LCD Terminal AppMod (#29121)
## 2 Line x 8 Character LCD Module with User Buttons

Introduction

The LCD Terminal AppMod provides a simple and convenient method of adding a standard character LCD and 4 user-input buttons to BASIC Stamp projects. Its 20-pin male header plugs into the 2x10 AppMod Header socket on the Parallax Board of Education (#28150, #28850) or Super Carrier Board (#27130).
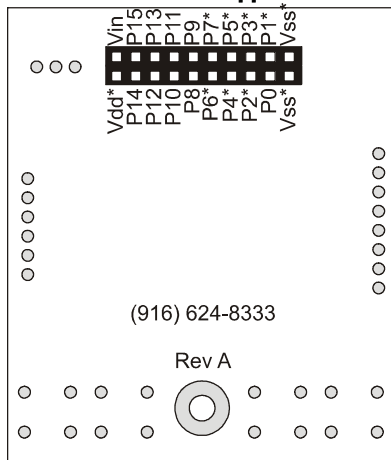
Features

- 2 x 8 LCD module, HD44780-compatible
- Parallel LCD uses 7 I/O pins
- Contrast control pot
- 4 buttons for user input
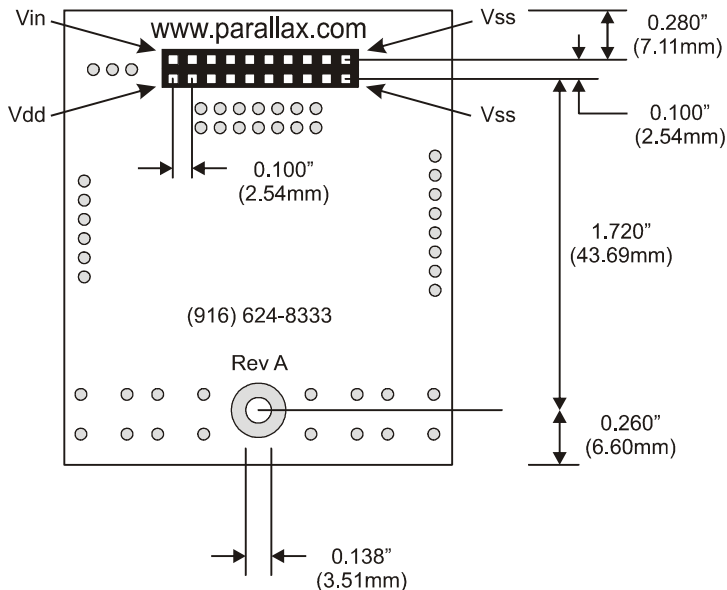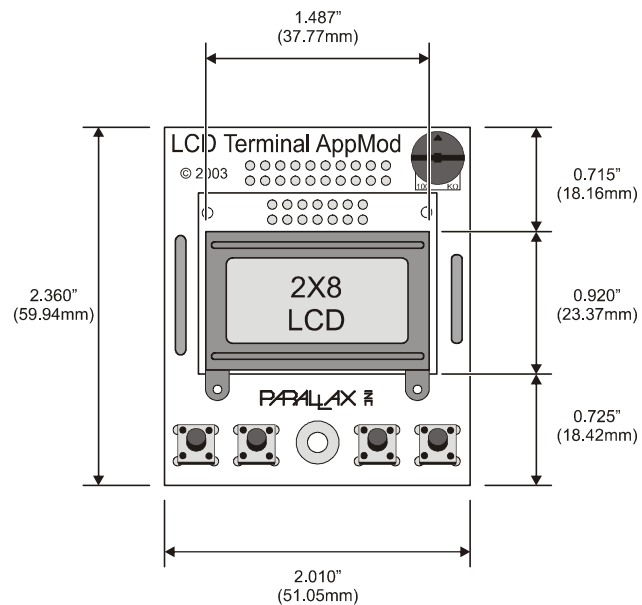- +5 VDC, supplied through AppMod Vdd

Appmod Header Pinout
(Board of Education and Super Carrier Board)
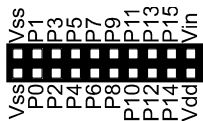
LCD Terminal AppMod Pinout

*Used by LCD Terminal AppMod

Board Installation

The LCD Terminal AppMod's 20-pin male header plugs directly into the 2x10 socket. Before installation, be sure to check the board silkscreen labels for proper orientation. The photos below show the proper orientation on a Super Carrier Board (left) and a Board of Education (right). Use caution: reversing the connection could result in applying power to the LCD's ground pins and could damage your unit.
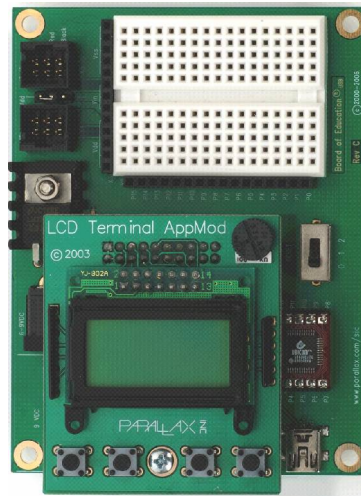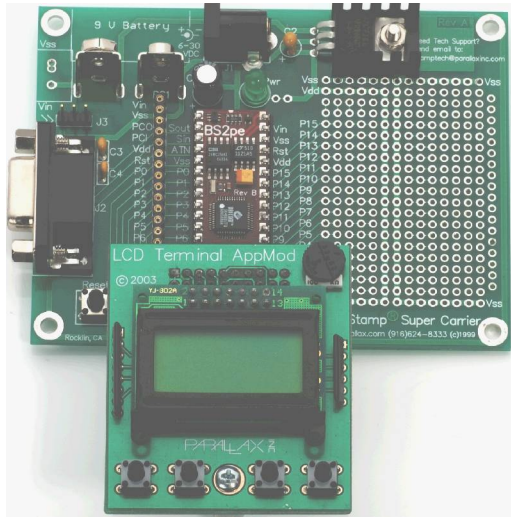
Schematic
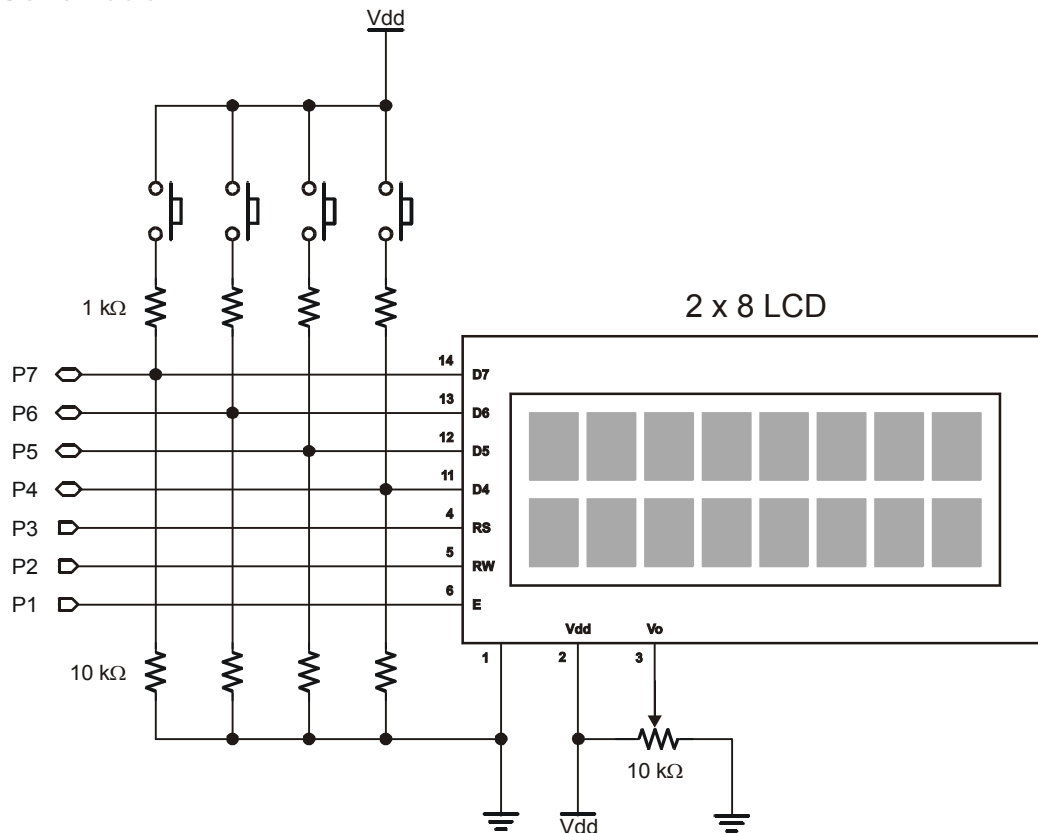
Circuit Notes

The resistor values are important for the proper operation of the circuit. You may be wondering if the LCD will be adversely affected if a button is pressed while the BASIC Stamp is writing to it. The answer is no. When no buttons are pressed, the signals from the BASIC Stamp microcontroller are felt "across" the 10K resistors, hence there is no concern. When a button is pressed, a high level will be exerted on the bus. If the state of that buss line is supposed to be low, the BASIC Stamp overcomes the button press and a small amount of current will flow through the 1K resistor and the low exerted by the BASIC Stamp pin will be seen by the LCD.

BASIC Stamp 2 Application

The following BASIC Stamp program demonstrates many of the capabilities of the LCD and how the user is able to read and debounce the module's user buttons. This program is somewhat unique in that it is compatible with every BASIC Stamp 2 module; no changes are required. If you attempt to run the program on something other than a standard BS2, the compiler will ask if you want to run on the installed Stamp (BS2e, BS2sx, BS2p, or BS2pe). If you do, the program will run without problems.

In order to allow this program to take advantage of the built-in LCD features of the BS2p family, conditional compilation directives are used. Conditional compilation directives are evaluated before the program is compiled and downloaded to the BASIC Stamp, so only those portions that pertain to the installed BASIC Stamp will be downloaded; not the entire listing.

The program is quite straightforward, and uses a simple software trick to scroll a string through the small window (eight characters) of the LCD. Entry and exit of the string is facilitated by padding the string with spaces on either end.

```
' =============================================================================
'
'   File....... LCD_AppMod_Demo.BS2
'   Purpose.... Demonstrates the LCD Terminal AppMod
'   Author..... Parallax, Inc.  (Copyright 2003-04, All Rights Reserved)
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 13 JAN 2004
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =============================================================================


' -----[ Program Description ]-------------------------------------------------
'
' This program demonstrates the use of the Parallax LCD Terminal AppMod
' with any BS2-series microcontroller. This program uses conditional
' compilation techniques which make it completely BS2-agnostic. Custom
' character generation and animation is demonstrated.



' -----[ I/O Definitions ]-----------------------------------------------------
```

```
E               PIN     1                       ' LCD Enable (1 = enabled)
RW              PIN     2                       ' Read/Write\
RS              PIN     3                       ' Reg Select (1 = char)
LcdDirs         VAR     DIRB                    ' dirs for I/O redirection
LcdBusOut       VAR     OUTB
LcdBusIn        VAR     INB



' -----[ Constants ]-----------------------------------------------

#DEFINE _LcdReady = ($STAMP = BS2P) OR ($STAMP = BS2PE)


LcdCls          CON     $01                     ' clear the LCD
LcdHome         CON     $02                     ' move cursor home
LcdCrsrL        CON     $10                     ' move cursor left
LcdCrsrR        CON     $14                     ' move cursor right
LcdDispL        CON     $18                     ' shift chars left
LcdDispR        CON     $1C                     ' shift chars right

LcdDDRam        CON     $80                     ' Display Data RAM control
LcdCGRam        CON     $40                     ' Character Generator RAM
LcdLine1        CON     $80                     ' DDRAM address of line 1
LcdLine2        CON     $C0                     ' DDRAM address of line 2


LcdScrollTm     CON     250                     ' LCD scroll timing (ms)



' -----[ Variables ]-----------------------------------------------

addr            VAR     Word                    ' address pointer
crsrPos         VAR     Byte                    ' cursor position
char            VAR     Byte                    ' character sent to LCD
idx             VAR     Byte                    ' loop counter
scan            VAR     Byte                    ' loop counter

buttons         VAR     Nib
btnA            VAR     buttons.BIT0            ' left-most button
btnB            VAR     buttons.BIT1
btnC            VAR     buttons.BIT2
btnD            VAR     buttons.BIT3            ' right-most

btnDemo         VAR     Byte                    ' loop counter



' -----[ EEPROM Data ]---------------------------------------------

CC0             DATA    $0E, $1F, $1C, $18, $1C, $1F, $0E, $00  ' char 0
CC1             DATA    $0E, $1F, $1F, $18, $1F, $1F, $0E, $00  ' char 1
```

```
CC2             DATA    $0E, $1F, $1F, $1F, $1F, $1F, $0E, $00  ' char 2
Smiley          DATA    $00, $0A, $0A, $00, $11, $0E, $06, $00  ' smiley

Msg1            DATA    "PARALLAX", 0
Msg2            DATA    "       BASIC STAMP  ", 3, "       ", 0
Msg3            DATA    "Type =", 0
Msg4            DATA    "Buttons:", 0

StampId0        DATA    "  BS2", 0
StampId1        DATA    " BS2e", 0
StampId2        DATA    "BS2sx", 0
StampId3        DATA    " BS2p", 0
StampId4        DATA    "BS2pe", 0


' -----[ Initialization ]-------------------------------------------------

Initialize:
  NAP 5                                       ' let LCD self-initialize
  DIRL = %11111110                            ' setup pins for LCD

LCD_Init:
  #IF _LcdReady #THEN
    LCDCMD E, %00110000 : PAUSE 5             ' 8-bit mode
    LCDCMD E, %00110000 : PAUSE 0
    LCDCMD E, %00110000 : PAUSE 0
    LCDCMD E, %00100000 : PAUSE 0             ' 4-bit mode
    LCDCMD E, %00101000 : PAUSE 0             ' 2-line mode
    LCDCMD E, %00001100 : PAUSE 0             ' no crsr, no blink
    LCDCMD E, %00000110                       ' inc crsr, no disp shift
  #ELSE
    LcdBusOut = %0011                         ' 8-bit mode
    PULSOUT E, 3 : PAUSE 5
    PULSOUT E, 3 : PAUSE 0
    PULSOUT E, 3 : PAUSE 0
    LcdBusOut = %0010                         ' 4-bit mode
    PULSOUT E, 3
    char = %00101000                          ' 2-line mode
    GOSUB LCD_Command
    char = %00001100                          ' on, no crsr, no blink
    GOSUB LCD_Command
    char = %00000110                          ' inc crsr, no disp shift
    GOSUB LCD_Command
  #ENDIF

Download_Chars:                               ' download custom chars
  char = LcdCGRam                             ' point to CG RAM
  GOSUB LCD_Command                           ' prepare to write CG data
  FOR idx = CC0 TO (Smiley + 7)               ' build 4 custom chars
```

```
    READ idx, char                            ' get byte from EEPROM
    GOSUB LCD_Write_Char                       ' put into LCD CG RAM
  NEXT



' -----[ Program Code ]----------------------------------------------------

Main:
  char = LcdCls                                ' clear the LCD
  GOSUB LCD_Command
  PAUSE 500

Write_Parallax:
  addr = Msg1                                  ' point to message
  GOSUB LCD_Put_String                         ' write it

Scroll_Message:
  crsrPos = LcdLine2                           ' scroll on line 2
  addr = Msg2                                  ' point to msg
  GOSUB LCD_Scroll_String                      ' scroll it

Pac_Man:                                       ' Pac-Man animation
  FOR idx = 0 TO 7                             ' cover 8 characters
    FOR scan = 0 TO 4                          ' 5 characters in animation
      char = LcdLine1 + idx                    ' position cursor
      GOSUB LCD_Command
      LOOKUP scan, [0, 1, 2, 1, " "], char     ' select "frame"
      GOSUB LCD_Write_Char                     ' write animation character
      PAUSE 75                                 ' delay between chars
    NEXT
  NEXT

Show_Stamp_Type:
  char = LcdCls                                ' clear the LCD
  GOSUB LCD_Command
  PAUSE 100
  addr = Msg3                                  ' display "Type ="
  GOSUB LCD_Put_String
  char = LcdLine2 + 3                          ' move cursor to 2nd line
  GOSUB LCD_Command

  #SELECT $STAMP                               ' check type at compile
    #CASE BS2
      addr = StampId0
    #CASE BS2E
      addr = StampId1
    #CASE BS2SX
      addr = StampId2
    #CASE BS2P
```

```
      addr = StampId3
    #CASE BS2PE
      addr = StampId4
   #ENDSELECT
   GOSUB LCD_Put_String                         ' display type on LCD
    PAUSE 2000

Show_Buttons:
  char = LcdCls                                 ' clear the LCD
  GOSUB LCD_Command
  PAUSE 100
  addr = Msg4                                   ' write "Buttons:"
  GOSUB LCD_Put_String

  FOR btnDemo = 1 TO 100
    GOSUB LCD_Get_Buttons                       ' read/debounce buttons
    char = LcdLine2 + 2                         ' show on 2nd line
    GOSUB LCD_Command
    FOR idx = 0 TO 3                            ' display buttons
      IF buttons.LOWBIT(idx) THEN
        char = "A" + idx                        ' button letter if pressed
      ELSE
        char = "-"                              ' otherwise dash
      ENDIF
      GOSUB LCD_Write_Char
    NEXT
  NEXT

  GOTO Main                                     ' run demo again
  END



' -----[ Subroutines ]-----------------------------------------------------

' Writes stored (in DATA statement) zero-terminated string to LCD
' -- position LCD cursor
' -- point to 0-terminated string (first location in 'addr')

LCD_Put_String:
  DO
    READ addr, char
    IF (char = 0) THEN EXIT
    GOSUB LCD_Write_Char
    addr = addr + 1
  LOOP
  RETURN



' Scroll a message across LCD line
```

```
' -- set starting position in 'crsrPos'
' -- point to 0-terminated string (first location in 'addr')
' -- strings should be padded with eight spaces on each end

LCD_Scroll_String:
  DO
    char = crsrPos                              ' move to start of window
    GOSUB LCD_Command
    FOR idx = 0 TO 7                            ' write chars in window
      READ (addr + idx), char
      IF (char = 0) THEN EXIT                   ' stop if end of string
      GOSUB LCD_Write_Char
    NEXT
    IF (char = 0) THEN EXIT
    addr = addr + 1                             ' scroll
    PAUSE LcdScrollTm
  LOOP
  RETURN


' Send command to LCD
' -- put command byte in 'char'

LCD_Command:                                    ' write command to LCD
  #IF _LcdReady #THEN
    LCDCMD E, char
    RETURN
  #ELSE
    LOW RS
    GOTO LCD_Write_Char
  #ENDIF


' Write character to current cursor position
' -- but byte to write in 'char'

LCD_Write_Char:                                 ' write character to LCD
  #IF _LcdReady #THEN
    LCDOUT E, 0, [char]
  #ELSE
    LcdBusOut = char.HIGHNIB                     ' output high nibble
    PULSOUT E, 3                                 ' strobe the Enable line
    LcdBusOut = char.LOWNIB                      ' output low nibble
    PULSOUT E, 3
    HIGH RS                                      ' return to character mode
  #ENDIF
  RETURN
```

```
' Reads byte from LCD
' -- put byte address in 'addr'
' -- returns byte read in 'char'

LCD_Read_Char:                                  ' read character from LCD
  #IF _LcdReady #THEN
    LCDIN E, addr, [char]
  #ELSE
    char = addr                                 ' move cursor
    GOSUB LCD_Command
    HIGH RS                                     ' data command
    HIGH RW                                     ' read
    LcdDirs = %0000                             ' make LCD bus inputs
    HIGH E
    char.HIGHNIB = LcdBusIn                      ' get high nibble
    LOW E
    HIGH E
    char.LOWNIB = LcdBusIn                       ' get low nibble
    LOW E
    LcdDirs = %1111                             ' return data lines to outputs
    LOW RW
  #ENDIF
  RETURN



' Read and debounce the LCD AppMod buttons

LCD_Get_Buttons:
  LcdDirs = %0000                               ' make LCD bus inputs
  buttons = %1111                               ' assume all pressed
  FOR scan = 1 TO 10
    buttons = buttons & LcdBusIn                 ' make sure button held
    PAUSE 5                                      ' debounce 10 x 5 ms
  NEXT
  LcdDirs = %1111                               ' return bus to outputs
  RETURN
```

Additional Resources

The following resources are available from www.parallax.com

- BASIC Stamp Manual or BASIC Stamp Editor Help file: syntax and reference for the LCDCMD, LCDIN, and LCDOUT commands for the BS2p-family of microcontroller modules.
- StampWorks Projects 11 – 14; more parallel LCD programming topics
- Nuts & Volts Stamp Applications #31: Demystifying Character-based LCDs
- BASIC Stamp 1 program version (limited features due to code space restrictions)
- Javelin Stamp program version, complete with LcdTerminal class file
- Hitachi HD44780 Datasheet