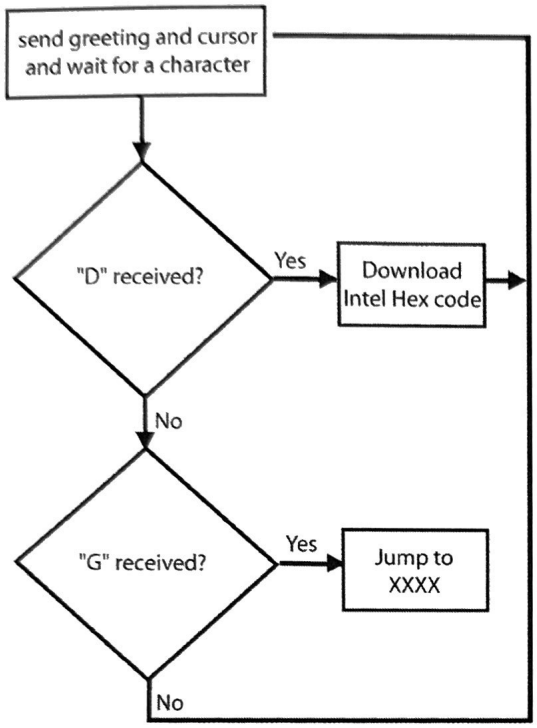
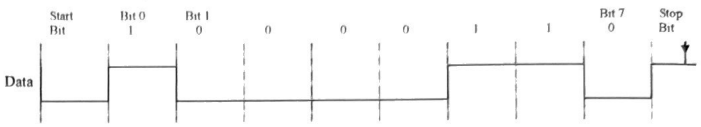
Lecture 3

Previously, we've examined the idea of operating the 8051 processor with a common linear memory space for code and data. We need an "operating system":



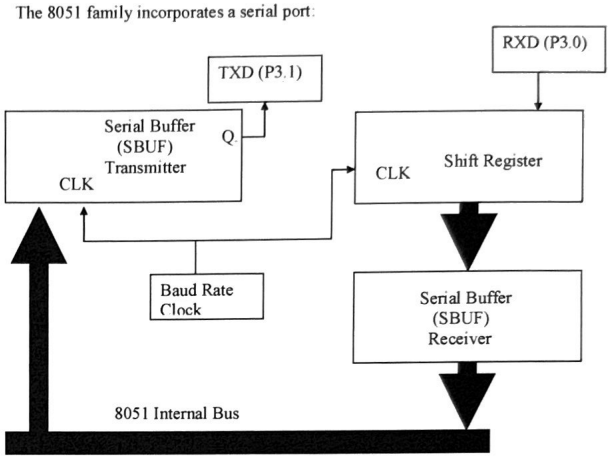
To develop this OS, we need a scheme for communicating with a personal computer/terminal.

RS232: (with asynchronous 8 bit serial data transmission)



For this pattern, the transmitted data is 61h. Is bit 1 the LSB or the MSB?

The 8051 family incorporates a serial port:



Data is transmitted and received using the serial port through the SBUF SFR. The behavior of the serial port is controlled or programmed through the SCON SFR:

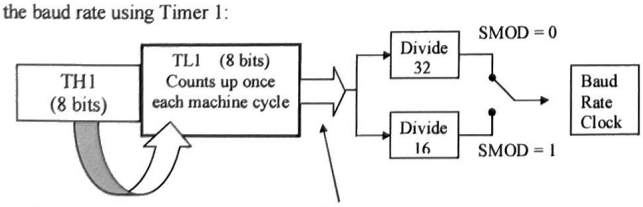
SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

We'll want to transmit 7 bit ASCII codes back and forth from the PC. So, we'll need to configure the serial port:

SM0	SM1	
0	0	Mode 0: Shift register, fixed clock
0	1	Mode 1: 8-bit UART, Baud rate set by Timer 1
1	0	Mode 2: 9-bit UART, Fixed baud rate (set by oscillator frequency)
1	1	Mode 3: 9-bit UART, Variable baud rate set by Timer 1

For ASCII transmissions without error correction, Mode 1 will be fine.

Set the baud rate using Timer 1:



When TL1 overflows, "auto re-load" the 8 bit TL1 counter register with the byte in TH1.

One pulse here every time TL1 overflows. (FFh to 00h transition)

What's the baud rate?

$$\text{Baud Rate} = \frac{\text{Clock Crystal}*2^{(\text{SMOD})}}{12*32*(256 - \text{TH1})}$$

So, for instance, pick TH1 = -3, i.e., TH1 = FDh, for a baud rate of 9600 with an 11.0592 crystal.

Typically can tolerate up to 4% error in baud rate.

TMOD Register

Time/Counter 1				Timer/Counter 0			
GATE#	C/T#	M1	M0	GATE#	C/T#	M1	M0

M0 and m1 (bits 0 and 1 of TMOD, respectively): *Mode Select*

The 2-bit field (M1, M0) selects one of four modes. Mode 0 is a 13-bit counter. An interrupt is generated when the counter overflows. Thus it takes 2^13 or 8192 input pulses to generate the next interrupt. Mode 1, similar to Mode 0, implements a 16-bit counter. It takes 2^16 or 65,526 input pulses to generate the next interrupt. Mode 2 operates in an 8-bit reload fashion. TLi serves as an 8-bit timer/counter. When the counter overflows, the number stored in THi is copied into TLi and the count continues. An interrupt is generated each time the counter overflows and a reload is performed. In Mode 3, Time 1 is inactive and simply holds its count. Timer 0 control bits and generates a Timer 0 interrupt at overflow. TH0 operates as a time driven by the system clock, prescaled by 12, and causes a Timer 1 interrupt at overflow.

C/T# (bit 2): *Conter/Timer Select*

When set, the timer/counter operates as a counter. The count increment is caused by external pulses through the T0 pin—the alternative function of P3.4. When cleared the timer/counter operates as a timer. The count increment is caused by every 12th system clock pulse. That is, the input to the counter is the system clock prescaled by 12.

GATE# (bit 3): *Gate*

Provided that TR0 of TCON (see next page) is set, clearing GATE enables (starts) counter/timer operation. If GATE is set, again provided that TR0 is set, the timer/counter operation is enabled if INT0# is high.

TCON Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0 (bit 0): *Interrupt 0 Type*
External interrupt 0 is received through bit 2 of Port 3 as an alternative function assignment. Setting IT0 causes the INT0# to be recognized at the falling edge of the input signal. Clearing IT0 causes an interrupt to be generated when the external signal is low, If the external signal stays low, INT0 will be generated over and over if IT0 is cleared but will not be generated if IT0 is set. IT0 is completely under software control.

IE0 (bit 1): *Interrrupt 0 Edge Flag*
Set by hardware when an external interrupt edge is detected. Cleared when a Return From Interrupt (RETI) instruction is executed.

IT1 (bit 2): *Interrupt 1 Type*
IT1 is associated with External Interrupt 1. Its function is similar to that of IT0 as described above.

IE1 (bit 3): *Interrupt 1 Edge Flag*
IE 1 is associated with External Interrupt 1. Its function is similar to that of IE0 as described above.

TR0 (bit 4): *Timer 0 Run Control Bit*
Timer/Counter 0 is disabled when TR0 is cleared. Setting TR0 is necessary but not sufficient to enable Timer/Counter 0 (see GATE# AND INT0#). TR0 is completely under software control.

TF0 (bit 5): *Timer 0 Overflow Flag*
TF0 is set by hardware when Timer/Counter 0 overflows. TF0 is cleared by hardware when the processor branches to the associate interrupt service routine. TF0, along with IE0, may be used to software to determine the state of the timer.

TR1 (bit 6): *Timer 1 Run Control Bit*
TR1 is associated with Timer/Counter 1. Its function is similar to that of TR0 described above.

TF1 (bit 7): *Timer 0 Overflow Flag*
TF1 is associated with Timer/Counter 1. Its function is similar to that of TF0 described above.

EXAMPLE: Let's set the serial port to operate at 2400 baud, 8 bits, using Timer 1 to provide the baud rate clock. Assume 11.0592 crystal. We need to set up a total of 4 registers!

Baud rate generation:

TMOD: set to #00100000b = #20h
TCON: set to #01000000b = #40h

After computation,

TH1: set to #-12 = #F4h (assuming SMOD = 0)

Finally, fire up the serial port:

SCON: set to #01010000b = #50h

How might we actually set these registers in assembly language?
Here's a code segment:

```
Init:
  MOV TMOD, #20h
  MOV TCON , #40h
  MOV TH1, #-12
  MOV SCON, #50h
  RET
```

In your MINMON handout, you'll find an initialization subroutine for 9600 baud:

```
;=============================================================
; subroutine init
; this routine initializes the hardware
;=============================================================
init:
; set up serial port with a 11.0592 MHz crystal,
; use timer 1 for 9600 baud serial communications
   mov   tmod, #20h        ; set timer 1 for auto reload - mode 2
   mov   tcon, #41h        ; run counter 1 and set edge trig ints
   mov   th1,  #0fdh       ; set 9600 baud with xtal=11.059mhz
   mov   scon, #50h        ; set serial control reg for 8 bit data
                           ; and mode 1
   ret
```