.
C_STS - console status. if char RXD, C=1, A=char, else C=0.
STROUT - Sends in-line character string to console. String is terminated by last character's MSB set. Hence, can only handle 7 bit ASCII.

The 8051 has 2 timers and can use only timer 1, operating as an 8 bit auto-reload timer to generate serial baud rates. The 8052 and supersets (80C52,'54, '58, 8051Fx,Gx ...) have three timers, and can use timer 2 and/or timer 1 to generate serial baud rates. Timer 2 is 16 bits and operates from a higher clock rate source than timer 1. This means that higher baud rates can be accurately generated over a wider range of oscillator frequencies than with timer 1. Lower baud rates can also be generated at higher oscillator frequencies.

The Auto Baud rate determining versions wait for a space character from the terminal, measuring its length, to determine the baud rate.

The manual-baud-rate programs are all written to generate 9600 baud from an 18.432 MHz crystal, except for the 110 baud example. Tables for other baud rates and crystals are given in the source files.

Timer 1 baud rate generation is controlled by the reload value of the timer, and also the value of the SMOD bit in the PCON SFR. If this bit is set, the baud rate is doubled. This accounts for the difference between the fast and slow mode examples.

Timer 1 can not normally be used to generate 110 baud at clock frequencies above 10MHz, due to being only 8 bits, but low baud rates can be generated by using it as a 16 bit timer with software reload on interrupt. This technique is used in SERIO110.A51, and is taken from the databook. This file also uses the serial port in mode 3 (9 bit), instead of mode 1 (8 bit), to generate the required 2 stop bits.

No examples are given for Mode 2, or for Mode 2 or 3 in the Multiprocessor communication mode, since these modes of operation need multiple processors and more complex software to demonstrate. Their

operation is otherwise similar to the standard modes of operation.

SERIO1_I.A51 is a simple interrupt driven example. It has only single character buffers for receive and transmit. A practical example of this would have larger buffers, and its design would depend more on the overall design of the program. For instance, the receive buffer might buffer a reasonable number of characters before the foreground program needed to read them, or may take care of all actions directly, and the transmit buffer would need to be big enough to hold a complete message, or make use of a queue of pointers to messages in XDATA or CODE space, to avoid the necessity of copying the strings from one place to another.

SERIO_SP.A51 is an example of split rate baud rate generation, using both Timer 1 and Timer 2. This example sets both to 9600 Baud.

MODE0.A51 is a simple demo of the use of Mode 0, the shift register mode. Again, real examples of this mode are dependent on the hardware configuration. With this example, you can look at the TX and RX pins with an oscilloscope to see the data and clock. Note that in Mode 0, the RXD pin (P3.0), which is normally an input, becomes an input/output for the data, while TXD (P3.1) provides the shift clock, so that in testing it for transmit mode, it must be disconnected from the serial line driver.

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
; Baud rate manually set to desired rate
; Using TIMER 1
; This version uses 1x baud rate
; This Version is interrupt driven on both the receive and transmit
ends.
;
;*****************************************
;
CODE_SEG    segment     code
data_seg    segment data
bit_seg           segment bit
;
public      S_init,C_IN,C_OUT,STROUT
;
cseg  at    sint
      ljmp  Ser_isr
;
      rseg  data_seg
rxbuf:      ds    1
txbuf:      ds    1

      rseg  bit_seg
txflag:     dbit  1
rxflag: dbit      1
txdone:     dbit  1
;
;Baud_Rate = Fosc/12 * (2^smod/32) / (256-TH1)
;     where 2^smod = 2 for smod=1, and 1 for smod=0
;     Fosc is oscillator frequency; and TH1 is timer 1 reload value.
;TH1 =  256 - (Fosc/12 * (2^smod/32) / Baud_Rate)
;
;Fosc =     12MHz 16MHz 20MHz 11.0592     14.7456     18.4320     smod
;Rate
;150  030H  --    --    040H  000H  --    0
;300  098H  075H  052H  0A0H  080H  060H  0
;600  0CCH  0BBH  0A9H  0D0H  0C0H  0B0H  0
;1200 0E6H  0DEH  0D5H  0E8H  0E0H  0D8H  0
;2400 0F3H  0EFH  0EAH  0F4H  0F0H  0ECH  0
;4800 *     *     0F5H  0FAH  0F8H  0F6H  0
;9600 --    --    *     0FDH  0FCH  0FBH  0
;19200      --    --    --    0FDH  0FCH  0FBH  1
;38400  --  --    --    --    0FEH  --    1
```

```
;76800      --    --    --    --    0FFH  --    1
;* These baud rates available by using the previous value, and setting
SMOD=1
BaudLoad    equ   0f6h  ;9600 baud @ 18.432
;*****************
;
      rseg  CODE_SEG
;
;*****************
; Serial Interrupt service routine.
;  This is a simple example of a serial interrupt service routine,
;  with only a 1 character buffer each for
;  reception and transmission.  A practical program would use a receive
;  buffer large enough to hold received characters without danger of
overflow
;  before being serviced,  and a transmit buffer large enough to hold
the
;  largest string to be sent, or some other buffering scheme, like
perhaps
;  a queue of pointers to constant strings and/or string buffers.
;
Ser_isr:
      jbc   ri,RXisr    ; incoming character?
TXisr:                          ; else a TX interrupt
      jbc   txflag,sendit     ; if no character waiting
      clr   ti          ;   then we just finished
      setb  txdone
      sjmp  ser_ret

sendit:     mov   sbuf,txbuf
      clr   ti
      clr   txdone
      sjmp  ser_ret
;
RXisr:      mov   rxbuf,sbuf  ; read incoming character
      setb  rxflag      ;  set received flag
ser_ret:
      reti
;*****************
S_INIT:
      CLR   TR1
      CLR   TXFLAG
      clr   rxflag
      setb  txdone
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN
      MOV   TMOD,#00100001B   ;Timer 1 is set to 8 bit auto reload mode
      orl   PCON,#SMOD  ; Set to double rate.
      mov   th1,#BaudLoad     ; Set reload value
      setb  tr1         ; start timer.
      setb  es          ; enable serial interrupts!
      setb  ea          ; Enable them all!
      ret
;
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
```

```
;     code in accumulator.
;
        JNB     RXFLAG,$        ;Wait until character received.
        MOV     A,RXBUF         ;Read input character.
      CLR   RXFLAG            ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
      JB    TXFLAG,$    ; not too hot, because it's blocking.

      MOV   TXBUF,A           ;Write out character.
      SETB  TXFLAG
      JNB   TXDONE,CORET
      SETB  TI
CORET:      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RXFLAG
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
;Automatically sets baud rate, using TIMER1
; This version uses 1x baud rate
;
;*****************************************
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;
      rseg  CODE_SEG
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN,
      MOV   TMOD,#00100001B   ;Timer 1 is set to auto-reload timer mode.
      ANL   PCON,#not SMOD    ; Set SMOD to single baud rate

;                ; wait for <space> from console, measure its
;                ; speed, and set baud rate in Timer 1 accordingly.
BAUDDET:
      MOV   TH1,#0      ;Assume fastest rate.
      MOV   R0,#144
      JB    RXD,$ ; wait for start bit
BAUDID:     DJNZ  R0,$
      DEC   TH1
      MOV   R0,#94
      JNB   RXD,BAUDID
      JB    RXD,$ ;Hang-up here until space char. over.
      JNB   RXD,$
      SETB  TR1
      ret
;
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
```

```
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;****************************************
;
;Automatically sets baud rate, using TIMER1
; This version uses 2X baud rate
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;
      rseg  CODE_SEG
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN,
      MOV   TMOD,#00100001B   ;Timer 1 is set to auto-reload timer mode.
      ORL   PCON,#SMOD  ; Set SMOD to double baud rate

;                ; wait for <space> from console, measure its
;                ; speed, and set baud rate in Timer 1 accordingly.
BAUDDET:
      MOV   TH1,#0      ;Assume fastest rate.
      MOV   R0,#72      ; wait 1.5 bit times.
      JB    RXD,$ ; wait for start bit
BAUDID:     DJNZ  R0,$
      DEC   TH1
      MOV   R0,#46      ; wait 1 bit time.
      JNB   RXD,BAUDID
      JB    RXD,$ ;Hang-up here until space char. over.
      JNB   RXD,$
      SETB  TR1
      ret
;
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
      JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
```

```
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
      JNB     TI,$            ;Wait until transmission completed.
      CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
; Baud rate manually set to desired rate
; Using TIMER 1
; This version uses 1x baud rate
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;public     baudload
;
      rseg  CODE_SEG
;
;Baud_Rate = Fosc/12 * (2^smod/32) / (256-TH1)
;     where 2^smod = 2 for smod=1, and 1 for smod=0
;     Fosc is oscillator frequency; and TH1 is timer 1 reload value.
;TH1 =  256 - (Fosc/12 * (2^smod/32) / Baud_Rate)
;
;Fosc =     12MHz 16MHz 20MHz 11.0592     14.7456     18.4320     smod
;Rate
;150  030H  --    --    040H  000H  --    0
;300  098H  075H  052H  0A0H  080H  060H  0
;600  0CCH  0BBH  0A9H  0D0H  0C0H  0B0H  0
;1200 0E6H  0DEH  0D5H  0E8H  0E0H  0D8H  0
;2400 0F3H  0EFH  0EAH  0F4H  0F0H  0ECH  0
;4800 *     *     0F5H  0FAH  0F8H  0F6H  0
;9600 --    --    *     0FDH  0FCH  0FBH  0
;19200      --    --    --    0FDH  0FCH  0FBH  1
;38400  --  --    --    --    0FEH  --    1
;76800      --    --    --    --    0FFH  --    1
;* These baud rates available by using the previous value, and setting
SMOD=1
BaudLoad    equ   0F6h  ;9600 baud @ 18.432
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN
      MOV   TMOD,#00100001B   ;Timer 1 is set to 8 bit auto reload mode
      orl   PCON,#SMOD  ; Set to double rate.
      mov   th1,#BaudLoad     ; Set reload value
      setb  tr1         ; start timer.
      ret
;
;=======
```

```
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
; Automatically sets baud rate, using TIMER 2
;
;*****************************************
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;
      rseg  CODE_SEG
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN,
      MOV   T2CON,#00110000B; Set to RCLK, TCLK
;                 ; wait for <space> from console, measure its
;                 ; speed, and set baud rate in Timer 2 accordingly.
Bauddet:
;
BG1:    CLR     A               ;DO BAUD RATE
        MOV     R3,A
        MOV     R1,A
        MOV     R0,#4
        JB      RXD,$           ;LOOP UNTIL A CHARACTER IS RECEIVED
        ;
BG2:    DJNZ    R0,$            ;Waste 8 clocks initially, 6 in loop
      CLR   C           ;1 clock
      MOV   A,R1        ;1
      SUBB  A,#1        ;1 dec(R3:R1)
      MOV   R1,A        ;1
      MOV   A,R3        ;1
      SUBB  A,#00H            ;1
      MOV   R3,A        ;1
        MOV     R0,#3           ;1
        JNB     RXD,BG2         ;2 CLOCKS, LOOP UNTIL DONE
        JB      RXD,$           ;WAIT FOR SPACE TO END
        JNB     RXD,$         ; WAIT FOR STOP BIT
        MOV RCAP2H,R3   ; LOAD THE TIMER2 REGISTER.
      MOV   RCAP2L,R1
      SETB  TR2         ; turn it on
        ;
      ret
;
```

```
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port. Baud rate manually set to desired
rate
;     using timer 2
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;public     baudload
;
      rseg  CODE_SEG
;
;Baud_Rate = ( Fosc / 32 ) / (65536 - (RCAP) )
;     Where RCAP = RCAP2H,RCAP2L, taken as a 16 bit unsigned integer.
;RCAP = 65536 - ( Fosc / 32 ) / Baud_Rate
;
;Fosc =     12MHz 16MHz 20MHz 11.0592     14.7456     18.4320
;Rate
;110  -3409 -4545 -5682 -3142 -3994 -5236
;150  -2500 -3333 -4167 -2304 -3072 -3840
;300  -1250 -1667 -2083 -1152 -1536 -1920
;600  -625  -833  -1042 -576  -768  -960
;1200 -312  -417  -521  -288  -384  -480
;2400 -156  -208  -260  -144  -192  -240
;4800 -78   -104  -130  -72   -96   -120
;9600 -39   -52   -65   -36   -48   -60
;19200      ---   -26   -33   -18   -24   -30
;38400  --- -13   -16   -9    -12   -15
;57600      ---   ---   ---   -6    -8    -10
;76800      ---   ---   ---   ---   -6    ---
;115200 ---     ---     ---   -3    -4    -5
;
BaudLoad    equ   -60   ;9600 at 18.432MHz
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN
      MOV   T2CON,#00110000B; Initialize Timer 2 as Baud Rate generator
;     ANL   PCON,#not SMOD    ; Set SMOD to single baud rate
      orl   PCON,#SMOD  ; Set to double rate.
      mov   rcap2h,#High BaudLoad   ; Set reload value
      mov   rcap2l,#low Baudload
      setb  tr2         ; start timer.
      ret
;
```

```
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;Baud rate manually set to 110 baud
;  using timer 1, by running timer 1 as a 16 bit timer, and using
;  Timer 1 interrupt to reload the timer.
;This version uses 1x baud rate
;*****************************************
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;public     baudload
;
cseg  at    timer1
T1Int:
      mov   th1,#high baudload
      mov   tl1,#low baudload
      reti
;
      rseg  CODE_SEG
;
;BaudLoad   equ   0FEEBH      ; 12 MHz, 110 baud, from handbook
;BaudLoad   equ   0FEE3H      ; 12 MHz, 110 baud, correct value
;BaudLoad   equ   0FE84H      ; 16 MHz, 110 baud
;BaudLoad   equ   0FEFAH      ; 11.0592 MHz, 110 baud
;BaudLoad   equ   0FEA2H      ; 14.7456 MHz, 110 baud
BaudLoad    equ   0FE4BH      ; 18.4320 MHz, 110 baud

;
S_INIT:
      CLR   TR1
      MOV   SCON,#11011010B   ;TI set indicates transmitter ready.
                        ; mode 3(9 bit),REN, bit 9 = 1 (stop bit).
      MOV   TMOD,#00010001B   ;Timer 1 is set to 16 bit timer mode.
      ANL   PCON,#not SMOD    ; Set SMOD to single baud rate
      mov   th1,#high BaudLoad
      mov   tl1,#low baudload
      setb  et1
      setb  tr1
      setb  ea
      ret
;
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
```

```
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A            ;Write out character.
      RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51    symbols     debug nolist
$INCLUDE(\i\rg51fb.pdf)
$INCLUDE(charname.a51)
$list
;
;     CONSOLE I/O ROUTINES AND DRIVERS:
;     ======= === ======== === =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD, C=1, A=char, else C=0.
; STROUT - Sends in-line character string to console. String is
terminated
;     by last character's MSB set.  Hence, can only handle 7 bit ASCII.
;
;*****************************************
;
; Split Baud rate manually set
;  using Timer 2 for XMIT and Timer 1 for RCV.
;
;*****************************************
;
CODE_SEG    segment     code
public      S_init,C_IN,C_OUT,STROUT
;public     baudload
;
      rseg  CODE_SEG
;
;BAUD rate table for Timer 1
;===========================
;Baud_Rate = Fosc/12 * (2^smod/32) / (256-TH1)
;     where 2^smod = 2 for smod=1, and 1 for smod=0
;     Fosc is oscillator frequency; and TH1 is timer 1 reload value.
;TH1 =  256 - (Fosc/12 * (2^smod/32) / Baud_Rate)
;
;Fosc =     12MHz 16MHz 20MHz 11.0592     14.7456     18.4320     smod
;Rate
;150  030H  --    --    040H  000H  --    0
;300  098H  075H  052H  0A0H  080H  060H  0
;600  0CCH  0BBH  0A9H  0D0H  0C0H  0B0H  0
;1200 0E6H  0DEH  0D5H  0E8H  0E0H  0D8H  0
;2400 0F3H  0EFH  0EAH  0F4H  0F0H  0ECH  0
;4800 *     *     0F5H  0FAH  0F8H  0F6H  0
;9600 --    --    *     0FDH  0FCH  0FBH  0
;19200      --    --    --    0FDH  0FCH  0FBH  1
;38400  --  --    --    --    0FEH  --    1
;76800      --    --    --    --    0FFH  --    1
;* These baud rates available by using the value above, and setting
SMOD=1
Baud1Load   equ   0f6h  ; 18.432MHz, 9600, 2x
;
;*********************************************
;
;BAUD rate table for Timer 2
;===========================
;Baud_Rate = ( Fosc / 32 ) / (65536 - (RCAP) )
;     Where RCAP = RCAP2H,RCAP2L, taken as a 16 bit unsigned integer.
;RCAP = 65536 - ( Fosc / 32 ) / Baud_Rate
;
```

```
;Fosc =     12MHz 16MHz 20MHz 11.0592     14.7456     18.4320
;Rate
;110  -3409 -4545 -5682 -3142 -3994 -5236
;150  -2500 -3333 -4167 -2304 -3072 -3840
;300  -1250 -1667 -2083 -1152 -1536 -1920
;600  -625  -833  -1042 -576  -768  -960
;1200 -312  -417  -521  -288  -384  -480
;2400 -156  -208  -260  -144  -192  -240
;4800 -78   -104  -130  -72   -96   -120
;9600 -39   -52   -65   -36   -48   -60
;19200      ---   -26   -33   -18   -24   -30
;38400  --- -13   -16   -9    -12   -15
;57600      ---   ---   ---   -6    -8    -10
;76800      ---   ---   ---   ---   -6    ---
;115200 ---     ---     ---   -3    -4    -5
;
Baud2Load   equ   -60   ; 18.432 MHz, 9600
;
S_INIT:
      CLR   TR1
      MOV   SCON,#01011010B   ;TI set indicates transmitter ready.
                        ; mode 1,REN
      MOV   T2CON,#00010000B; Initialize Timer 2 as Baud Rate generator
                        ; for transmit only
;     ANL   PCON,#not SMOD    ; Set SMOD to single baud rate
      orl   PCON,#SMOD  ; Set to double rate.
      mov   rcap2h,#High Baud2Load  ; Set reload value
      mov   rcap2l,#low Baud2Load
      MOV   TMOD,#00100001B   ;Timer 1 is set to 8 bit auto reload mode
      orl   PCON,#SMOD  ; Set to double rate.
      mov   TH1,#baud1load
      setb  tr2         ; start timer.
      setb  tr1
            ret
;
;=======
;
C_IN:
;     Console character input routine.
;     Waits for next input from console device and returns with
character
;     code in accumulator.
;
        JNB     RI,$            ;Wait until character received.
        MOV     A,SBUF          ;Read input character.
      CLR   RI          ;Clear reception flag.
      ret               ;
;
;=======
;
C_OUT:
;     Console character output routine.
;     Outputs character received in accumulator to console output
device.
;
        JNB     TI,$            ;Wait until transmission completed.
        CLR     TI              ;Clear interrupt flag.
      MOV   SBUF,A           ;Write out character.
      RET
;
```

```
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and character in A
; Note: Serial input status can be checked by RI bit, also.
;
C_STS:      MOV   C,RI
      JNC   CNTRET            ;Poll whether character has been typed.
      CALL  C_IN
CNTRET:     RET
;
;=======
;
;STROUT
;     Copy in-line character string to console output device.
;     uses: DPTR,ACC
;
STROUT:     POP   DPH         ;get in-line string address from stack
      POP   DPL
STRO_1:     CLR   A
      MOVC  A,@A+DPTR   ;Read next byte.
      INC   DPTR        ;Bump pointer.
      JBC   ACC.7,STRO_2      ;Escape after last character.
      CALL  C_OUT       ;Output character.
      SJMP  STRO_1            ;Loop until done.
;
STRO_2:     CALL  C_OUT       ;Output character.
      CLR   A
      JMP   @A+DPTR           ;Return to program.
;
      end
```

```
$nomod51
$symbols
$debug
$nolist
$include(\i\rg51fb.pdf)
$list
;*******************************************
;
;
;
;
;*******************************************
;
;
$include(charname.a51)
;
EXTRN CODE  (S_init,C_IN,C_OUT,STROUT)
;
;data_seg   segment     data
code_seg    segment     code
?stack            segment idata
;
rseg  ?stack      ; This will allow RL51 to automatically place the
stack
stack:      ds    1
;
;
cseg at reset
      ljmp  start
;reserve interrupt vectors
;
cseg at exti0
;
cseg at timer0
;
cseg at exti1
;
cseg at timer1
;
rseg  code_seg
;
;
Start:      mov   sp,#stack
      call  s_init
      nop
      CALL  STROUT
        DB      CRet,'This is a test of the 8051 Serial port.'
      db    CRet,(LF OR 80H)
ECHO:
      call  C_IN
      call  C_OUT
      jmp   ECHO
;
      end
```