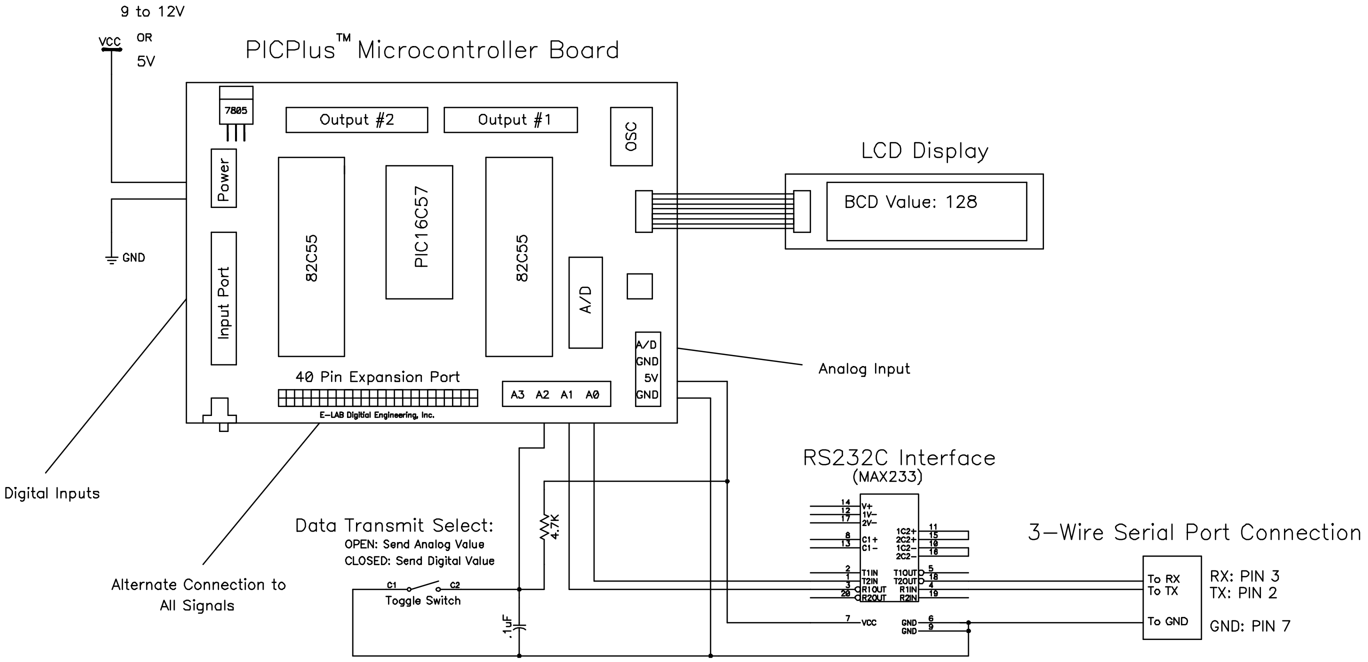
PICPlus™ Application Example #1



This application example illustrates the connection of the PICPlus™ board to a PC via a RS232, 9600 baud serial connection. A Maxim 233 IC is used to make the physical connection.

The liquid crystal screen simply plugs into the PICPlus™ board - no additional hardware is necessary. In this example, the screen in used to view the binary-coded data send to the serial port. The position of the toggle switch determines whether a byte of data from the digital input port or the data from the on-board analog to digital converter is sent to the serial port.

The attached assembly language program, written in Parallax™ assembly language, illustrates the use of both the driver routines provided with the PICPlus™ board as well as the use of routines specific to this application, such as 'to_BCD' and 'xmit'. These particular routines convert a binary value to its binary-coded decimal equivalent (11111111 binary is sent as '2' '5' '5') and transmit a byte of data as 9600 baud, respectively.

The connections to the PICPlus™ board are being made to the terminal blocks in this example. All connections could, if desired, be made through the 40 pin expansion port. Using this method, any additional custom circuitry, such as the MAX233 shown in this example, could be on one card and simply plug into the PICPlus™ board via a ribbon cable. This approach greatly simplifies development.

```
; PROGRAM: out_232.asm
; This program, written in Parallax(TM) assembly language, is for use on
; the PICPlus(TM) Board manufactured by E-LAB Digital Engineering, Inc.
; It samples a toggle switch and sends, depending upon the position of the
; switch, either the BCD value of the digital input port or the BCD value
; of the converted A/D value.  The data is send at 9600 baud using RS232C.
; In addition, the transmitted data is also written to the LCD port.
; This allows a visual conformation that valid data is being sent to the PC.
; A terminal program can be used to receive the data, or some simple software
; could be written to sample the PC's serial port.


bit_K                           =               128                             ;9600 baud operation
serial_out                      =               ra.0                            ;serial out port A pin 0
toggle_in                       =               ra.2                            ;data select (toggle switch input)

; Variable storage above special-purpose registers.
                                org             8

first                           ds              1                               ;first number in BCD string
second                          ds              1                               ;second number in BCD string
third                           ds              1                               ;third number in BCD string
cycle                           ds              1                               ;used in the BCD conversion
delay_cntr                      ds              1                               ;Counter for serial delay routines
bit_cntr                        ds              1                               ;Number of transmitted bits
xmt_byte                        ds              1                               ;The transmitted byte
length                          ds              1                               ;LCD length coulter

                                device  pic16c57,hs_osc,wdt_off,protect_off
                                include 'driver.asm'                            ;link in driver routine!

                                mov             !ra, #00000100b                 ;set A0 to output, A2 to input
                                jmp             start                           ;skip ahead to main loop

;------------------------------------------------------------
;lcd text string listed here:
string1                         mov             w,length                        ;these 3 lines return string #1
                                jmp             pc+w
                                retw            'B','C','D',' ','V','A','L','U','E',':',' '

;------------------------------------------------------------
; this subroutine converts a binary number to its binary-coded
; decimal (BCD) equivalent: (Ex. 11111111 binary -> 2,5,5)
to_BCD                          mov             first,#000h
                                mov             second,#000h
                                mov             third,#000h
                                cjb             data,#100,tens_start
                                sub             data,#100
                                inc             first
                                lset            $
                                cjb             data,#100,tens_start
                                sub             data,#100
                                inc             first
tens_start                      mov             cycle,#009
```

```
                        lset        $
tens                    cjb         data,#010,ones
                        sub         data,#010
                        inc         second
                        lset        $
                        djnz        cycle,tens
ones                    mov         third,data
                        ret

;------------------------------------------------------------
;this subroutine sends 1 byte out A0 serially at 9600 baud:
xmit                    mov         bit_cntr,#8             ;eight bits in a byte.
                        mov         xmt_byte,rc             ;put character into the transmit byte.
                        clrb        serial_out              ;hold line high
bit_delay1              mov         delay_cntr,#bit_K
:loop                   nop
                        djnz        delay_cntr, :loop
send                    rr          xmt_byte                ;rotate right moves data bits into
                                                            ;carry, starting with bit 0.
                        movb        serial_out,c
bit_delay2              mov         delay_cntr,#bit_K
:loop                   nop
                        djnz        delay_cntr, :loop
                        djnz        bit_cntr,send           ;Not eight bits yet? Send next data bit
                        setb        serial_out
bit_delay3              mov         delay_cntr,#bit_K
:loop                   nop
                        djnz        delay_cntr, :loop
bit_delay4              mov         delay_cntr,#bit_K
:loop                   nop
                        djnz        delay_cntr, :loop
                        ret

;-------------------------------------------------------------
;initialize LCD:
start                   mov         rc,#038h                ;8-bit, 2-line, 5x7 font
                        lcall       LCD_ctrl                ;write to LCD control register
                        lset        $                       ;set proper page (in larger code)
                        mov         rc,  #00Ch              ;display on, cursor off, blink off
                        lcall       LCD_ctrl
                        lset        $
                        mov         rc,  #006h              ;increment cursor, no shifting
                        lcall       LCD_ctrl
                        lset        $
                        mov         rc,  #001h              ;clear display, homes cursor
                        lcall       LCD_ctrl
                        lset        $

;----------------------------------------------------------------------------
; this loop is the main program:
loop                    lcall       input                   ;read digital input into 'data'
                        lset        $
                        jnb         toggle_in,use_dig       ;read toggle switch
                        lcall       a2d                     ;read A/D converter into 'data'
                        lset        $
```

```
use_dig                 lcall       to_BCD                  ;convert value in 'data' to BCD
                        lset        $
                        add         first,#030h             ;convert to ASCII
                        add         second,#030h            ;convert to ASCII
                        add         third,#030h             ;convert to ASCII

                        mov         rc,  #080h              ;home cursor
                        lcall       LCD_ctrl
                        lset        $

;write text to LCD screen:
                        mov         length,#00              ;clear length counter
print1                  lcall       string1                 ;get next character
                        lset        $
                        mov         rc,w                    ;move character from 'w' to 'rc'
                        lcall       LCD_print               ;print character to LCD port
                        lset        $
                        inc         length                  ;add one to 'length' counter
                        cjb         length,#11,print1       ;'11' is the length of string #1

                        mov         rc,first
                        lcall       xmit                    ;send 'first' out serially
                        lset        $
                        lcall       LCD_print               ;print 'first' to LCD port
                        lset        $

                        mov         rc,second
                        lcall       xmit                    ;send 'second' out serially
                        lset        $
                        lcall       LCD_print               ;print 'second' to LCD port
                        lset        $

                        mov         rc,third
                        lcall       xmit                    ;send 'third' out serially
                        lset        $
                        lcall       LCD_print               ;print 'third' to LCD port
                        lset        $

                        mov         rc,#00dh                ;ASCII for carriage return
                        lcall       xmit                    ;send carriage return to serial port
                        lset        $

                        mov         !ra, #00000100b         ;set A0 to output, A2 to input
                        jmp         loop                    ;start loop over
```