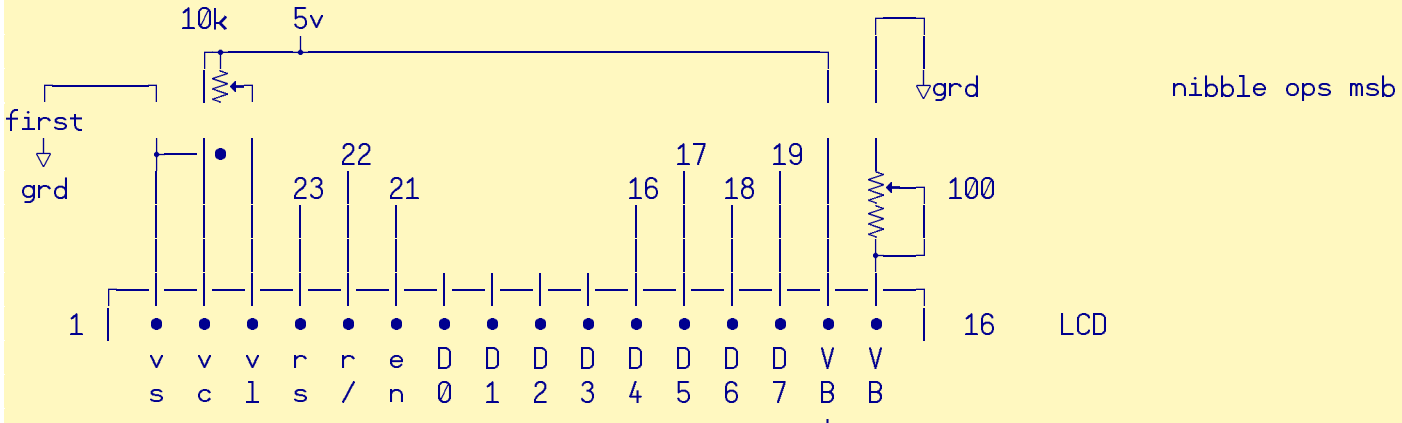
```
'' parallel lcd driver
'' for using a HD44780 based 4 x 20 parallel lcd
'' dan miller 23 july 2006
''
{{
```




```
}}
con


  _clkmode      = xtal1 + pll16x                        ' use crystal x 16
  _xinfreq      = 5_200_000


' ------------------------------------------------------------------------
' IO Pins
' ------------------------------------------------------------------------

                                                ' LCD bus is 16..19
E               =     21                         ' Enable Bit
RW              =     22                        ' Read Write Bit
RS              =     23                       ' Register Select Bit
obj

 delay    : "timing"
 bs2      : "BS2_Functions"
```

```
   dira[RW]~~
   dira[RS]~~
' initialization code here
  waitcnt(clkfreq * 1 + cnt)                    ' let lcd come up
   dira[16..19] := %11111111                    ' Make  Output.

    outa[RS] := 0
    outa[RW] := 0
   outa[E] := 1

    Init_Lcd
     'tester


pub tester  | temp

  repeat  temp  from 0 TO 13                    ' 14 Characters
    IF temp == 19                               ' Check For End Of Line
       Next_Line                                ' Jump To Next Line

    char := Text[temp]
                              ' Read Next Character From EEPROM
    Send_Text

                                  ' Send Character To LCD Display

PUB str(stringptr)

'' Print a zero-terminated string

  repeat strsize(stringptr)
    out(byte[stringptr++])


PUB dec(value) | i

'' Print a decimal number

  if value < 0
    -value
    out("-")
```

```
118     out(lookupz((value <-= 4) & $F : "0".."9", "A".."F"))
119
120
121   PUB bin(value, digits)
122
123   '' Print a binary number
124
125     value <<= 32 - digits
126     repeat digits
127       out((value <-= 1) & 1 + "0")
128
129
130   PUB out(c) | i, k
131
132   '' Output a character
133   ''
134   ''     $00 = clear screen
135   ''     $01 = home
136   ''     $08 = backspace
137   ''     $09 = tab (8 spaces per)
138   ''     $0D = return
139   ''   others = printable characters
140
141       case c
142             $00:
143               Inst := %0000_0001                    ' cursor home , clear display
144               Send_Inst
145
146             $01:
147               Inst := %0000_0010                    ' cursor home
148               Send_Inst
149
150             $08:
151               Inst := %0001_0000                    ' backspace
152               Send_Inst
153
154             $09: repeat
155                     char := 32
156                     Send_Text
157
158
```

```
  outa[16..19] := %0010                    ' Set To 4-bit Operation
  bs2.PULSOUT(E,1)

  delay.pause1ms(200)

  Inst := %00101000                       ' Function Set (4-Line Mode)
  Send_Inst
  Inst := %0000_0001                       ' cursor home , clear display
  Send_Inst
  Inst := %0000_1110                       ' Turn On Cursor, no blink
  Send_Inst
  Inst := %00000110                       ' Set Auto-Increment
  Send_Inst
 Inst := %00000010                      ' Clears LCD
  Send_Inst
  Inst := 13                              ' Set Cursor To Underline,blink
  Send_Inst


pri Send_Inst | temp1              ' can we use byte move, then shift up 4 times ?

    temp1 := Inst
    temp1 := temp1>>4

  outa[RS] := 0                             ' Set Instruction Mode
  outa[16..19] := temp1                    ' Send High Nibble
  bs2.PULSOUT(E,3)
  Wait_Busy
  outa[16..19] := Inst                     ' Send Low Nibble
  bs2.PULSOUT(E,3)
  Wait_Busy
  outa[RS] := 1                            ' Set LCD Back To Text Mode


pri Send_Text   | temp1


    temp1 := Char
    temp1 := temp1>>4

  outa[RS] := 1                            ' Set data Mode
```

```
  Send_Inst


pub Wait_Busy

  delay.pause1ms(15)


{{
Older 4x20 LCD displays, like the Optrex I'm using, have different line layout:

0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
40 .. .. .. .. .. .. .. .. .. 49 .. .. .. .. .. .. .. .. 59
20 .. .. .. .. .. .. .. .. .. 30 .. .. .. .. .. .. .. .. 39
60 .. .. .. .. .. .. .. .. .. 70 .. .. .. .. .. .. .. .. 79

Notice that the lines 1 and 3 follow each other. Line 2 is followed by line 4.
So basically when filling all character positions, the lines will be filed in this order:
 line 1, line 3, line 2 and finally line 4.  }}

pub pos(x,y) | counter

                                                      '' ============================
                                                      '' position cursor at position (X,
Y)
                                                      '' ============================
                                                      '' Usage:
                                                      '' horizontal position or column (
X)
                                                      '' vertical position or line (Y)

  Inst := %0000_0010                    ' cursor home
  Send_Inst
  counter := 0                          ' reset counter

  case y

    1 :   '4x20: row1
                counter := %00000000
                counter := counter + X
                Inst := %10000000 + counter
```

```
293                 Inst := %10000000 + counter
294                 Send_Inst
295
296
297
298
299
300 dat
301   Text byte "dan miller "      ' Message To Send To LCD
302
```