

```

;*****
;NAME: AUTOBAUD
; Initialize timer2 for serial clock. Initialize
; serial port. Expect to receive a space char (20H)
; Thus, measure width of six zero bits.
; Check for valid baud rate.
;REGS: R0, R1, R3
;-----
AUTOBAUD:
    MOV     T2CON, #34H      ; Use Timer 2 for serial clock.
    CLR     ET2

    MOV     A, #100         ; setup count for 30 seconds
    MOV     B, #30
    MUL     AB
    CALL    TIME

;
AUTO1:
    JNB     RXD, AUTO2      ; Wait for start bit
    JB      TIME_FLG, AUTO1
    JMP     SLEEP           ; power down

;
; start bit found so start timer to measure its duration
;
AUTO2:
    CLR     EA              ; disable interrupts
    CLR     TR1             ; disable timer 1
    MOV     TH1, #00h       ; clear timer 1
    MOV     TL1, #00h
    JNB     RXD, $          ; wait for next RXD going high
    JB      RXD, $          ; wait for next RXD going low
    SETB    TR1            ; enable the TIMER 1 running
    JNB     RXD, $          ; wait for next RXD going high
    CLR     TR1            ; stop the timer
    SETB    EA             ; re-allow interrupts

;
; TH1::TL1 have the pulse period in there. Provided the PC user has
; sent a space bar 0x20 this number will be directly related to the
; baud rate divisor needed according to the formula:
;
; DIVISOR = 65536 - ((TH1:TL1 * 16)/84)
;
; See the spreadsheet AutoBaud.XLS with the source code for a full
; analysis of the computation.
;
    MOV     R1, TH1         ; get timer value
    MOV     R0, TL1
    MOV     R3, #0          ; set up multiplier 16
    MOV     R2, #16D
    CALL    UMUL16         ; multiply timer * 16
    MOV     R3, #0         ; setup divisor of 84
    MOV     R2, #84D
    CALL    UDIV16         ; divide result to R1:R0

;
    MOV     A, R0           ; complement for up counter
    CPL     A
    ADD     A, #1
    MOV     R0, A
    MOV     A, R1
    CPL     A
    ADDC    A, #0
    MOV     R1, A

;
    MOV     RCAP2H, R1     ; program the baud rate divisor
    MOV     RCAP2L, R0
    MOV     SCON, #52H     ; Initialize serial interface
                          ; (mode 1, set REN, set TI)
                          ; (setting TI enables first UART out polling OK)

    RET

;=====
; subroutine UMUL16
; 16-Bit x 16-Bit to 32-Bit Product Unsigned Multiply
;
; input:  r1, r0 = multiplicand X
;         r3, r2 = multiplier Y
;
; output: r3, r2, r1, r0 = product P = X x Y
;

```

```

; alters:  acc, C
;=====
UMUL16:
    PUSH    B
    PUSH    DPL
    MOV     A, R0
    MOV     B, R2
    MUL     AB          ; multiply XL x YL
    PUSH    ACC         ; stack result low byte
    PUSH    B           ; stack result high byte
    MOV     A, R0
    MOV     B, R3
    MUL     AB          ; multiply XL x YH
    POP     ARO
    ADD     A, R0
    MOV     R0, A
    CLR     A
    ADDC    A, B
    MOV     DPL, A
    MOV     A, R2
    MOV     B, R1
    MUL     AB          ; multiply XH x YL
    ADD     A, R0
    MOV     R0, A
    MOV     A, DPL
    ADDC    A, B
    MOV     DPL, A
    CLR     A
    ADDC    A, #0
    PUSH    ACC         ; save intermediate carry
    MOV     A, R3
    MOV     B, R1
    MUL     AB          ; multiply XH x YH
    ADD     A, DPL
    MOV     R2, A
    POP     ACC         ; retrieve carry
    ADDC    A, B
    MOV     R3, A
    MOV     R1, 00H
    POP     ARO        ; retrieve result low byte
    POP     DPL
    POP     B
    RET

```

```

;=====
; subroutine UDIV16
; 16-Bit / 16-Bit to 16-Bit Quotient & Remainder Unsigned Divide
;
; input:  r1, r0 = Dividend X
;         r3, r2 = Divisor Y
;
; output: r1, r0 = quotient Q of division Q = X / Y
;         r3, r2 = remainder
;
; alters: acc, B, dpl, dph, r4, r5, r6, r7, flags
;=====

```

```

UDIV16:
    MOV     R7, #0      ; clear partial remainder
    MOV     R6, #0
    MOV     B, #16     ; set loop count
;
DIV_LOOP:
    CLR     C           ; clear carry flag
    MOV     A, R0      ; shift the highest bit of
    RLC     A           ; the dividend into...
    MOV     R0, A
    MOV     A, R1
    RLC     A
    MOV     R1, A
    MOV     A, R6      ; ... the lowest bit of the
    RLC     A           ; partial remainder
    MOV     R6, A
    MOV     A, R7
    RLC     A
    MOV     R7, A
    MOV     A, R6      ; trial subtract divisor
    CLR     C           ; from partial remainder

```

```

SUBB  A, R2
MOV   DPL, A
MOV   A, R7
SUBB  A, R3
MOV   DPH, A
CPL   C           ; complement external borrow
JNC   DIV_1       ; update partial remainder if
                        ; borrow
MOV   R7, DPH     ; update partial remainder
MOV   R6, DPL

DIV_1:
MOV   A, R4       ; shift result bit into partial
RLC   A           ; quotient
MOV   R4, A
MOV   A, R5
RLC   A
MOV   R5, A
DJNZ  B, DIV_LOOP
MOV   A, R5       ; put quotient in r0, and r1
MOV   R1, A
MOV   A, R4
MOV   R0, A
MOV   A, R7       ; get remainder, saved before the
MOV   R3, A       ; last subtraction
MOV   A, R6
MOV   R2, A
RET

```