LCD Serial Backpack App Note No. 1

Beyond the Basics: Advanced Techniques with LCDs

Giant Characters, LED Backlighting, and Extended Temperature Operation

A FEW SIMPLE TRICKS can make liquid-crystal displays (LCDs) much more versatile. This application note will present three techniques of interest to LCD users:

- Displaying giant characters on 4-line LCDs.
- Making LED backlighting more efficient.
- Driving extended-temperature LCDs.

Background. This application note is intended primarily for displays equipped with the LCD Serial Backpack™, a daughterboard that gives 14-pin alphanumeric LCDs an easy-to-use serial interface. However, the priniciples described here apply to standard LCD modules as well.

Giant Custom Characters. Many display applications have to meet conflicting design requirements: presenting important data in symbols that can be read from a distance, and showing lots of detailed information when the user moves in close. Many designers faced with this dilemma throw up their hands and specify an expensive graphical display.

There's a better way, using nothing more than a standard four-line alphanumeric display. See figure 1.1 (photos of the 4x20 serial LCD module, BPP-420L, made by Scott Edwards Electronics). By defining custom graphics in the LCD's character-generator (CG) RAM, you can create four-row-tall characters.

Figure 1.1. Combining custom graphics symbols to produce 1-inch-high characters.

Since most four-line displays' characters are 0.19 inches tall with a pixel space between lines, these characters can be almost an inch tall.

Figure 1.2 shows how to build the necessary symbols in CG RAM. Figure 1.3 is a complete 36-character alphanumeric symbol set.

Listings 1.1 and 1.2 at the end of this application note are programs for the BASIC Stamp II and QBASIC that implement a large numeric display on the 4x20 serial LCD module BPP-420L. Note that the custom symbol set of figure 1.2 is defined in firmware by newer Backpacks (all 4x20 modules sold after July '96), so certain portions of the program listings may be omitted to save space.

#	Graphic	Data	#	Graphic	Data
0		0,0,0,1,3,7,15,31	4		0,0,0,0,31,31,31,31
1		0,0,0,16,24,28,30,31	5		31,31,31,31,0,0,0,0
2		31,15,7,3,1,0,0,0	6		31,31,31,31,31,31,31,31
3		31,30,28,24,16,0,0,0	7		0,0,0,0,0,0,0,0

These graphics symbols are used to create the giant characters. To define the symbols you must write the data shown above (8 bytes per symbol; 64 bytes total) to the LCD's character-generator (CG) RAM. The procedure is as follows:

Clear (write 0 to) the LCD register-select (RS) input.
Write 64 to the LCD to send subsequent writes to CG RAM, starting at address 0.
Set (write 1 to) RS.
Write the 64 bytes shown in the table above to the LCD.
Clear RS and write 128 to the LCD to send subsequent writes to display RAM, starting at address 0.
Set RS. You may now use the symbols to construct giant characters. Figure 3 shows how the charcters are made by placing symbols on each of the LCD's four lines. For example, the 0 is created by placing symbols 0, 5, 5, 1 on the top line; 6, 7, 7, 6 on the second line and so on.

Figure 1.2. Custom symbols for drawing large characters. (Predefined in newer Backpacks.)

This technique is so useful that it was added to the instrustion set of the 4x40 serial LCD modules. A single-byte instruction, ctrl-B, puts the display into "big-number" mode. Subsequent numeric characters—0 through 9, decimal point, colon and minus—are printed spanning the four lines of the display.

Efficient LED Backlighting. In the past, most backlit LCDs were equipped with some form of electroluminescent (EL) panel. While reasonably efficient and inexpensive, these panels require a source of 100+ volts AC at 100 to 1000 Hz. Manufacturers specify a proprietary inverter to drive the EL backlight. Even as LCD prices have dropped, the inverters remain expensive. They are also a source of electromagnetic noise.

Many LCDs are now available with LED backlighting. LCDs with this option generally cost slightly more than those with EL panels, but far less than an EL-equipped LCD *plus* the required inverter.

Driving an LED backlight is a piece of cake—just tap into your system's regulated 5V supply and add a series resistor for current limiting. See figure 1.4, which also shows an active current limiter that holds LED brightness steady over a range of input voltages.

Another way to regulate the average current through an LED backlight is to control the duty cycle of the current to the LEDs as in figure 1.5.

With an LED-backlit 4x20 display we found that 20mA of backlight current was sufficient to make the display readable in darkness; 50 mA even enhanced display contrast under bright indoor lighting. These current levels make the backlight usable even with minimal power supplies, such as the common combination of a 9V alkaline battery and 100-mA, three-terminal voltage regulator.

Extended Temperature Operation. Most LCDs will operate over a temperature range of 0° to 50°C (32° to 122°F). When cold, the pixels fade; when hot, they darken. In either case, there may not be enough contrast between on and off pixels.

Between the extremes of temperature, contrast control is provided by a user-adjustable pot that dials in a LCD bias voltage between ground and 5 volts.

Many manufacturers offer "extended-temperature" LCD modules that can operate from –20° to 70°C (–4° to 158°F). The only catch is that you need a negative power supply of approximately –7 volts to drive the bias input.

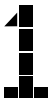
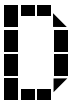
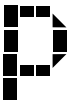
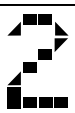
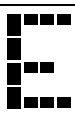
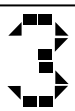
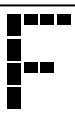
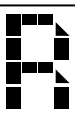
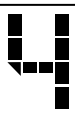
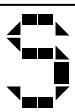
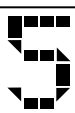
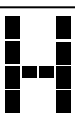
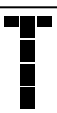
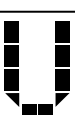
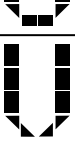
Data	Character	Data	Character	Data	Character
0,5,5,1 6,7,7,6 6,7,7,6 2,4,4,3		0,5,5,1 6,7,7,7 6,7,7,7 2,4,4,3		0,5,5,1 6,7,7,6 6,7,7,6 2,4,4,3	
7,0,6,7 7,7,6,7 7,7,6,7 7,4,6,4		6,5,5,1 6,7,7,6 6,7,7,6 6,4,4,3		6,5,5,1 6,7,7,6 6,4,4,3 6,7,7,7	
0,5,5,1 7,7,4,3 0,5,7,7 6,4,4,4		6,5,5,5 6,7,7,7 6,5,5,7 6,4,4,4		0,5,5,1 6,7,7,6 6,7,7,6 2,4,2,1	
0,5,5,1 7,7,4,3 7,7,5,1 2,4,4,3		6,5,5,5 6,7,7,7 6,5,5,7 6,7,7,7		6,5,5,1 6,7,7,6 6,5,5,1 6,7,7,6	
6,7,7,6 6,7,7,6 2,5,5,6 7,7,7,6		0,5,5,1 6,7,7,7 6,7,5,6 2,4,4,3		0,5,5,1 2,4,4,1 7,7,76 2,4,4,3	
6,5,5,5 2,4,4,1 7,7,7,6 2,4,4,3		6,7,7,6 6,7,7,6 6,5,5,6 6,7,7,6		5,6,5,7 7,6,7,7 7,6,7,7 7,6,7,7	
0,5,5,7 6,0,4,1 6,7,7,6 2,4,4,3		7,5,6,5 7,7,6,7 7,7,6,7 7,4,6,4		6,7,7,6 6,7,7,6 6,7,7,6 2,4,4,3	
2,5,5,6 7,7,0,3 7,7,6,7 7,7,6,7		7,5,6,5 7,7,6,7 7,7,6,7 2,4,3,7		6,7,7,6 6,7,7,6 6,7,7,6 2,1,0,3	
0,5,5,1 2,4,4,3 0,5,5,1 2,4,4,3		6,7,0,3 6,4,3,7 6,5,1,7 6,7,2,1		6,7,7,6 6,7,7,6 6,0,1,6 6,3,2,6	
0,5,5,1 6,7,7,6 2,5,3,6 7,4,4,3		6,7,7,7 6,7,7,7 6,7,7,7 6,4,4,4		6,7,7,6 2,1,0,3 0,3,2,1 6,7,7,6	
0,5,5,1 6,7,7,6 6,5,5,6 6,7,7,6		6,7,7,6 6,1,0,6 6,2,3,6 6,7,7,6		6,7,7,6 2,1,0,3 7,7,6,7 7,7,6,7	
6,5,5,1 6,4,4,3 6,5,5,1 6,4,4,3		6,1,7,6 6,2,1,6 6,7,2,6 6,7,7,6		5,5,5,6 7,7,0,3 0,3,7,7 6,4,4,4	

Figure 1.3. The giant character set.

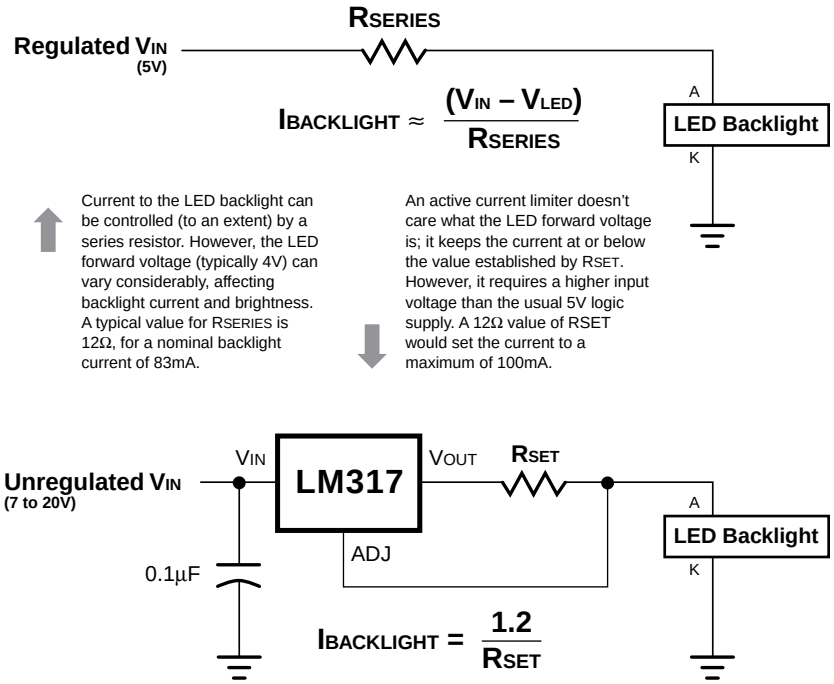


Figure 1.4. Current-limiting techniques for LED backlights.

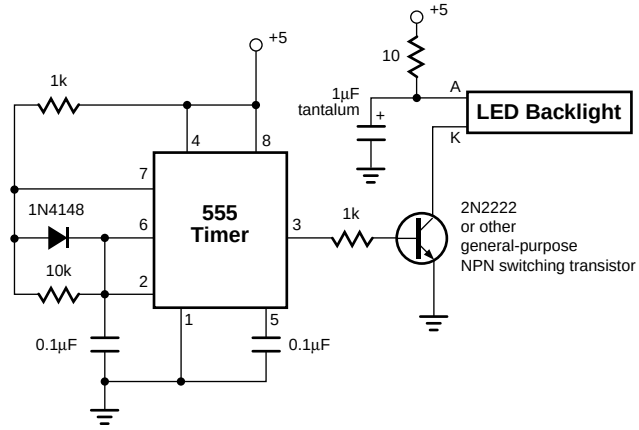


Figure 1.5. Duty-cycle control of LED backlight for ultra efficiency. The circuit shown produces a 10% duty cycle, limiting average LED current to just 20mA.

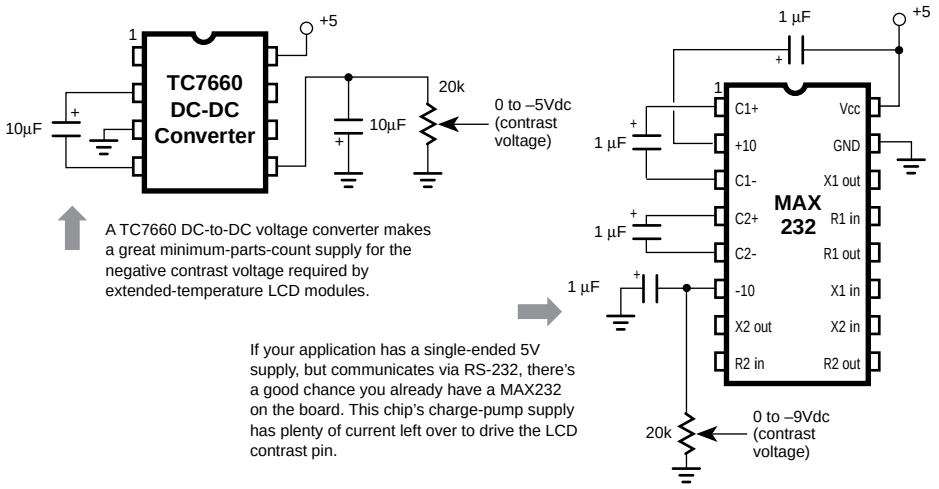


Figure 1.6. Obtaining a negative voltage source for extended-temperature LCDs.

This can be hard to come by in a typical 5-volt digital system. Figure 6 offers two suggestions, one obvious (a DC-to-DC converter chip) the other downright devious (thieving the voltage from an existing serial-port driver).

The Backpack manual (page 3, top) shows how to connect a negative supply through the Backpack circuit board.

Sources

Scott Edwards Electronics (SEE) manufactures and sells the LCD Serial Backpack™, a daughterboard that gives 14-pin alphanumeric LCDs an easy-to-use serial interface. The Backpack automatically initializes the LCD, receives serial data at a user-selectable rate of 2400 or 9600 baud and converts received data into LCD-compatible parallel output. The Backpack supports all standard alphanumeric LCDs up to 80 characters total (4x20 or 2x40).

The Backpack is available by itself for installation on a user-supplied LCD, or preinstalled to one of several high-quality supertwist LCDs. See the catalog for current offerings.

SEE also offers 4x40 serial LCD modules with additional terminal-emulation and special-features including automatic large-number display.

You may download a current SEE catalog from the customer-support FTP archive: ftp.nutsvolts.com, in pub/nutsvolts/scott. Or contact—

Scott Edwards Electronics

PO Box 160

Sierra Vista, AZ 85635

ph: 520-459-4802 fax: 520-459-0623

```
' Listing 1.1: BIGNUMS.BS2 (Display four 1" digits on 4x20 display)
' Connect the serial input of a Backpack-equipped 4x20 display to
' BS2 pin P0 and run this program. The program will define a set
' of symbols that allow it to display 4-line-tall numerals on the
' LCD. To incorporate this capability into your own programs, just
' substitute your code for the demo loop. When you want to display
' a value (0-9999) in big numerals, write it to dispVal, then
' gosub bigLCD.

I       con    254      ' Instruction prefix.
ClrLCD  con    1        ' Clear-LCD instruction.
N96N    con    $4054    ' 9600 baud, inverted, no parity.
cgRAM   con    64       ' Address 0 of CG RAM.

EEptr   var    word     ' Pointer into EEPROM.
pat     var    EEptr    ' Alias for EEptr.
dispVal var    word     ' Value to be displayed as big digits.
temp    var    byte     ' Temporary byte variable.
decade  var    word
nbl     var    nib      ' Index into number-pattern tables.
digit   var    nib      ' Current digit to display
line    var    nib      ' LCD line

' ====This section may be omitted with newer (post July 96)
' ====4x20 Serial LCD modules. Cut from here...============== >>>
bitPat0 DATA   0,0,0,1,3,7,15,31         ' Left-right up-ramp shape.
bitPat1 DATA   0,0,0,16,24,28,30,31      ' Right-left "       "
bitPat2 DATA   31,15,7,3,1,0,0,0         ' Left-right down ramp.
bitPat3 DATA   31,30,28,24,16,0,0,0      ' Right-left "    "
bitPat4 DATA   0,0,0,0,31,31,31,31       ' Lower block.
bitPat5 DATA   31,31,31,31,0,0,0,0       ' Upper block.
bitPat6 DATA   31,31,31,31,31,31,31,31   ' Full block.
bitPat7 DATA   0,0,0,0,0,0,0,0           ' Full blank
' <<<...to here. ==============================================

low 0                   ' Make the serial output low
pause 1000              ' Let the LCD wake up.

' ====This section may be omitted with newer (post July 96)
' ====4x20 Serial LCD modules. Cut from here...============= >>>
serout 0,N96N,[I,cgRAM]         ' Enter CG RAM.
for EEptr = 0 to 63             ' Write the bit patterns
  Read EEptr,temp               '   to the LCD.
  serout 0,N96N,[temp]
next
' <<<...to here. ==============================================

serout 0,N96N,[I,ClrLCD]        ' Clear the LCD.
pause 1
```

```
' ================================================================================
'       Demo Loop: Show dispVal in Big Numerals, Increment, Loop
' ================================================================================
again:
  gosub bigLCD
  dispVal = dispVal + 1
  pause 500
goto again


' ================================================================================
'                  Subroutine Displaying Large Numbers
' ================================================================================
bigLCD:
for line = 0 to 3
decade = 1000
  lookup line,[128,192,148,212],temp
  serout 0,N96N,[I,temp]
  for digit = 3 to 0
    nbl = dispVal dig digit
    gosub getPattern
    if dispVal = 0 and digit = 0 then skip0
    if dispVal < decade then blankIt
skip0:
    serout 0,N96N,[pat.nib3,pat.nib2,pat.nib1,pat.nib0]
    goto cont
blankIt:
   serout 0,N96N,["    "]
cont:
    if digit = 0 then skip1
    serout 0,N96N,[32]
skip1:
  decade = decade/10
  next
next
return
```

```
'=========================================================================
'               Subroutines Defining Big-Character Patterns
' ========================================================================
getPattern:
branch line,[first,second,third,fourth]

'             0     1     2     3     4     5     6     7     8     9
'            ---   ---   ---   ---   ---   ---   ---   ---   ---   ---
first:
lookup nbl,[$0551,$7067,$0551,$0551,$6776,$6555,$0557,$2556,$0551,$0551],pat
return

second:
lookup nbl,[$6776,$7767,$7743,$7743,$6776,$2441,$6041,$7703,$2443,$6776],pat
return

third:
lookup nbl,[$6776,$7767,$0577,$7751,$2556,$7776,$6776,$7767,$0551,$2536],pat
return

fourth:
lookup nbl,[$2443,$7464,$6444,$2443,$7776,$2443,$2443,$7767,$2443,$7443],pat
return
```

```
' Listing 1.2: QBIGNUM.BAS (for QBASIC, DOS 5.0+)
' (Display 4-line-tall digits on 4x20 Serial LCD Module)
' This program shows how to display 1" numbers on the 4x20 Serial
' LCD Module (4x20 display with LCD Serial Backpack) from Scott
' Edwards Electronics. Feel free to modify/use this code for your
' own applications. Note that this program assumes the use of
' a Serial LCD made after July '96, when SEE added special
' graphics symbols to the firmware.

' >> This program runs at 9600 bps-- be sure there's a jumper
' installed at "B" on the Backpack circuit board, or change the
' baud rate to 2400 in the OPEN statement below.

'============================
' CONSTANTS, SUBS, FUNCTIONS
' ============================
' The program uses the function DIG% to get a particular decimal
' digit of an integer from 0 to 9999. It uses the subroutine
' bigLCD to display a decimal value from 0 to 9999 in big #s on
' the serial LCD screen.

DECLARE FUNCTION DIG% (theValue AS INTEGER, digit AS INTEGER)
DECLARE SUB bigLCD (dispVal AS INTEGER)
DEFINT A-Z              ' All integers unless otherwise defined.
CONST iPre = 254        ' Backpack instruction prefix.
CONST LCDcls = 1        ' Instruction to clear LCD.

'============================
'    INITIALIZATION
' ============================
' The bigLCD subroutine uses the array symbSeq() as a lookup table
' holding 4-byte sequences of symbols used to draw the big-number
' characters. The "FOR i=..FOR j=.." loop below loads the symbols
' from DATA statements at the end of the program into the array.
' The array lineOrg holds the beginning addresses of each of the
' LCD's four lines, 0-3.
CLS
DIM SHARED symSeq(39) AS STRING * 4
DIM SHARED lineOrg(3) AS INTEGER
lineOrg(0) = 128: lineOrg(1) = 192
lineOrg(2) = 148: lineOrg(3) = 212

' Open COM1 at 9600 baud with all handshaking turned off. Then
' print the "clear-screen" sequence to the LCD.
OPEN "com1:9600,N,8,1,CD0,CS0,DS0,OP0" FOR OUTPUT AS #1
PRINT #1, CHR$(iPre); CHR$(LCDcls);
```

```
' Load the symSeq array with symbols stored in the DATA statements.
FOR i = 0 TO 39
  FOR j = 0 TO 3
    READ temp
    symSeq(i) = CHR$(temp) + symSeq(i)
  NEXT
NEXT

'============================
'         DEMO LOOP
' ============================
' Once the program is set up, all it takes to display an integer
' from 0 to 9999 in 1" tall numbers on the display is to feed
' that number to the subroutine "bigLCD." The loop below shows
' this by counting from 0 to 9999.

FOR i = 0 TO 9999
  bigLCD (i)            ' Display the value of i in big numbers.
  SLEEP 1               ' Pause 1 second.
NEXT
CLOSE                   ' When program is done, close serial port.

'============================
'        GRAPHICS DATA
' ============================
' The numbers below are the graphics symbols (0-7) that the program
' uses to 'draw' the big numbers on the LCD screen.
DATA 1,5,5,0,7,6,0,7,1,5,5,0,1,5,5,0,6,7,7,6
DATA 5,5,5,6,7,5,5,0,6,5,5,2,1,5,5,0,1,5,5,0
DATA 6,7,7,6,7,6,7,7,3,4,7,7,3,4,7,7,6,7,7,6
DATA 1,4,4,2,1,4,0,6,3,0,7,7,3,4,4,2,6,7,7,6
DATA 6,7,7,6,7,6,7,7,7,7,5,0,1,5,7,7,6,5,5,2
DATA 6,7,7,7,6,7,7,6,7,6,7,7,1,5,5,0,6,3,5,2
DATA 3,4,4,2,4,6,4,7,4,4,4,6,3,4,4,2,6,7,7,7
DATA 3,4,4,2,3,4,4,2,7,6,7,7,3,4,4,2,3,4,4,7

SUB bigLCD (dispVal AS INTEGER)
FOR LCDline = 0 TO 3
  PRINT #1, CHR$(iPre); CHR$(lineOrg(LCDline));
    FOR digit = 3 TO 0 STEP -1
      currentDigit = DIG%(dispVal, digit)
      IF (dispVal < 10 ^ digit) AND (digit <> 0) THEN
        PRINT #1, "     ";
      ELSE
        index = (10 * LCDline) + currentDigit
        PRINT #1, symSeq(index); " ";
      END IF
    NEXT
NEXT
END SUB
```

```
FUNCTION DIG% (theValue AS INTEGER, digit AS INTEGER)
DEFINT A-Z
' This function takes an integer value (-32767 to +32767) and a digit
' number (0 to 4). It returns an integer with the appropriate decimal
' digit, where 0 is the ones place, 1 the tens, 2 the hundreds, etc.
DIG = (ABS(theValue) \ (10 ^ ABS(digit))) MOD 10
END FUNCTION
```