



Column #32, October 1997 by Jon Williams:

Custom Characters

While character-based LCDs do not allow us to control the individual bits in the display, they do allow us to generate our own characters. And these characters don't have to be letters and numbers; they can be anything we desire and our imagination can create. The only restraint is that our imagination must fit into a 5x7 bit character cell. Our custom character patterns are stored in the display area called CGRAM (Character Generator RAM). There are 64 bytes of CGRAM, thus allowing us to create and display up to eight custom characters.

So, what's the trick? It starts with the design. As I just stated, your custom character must fit into a 5x7 grid. Each character actually uses eight bytes. The first seven bytes (lower five bits only) are the character pattern, and the eighth byte should be set to \$00. The eighth line of the display is where the underline cursor resides. If you use this area in your character, the underline cursor could become lost in the display.

Figure 32.1 shows the creation of a custom character that is used in the demo program (Program Listing 32.1). Notice that the boxes with the heavy outlines are the ones where our pixels will go. The diagram also shows the logical progression from design to code: fill in the desired boxes, write the binary value of the character (1 for a pixel on, 0 for off), then transfer to hex for your program. You don't have to program your EEPROM or

Figure 32.1: Creation of custom character used in the demo program

Byte 1	0	0	0	□	■	■	■	□	%00001110 or \$OE
Byte 2	0	0	0	■	■	■	■	■	%00001111 or \$OF
Byte 3	0	0	0	■	■	■	□	□	%00011100 or \$1C
Byte 4	0	0	0	■	■	□	□	□	%00011000 or \$18
Byte 5	0	0	0	■	■	■	□	□	%00011100 or \$1C
Byte 6	0	0	0	■	■	■	■	■	%00001111 or \$OF
Byte 7	0	0	0	□	■	■	■	□	%00001110 or \$OE
Byte 8	0	0	0	0	0	0	0	0	%00000000 or \$00

DATA statement in hex. In fact, expressing this information in binary can help others to visualize your characters before they actually see them on the LCD.

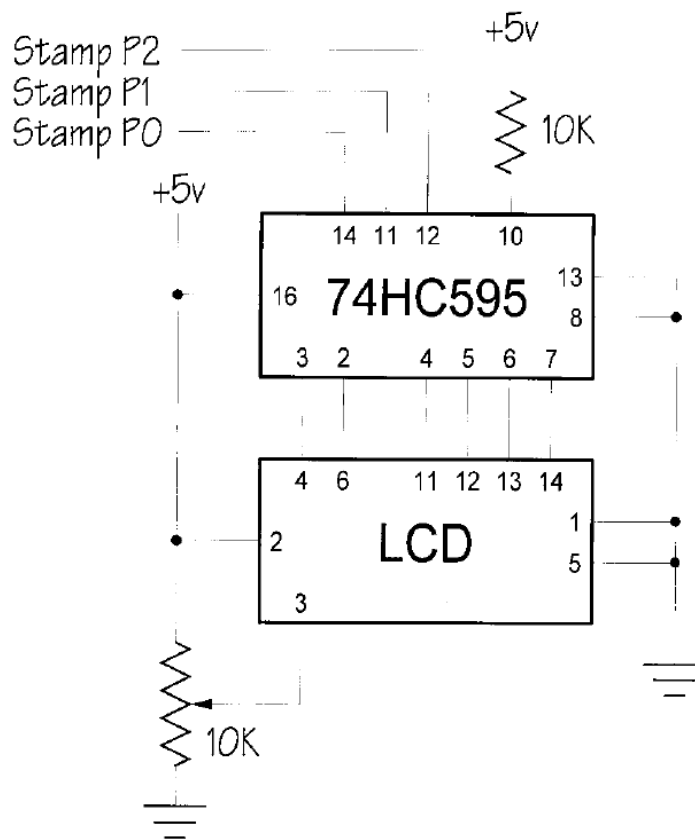
Once we've determined what our character(s) will look like and have stored the patterns in EEPROM, it's a simple matter of downloading this data to the LCD. You'll see in Listing 32.1 that when the LCD initialization is complete, the character patterns are downloaded in one block using a loop. All we have to do is set the starting address in CGRAM (usually \$00) and write the data. The process is identical to writing to the display (DDRAM). A nice byproduct of setting the display address to auto increment is that it works for the CGRAM, too. This is why we only need to set the CGRAM address once before downloading all of our custom character data.

Once your custom characters are downloaded, you need to perform an operation that accesses the DDRAM so that you can start displaying characters. I usually do this by moving the ClrLCD operation from the end of the initialization code to just after the custom character downloading. To display any of your new characters, you just write the character code (0 to 7) to the desired position in the display (DDRAM).

Animation

Most everyone knows that animated cartoons are created by painting individual cells and displaying them in rapid succession to create the illusion of motion. With a bit of imagination, we can do the same thing in our LCD. While we're not going to worry the good folks at Disney, we can certainly spice up our projects.

Figure 32.1: Hookup diagram with 74HC595 and LCD



The program in Listing 32.1 creates an animation using three custom characters. These characters depict a Pacman-type creature with its mouth open in varying stages of aggression. We create the illusion of motion by rapidly displaying the different characters in the same location.

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

The program is very straightforward. First, we use a loop to display a message. Then, the fun begins. The animation is handled by two loops; one embedded within the other: `index1` determines the display position for the creature and `index2` determines what character — “cell” if you will — to display. We use `LOOKUP` to get the next animation character and then we add a twist. Notice that the last character in the animation cycle is actually a character from the second message. The effect is pretty nifty. As the chomping mouth moves from left to right, it eats one message and reveals another.

Not bad, huh? You can do it, too. It just takes a little planning (where have you heard me say that before?) and a bit of code. Let your imagination run wild and you’ll be happily surprised with what you come up with.

Using CGRAM As External Memory

There are times when even Stamp 2 users wish they had more RAM. If your project uses an LCD, you have it — up to 64 bytes! That’s quite a lot in Stamp terms. By now we’ve had plenty of practice writing to the DDRAM and CGRAM areas; reading it back is no more complicated. The tradeoff for the extra memory space is the use of an additional Stamp pin (to control the LCD Read/Write line [pin 5]) and a little bit of code.

Listing 32.2 demonstrates the use of CGRAM as general-purpose memory. To read from the memory, you simply set the address and call `LCDrd`. This routine reads back the data — just like a write — one nibble at a time. The data from the memory location will be contained in the byte variable `char`. Note that the read routine is general purpose in nature and can also be used to read the contents of the display (DDRAM). You can use this technique to determine what character is in a specific location.

Be a little cautious if you decide to mix custom characters and use some of the CGRAM as external memory. If you write to a CGRAM location that is part of a displayed character, the display will change to reflect the new contents of the CGRAM. This effect can actually be very useful for complex animations or displaying more than eight custom characters (but not all at once). Again, be careful if you decide to mix.

Let’s Get Fancy

While looking back through all the LCD code I’ve written over the past few years, I came across a program that I think is worth sharing in light of my fanaticism toward formatting. One of the things that you may have noticed is that most of my LCD messages are completely capitalized. Thankfully, LCD messages are not governed by the rules of Netiquette — or any rules for that matter — otherwise, I’d certainly be flamed

for SHOUTING. My problem is the look of lowercase characters with decenders; the letters “g,” “j,” “q,” “p,” and “y.” The 5x7 font format does not do these characters justice and, in my opinion, they look awful. What to do?

Admittedly, they’re a bit difficult to find anymore, but there are LCDs available with a 5x10 font format. The uppercase characters are mapped just like the 5x7 models and the lower case characters with decenders are mapped to look like you’d expect to see them — no more “scrunchy” decenders. Access to the 5x10 font requires just a small addition to the initialization sequence (see Listing 32.3).

If you use this code with a 5x7 type display, the decenders will be cut off, making them less readable than the 5x7 versions. So, be careful. If you don’t have a 5x10 display, do you have a two-line 5x7 display? If you do, this code will work. Keep in mind that you will lose the two-line capability (the upper bits of the second line are used for the decenders) and that the decenders will have a small break in them due to the space between the lines.

The only problem with using the 5x10 font is that the characters with decenders are not mapped to their ASCII codes. To get around this, the program defines the new characters in constant declarations to make them more accessible. The first message in Listing 32.3 uses one of the constants (‘_y’ instead of “y”).

A problem that the declarations do not directly solve, however, is properly displaying data that is not embedded in the program (i.e., a character provided by SERIN). Since there are only five characters to deal with, I created a little translation routine with LOOKDOWN and LOOKUP. The LOOKDOWN converts a decender character to an index that is used by the LOOKUP to get the new character code. A nice side effect of LOOKDOWN and LOOKUP is that the target variable is not modified if the search variable is out of range. This allows a non-decender character (with an ASCII code greater than 4) to pass through the routine unchanged. If you’re going to embed all your messages, you should remove the Xlate routine to save space.

In case you’re wondering, yes, you can generate custom characters with the 5x10 font. Since they’re bigger, there is only room for four characters. You should download 16 bytes for each character: the first 10 containing your character map, the last six padded with zeros. This will allow you to download all of your data in one block without having to worry about the character map boundaries or resetting the CGRAM address. If your program only needs one custom character, just download 11 bytes: 10 for the character and \$00 to make sure that the cursor line is clear.

A Three-Wire Interface

The Scott Edwards LCD backpack uses one I/O line and direct connection uses six. Is there anything in between? It turns out that there is. While working on a recent BS2 project, I hit a bit of a snag; I ran out of I/O lines! Not completely, but I didn't have enough for direct connection to an LCD. Since cost was a concern for my customer, I was not able to use an LCD backpack. After stewing for a while, I remembered that Karl Lunt had connected an LCD to the Motorola SPI by using a \$2.00 74HC595 (see "Amateur Robotics," *Nuts & Volts*, June '94).

I plugged a '595 into my breadboard and connected it to a BS2 based on Karl's schematic. With some code modifications to the LCD routines, it works great. The best part for my particular project is that it was already using SHIFTOUT and SHIFTIN to communicate with other devices. And, since the clock and data lines can be shared, my project needed only one additional Stamp pin to send data to the LCD.

When using the 74HC595, you'll use three pins: one for the shift clock; one for the shifted data; and one for the latch clock. The line for the latch clock is the only one that cannot be shared with other devices (except daisy-chained '595s). A data byte is transferred to the '595 using SHIFTOUT. The byte will appear on the outputs of the 74HC595 when the latch clock line is brought high. This line is normally held low so that the outputs do not ripple during the data shift. PULSOUT is used to latch the outputs.

Listing 32.4 shows how to send data to the LCD through a 74HC595. The program demonstrates all the typical features we'd use, and even downloads a custom character. Notice that there are changes in the LCD initialization section and that the LCD output routine was completely rewritten for the 74HC595.

You may recall from my article last month that I place a nibble on the LCD bus and then cause a high-to-low transition on the Enable line. Since the enable line is part of the 74HC595 outputs, we are forced to send each nibble twice. The first transfer places the nibble on the data bus, the second causes the Enable transition so that the nibble can be read by the LCD. Just keep in mind that when using this technique, you cannot read from the LCD.

LCD Wrap-Up

Scott Edwards has covered the use of LCD serial backpacks in previous issues and, over the last two installments, I've shown you how to connect and control an LCD directly or through a 74HC595. The point is that, when it comes to using LCDs, you have choices. The choice you make will depend on the circumstances of your project.

Connecting directly is certainly inexpensive — dollar-wise — but, as you've seen, you'll pay for the financial savings in I/O pin use and code space. Using a serial backpack is incredibly easy and doesn't use much code, but costs you more money. Using a 74HC595 falls somewhere in between. So, what should you use? Use whatever works best for the project in terms of I/O use, code space, and cost. Experiment with all of the techniques we've covered here so that you're ready to use them when the need arises.

```
' Program Listing 32.1
' Nuts & Volts: Stamp Applications, October 1997

' ---- [ Title ]-----
'
' File..... LCDCHRS.BAS
' Purpose... Stamp 1 -> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 27 AUG 1997

' ---- [ Program Description ]-----
'
' This program demonstrates the generation of custom characters for an
' LCD display that uses the Hitachi HD44780 controller. The LCD used
' to test this program was the Optrex DMC-16106 (16x1).
'
' LCD Connections:
'
' LCD      (Function)      Stamp
' -----
' pin 1      Vss           Gnd
' pin 2      Vdd           +5
' pin 3      Vo            Gnd (or wiper of 10K pot)
' pin 4      RS            Pin 4
' pin 5      R/W           Gnd
' pin 6      E             Pin 5
' pin 7      DB0           Gnd
' pin 8      DB1           Gnd
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' pin 9          DB2          Gnd
' pin 10         DB3          Gnd
' pin 11         DB4          Pin 0
' pin 12         DB5          Pin 1
' pin 13         DB6          Pin 2
' pin 14         DB7          Pin 4

' ---- [ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 27 AUG 97 : Updated for Nuts & Volts

' ---- [ Constants ]-----
'
SYMBOL  RS      = 4          ' Register Select (1 = char)
SYMBOL  E       = 5          ' LCD enable pin (1 = enabled)

' LCD control characters
'
SYMBOL  ClrLCD  = $01        ' clear the LCD
SYMBOL  CrsrHm  = $02        ' move cursor to home position
SYMBOL  CrsrLf  = $10        ' move cursor left
SYMBOL  CrsrRt  = $14        ' move cursor right
SYMBOL  DispLf  = $18        ' shift displayed chars left
SYMBOL  DispRt  = $1C        ' shift displayed chars right
SYMBOL  DDRam   = $80        ' Display Data RAM control
SYMBOL  CGRam   = $40        ' Char Gen RAM control

' ---- [ Variables ]-----
'
SYMBOL  char     = B1        ' character sent to LCD
SYMBOL  addr     = B2        ' EEPROM address pointer
SYMBOL  newChr   = B3        ' new character for animation
SYMBOL  index1   = B4        ' loop counter
SYMBOL  index2   = B5        ' loop counter

' ---- [ EEPROM Data ]-----
'
EEPROM ("THE BASIC STAMP ") ' preload EEPROM with messages
EEPROM (" IS VERY COOL! ")

' custom character data

EEPROM ($0E,$1F,$1C,$18,$1C,$1F,$0E,$00) ' character 0
EEPROM ($0E,$1F,$1F,$18,$1F,$1F,$0E,$00) ' character 1
EEPROM ($0E,$1F,$1F,$1F,$1F,$1F,$0E,$00) ' character 2
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

' ----[ Initialization ]-----
'
Init:  Dirs = %00111111          ' set 0-5 as outputs
       Pins = %00000000          ' clear the pins

' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500                ' let the LCD settle
        Pins = %0011            ' 8-bit mode
        PULSOUT E, 1
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        Pins = %0010            ' 4-bit mode
        PULSOUT E, 1
        char = %00001100        ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110        ' inc crsr, no disp shift
        GOSUB LCDcmd

        ' download custom character maps to LCD

        char = CGRam            ' point to CG RAM
        GOSUB LCDcmd            ' prepare to write CG data
        FOR index1 = 32 TO 55   ' build 3 custom chars
            READ index1, char    ' get byte from EEPROM
            GOSUB LCDwr          ' put into LCD CG RAM
        NEXT index1

' ----[ Main Code ]-----
'
Start: char = ClrLCD
       GOSUB LCDcmd

       FOR index1 = 0 TO 15
           READ index1, char     ' get character from EEPROM
           GOSUB LCDwr           ' write it
       NEXT
       PAUSE 2000                ' wait 2 seconds

       ' Animation by character replacement
       ' Eat first message and expose second

       FOR index1 = 0 TO 15     ' cover 16 characters
           addr = index1 + 16   ' get new char from 2nd message
           READ addr, newChr
           FOR index2 = 0 TO 4   ' 5 characters in animation cycle
               char = DDRam | index1 ' set new DDRAM address
               GOSUB LCDcmd
           NEXT index2
       NEXT index1

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
        LOOKUP index2, (2,1,0,1,newChr), char
        GOSUB LCDwr          ' write animation character
        PAUSE 50             ' delay between animation chars
    NEXT index2
NEXT index1
PAUSE 1000

GOTO Start          ' do it all over

' ----[ Subroutines ]-----
'
' Send command to the LCD
'
LCDcmd: LOW RS          ' enter command mode
                    ' then write the character

' Write ASCII char to LCD
'
LCDwr:  Pins = Pins & %11010000 ' save 7, 6 and RS; clear bus
        Pins = char / 16 | Pins ' output high nibble
        PULSOUT E, 1           ' strobe the Enable line
        Pins = Pins & %11010000
        Pins = char & $0F | Pins ' output low nibble
        PULSOUT E, 1
        HIGH RS                ' return to character mode
        RETURN

' ----[ Title ]-----
'
' File..... LCDCCHRS.BS2
' Purpose... Stamp 2 -> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 27 AUG 1997

' ----[ Program Description ]-----
'
' This program demonstrates the generation of custom characters for an
' LCD display that uses the Hitachi HD44780 controller. The LCD used
' to test this program was the Optrex DMC-16106 (16x1).
'
' LCD Connections:
'
' LCD          (Function)          Stamp
' -----
' pin 1        Vss                  Gnd
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' pin 2      Vdd      +5
' pin 3      Vo       Gnd (or wiper of 10K pot)
' pin 4      RS       Pin 4
' pin 5      R/W      Gnd
' pin 6      E        Pin 5
' pin 7      DB0      Gnd
' pin 8      DB1      Gnd
' pin 9      DB2      Gnd
' pin 10     DB3      Gnd
' pin 11     DB4      Pin 0
' pin 12     DB5      Pin 1
' pin 13     DB6      Pin 2
' pin 14     DB7      Pin 4

' ---- [ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 27 AUG 97 : Updated for Nuts & Volts

' ---- [ Constants ]-----
'
RS      CON      4      ' Register Select (1 = char)
E       CON      5      ' LCD Enable pin (1 = enabled)

' LCD control characters
'
ClrLCD  CON      $01    ' clear the LCD
CrsrHm  CON      $02    ' move cursor to home position
CrsrLf  CON      $10    ' move cursor left
CrsrRt  CON      $14    ' move cursor right
DispLf  CON      $18    ' shift displayed chars left
DispRt  CON      $1C    ' shift displayed chars right
DDRam   CON      $80    ' Display Data RAM control
CGRam   CON      $40    ' Char Gen RAM control

' ---- [ Variables ]-----
'
char    VAR      Byte   ' character sent to LCD
newChr  VAR      Byte   ' new character for animation
index1  VAR      Byte   ' loop counter
index2  VAR      Byte   ' loop counter

' ---- [ EEPROM Data ]-----
'
Msg1    DATA    "THE BASIC STAMP "  ' preload EEPROM with messages
Msg2    DATA    " IS VERY COOL!  "
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' custom character data

CC0  DATA  $0E,$1F,$1C,$18,$1C,$1F,$0E,$00  ' character 0
CC1  DATA  $0E,$1F,$1F,$18,$1F,$1F,$0E,$00  ' character 1
CC2  DATA  $0E,$1F,$1F,$1F,$1F,$1F,$0E,$00  ' character 2

' -----[ Initialization ]-----
'
Init:  DirL = %00111111  ' set pins 0-5 as outputs
      Outs = $0000      ' clear the pins

' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500        ' let the LCD settle
      OutA = %0011      ' 8-bit mode
      PULSOUT E, 1
      PAUSE 5
      PULSOUT E, 1
      PULSOUT E, 1
      OutA = %0010      ' 4-bit mode
      PULSOUT E, 1
      char = %00001100  ' disp on, crsr off, blink off
      GOSUB LCDcmd
      char = %00000110  ' inc crsr, no disp shift
      GOSUB LCDcmd
      char = ClrLCD
      GOSUB LCDcmd

' download custom character maps to LCD

char = CGRAM            ' point to CG RAM
GOSUB LCDcmd           ' prepare to write CG data
FOR index1 = CC0 TO (CC2 + 7) ' build 3 custom chars
  READ index1, char    ' get byte from EEPROM
  GOSUB LCDwrt        ' put into LCD CG RAM
NEXT

' ----[ Main Code ]-----
'
Start: char = ClrLCD
      GOSUB LCDcmd

      FOR index1 = 0 TO 15
        READ Msg1 + index1, char  ' get character from EEPROM
        GOSUB LCDwrt            ' write it
      NEXT
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
    PAUSE 2000                                ' wait 2 seconds

    ' Animation by character replacement
    ' Eat first message and expose second

    FOR index1 = 0 TO 15                      ' cover 16 characters
      READ Msg2 + index1, newChr             ' get new char from 2nd message
      FOR index2 = 0 TO 4                    ' 5 characters in animation cycle
        char = DDRam | index1               ' set new DDRAM address
        GOSUB LCDcmd
        LOOKUP index2, [2,1,0,1,newChr], char
        GOSUB LCDwr                          ' write animation character
        PAUSE 50                             ' delay between animation chars
      NEXT
    NEXT
    PAUSE 1000

    GOTO Start                                ' do it all over

' ----[ Subroutines ]-----
'
' Send command to the LCD
'
LCDcmd: LOW RS                                ' enter command mode
                                              ' then write the character

' Write ASCII char to LCD
'
LCDwr:  OutA = char.HIGHNIB                  ' output high nibble
        PULSOUT E, 1                        ' strobe the Enable line
        OutA = char.LOWNIB                  ' output low nibble
        PULSOUT E, 1
        HIGH RS                             ' return to character mode
        RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' Program Listing 32.2
' Stamp Applications: Nuts & Volts, October 1997

' ----[ Title ]-----
'
' File..... LCDREAD.BS2
' Purpose... Stamp 2 <-> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 27 AUG 1997

' ----[ Program Description ]-----
'
' This program demonstrates the ability to use the CGRAM area as external
' memory for the Stamp 2. When used as external memory, this RAM should'nt
' be used for custom characters. The Hitachi HD44780 display driver
' provides 64 bytes of CGRAM.
'
' LCD Connections:
'
' LCD          (Function)          Stamp
' -----
' pin 1        Vss                 Gnd
' pin 2        Vdd                 +5
' pin 3        Vo                  Gnd (or wiper of 10K pot)
' pin 4        RS                  P4
' pin 5        R/W                 P6
' pin 6        E                   P5
' pin 7        DB0                 Gnd
' pin 8        DB1                 Gnd
' pin 9        DB2                 Gnd
' pin 10       DB3                 Gnd
' pin 11       DB4                 P0
' pin 12       DB5                 P1
' pin 13       DB6                 P2
' pin 14       DB7                 P4

' ----[ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 27 AUG 97 : Added LCDrd routine

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

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

RS      CON      4          ' Register Select      (1 = char)
E       CON      5          ' LCD Enable pin      (1 = enabled)
RW      CON      6          ' Read/Write control  (0 = write)

' LCD control characters
'
ClrLCD  CON      $01        ' clear the LCD
CrsrHm  CON      $02        ' move cursor to home position
CrsrLf  CON      $10        ' move cursor left
CrsrRt  CON      $14        ' move cursor right
DispLf  CON      $18        ' shift displayed chars left
DispRt  CON      $1C        ' shift displayed chars right
DDRam   CON      $80        ' Display Data RAM control
CGRam   CON      $40        ' Char Gen RAM control

' ---- [ Variables ] -----
'
char    VAR      Byte       ' character sent to LCD
addr    VAR      Byte       ' address to write to / read from
index   VAR      Byte       ' loop counter

rVar    VAR      Word       ' for random number
tVal    VAR      Byte       ' test value to write / read

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

' ---- [ Initialization ] -----
'
Init:   DirL = %01111111    ' set pins 0-6 as outputs
        Outs = $0000        ' clear the pins

' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500           ' let the LCD settle
        OutA = %0011        ' 8-bit mode
        PULSOUT E, 1
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        OutA = %0010        ' 4-bit mode
        PULSOUT E, 1
        char = %00001100    ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110    ' inc crsr, no disp shift
        GOSUB LCDcmd

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' ---- [ Main Code ] -----
'
Start: char = ClrLCD           ' clear the display
      GOSUB LCDcmd

      RANDOM rVar             ' generate random number
      addr = rVar.LOWBYTE & $3F ' create address (0 to 63)
      tVal = rVar.HIGHBYTE     ' create test value (0 to 255)

      FOR index = 0 TO 3      ' put "OUT=" in LCD
        LOOKUP index, ["OUT="], char
        GOSUB LCDwr
      NEXT

      GOSUB WrVal             ' write tVal to LCD

      DEBUG "Addr=", DEC2 addr, " " ' show address in debug window
      DEBUG "Out=" , DEC3 tVal, " " ' show outgoing value

      char = CGRAM + addr     ' set CGRAM pointer
      GOSUB LCDcmd
      char = tVal
      GOSUB LCDwr             ' move the value to CGRAM

      PAUSE 100               ' wait a bit, then go get it

      char = CGRAM + addr     ' set CGRAM pointer
      GOSUB LCDcmd
      GOSUB LCDrd
      tVal = char

      char = DDRam + 8        ' move to 9th column
      GOSUB LCDcmd

      FOR index = 0 TO 2      ' put "IN=" in LCD
        LOOKUP index, ["IN="], char
        GOSUB LCDwr
      NEXT

      GOSUB WrVal

      DEBUG "In=", DEC3 tVal, cr ' display ncoming value

      PAUSE 1000
      GOTO Start

' ---- [ Subroutines ] -----
'
' Write byte value (3 digits) to LCD
' -- be sure to set DDRAM address before calling
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
'  
WrVal: FOR index = 2 TO 0          ' display digits left to right  
      char = tVal DIG index + 48  ' convert digit to ASCII  
      GOSUB LCDwr  
    NEXT  
    RETURN  
  
' Send command to the LCD  
'  
LCDcmd: LOW RS                    ' enter command mode  
                                           ' then write the character  
  
' Write ASCII char to LCD  
'  
LCDwr:  OutA = char.HIGHNIB        ' output high nibble  
        PULSOUT E, 1              ' strobe the Enable line  
        OutA = char.LOWNIB        ' output low nibble  
        PULSOUT E, 1  
        HIGH RS                   ' return to character mode  
        RETURN  
  
LCDrd:  HIGH RS                   ' data command  
        HIGH RW                   ' read  
        DirA = %0000              ' make data lines inputs  
        HIGH E                    ' get high nibble  
        char.HIGHNIB = InA  
        LOW E  
        HIGH E                    ' get low nibble  
        char.LOWNIB = InA  
        LOW E  
        DirA = %1111              ' return data lines to outputs  
        LOW RW  
        RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' Listing 32.3
' Nuts & Volts: Stamp Applications, October 1997

' ----[ Title ]-----
'
' File..... LCD_5x10.BAS
' Purpose... Stamp 1 -> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 28 AUG 1997

' ----[ Program Description ]-----
'
' This program demonstrates the various standard features of a 1x16 LCD
' display that uses the Hitachi HD44780 controller. The display is con-
' figured to use the 5x10 character set, thus allowing proper decenders on
' the characters "g", "j", "q", "p" and "y".
'
' LCD Connections:
'
' LCD          (Function)          Stamp
' -----
' pin 1        Vss                  Gnd
' pin 2        Vdd                  +5
' pin 3        Vo                    Gnd (or wiper of 10K pot)
' pin 4        RS                    Pin 4
' pin 5        R/W                   Gnd
' pin 6        E                      Pin 5
' pin 7        DB0                   Gnd
' pin 8        DB1                   Gnd
' pin 9        DB2                   Gnd
' pin 10       DB3                   Gnd
' pin 11       DB4                   Pin 0
' pin 12       DB5                   Pin 1
' pin 13       DB6                   Pin 2
' pin 14       DB7                   Pin 4

' ----[ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 25 JUL 97 : Updated for Nuts & Volts
' 28 AUG 97 : Added ASCII translation routine of decender characters
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' -----[ Constants ]-----
-
'
SYMBOL E      = 5          ' LCD enable pin (1 = enabled)
SYMBOL RS     = 4          ' Register Select (1 = char)

' LCD control characters
'
SYMBOL ClrLCD = $01        ' clear the LCD
SYMBOL CrsrHm = $02        ' move cursor to home position
SYMBOL CrsrLf = $10        ' move cursor left
SYMBOL CrsrRt = $14        ' move cursor right
SYMBOL DispLf = $18        ' shift displayed chars left
SYMBOL DispRt = $1C        ' shift displayed chars right
SYMBOL DDRam  = $80        ' Display Data Ram control
SYMBOL CGRam  = $40        ' Char Gen Ram control

' DDROM codes for characters with decenders
'
SYMBOL _g     = $E7
SYMBOL _j     = $EA
SYMBOL _p     = $F0
SYMBOL _q     = $F1
SYMBOL _y     = $F9

' ----[ Variables ]-----
'
SYMBOL char    = B1          ' character sent to LCD
SYMBOL index   = B2          ' loop counter

' ----[ EEPROM Data ]-----
'
' test messages with decenders

EEPROM ("Heeeere's Jonn",_y,"!")
EEPROM ("Jolly good!")      ' needs ASCII translation
EEPROM ("quick... jump!")   ' " " "

' ----[ Initialization ]-----
'
Init:  Pins = %00000000      ' clear the pins
      Dirs = %00111111      ' set 0-5 as outputs

' Initialize the LCD (Hitatchi HD44780 controller)
'
LCDini: PAUSE 500           ' let the LCD settle
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
Pins = %0011          ' 8-bit mode
PULSOUT E, 1
PAUSE 5
PULSOUT E, 1
PULSOUT E, 1
Pins = %0010          ' 4-bit mode
PULSOUT E, 1
char = %00100100      ' 5x10 font
GOSUB LCDcmd
char = %00001100      ' disp on, crsr off, blink off
GOSUB LCDcmd
char = %00000110      ' inc crsr, no disp shift
GOSUB LCDcmd

' ----[ Main Code ]-----
'
Start: char = ClrLCD
GOSUB LCDcmd

' message 1

FOR index = 0 TO 15
  READ index,char      ' get char from EEPROM
  GOSUB LCDwr          ' write it
NEXT index

PAUSE 1500

char = ClrLCD
GOSUB LCDcmd

' message 2

FOR index = 16 TO 26
  READ index,char
  GOSUB XLate          ' translate from ASCII
  GOSUB LCDwr
NEXT index

PAUSE 1500

char = ClrLCD
GOSUB LCDcmd

' message 3

FOR index = 27 TO 40
  READ index,char
  GOSUB XLate          ' translate from ASCII
  GOSUB LCDwr
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
        NEXT index

        PAUSE 1500

        GOTO Start                ' do it all over

' ----[ Subroutines ]-----
'
' Translate ASCII code for characters with decenders
' -- has no affect on other characters
'
Xlate: LOOKDOWN char, ("g","j","q","p","y"), char
        LOOKUP char, (_g,_j,_q,_p,_y), char
        RETURN

' Send command to the LCD
'
LCDcmd: LOW RS                    ' enter command mode
                                     ' then write the character

' Write ASCII char to LCD
'
LCDwr:  Pins = Pins & %11010000    ' save 7, 6 and RS; clear bus
        Pins = char / 16 | Pins    ' output high nibble
        PULSOUT E, 1              ' strobe the Enable line
        Pins = Pins & %11010000
        Pins = char & $0F | Pins   ' output low nibble
        PULSOUT E, 1
        HIGH RS                    ' return to character mode
        RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' ---- [ Title ]-----  
,  
' File..... LCD_5x10.BS2  
' Purpose... Stamp 2 -> LCD (4-bit interface)  
' Author.... Jon Williams  
' E-mail.... jonwms@aol.com  
' WWW..... http://members.aol.com/jonwms  
' Started... 16 JUL 1994  
' Updated... 27 AUG 1997  
  
' ---- [ Program Description ]-----  
,  
' This program demonstrates the various standard features of a 1x16 LCD  
' display that uses the Hitachi HD44780 controller. The display is con-  
' figured to use the 5x10 character set, thus allowing proper decenders on  
' the characters "g", "J", "q", "p" and "y".  
,  
' LCD Connections:  
,  
' LCD          (Function)          Stamp  
' -----  
' pin 1         Vss                 Gnd  
' pin 2         Vdd                 +5  
' pin 3         Vo                  Gnd (or wiper of 10K pot)  
' pin 4         RS                  Pin 4  
' pin 5         R/W                 Gnd  
' pin 6         E                   Pin 5  
' pin 7         DB0                 Gnd  
' pin 8         DB1                 Gnd  
' pin 9         DB2                 Gnd  
' pin 10        DB3                 Gnd  
' pin 11        DB4                 Pin 0  
' pin 12        DB5                 Pin 1  
' pin 13        DB6                 Pin 2  
' pin 14        DB7                 Pin 4  
  
' ---- [ Revision History ]-----  
,  
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months  
' 08 AUG 96 : Trimmed code to save space -- no performance changes!  
' 27 AUG 97 : Updated for Nuts & Volts  
  
' ---- [ Constants ]-----  
,  
RS      CON      4          ' Register Select (1 = char)  
E       CON      5          ' LCD Enable pin (1 = enabled)  
  
' LCD control characters
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

'
ClrLCD  CON    $01                ' clear the LCD
CrsrHm  CON    $02                ' move cursor to home position
CrsrLf  CON    $10                ' move cursor left
CrsrRt  CON    $14                ' move cursor right
DispLf  CON    $18                ' shift displayed chars left
DispRt  CON    $1C                ' shift displayed chars right
DDRam   CON    $80                ' Display Data RAM control

' DDROM codes for characters with decenders
'
_g      CON    $E7
_j      CON    $EA
_p      CON    $F0
_q      CON    $F1
_y      CON    $F9

' ---- [ Variables ]-----
'
char    VAR    Byte                ' character sent to LCD
index   VAR    Byte                ' loop counter

' ---- [ EEPROM Data ]-----
'
Msg     DATA   "Heeeere's Jonn",_y,"!" ' preload EEPROM with message

' ---- [ Initialization ]-----
'
Init:   DirL = %00111111          ' set pins 0-5 as outputs
        Outs = $0000

' Initialize the LCD (Hitatchi HD44780 controller)
'
LCDini: PAUSE 500                ' let the LCD settle
        OutA = %0011              ' 8-bit mode
        PULSOUT E, 1
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        OutA = %0010              ' 4-bit mode
        PULSOUT E, 1
        char = %00100100          ' 5x10 font
        GOSUB LCDcmd
        char = %00001100          ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110          ' inc crsr, no disp shift
        GOSUB LCDcmd

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
char = ClrLCD
GOSUB LCDcmd

' ----[ Main Code ]-----
'
Start:  FOR index = 0 TO 15
        READ Msg + index, char      ' get character from EEPROM
        GOSUB LCDwrt                ' write it
        PAUSE 50                    ' delay between chars
    NEXT
    PAUSE 2000                      ' wait 2 seconds

    char = CrsrHm                    ' move the cursor home
    GOSUB LCDcmd

    char = %00001110                ' turn the cursor on
    GOSUB LCDcmd
    PAUSE 500

    char = CrsrRt
    FOR index = 1 TO 15              ' move the cursor accross display
        GOSUB LCDcmd
        PAUSE 100
    NEXT

    FOR index = 14 TO 0              ' go backward by moving to
        char = DDRam + index        ' a specific address
        GOSUB LCDcmd
        PAUSE 100
    NEXT

    char = %00001101                ' cursor off, blink on
    GOSUB LCDcmd
    PAUSE 2000

    char = %00001100                ' blink off
    GOSUB LCDcmd

    FOR index = 1 TO 10              ' flash display
        char = char ^ %00000100    ' toggle display bit
        GOSUB LCDcmd
        PAUSE 250
    NEXT
    PAUSE 1000

    FOR index = 1 TO 16              ' shift display
        char = DispRt
        GOSUB LCDcmd
        PAUSE 100
    NEXT
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
    PAUSE 1000

    FOR index = 1 TO 16          ' shift display back
      char = DispLf
      GOSUB LCDcmd
      PAUSE 100
    NEXT
    PAUSE 1000

    char = ClrLCD                ' clear the LCD
    GOSUB LCDcmd
    PAUSE 500

    GOTO Start                  ' do it all over

' ----[ Subroutines ]-----
'
' Translate ASCII code for characters with decenders
' -- has no affect on other characters
'
Xlate: LOOKDOWN char, ["g","j","q","p","y"], char
      LOOKUP char, [_g,_j,_q,_p,_y], char
      RETURN

' Send command to the LCD
'
LCDcmd: LOW RS                  ' enter command mode
      ' then write the character

' Write ASCII char to LCD
'
LCDwr:  OutA = char.HIGHNIB      ' output high nibble
      PULSOUT E, 1              ' strobe the Enable line
      OutA = char.LOWNIB        ' output low nibble
      PULSOUT E, 1
      HIGH RS                   ' return to character mode
      RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' Program Listing 32.3
' Stamp Applications: Nuts & Volts, September 1997

' ----[ Title ]-----
'
' File..... LCDDEMO2.BAS
' Purpose... Stamp 1 -> Multi-line LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 25 JUL 1997

' ----[ Program Description ]-----
'
' This program demonstrates the various standard features of a 2x16 LCD
' display that uses the Hitachi HD44780 controller.
'
' LCD Connections:
'
' LCD          (Function)          Stamp
' -----
' pin 1        Vss                  Gnd
' pin 2        Vdd                  +5
' pin 3        Vo                    wiper of 10K pot
' pin 4        RS                    Pin 4
' pin 5        R/W                   Gnd
' pin 6        E                      Pin 5
' pin 7        DB0                   Gnd
' pin 8        DB1                   Gnd
' pin 9        DB2                   Gnd
' pin 10       DB3                   Gnd
' pin 11       DB4                   Pin 0
' pin 12       DB5                   Pin 1
' pin 13       DB6                   Pin 2
' pin 14       DB7                   Pin 4

' ----[ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 25 JUL 97 : Updated for Nuts & Volts

' ----[ Constants ]-----
'
SYMBOL  RS      = 4                ' Register Select (1 = char)
SYMBOL  E      = 5                ' LCD enable pin (1 = enabled)
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' LCD control characters
'
SYMBOL ClrLCD = $01           ' clear the LCD
SYMBOL CrsrHm = $02         ' move cursor to home position
SYMBOL CrsrLf = $10        ' move cursor left
SYMBOL CrsrRt = $14        ' move cursor right
SYMBOL DispLf = $18        ' shift displayed chars left
SYMBOL DispRt = $1C        ' shift displayed chars right
SYMBOL DDRam = $80         ' Display Data RAM control
SYMBOL Line1 = $00         ' starting address of line 1
SYMBOL Line2 = $40         ' starting address of line 2

' ----[ Variables ]-----
'
SYMBOL char = B1           ' character sent to LCD
SYMBOL index = B2         ' loop counter

' ----[ EEPROM Data ]-----
'
      EEPROM ("This is Line 1")   ' preload EEPROM with messages
      EEPROM ("This is Line 2")

' ----[ Initialization ]-----
'
Init:  Dirs = %00111111       ' set 0-5 as outputs
       Pins = %00000000     ' clear the pins

' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500             ' let the LCD settle
        Pins = %0011         ' 8-bit mode
        PULSOUT E, 1
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        Pins = %0010         ' 4-bit mode
        PULSOUT E, 1
        char = %00101000     ' 2-line mode
        GOSUB LCDcmd
        char = %00001100     ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110     ' inc crsr, no disp shift
        GOSUB LCDcmd

' ----[ Main Code ]-----
'
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
Start: char = ClrLCD           ' clear LCD, home cursor
      GOSUB LCDcmd
      PAUSE 1000

L1:   FOR index = 0 TO 13
      READ index, char         ' get character from EEPROM
      GOSUB LCDwr              ' write it
      PAUSE 50                 ' delay between chars
    NEXT
    PAUSE 2000                 ' wait 2 seconds

    char = DDRam + Line2      ' move to line 2
    GOSUB LCDcmd

L2:   FOR index = 14 TO 27
      READ index, char
      GOSUB LCDwr
      PAUSE 50
    NEXT
    PAUSE 2000

    GOTO Start

' ----[ Subroutines ]-----
'
' Send command to the LCD
'
LCDcmd: LOW RS                 ' enter command mode
      ' then write the character

' Write ASCII char to LCD
'
LCDwr: Pins = Pins & %11010000 ' save 7, 6 and RS; clear bus
      Pins = char / 16 | Pins   ' output high nibble
      PULSOUT E, 1             ' strobe the Enable line
      Pins = Pins & %11010000
      Pins = char & $0F | Pins  ' output low nibble
      PULSOUT E, 1
      HIGH RS                  ' return to character mode
      RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' ---- [ Title ]-----
'
' File..... LCDDEMO2.BS2
' Purpose... Stamp 2 -> Multi-line LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 25 JUL 1997
'
' ---- [ Program Description ]-----
'
' This program demonstrates the various standard features of a 2x16 LCD
' display that uses the Hitachi HD44780 controller.
'
' LCD Connections:
'
' LCD          (Function)          Stamp
' -----
' pin 1        Vss                 Gnd
' pin 2        Vdd                 +5
' pin 3        Vo                   wiper of 10K pot
' pin 4        RS                   P4
' pin 5        R/W                  Gnd
' pin 6        E                     P5
' pin 7        DB0                  Gnd
' pin 8        DB1                  Gnd
' pin 9        DB2                  Gnd
' pin 10       DB3                  Gnd
' pin 11       DB4                  P0
' pin 12       DB5                  P1
' pin 13       DB6                  P2
' pin 14       DB7                  P4
'
' ---- [ Revision History ]-----
'
' 16 JUL 94 : Version 1.0 - compilation of code from last 3 months
' 08 AUG 96 : Trimmed code to save space -- no performance changes!
' 25 JUL 97 : Updated for Nuts & Volts
'
' ---- [ Constants ]-----
'
RS      CON      4                ' Register Select (1 = char)
E       CON      5                ' LCD Enable pin (1 = enabled)
' LCD control characters
'
ClrLCD  CON      $01              ' clear the LCD
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

CrsrHm CON    $02      ' move cursor to home position
CrsrLf CON    $10      ' move cursor left
CrsrRt CON    $14      ' move cursor right
DispLf CON    $18      ' shift displayed chars left
DispRt CON    $1C      ' shift displayed chars right
DDRam  CON    $80      ' Display Data RAM control
Line1  CON    $00      ' starting address of line 1
Line2  CON    $40      ' starting address of line 2

' ----[ Variables ]-----
'
char    VAR    Byte    ' character sent to LCD
index  VAR    Byte    ' loop counter

' ----[ EEPROM Data ]-----
'
Msg1    DATA  "This is Line 1"  ' preload EEPROM with messages
Msg2    DATA  "This is Line 2"

' ----[ Initialization ]-----
'
Init:   DirL = %00111111      ' set pins 0-5 as outputs
        Outs = $0000         ' clear the pins

' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500             ' let the LCD settle
        OutA = %0011         ' 8-bit mode
        PULSOUT E, 1
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        OutA = %0010         ' 4-bit mode
        PULSOUT E, 1
        char = %00101000     ' 2-line mode
        GOSUB LCDcmd
        char = %00001100     ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110     ' inc crsr, no disp shift
        GOSUB LCDcmd
        char = ClrLCD
        GOSUB LCDcmd

' ----[ Main Code ]-----
'
Start:  char = ClrLCD         ' clear LCD, home cursor
        GOSUB LCDcmd

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
        PAUSE 1000

L1:    FOR index = Msg1 TO (Msg1 + 13)
        READ index, char           ' get character from EEPROM
        GOSUB LCDwr                ' write it
        PAUSE 50                   ' delay between chars
    NEXT
    PAUSE 2000                      ' wait 2 seconds

    char = DDRam + Line2           ' move to line 2
    GOSUB LCDcmd

L2:    FOR index = Msg2 TO (Msg2 + 13)
        READ index, char
        GOSUB LCDwr
        PAUSE 50
    NEXT
    PAUSE 2000

    GOTO Start

' ---- [ Subroutines ] -----
'
' Send command to the LCD
'
LCDcmd: LOW RS                      ' enter command mode
        ' then write the character

' Write ASCII char to LCD
'
LCDwr:  OutA = char.HIGHNIB         ' output high nibble
        PULSOUT E, 1               ' strobe the Enable line
        OutA = char.LOWNIB        ' output low nibble
        PULSOUT E, 1
        HIGH RS                    ' return to character mode
    RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' Listing 32.4
' Stamp Applications: Nuts & Volts, October 1997

' ----[ Title ]-----
'
' File..... LCD_595.BS2
' Purpose... Stamp 2 -> 74HC595 -> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 27 AUG 1997

' ----[ Program Description ]-----
'
' This program demonstrates the various standard features of a 1x16 LCD
' display that uses the Hitachi HD44780 controller. The LCD is connected
' to the Stamp 2 through a 74HC595 shift register.
'
' Connections:
'   NC = Not Connected
'   PU = Pulled Up to +5V through 10K resistor
'
' Stamp 2   74HC595   LCD       Notes
' -----
' 1          NC
' 2          6       (E)
' 3          4       (RS)
' 4          11      (D4)
' 5          12      (D5)
' 6          13      (D6)
' 7          14      (D7)
' 8          GND
' 9          NC
' 10         PU
' P1         11          Shift clock
' P2         12          Output latch
' 13         GND
' P0         14          Shift data
' 15         NC
' 16         Vcc

' ----[ Revision History ]-----
'
' 25 AUG 97 : Modified standard LCD demo to use 74HC595
' 27 AUG 97 : Added custom character
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

' ---- [ Constants ]-----
'
SData  CON    0           ' 74HC595 serial data (14)
Clk    CON    1           ' 74HC595 shift clock (11)
Latch  CON    2           ' 74HC595 output latch (12)

' LCD control characters
'
ClrLCD  CON    $01        ' clear the LCD
CrsrHm  CON    $02        ' move cursor to home position
CrsrLf  CON    $10        ' move cursor left
CrsrRt  CON    $14        ' move cursor right
DispLf  CON    $18        ' shift displayed chars left
DispRt  CON    $1C        ' shift displayed chars right
DDRam   CON    $80        ' Display Data RAM control
CGRam   CON    $40        ' Char Gen RAM control

' ---- [ Variables ]-----
'
char    VAR    Byte       ' character sent to LCD
temp    VAR    Byte       ' work variable for LCD routines
index   VAR    Byte       ' loop counter

lcd_E   VAR    temp.Bit2   ' LCD Enable pin
lcd_RS  VAR    temp.Bit3   ' Register Select (1 = char)

' ---- [ EEPROM Data ]-----
'
Msg      DATA    0," NUTS & VOLTS ",0 ' preload EEPROM with message

' custom character map
' character code will be 0

Grin0   DATA    %00000000      ' . . . . .
Grin1   DATA    %00001010      ' . . . . * . * .
Grin2   DATA    %00001010      ' . . . . * . * .
Grin3   DATA    %00000000      ' . . . . .
Grin4   DATA    %00010001      ' . . . * . . . *
Grin5   DATA    %00001110      ' . . . . * * * .
Grin6   DATA    %00000110      ' . . . . . * * .
Grin7   DATA    %00000000      ' . . . . .

' ---- [ Initialization ]-----
'
' Initialize the LCD (Hitachi HD44780 controller)
'
LCDini: PAUSE 500                ' let the LCD settle

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
char = %0011          ' 8-bit mode
GOSUB LCDcmd
PAUSE 5
GOSUB LCDcmd
GOSUB LCDcmd
char = %0010          ' put in 4-bit mode
GOSUB LCDcmd
char = %00001100      ' disp on, crsr off, blink off
GOSUB LCDcmd
char = %00000110     ' inc crsr, no disp shift
GOSUB LCDcmd

' download custom character map to LCD

char = CGRam          ' point to CG RAM
GOSUB LCDcmd          ' prepare to write CG data
FOR index = Grin0 TO Grin7
  READ index, char    ' get byte from EEPROM
  GOSUB LCDwr         ' put into LCD CG RAM
NEXT

' ----[ Main Code ]-----
'
Start: char = ClrLCD
GOSUB LCDcmd

FOR index = 0 TO 15
  READ Msg + index, char ' get character from EEPROM
  GOSUB LCDwr           ' write it
  PAUSE 50              ' delay between chars
NEXT
PAUSE 2000              ' wait 2 seconds

char = CrsrHm          ' move the cursor home
GOSUB LCDcmd

char = %00001110      ' turn the cursor on
GOSUB LCDcmd
PAUSE 500

char = CrsrRt
FOR index = 1 TO 15   ' move the cursor accross display
  GOSUB LCDcmd
  PAUSE 100
NEXT

FOR index = 14 TO 0   ' go backward by moving to
  char = DDRam + index ' a specific address
  GOSUB LCDcmd
  PAUSE 100
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

NEXT

char = %00001101          ' cursor off, blink on
GOSUB LCDcmd
PAUSE 2000

char = %00001100          ' blink off
GOSUB LCDcmd

FOR index = 1 TO 10        ' flash display
  char = char ^ %00000100  ' toggle display bit
  GOSUB LCDcmd
  PAUSE 250
NEXT
PAUSE 1000

FOR index = 1 TO 16        ' shift display
  char = DispRt
  GOSUB LCDcmd
  PAUSE 100
NEXT
PAUSE 1000

FOR index = 1 TO 16        ' shift display back
  char = DispLf
  GOSUB LCDcmd
  PAUSE 100
NEXT
PAUSE 1000

char = ClrLCD              ' clear the LCD
GOSUB LCDcmd
PAUSE 500

GOTO Start                 ' do it all over

' ----[ Subroutines ]-----
'
' Send command to the LCD
'
LCDcmd: lcd_RS = 0          ' command mode
        GOTO LCDout

' Write ASCII char to LCD
'
LCDwr:  lcd_RS = 1          ' character mode
        GOTO LCDout

LCDout: temp.HIGHNIB = char.HIGHNIB  ' get high nibble
        lcd_E = 1
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
SHIFTOUT SData, Clk, MSBFIRST, [temp]
PULSOUT Latch, 1
lcd_E = 0 ' drop Enable line low
SHIFTOUT SData, Clk, MSBFIRST, [temp]
PULSOUT Latch, 1
temp.HIGHNIB = char.LOWNIB ' get low nibble
lcd_E = 1
SHIFTOUT SData, Clk, MSBFIRST, [temp]
PULSOUT Latch, 1
lcd_E = 0
SHIFTOUT SData, Clk, MSBFIRST, [temp]
PULSOUT Latch, 1
RETURN
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
' ----[ Title ]-----
'
' File..... LCD_595.BAS
' Purpose... Stamp 1 -> 74HC595 -> LCD (4-bit interface)
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW..... http://members.aol.com/jonwms
' Started... 16 JUL 1994
' Updated... 29 AUG 1997
'
' ----[ Program Description ]-----
'
' This program demonstrates the the connection of a standard LCD to the
' Stamp through a 74HC595 shift register. Note that this code chews up
' nearly all of the BS1 GOSUB stack.
'
' Connections:
'   NC = Not Connected
'   PU = Pulled Up to +5V through 10K resistor
'
' Stamp 1   74HC595   LCD       Notes
' -----
'           1 NC
'           2         6 (E)
'           3         4 (RS)
'           4        11 (D4)
'           5        12 (D5)
'           6        13 (D6)
'           7        14 (D7)
'           8         GND
'           9         NC
'          10         PU
' Pin 1     11         Shift clock
' Pin 2     12         Output latch
'           13 GND
' Pin 0     14         Shift data
'           15 NC
'           16 Vcc
'
' ----[ Revision History ]-----
'
' 25 AUG 97 : Modified standard LCD demo to use 74HC595
' 27 AUG 97 : Added custom character
' 29 AUG 97 : Ported from BS2 to BS1
'
' ----[ Constants ]-----
'
SYMBOL SData = Pin0           ' 74HC595 serial data (14)
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```

SYMBOL Clk      = 1           ' 74HC595 shift clock (11)
SYMBOL Latch   = 2           ' 74HC595 output latch (12)

' LCD control characters
'
SYMBOL ClrLCD  = $01         ' clear the LCD
SYMBOL CrsrHm = $02         ' move cursor to home position
SYMBOL CrsrLf = $10         ' move cursor left
SYMBOL CrsrRt = $14         ' move cursor right
SYMBOL DispLf = $18         ' shift displayed chars left
SYMBOL DispRt = $1C         ' shift displayed chars right
SYMBOL DDRam  = $80         ' Display Data RAM control
SYMBOL CGRam  = $40         ' Char Gen RAM control

' ---- [ Variables ] -----
'
SYMBOL temp    = B0         ' work variable for LCD routines
SYMBOL char    = B2         ' character sent to LCD
SYMBOL index   = B3         ' loop counter
SYMBOL shift   = B4         ' loop counter for SOut

SYMBOL lcd_E   = Bit2       ' LCD Enable pin
SYMBOL lcd_RS  = Bit3       ' Register Select (1 = char)
SYMBOL RS      = Bit8       ' holds our RS value

' ---- [ EEPROM Data ] -----
'
      EEPROM (0," NUTS & VOLTS ",0) ' preload EEPROM with message

' custom character map
' -- character code will be 0

      EEPROM (%00000000)      ' . . . . .
      EEPROM (%00001010)      ' . . . . * . * .
      EEPROM (%00001010)      ' . . . . * . * .
      EEPROM (%00000000)      ' . . . . .
      EEPROM (%00010001)      ' . . . * . . . *
      EEPROM (%00001110)      ' . . . . * * * .
      EEPROM (%00000110)      ' . . . . . * * .
      EEPROM (%00000000)      ' . . . . .

' ---- [ Initialization ] -----
'
Init:  Dirs = %00000111
       Pins = %00000000

' Initialize the LCD (Hitachi HD44780 controller)
'

```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
LCDini: PAUSE 500           ' let the LCD settle
        char = %0011       ' 8-bit mode
        GOSUB LCDcmd
        PAUSE 5
        GOSUB LCDcmd
        GOSUB LCDcmd
        char = %0010       ' put in 4-bit mode
        GOSUB LCDcmd
        char = %00001100   ' disp on, crsr off, blink off
        GOSUB LCDcmd
        char = %00000110   ' inc crsr, no disp shift
        GOSUB LCDcmd

        ' download custom character map to LCD

        char = CGRam       ' point to CG RAM
        GOSUB LCDcmd       ' prepare to write CG data
        FOR index = 16 TO 23
            READ index, char ' get byte from EEPROM
            GOSUB LCDwr     ' put into LCD CG RAM
        NEXT

' ---- [ Main Code ] -----
'
Start:  char = ClrLCD
        GOSUB LCDcmd

        FOR index = 0 TO 15 ' put message on LCD
            READ index, char ' get character from EEPROM
            GOSUB LCDwr     ' write it
        NEXT
        PAUSE 1000

        FOR index = 1 TO 16 ' shift message off screen
            char = DispRt
            GOSUB LCDcmd
        NEXT
        PAUSE 500

        GOTO Start

' ---- [ Subroutines ] -----
'
' Send command to the LCD
'
LCDcmd: RS = 0           ' command mode
        GOTO LCDout

' Write ASCII char to LCD
```

Column #32: More LCDs: Custom Characters, Animation and Another Connection Method

```
'
LCDwr: RS = 1                                ' character mode

LCDout: temp = char & $F0                    ' char.HIGHNIB -> temp.HIGHNIB
        lcd_RS = RS                          ' set RS
        lcd_E = 1
        GOSUB SOut
        lcd_E = 0                            ' blip Enable line
        GOSUB SOut
        temp = char * 16                      ' char.LOWNIB -> temp.HIGHNIB
        lcd_RS = RS                          ' set RS
        lcd_E = 1
        GOSUB SOut
        lcd_E = 0
        GOSUB SOut
        RETURN

SOut:   FOR shift = 1 TO 8                    ' shift 8 bits
        sData = Bit7                          ' MSB first
        PULSOUT Clk, 1                       ' clock the bit
        temp = temp * 2                       ' get next bit
    NEXT
    PULSOUT Latch, 1
    RETURN
```