Summary

The following software shows how to display a bar graph.

```
    \ MI-AN-023            Bar Graph Display
    ANEW BAR.GRAPH.CODE
    HEX
    \ CHARACTER.PATTERN is a defining word which defines <name> to return an xaddr
    \ which is the location of the last byte of a custom character pattern. The
    \ reason characters are stored in reverse order is to allow greater clarity in
    \ the custom pattern definition, code simplicity, and efficiency. To use this
    \ function, supply eight bytes which define the desired binary pattern
    \ then call CHARACTER.PATTERN followed by <name>. Afterward, whenever <name> is
    \ executed,the xaddress of the pattern will be left on the stack.
:   CHARACTER.PATTERN
    <DBUILDS     ( n1..n8\<name> -- )
       C,  C,  C,  C,  C,  C,  C,  C,    \ store each char byte associated w/name
    DOES>        ( -- xaddr )             \ when name is execute the xaddr of the
;                                        \ first character byte is returned.

\ BINARY changes the BASE to binary to allow custom character patterns to be
\ expressed as 1s and 0s.
:  BINARY ( -- | Changes BASE to binary )
   2 BASE !
;

\ INIT.CUSTOM.CHAR when provided with a custom character pattern xaddress and a
\ character code (0..4) will initialize the display with the pattern. After
\ executing this function, <character code> CHAR>DISPLAY commands will result in
\ the display of the custom character.
:  INIT.CUSTOM.CHAR ( xaddr.char.pattern\char.code -- )
   ROT  7  +  -ROT                  \ addr+7\page\custom.char#
   3  SCALE  40  OR                 \ convert char code to command byte addr
   DUP  8  +  SWAP  DO          \ Each custom character is made of 8 bytes, for each
one:
      I  COMMAND>DISPLAY        \ write the location of the current byte
      XDUP  C@  CHAR>DISPLAY    \ write the contents of the current byte
      1XN-                      \ decrement for the next content byte
   LOOP  XDROP
;
```

```
BINARY      \ Binary mode helps clarify the custom character bit patterns.

10000
10000
10000
10000
10000
10000
10000
10000 CHARACTER.PATTERN   1.BAR.PATTERN   \ Define this pattern as 1 bar

10100
10100
10100
10100
10100
10100
10100
10100 CHARACTER.PATTERN   2.BAR.PATTERN   \ Define this pattern as 2 bar

10101
10101
10101
10101
10101
10101
10101
10101 CHARACTER.PATTERN   3.BAR.PATTERN   \ Define this pattern as 3 bar

\ *** Define constants to assign patterns to display char codes 0..4
HEX
0   CONSTANT  1.BAR           \ 0 will be the 1 bar character
1   CONSTANT  2.BARS     \ 1 will be the 2 bar character
2   CONSTANT  3.BARS     \ 2 will be the 3 bar character

\ BAR GRAPH:
\ use 3 special characters: I   I I     I I I
\ the bar graph is a set of stripes, black alternating with white.
\ this gives a resolution of 3*20 = 1 part in 60.
\ 10000           \ 1.bar
\ 10100           \ 2.bars
\ 10101           \ 3.bars


HEX
:  DEFINE.LCD.SPECIAL.CHARS   ( -- )      \ must be executed at startup
      1.BAR.PATTERN 1.BAR  INIT.CUSTOM.CHAR
      2.BAR.PATTERN 2.BARS INIT.CUSTOM.CHAR
      3.BAR.PATTERN 3.BARS INIT.CUSTOM.CHAR
;
DECIMAL

DEFINE.LCD.SPECIAL.CHARS      \ execute this upon each restart to init display!
```

```
60 CONSTANT MAX#STRIPES \ in bar graph; max 3 per character, 20 characters
2 CONSTANT MIN#STRIPES  \ in bar graph; lets user see that graph is present

VHERE   XCONSTANT   LINE.BUFFER     CHARS/DISPLAY.LINE @ 1+ VALLOT
      \ 20 character buffer + cnt, holds bar graph

:  INIT.LINE.BUFFER     ( -- )
      \ 20 character buffer + cnt, holds bar graph
      CHARS/DISPLAY.LINE @ DUP>R LINE.BUFFER C! \ init count
      LINE.BUFFER 1XN+ R> BLANK                 \ init to blanks
;

:   BAR.GRAPH>LINE.BUFFER     ( #stripes -- )
      \ writes to line buffer.
      LINE.BUFFER 1XN+                    \ skip the count byte
      LOCALS{ x&buffer.addr &#stripes }
      INIT.LINE.BUFFER                    \ set count and blank it
      &#stripes 0 MAX MAX#STRIPES MIN     \ clamp
      3 /MOD                              ( --remainder\quotient )
      0 DO  3.BARS x&buffer.addr C!       \ quotient * 3.bars
            x&buffer.addr 1XN+ TO x&buffer.addr
      LOOP
      CASE                                \ handle the remainder...
            1     OF    1.BAR  x&buffer.addr  C!      ENDOF
            2     OF    2.BARS x&buffer.addr C! ENDOF
      ENDCASE
;

1 CONSTANT #1.BAR.GRAPH.ROW   \ row# on lcd display
2 CONSTANT #2.BAR.GRAPH.ROW   \ row# on lcd display

:   DOUBLE.BAR.GRAPH    ( #stripes -- )
      \ writes a 2-line bar graph to display buffer.
      BAR.GRAPH>LINE.BUFFER                       ( -- )
      LINE.BUFFER #1.BAR.GRAPH.ROW 0 $>DISPLAY
      LINE.BUFFER #2.BAR.GRAPH.ROW 0 $>DISPLAY
;

\ *** Demonstration of use:
\ First define a function to initialize the display and bar graph buffer.
: INIT.BAR.GRAPH ( -- )
INIT.DISPLAY
DEFINE.LCD.SPECIAL.CHARS
INIT.LINE.BUFFER
;

\ Execute init, make a 12 bar graph, display the bar graph:
INIT.BAR.GRAPH
12 DOUBLE.BAR.GRAPH
UPDATE.DISPLAY
```


Mosaic Industries

5437 Central Ave Suite 1, Newark, CA 94560 Telephone: (510) 790-8222 Fax: (510) 790-0925