

```

      LB Essential LCD Driving Code V4.bas
|
|*****
| *
| * refer to previous code in program by The Happy Hippy
| * www.psynet.net/hippy
| *
| * SIMPLE SERIAL LCD DRIVER CODE FOR THE PICAXE-18      LCD-18.BAS
| *
|*****
| *
| *   TO DRIVE A LCD CONSIDER THE FOLLOWING STEPS
| *
| *       GOSUB Initialiselcd
| *
| *       store a byte in variable byte then send it by...
| *       byte = $01
| *       GOSUB SendCmdByte
| *
| *       'NOW SEND "H"
| *       byte = $48
| *       GOSUB SendDataByte
| *
|*****
| *
| *   DESCRIPTION
| *
| *       This project component allows a standard Hitachi HD44780 based,
| *       or compatible, parallel LCD display to be interface to the PICAXE
|*****
|
|*****
| *
| *   CIRCUIT DIAGRAM
| *
| *       This circuit uses the 'recommended' PC interfacing for the PICAXE
| *       as suggested by Revolution Education Ltd in their data sheets and
| *       should be suitable for most PC RS232C interfaces.
| *
| *       When connecting to another PICAXE; the 10K should be removed and
| *       the 180R and 22K replaced by 0R or wire links. These components
| *       are only required for interfacing to a PC RS232C serial line.
| *
| *       The Serial Programming connections are not included for clarity
| *       but may be easily added as per the Revolution Education Ltd data
| *       sheets. In order to use the PICAXE without the Serial Programming
| *       connections; Pin 3 of the PICAXE must be connected to 0v.
| *
| *
| *

```


LB Essential LCD Driving Code V4.bas

*
*
* Bytes with values \$20 to \$7F (ASCII) \$A0 to \$FF (JAP CHR)
* will cause a preset character to be displayed, and once the last
* character on the LCD has been written to, the first will be
* overwritten with subsequent data.
*

*
*
* LCD HANDLER COMMANDS
*
*

- * \$00..\$07 Show CGRAM char 0..7
- * \$08..\$0C Show CGRAM char 0..4
- * \$0D Command line terminator
- * \$0E Escape Code
- * \$0F Show CGRAM char 7
- * \$10..\$1F Invalid / Don't use
- * \$20..\$7F Show ASCII chars ("\" = Yen symbol)
- * \$80..\$9F Invalid / Don't use
- * \$A0..\$FF Show Japanese char

*
*
* LCD DISPLAY STORAGE SET ADDRESS COMMANDS
*
*

- * 1 x 8 \$80..\$87
- * 1 x 16 \$80..\$87 + \$C0..\$C7
- * 2 x 16 \$80..\$8F / \$C0..\$CF
- * 4 x 16 \$80..\$8F / \$C0..\$CF / \$A0..\$AF / \$E0..\$EF
- * 1 x 32 * \$80..\$8F + \$C0..\$CF
- * 2 x 32 \$80..\$9F / \$C0..\$DF
- * 4 x 32 \$80..\$9F / \$C0..\$DF / \$A0..\$BF / \$E0..\$FF

* These addresses have not been verified nor tested

LB Essential LCD Driving Code V4.bas

```
*****
' *
*
' *   Define Input Pin Usage
*
' *
*
'
*****

'   The input pin allocations reflect those used on my own development
'   board and may be adjusted as required.
'
'   Note that re-allocation of pins may cause a change in program code
'   size. Using pins 0 or 1 can often reduce code size, while moving
'   connections from pin 0 or 1 may increase it. The gain, or loss,
'   will depend upon other changes made to the code, and how many
'   accesses are made to the pin in the code, and which commands use
'   the pin. Pin allocation to reduce code size will often be a matter
'   of trial and error after the code is designed.

SYMBOL  I0          = 0          ; Unused - Input 0
SYMBOL  I1          = 1          ; Unused - Input 1
SYMBOL  I2          = 2          ; Unused - Input 2
SYMBOL  RX          = 6          ; RX from PC
SYMBOL  I7          = 7          ; Unused - Input 7

'
*****
' *
*
' *   Define Output Pin Usage
*
' *
*
'
*****

'   D4 - D7 is required as specified by the 'SendDataByte' routine
'   Changing those pin allocations will require a rewrite of
'   the 'SendDataByte' routine.
'
'   If the pin allocated for the RS line is changed;the 'RSDATmask'
'   constant must be updated to reflect the change.

SYMBOL  O0          = 0          ; Unused - Output 0 pin 6
SYMBOL  O1          = 1          ; Unused - Output 1 pin 7
- Output 2 pin 8  SYMBOL  E          = 2          ; 0 = Idle      1 = Active
- Output 3 pin 9  SYMBOL  RS         = 3          ; 0 = Command  1 = Data
4 pin 10          SYMBOL  D4         = 4          ; LCD Data Line 4 - Output
5 pin 11          SYMBOL  D5         = 5          ; LCD Data Line 5 - Output
6 pin 12          SYMBOL  D6         = 6          ; LCD Data Line 6 - Output
7 pin 13          SYMBOL  D7         = 7          ; LCD Data Line 7 - Output

SYMBOL  RSCMDmask   = %00000000    ; RS = 0 - Command Register
SYMBOL  RSDATmask   = %00001000    ; RS = 1 - Data Register

'
*****
' *
*
' *   Define LCD Configuration
*
' *
*
'
*****
```

LB Essential LCD Driving Code V4.bas

' This code can be used with most one, two and four line displays.
 The following constants must be edited to reflect the type of display
 ' being used. The settings in this source code as delivered are for
 ' a 2 x 16 display.

LCD	ADRMASK	ADRCOMP	LCDSETUP
1 x 8	\$08 = %00001000	\$08 = %00001000	
1 x 16	\$08 = %00001000	\$48 = %01001000	
2 x 16	\$10 = %00010000	\$50 = %01010000	
4 x 16	\$10 = %00010000	\$50 = %01010000	
1 x 32	\$10 = %00010000	\$50 = %01010000	
2 x 32	\$20 = %00100000	\$60 = %01100000	
4 x 32	\$20 = %00100000	\$60 = %01100000	

' Some of these settings have not been verified nor tested

SYMBOL	ADRMASK	= \$10	; 2 x 16
SYMBOL	ADRCOMP	= \$50	; 2 x 16
SYMBOL	LCDSETUP	= \$28	; 2 x 16

' *
 *
 ' * Define Variables Used
 *
 ' *
 *
 ' *

SYMBOL	byte	= b0
SYMBOL	get	= b2
SYMBOL	rsbit	= b4
SYMBOL	mycount	= b1

' *
 *
 ' * Program Initialisation
 *
 ' *
 *
 ' *

MAIN:

```
GOSUB InitialiseLcd ; Initialise the LCD

byte = "A" ; Display "A" on the LCD
GOSUB SendDataByte

byte = $80 | $00 ; Put cursor at start of Line 1
GOSUB SendCmdByte

byte = $80 | $40 ; Put cursor at start of Line 2
GOSUB SendCmdByte

byte = 2 * 8 | $40 ; Program User Defined Character 2
GOSUB SendCmdByte
byte = %0001100 : GOSUB SendDataByte ; ##
byte = %0011110 : GOSUB SendDataByte ; ####
```

```

                LB Essential LCD Driving Code V4.bas
byte = %01111111 : GOSUB SendDataByte ; #####
byte = %0001100 : GOSUB SendDataByte ; ##
byte = %0001100 : GOSUB SendDataByte ; ##
byte = %0001100 : GOSUB SendDataByte ; ##
byte = %0001100 : GOSUB SendDataByte ; ##
byte = %0000000 : GOSUB SendDataByte ;

byte = 2 ; Display User Defined Character 2
GOSUB SendCmdByte

END

'
*****
' *
*
' *   Initialise LCD
*
' *
*
'
*****
'
'   To initialise the LCD in 4-bit mode, we send a sequence of nibbles
'   followed by a sequence of bytes. In reality, we can send these as
'   a sequence of bytes.
'
InitialiseLcd:
    FOR get = 0 TO 5
        READ get,byte
        GOSUB SendInitCmdByte
    NEXT

    ' Nibble commands - To initialise 4-bit mode

    EEPROM 0,( $33 )
    EEPROM 1,( $32 )

    ' Byte commands - To configure the LCD

    EEPROM 2,( $28 ) ; %00101000 %001LNF00 Display Format
    EEPROM 3,( $0C ) ; %00001100 %00001DCB Display On
    EEPROM 4,( $06 ) ; %00000110 %000001IS Display Shift

                    ; L : 0 = 4-bit Mode      1 = 8-bit Mode
                    ; N : 0 = 1 Line          1 = 2 Lines
                    ; F : 0 = 5x7 Pixels     1 = N/A
                    ; D : 0 = Display Off    1 = Display On
                    ; C : 0 = Cursor Off     1 = Cursor On
                    ; B : Cursor Flash
                    ; I : 0 = Dec Cursor     1 = Inc Cursor
                    ; S : 0 = Cursor Move    1 = Display Shift

    EEPROM 5,( $01 ) ; Clear Screen

'
RETURN
'
*****
' *
*
' *   Send an initialisation byte to the LCD
*
' *
*
'
*****

```

LB Essential LCD Driving Code V4.bas

' According to the data sheets for standard-style LCD displays; the
' initialisation codes to put an LCD into 4-bit mode should be sent
as
' 4-bit nibbles, each with RS set to 0, with E pulsed after each code
' is sent. Analysing this, shows that sending two nibbles is the same
' as sending a byte in 4-bit mode to the LCD Command Register. Pairs
' of nibbles are therefore sent as single bytes.
'
' The data sheets also indicate that there must be a delay after the
' LCD is first powered up and between initialisation codes. Although
' the delays reduce as initialisation proceeds, it is simpler to use
' a single, long, delay; this does not increase the initialisation of
' the LCD by any appreciable amount.
'
' According to the data sheet; there should be delays between every
' initialisation code, however, the method used here only puts delays
' between every two initialisation codes. This works perfectly for
the
' LCD being used to test this project, but may prove problematic for
' other LCD models.

SendInitCmdByte:

```
        PAUSE 15                ; Delay 15mS  
'RETURN
```

'

' *
' * Send a command byte to the LCD
' *
' *
' *

'
' The only difference between sending an LCD Control Command and an
' LCD Data Byte is that the RS line on the LCD needs to be 0 not 1.
' We use the 'rsbit' variable to store the bit which is sent to the
' LCD when we output to the LCD to save having to have two separate
' routines. We have plenty of variables free so this is a compromise
' which is perfectly acceptable.
'
' Whenever a byte is sent to the LCD, 'rsbit' is reset so a
subsequent
' call to 'SendDataByte' will be sent as data. We need to set 'rsbit'
' whenever we need to send a Control Command. Because we always send
' Control Commands to the LCD during initialisation, it is
unnecessary
' to initialise 'rsbit' before the first call to 'SendCmdByte', nor
' before a call to 'SendDataByte'.

SendCmdByte:

```
        rsbit = RSCMDmask       ; Send to Command register  
'RETURN
```

'

' *
' * Send a data byte to the LCD
' *
' *
' *

'
' Note that if you change the pin allocations for D4..D7 then this
' routine must be rewritten to cater for the changes made.

LB Essential LCD Driving Code V4.bas

SendDataByte:

```
pins = byte & %11110000 | rsbit ; Put MSB out first
PULSOUT E,1 ; Give a 10uS pulse on E
pins = byte * %00010000 | rsbit ; Put LSB out second
PULSOUT E,1 ; Give a 10uS pulse on E

rsbit = RSDATmask ; Send to Data register next

RETURN
```

```
!
*****
! *
*
! * End of source code
*
! *
*
!
*****
```