

Appendix B: PIC16F84-1 Serial Transmission Code

```
;; serial2.asm
;;
;; Transmits PORTB to serial port at various speeds.
;;
;; Requires 4.9152 MHz crystal.
;;
;; 04/27/99
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LIST p=16F84
include "p16f84.inc"
__CONFIG _HS_OSC & _CP_OFF & _WDT_OFF & _PWRTE_OFF
errorlevel -302

delay_count EQU 0x20
send EQU 0x21
send_count EQU 0x22
number EQU 0x23
hundreds EQU 0x24
tens EQU 0x25
ones EQU 0x26

init
;; Set ports for I/O
;;;;;;;;;;;;;;;;;;;;;;;;

BSF STATUS, RP0

;; PORTA outputs
CLRF TRISA

;; PORTB inputs
MOVLW 0xFF
MOVWF TRISB

BCF STATUS, RP0

;; Clear outputs
CLRF PORTA

;; Data format
;;
;; Start 0123 4567 Stop
;; 0 0000 0010 1
;;;;;;;;;;;;;;;;;;;;;;;;
```

```

main
    ;; Transmit radius info
    MOVLW    A'r'
    CALL     send_data
    MOVLW    A'='
    CALL     send_data

    ;; Select proper 74373 latch
    BSF     PORTA, 2
    BCF     PORTA, 3

    MOVF     PORTB, w
    CALL     send_number

    ;; Comma
    MOVLW    A', '
    CALL     send_data

    ;; Transmit angle info
    MOVLW    A't'
    CALL     send_data
    MOVLW    A'='
    CALL     send_data

    ;; Select proper 74373 latch
    BSF     PORTA, 2
    BCF     PORTA, 3

    MOVF     PORTB, w
    CALL     send_number

    ;; Send CR/LF
    MOVLW    0x0D
    CALL     send_data
    MOVLW    0x0A
    CALL     send_data

    ;; Wait a while
    MOVLW    0xFF
    MOVWF   delay_count
    CALL     delay_loop
    MOVLW    0xFF
    MOVWF   delay_count
    CALL     delay_loop

```


send_number

```
;; Sends ASCII numbers to terminal  
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
MOVWF    number  
CLRF     hundreds  
CLRF     tens  
CLRF     ones
```

get_hundreds

```
MOVLW    D'100'  
INCF     hundreds, f  
SUBWF    number, f
```

```
;; If result negative, C=0  
BTFSC    STATUS, C  
GOTO     get_hundreds
```

```
ADDWF    number, f  
DECF     hundreds, f
```

get_tens

```
MOVLW    D'10'  
INCF     tens, f  
SUBWF    number, f
```

```
;; If result negative, C=0  
BTFSC    STATUS, C  
GOTO     get_tens
```

```
ADDWF    number, f  
DECF     tens, f
```

get_ones

```
MOVLW    D'1'  
INCF     ones, f  
SUBWF    number, f
```

```
;; If result negative, C=0  
BTFSC    STATUS, C  
GOTO     get_ones
```

```
DECF     ones, f
```

```

;; Now send the data
;; Add 0x30 to each digit (0x30=ASCII 0)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

```

    MOVLW    0x30
    ADDWF   hundreds, w
    CALL    send_data

```

```

    MOVLW    0x30
    ADDWF   tens, w
    CALL    send_data

```

```

    MOVLW    0x30
    ADDWF   ones, w
    CALL    send_data

```

```

    RETURN

```

```

send_data

```

```

    MOVWF   send
    MOVLW   0x09
    MOVWF   send_count

```

```

    ;; Put start bit first (in carry)
    BCF    STATUS, C

```

```

send_loop

```

```

    BTFSS   STATUS, C
    GOTO    $+2
    GOTO    $+4

```

```

    ;; If bit not set
    NOP
    BCF    PORTA, 0
    GOTO    $+3

```

```

    ;; If bit set
    BSF    PORTA, 0
    GOTO    $+1

```

```

    CALL    delay

```

```

    RRF    send, f

```

```

    DECFSZ send_count

```

```

GOTO send_loop

;; Send stop bit
NOP
NOP
NOP
NOP
NOP

BSF      PORTA, 0
MOVLW   0xFF
MOVWF   delay_count
CALL    delay_loop

RETURN

;; Main delay loop
;;
;; Multiplies value in W register by 32, then waits
;; that many instruction cycles.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

delay

;; 2400   0x10
;; 4800   0x08
;; 9600   0x04
;; 19200  0x02
;; 38400  0x01
;;;;;;;;;;;;;;;;;;;;;;;;;;

MOVLW   0x08
MOVWF   delay_count

delay_loop
DECFSZ  delay_count, f
GOTO    delay_nop

NOP
NOP
NOP
NOP
NOP

```

NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF

RETURN

delay_nof

NOF
NOF
NOF

NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF

NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF
NOF

GOTO delay_loop

```
;; End of code  
;;;;;;;;;;;;;
```

END