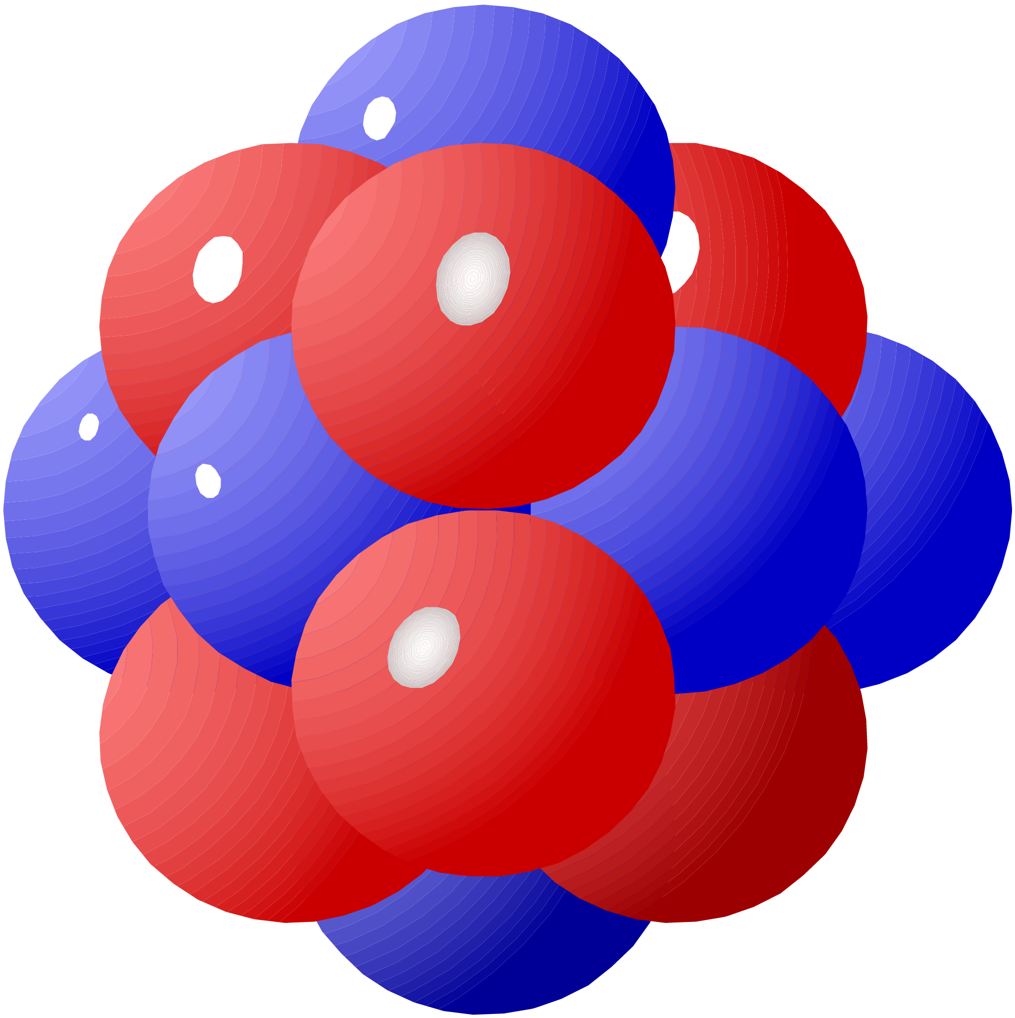
PROTON

Serial Graphic LCD Interface



Crownhill Associates
smart electronic solutions

Disclaimer

In order to comply with EMC directive 89/336/EEC, this product should not be used outside of a classroom or laboratory environment.

Any software supplied with this product is provided in an “as is” condition. No warranties, whether express, implied or statutory, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose apply to this software. The company shall not, in any circumstances, be liable for special, incidental or consequential damages, for any reason whatsoever.

This product should not be used as a critical component in life support systems. Crownhill does not recommend the use of any of its products in life support applications where the failure or malfunction of the product can reasonably be expected to cause failure in the life support system or to significantly affect its safety or effectiveness.

In producing this, document and the associated hardware we have spent a great deal of time to ensure the accuracy of the information presented. We constantly strive to improve the quality of our products and documentation. Crownhill operates a quality system conforming to the requirements of BS EN ISO9001: 2000. Crownhill reserves the right to change the functionality and specifications of its products from time to time without prior notice.

If you should find any errors or omission in these documents or hardware, please contact us, we appreciate your assistance in improving our products and services.

Crownhill reserves the right to make changes to the products contained in this publication in order to improve design, performance or reliability. Crownhill assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representation that the circuits are free of patent infringement. Charts and schedules contained herein reflect representative operating parameters, and may vary depending upon a user’s specific application. While the information in this publication has been carefully checked, Crownhill shall not be liable for any damages arising as a result of any error or omission.

PICmicro^{tm} is a trade name of Microchip Technologies Inc.
PROTON^{tm} is a trade name of Crownhill Associates Ltd.
EPIC^{tm} is a trade name of microEngineering Labs Inc.

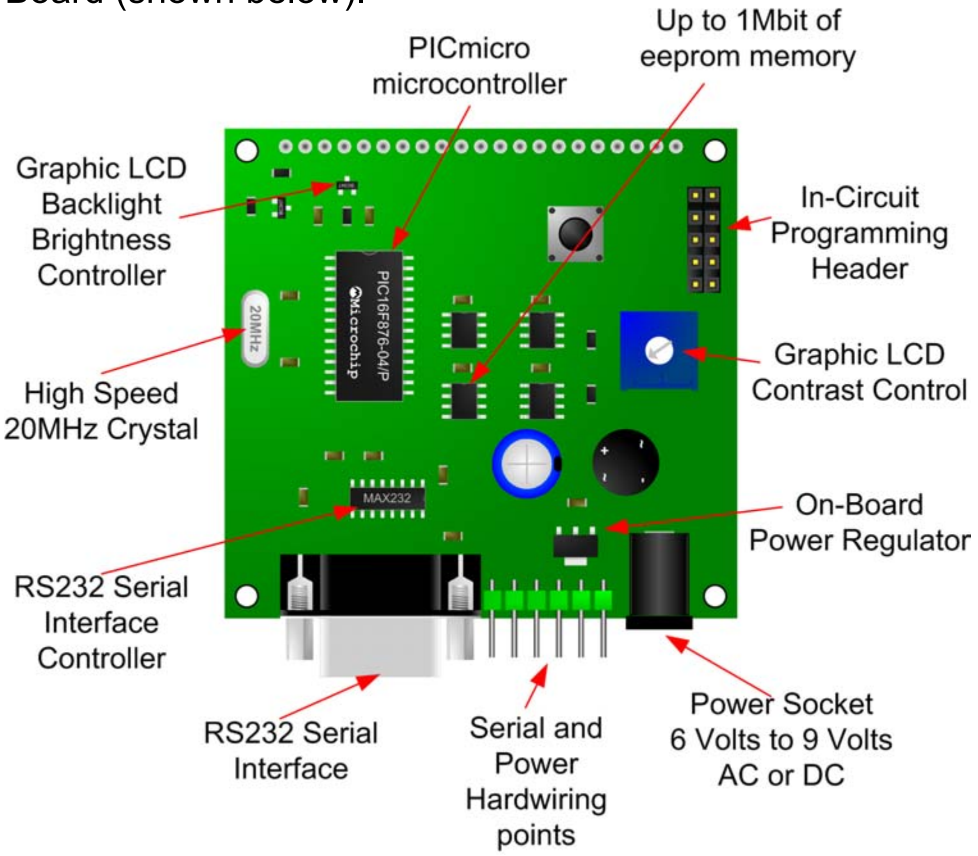
This document was written by Les Johnson, and published by Crownhill associates limited, Cambridge England, 2003

Introduction

Graphic LCDs are not only great fun to use, they also offer a very professional finish to an end product. However, cost has always been a issue, as graphic LCDs are frequently many times more expensive than their alphanumeric counterparts. But thanks to the graphic LCDs that Crownhill supplies, this is now not an issue, as they offer both affordability and flexibility.

This still leaves two chief problems when interfacing to graphic LCDs, board layout and controlling software. Board layout can be a problem because the LCD requires up to 20 pins to be wired. Controlling software can be a problem because of the complex nature of displaying anything of any use on the LCD.

The PROTON+ BASIC compiler takes the hardship out of controlling the LCD, with commands such as PRINT, PLOT, LINE, CIRCLE etc, and now the complex interface is not a problem thanks to the PROTON Graphic LCD Serial Development Board (shown below).

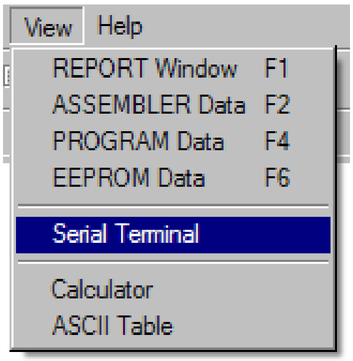

Up to 1Mbit of eeprom memory
PICmicro microcontroller
Graphic LCD Backlight Brightness Controller
In-Circuit Programming Header
PIC16F876-04/P
Microchip
20MHz
Graphic LCD Contrast Control
High Speed 20MHz Crystal
MAX232
On-Board Power Regulator
RS232 Serial Interface Controller
RS232 Serial Interface
Serial and Power Hardwiring points
Power Socket 6 Volts to 9 Volts AC or DC

Using a simple RS232 serial interface, you can PRINT, SCROLL, PLOT, SAVE and LOAD screens, (*plus much more*) on the LCD. And not only that, but you can develop your own software then program the on-board PICmicro^{tm} using either the PROTON+ compiler's Bootloader, or a conventional Device programmer such as the EPIC^{tm}. These features coupled with full source code for the serial interface commands offers a truly enjoyable, educational and time saving development platform, as well as a finished product for your final design.

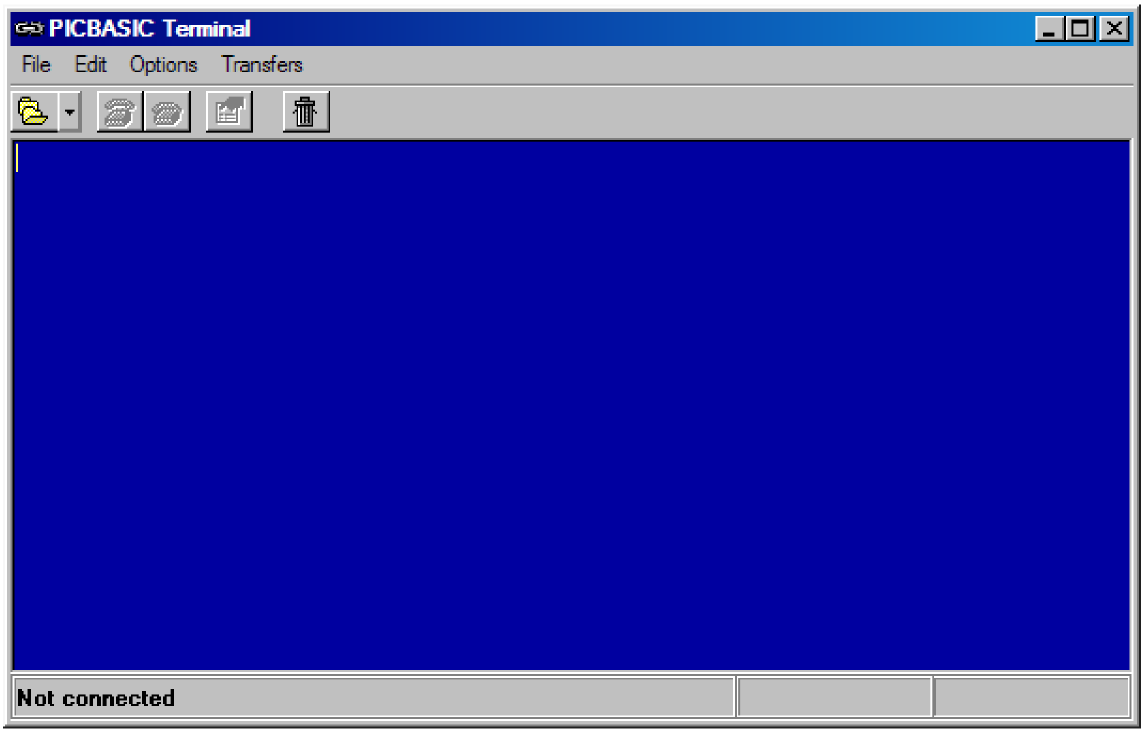
Testing the Development Board

There are two software sources supplied with the development board, one for controlling it using the PROTON+ compiler's Serial Terminal that uses standard ASCII text; which is ideal for testing and demonstrating some of its capabilities. And one that is intended to be controlled from another microcontroller using less cumbersome binary controls. We'll take a look at the Serial Terminal controlled software to start with, as this is already loaded on the development board when supplied.

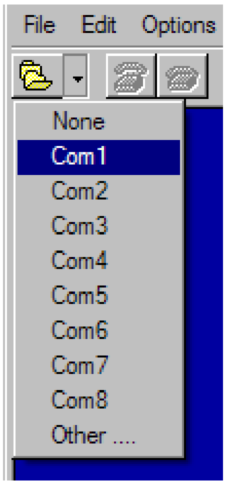
Supply 6 to 9 Volts to the board via its power socket, and a short splash screen will be displayed. While the splash screen is busy, open the serial terminal built into the compiler's IDE, by clicking on VIEW->SERIAL TERMINAL (see below).



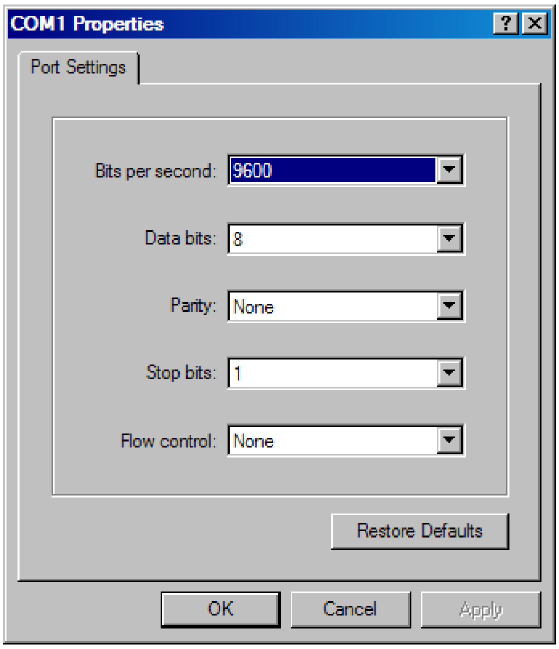
You will be greeted with a window looking something like the screenshot below.



Now the Com port and Baud rate requires setting up. Click on the open Com icon, and a small menu will appear (see below).



Choose the appropriate Com port, according to the setup of your PC. The illustration above shows Com1 being chosen. Note: that the Com port chosen should be the same as the Com port used to download the program to the PROTON SGLCD (more details later). When the Com port is chosen, another window will appear that allows the Baud rate to be set (see below).


COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

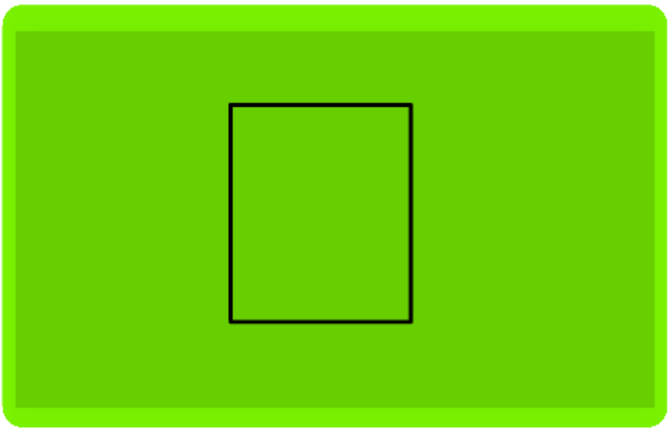
Flow control: None

Set the Baud rate to 9600 (as above), and we're ready to send some commands to the graphic LCD.

Just as a starter, type the following text on the terminal all in UPPER CASE and in quick succession.

B3F1F1001

And displayed on the graphic LCD should be a square, as shown below. OK, so it's not an exact square, but that's because of the aspect ratio of the graphic LCD's pixels. They are approx 1.5 times taller than they are wide.



If a square did not appear, then try again but type quicker, and don't forget to type in UPPER CASE.

The collection of characters told the serial LCD to draw a square "B" at X position 3F (decimal 63) and Y position 1F (decimal 31), with a RADIUS of 10 (decimal 16), and SET the pixels "01"

Note that all the values are formatted as 2 character HEX values. This is true for all the commands that have parameters, and all future discussions will be based on **Hexadecimal** values unless otherwise stated.

Now type 'B3F1F1000' and the square will disappear because the same XPOS, YPOS and RADIUS have been entered, but now a command to CLEAR the pixels was sent "00"

With that small (but crucial) test carried out, we can continue with a detailed discussion of the commands available.

PROTON Graphic LCD Serial Interface Commands

List of commands: -

A… Set or Clear a Single Pixel.
B… Draw a Square.
C… Clear the LCD.
D… Scroll Display Down One Line.
E… Scroll the Display Left a Single Pixel.
F… Scroll the Display Right a Single Pixel.
G… Adjust LCD Backlight Brightness.
H… Change Serial Interface Baud Rate.
I… Rotate the Display Right a Single Pixel.
J… Rotate Part of the Display Right a Single Pixel.
K… Rotate the Display Left a Single Pixel.
L… Draw a Line.
M… Rotate Part of the Display Left a Single Pixel.
N… Scroll the Display Up a Single Pixel.
O… Scroll the Display Down a Single Pixel.
P… Position the Cursor.
Q… Set Response.
R… Draw a Circle.
S… Scroll Display Up One Line.
T… Display Text.
U… Upload a Screen Serially.
V… Rotate the Display Down a Single Pixel.
W… Downlaod a Screnn Serially.
X… Load a Screen from Eeprom Memory.
Y… Rotate the Display Up a Single Pixel.
Z… Save a Screen to Eeprom Memory.

DISPLAY TEXT

Command: - **T**

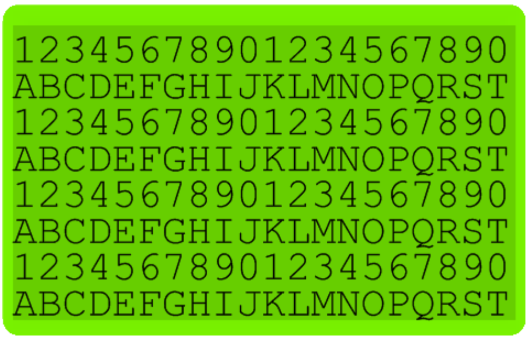
Syntax: - **T** text to display

Example: -
THELLO WORLD

The above example will display the text “HELLO WORLD”. Notice that the text did not appear until typing had finished. This is because the text is stored in a 200 element BYTE ARRAY before being sent to the LCD. However, when typing stops, a TIMEOUT occurs and the text is written to the LCD. This means that up to 200 characters can be entered in one ‘T’ command.

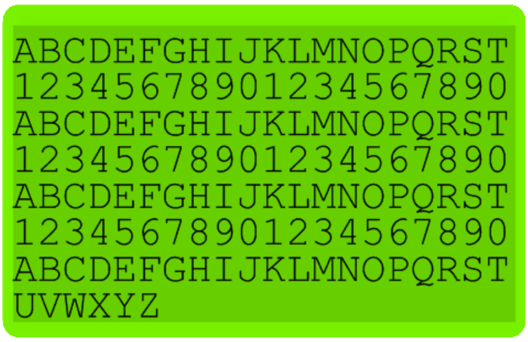
Notes: -
When any text reaches the end of the line, it will move down one and move the X position to 0 and carry on displaying text. Essentially performing a Carriage Return with Line Feed. When the text reaches the end of the display i.e. the end of line on the bottom line (line 07), the text will SCROLL upwards one line, and the top line will disappear.

Try typing ‘T’ followed by many characters until the bottom line is reached. The picture below shows what it could look like.



```
12345678901234567890
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
```

Now type in a few more characters. i.e. **T**UVWXYZ, and the display will scroll up one line before displaying the new text (shown below).



```
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
12345678901234567890
ABCDEFGHIJKLMNOPQRST
UVWXYZ
```

CLEAR the LCD

Explanation.

Clears the LCD display and positions the cursor at 0,0.

Command: - **C**

Syntax: -

C

Example: -

B3F1F1001

C

The above example will draw a square (shown earlier) , then typing the single character "C" will clear the LCD.

POSITION the CURSOR

Explanation.

Positions the cursor on the LCD normally prior to a text command being issued.

Command: - **P**

Syntax: -

P XPOS (*00 to 14*) LINE (*00 to 07*)

Example: -

C

P0003

THELLO WORLD

The above example will clear the LCD, then position the cursor at XPOS 00 and LINE 03 (*remember the LCD lines count from 00 to 07*) before displaying the text "HELLO WORLD".

SET or CLEAR a SINGLE PIXEL

Explanation.

Sets or Clears a single pixel on the LCD.

Command: - **A**

Syntax: -

A XPOS (*00 to 7F*) YPOS (*00 to 3F*) SET or CLEAR (*00 or 01*)

Example: -

C

A3F2001

The above example will clear the LCD then move to position XPOS 3F and YPOS 20 before setting the pixel near to the centre of the LCD. Altering the last two digits to 00 will clear the pixel: - **A**3F2000

The same pixel has now been cleared.

DRAW a CIRCLE

Explanation.

Draws or Erases a Circle on the LCD at the given X and Y coordinates.

Command: - **R**

Syntax: -

R XPOS (*00 to 7F*) YPOS (*00 to 3F*) RADIUS (*00 to FF*) SET or CLEAR (*00 or 01)*

Example: -

R3F1F1001

The above example will draw a CIRCLE at XPOS 3F and YPOS 1F with a RADIUS of 10. The SET or CLEAR parameter is set to 01, so the pixels will be set. Changing the last parameter to 00 will clear the pixels. For example: - **R**3F1F1000 will clear the circle.

DRAW a SQUARE

Explanation.

Draws or Erases a Square on the LCD at the given X and Y coordinates.

Command: -   B

B XPOS (*00 to 7F*) YPOS (*00 to 3F*) RADIUS (*00 to FF*) SET or CLEAR (*00 or 01)*

Example: -

B3F1F1001

The above example will draw a SQUARE at XPOS 3F and YPOS 1F with a RADIUS of 10. The SET or CLEAR parameter is set to 01, so the pixels will be set. Changing the last parameter to 00 will clear the pixels. For example: - **B**3F1F1000 will clear the square.

DRAW a LINE

Explanation.

Draws or Erases a Line on the LCD at the given X and Y coordinates.

Command: -   L

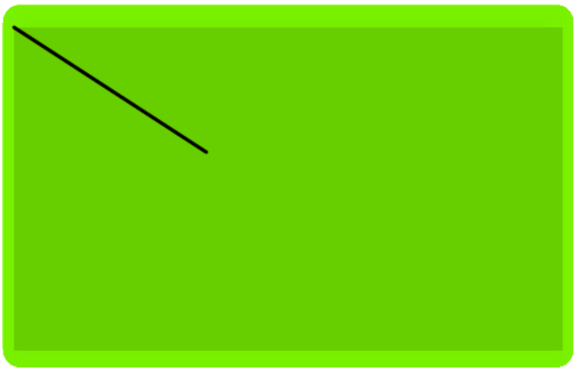
Syntax: -

L XPOS_START (*00 to 7F*) YPOS_START (*00 to 3F*) XPOS_END (*00 to 7F*) YPOS_END (*00 to 3F*) SET or CLEAR (*00 or 01*)

Example: -

L00002F1F01

The above example will draw a line from XPOS_START 00 and YPOS_START 00 to XPOS_END 2F and YPOS_END 1F.



As with all of the drawing commands, the last 2 HEX digits represent SET or CLEAR the pixel. To erase the line drawn, use: - **L**00002F1F00

SCROLL the DISPLAY UP

Explanation.

Scrolls the LCD up a single line. Any Text or data on the top line will disappear to make room for text or data on the bottom line. The cursor is also set to the beginning of the bottom line.

Command: - **S**

Syntax: -

S

Example:-

```
C
P0003
THELLO WORLD
S
```

The above example will first clear the display (command **C**), then position the cursor at the beginning of line 3 (command **P**) and display the text “HELLO WORLD” (command **T**). The final character, “S”, will SCROLL the display UP one line.

SCROLL the DISPLAY DOWN

Explanation.

Scrolls the LCD down a single line. Any Text or data on the bottom line will disappear to make room for text or data on the top line.

Command: - **D**

Syntax: -

D

Example:-

```
C
P0003
THELLO WORLD
D
```

The SCROLL DOWN example will first clear the display (command **C**), then position the cursor at the beginning of line 3 (command **P**) and display the text “HELLO WORLD” (command **T**). The final character, “D”, will SCROLL the display DOWN one line.

SMOOTH SCROLL the DISPLAY UP

Explanation.

Scrolls the LCD up by a single pixel. Any data on the top row of pixels will disappear to make room for data on the bottom row of pixels.

Command: - N

Syntax: -

N AMOUNT of SCROLLS (*00 to FF*)

Example:-

```
C
P0005
THELLO WORLD
N3F
```

The SMOOTH SCROLL UP example will first clear the display (command **C**), then position the cursor at the beginning of line 5 (command **P**) and display the text “HELLO WORLD” (command **T**). It will then smooth scroll the text up the display 3F (63) times.

SMOOTH SCROLL the DISPLAY DOWN

Explanation.

Scrolls the LCD down by a single pixel. Any data on the bottom row of pixels will disappear to make room for data on the top row of pixels.

Command: - O

Syntax: -

O AMOUNT of SCROLLS (*00 to FF*)

Example:-

```
C
P0002
THELLO WORLD
O3F
```

The SMOOTH SCROLL UP example will first clear the display (command **C**), then position the cursor at the beginning of line 2 (command **P**) and display the text “HELLO WORLD” (command **T**). It will then smooth scroll the text down the display 3F (63) times.

SMOOTH SCROLL the DISPLAY LEFT

Explanation.

Scrolls the LCD left by a single pixel. Any data on the right column of pixels will disappear to make room for data on the left column of pixels.

Command: -   E

Syntax: -

E AMOUNT of SCROLLS (*00 to FF*)

Example:-

C

THELLO WORLD <CR>

HOW ARE YOU

E3F

Note. Do not type <CR> this means press return (Carriage Return).

The SMOOTH SCROLL UP example will first clear the display (command **C**), then display some text (command **T**). It will then smooth scroll the text left across the display 3F (63) times.

SMOOTH SCROLL the DISPLAY RIGHT

Explanation.

Scrolls the LCD right by a single pixel. Any data on the left column of pixels will disappear to make room for data on the right column of pixels.

Command: -   F

Syntax: -

F AMOUNT of SCROLLS (*00 to FF*)

Example:-

C

THELLO WORLD <CR>

HOW ARE YOU

F3F

Note. Do not type <CR> this means press return (Carriage Return).

The SMOOTH SCROLL UP example will first clear the display (command **C**), then display some text (command **T**). It will then smooth scroll the text right 3F (63) times.

SMOOTH ROTATE the DISPLAY UP

Explanation.

Rotates the LCD upwards by a single pixel. Any data on the top row of pixels will appear on the bottom row of pixels.

Command: - **Y**

Syntax: -

Y AMOUNT of ROTATES (*00 to FF*)

Example:-

C
P0005
THELLO WORLD
Y3F

The SMOOTH ROTATE UP example will first clear the display (command **C**), then position the cursor at the beginning of line 5 (command **P**) and display the text “HELLO WORLD” (command **T**). It will then smoothly rotate the text in an upward direction 3F (63) times.

SMOOTH ROTATE the DISPLAY DOWN

Explanation.

Rotates the LCD downwards by a single pixel. Any data on the bottom row of pixels will re-appear on the top row of pixels.

Command: - **V**

Syntax: -

V AMOUNT of ROTATES (*00 to FF*)

Example:-

C
P0002
THELLO WORLD
V3F

The SMOOTH ROTATE DOWN example will first clear the display (command **C**), then position the cursor at the beginning of line 2 (command **P**) and display the text “HELLO WORLD” (command **T**). It will then smoothly rotate the text in a downward direction 3F (63) times.

SMOOTH ROTATE the DISPLAY LEFT

Explanation.
Rotates the LCD anticlockwise by a single pixel. Any data on the left column of pixels will re-appear on the right column of pixels.

Command: - **K**

Syntax: -
K AMOUNT of ROTATES (*00 to FF*)

Example:-
C
THELLO WORLD <CR>
HOW ARE YOU
K7F

Note. Do not type <CR> this means press return (Carriage Return).

The SMOOTH ROTATE LEFT example will first clear the display (command **C**), then display some text (command **T**). It will then smooth rotate the display in an anticlockwise direction 80 (128) times. And the text will end where it started.

SMOOTH ROTATE the DISPLAY RIGHT

Explanation.
Rotates the LCD clockwise by a single pixel. Any data on the right column of pixels will re-appear on the left column of pixels.

Command: - **I**

Syntax: -
I AMOUNT of ROTATES (*00 to FF*)

Example:-
C
THELLO WORLD <CR>
HOW ARE YOU
I80

Note. Do not type <CR> this means press return (Carriage Return).

The SMOOTH ROTATE RIGHT example will first clear the display (command **C**), then display some text (command **T**). It will then smooth rotate the display in a clockwise direction 80 (128) times. And the text will end where it started.

SMOOTH ROTATE PART of the DISPLAY LEFT

Explanation.

Rotates a user defined section of the LCD anticlockwise by a single pixel. Any data on the left column of pixels will re-appear on the right column of pixels.

Command: - **M**

Syntax: -

M XPOS START (*00 to 7F*) YPOS START (*00 to 3F*) XPOS END (*00 to 7F*) YPOS END (*00 to 3F*) AMOUNT of ROTATES (*00 to FF*)

Example:-

```
C
THELLO WORLD <CR>
HOW ARE YOU
M05003B0237
```

Note. Do not type <CR> this means press return (Carriage Return).

The above example will first clear the display (command **C**), then display some text (command **T**). It will then smooth rotate part of the display in an anticlockwise direction 37 (decimal 55) times. And the text will end where it started.

SMOOTH ROTATE PART of the DISPLAY RIGHT

Explanation.

Rotates a user defined section of the LCD clockwise by a single pixel. Any data on the right column of pixels will re-appear on the left column of pixels.

Command: - **J**

Syntax: -

J XPOS START (*00 to 7F*) YPOS START (*00 to 3F*) XPOS END (*00 to 7F*) YPOS END (*00 to 3F*) AMOUNT of ROTATES (*00 to FF*)

Example:-

```
C
THELLO WORLD <CR>
HOW ARE YOU
J05003B0237
```

Note. Do not type <CR> this means press return (Carriage Return).

The above example will first clear the display (command **C**), then display some text (command **T**). It will then smooth rotate part of the display in a clockwise direction 37 (decimal 55) times. And the text will end where it started.

SAVE a SCREEN to EEPROM MEMORY

Explanation.

Save the current LCD display to a particular portion of the I^2C eeprom memory. Within a single 24LC256 serial eeprom, which contains 32 Kbytes of memory, Thirty two screens can be saved. If all 4 eeproms are fitted to the PROTON SGLCD board, then a huge 1Mbit of memory is available (128 Kbytes) which is capable of holding 128 screens. As you've probably gathered, a screen consists of 1 Kbytes (1024 bytes). Because of the delay required for writing to eeproms, a screen save may take a few seconds. The PROTON SGLCD board's serial interface will acknowledge when a save is complete.

Command: - **Z**

Syntax: -

Z LOCATION in MEMORY (*00 to 7F*)

Example: -

C
THELLO WORLD

Z10

In the example above, the LCD is first cleared (command C), and text is displayed (command T). Once the text has appeared on the LCD, the SAVE SCREEN command is issued (Z) with the location to save the screen. While the screen is being saved to eeprom memory, the PROTON SGLCD will not respond to any further commands. After a few seconds, an acknowledge character "A" will be transmitted from the PROTON SGLCD to signal the screen has been saved.

LOAD a SCREEN from EEPROM MEMORY

Explanation.

Load a screen (saved earlier) from a particular portion of the I^{2}C eeprom memory. Within a single 24LC256 serial eeprom, which contains 32 Kbytes of memory, Thirty two screens can be stored. If all 4 eeproms are fitted to the PROTON SGLCD board, then a huge 1Mbit of memory is available (128 Kbytes) which is capable of holding 128 screens. As you've probably guessed, a screen consists of 1 Kbytes (1024 bytes).

Command: - **X**

Syntax: -

X LOCATION in MEMORY (*00 to 7F*)

Example: -

C

X10

The example above should be used after the SAVE to SCREEN example has been carried out. In the example, the LCD is first cleared (command C), and the LOAD SCREEN command is issued (X) with the location to load the screen from. The previously saved screen will then be loaded on to the LCD.

Example2: -

C

X01

The PROTON SGLCD's eeprom is preloaded with several sample screens. Example2 will load the screen stored at location 01 and the display shown below should appear.

UPLOAD a SCREEN

Explanation.

Most complex images displayed on the LCD are usually created using the PC, so a means of uploading the images to the LCD is required. That's what the UPLOAD SCREEN command is for. It accepts a list of HEX data; where each HEX byte represents a byte on the LCD working from top left of the LCD to bottom right. Once the image is displayed on the LCD, the SAVE SCREEN command can store it in eeprom memory.

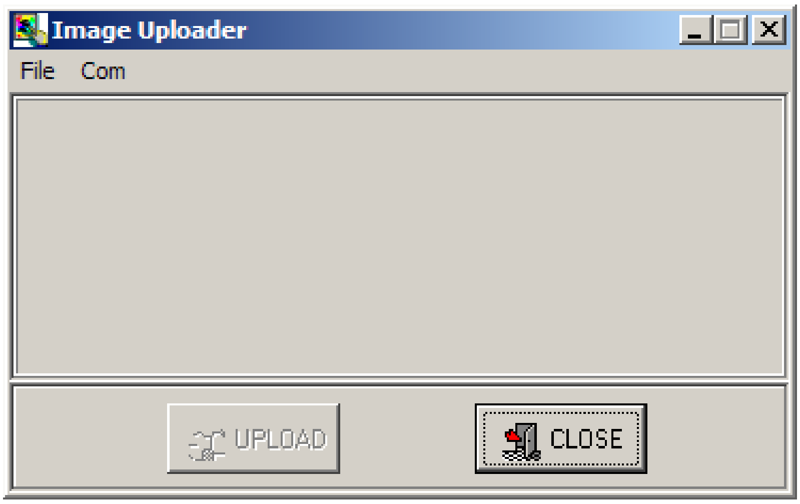
Command: - U

Syntax: -
U

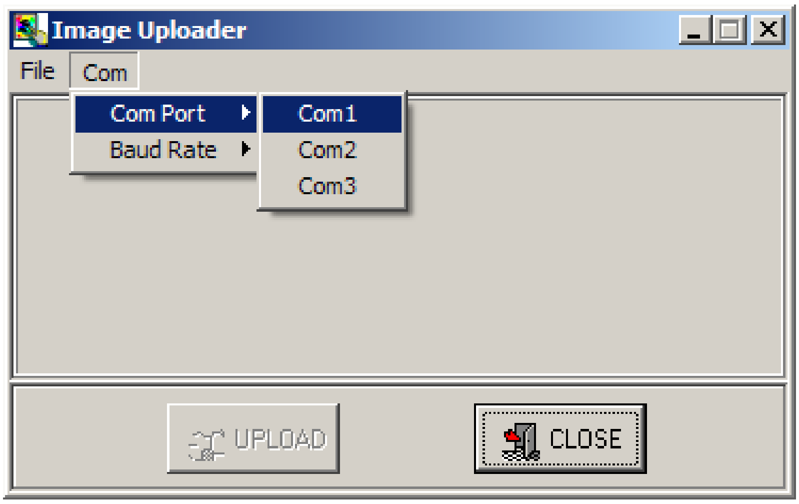
Example: -

In order to demonstrate uploading an image to the LCD, a small executable program has been created that will load an image from the PC, and upload the image to the LCD via the serial port.

Locate and run the program **IMAGE_UPLOAD.EXE** (found on the accompanying CDROM).

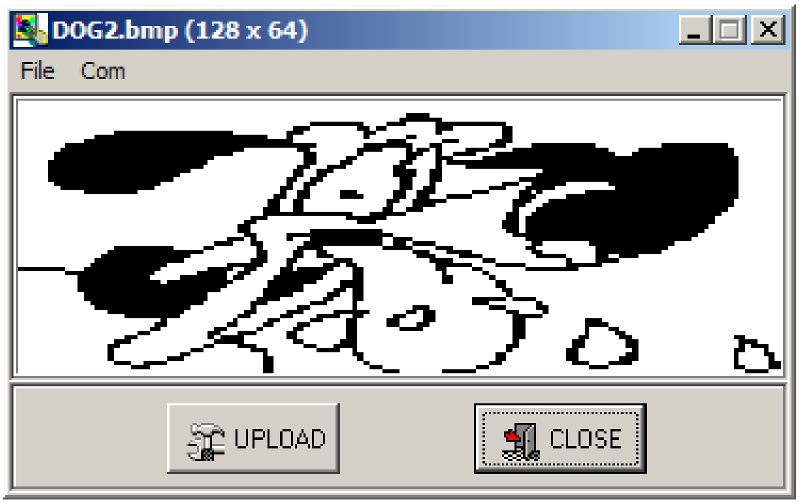


Choose the com port that the PROTON SGLD is attached to (see overleaf).



In the screen shot shown above, com port 1 was chosen. Leave the baud rate as it is for now, because it defaults to 9600, which is the default baud rate of the PROTON SGLCD. Once the com port is chosen, the UPLOAD button will be enabled.

Click on FILE, and load one of the pictures located in the same folder as the executable. The screen shot below shows the picture DOG1.BMP.



Ensure power is applied to the PROTON SGLCD, and that the splash screen has completed its journey to the bottom of the display, then press the UPLOAD button.

An image will be produced on the PROTON SGLCD's display, scanning from top left to bottom right. Once this has completed, close the IMAGE UPLOADER, and save the screen to eeprom memory using the SAVE SCREEN command. Remember where you placed it, as we'll use this in the next command's example.

DOWNLOAD a SCREEN

Explanation.

Images produced on the LCD's display can be saved to eeprom memory for long term storage, or downloaded back to the PC for inclusion into other programs. The DOWNLOAD IMAGE command, sends the data from the display in the format of CDATA statements, ready to be copied and pasted into another PROTON+ BASIC program.

Command: - **W**

Syntax: -
W

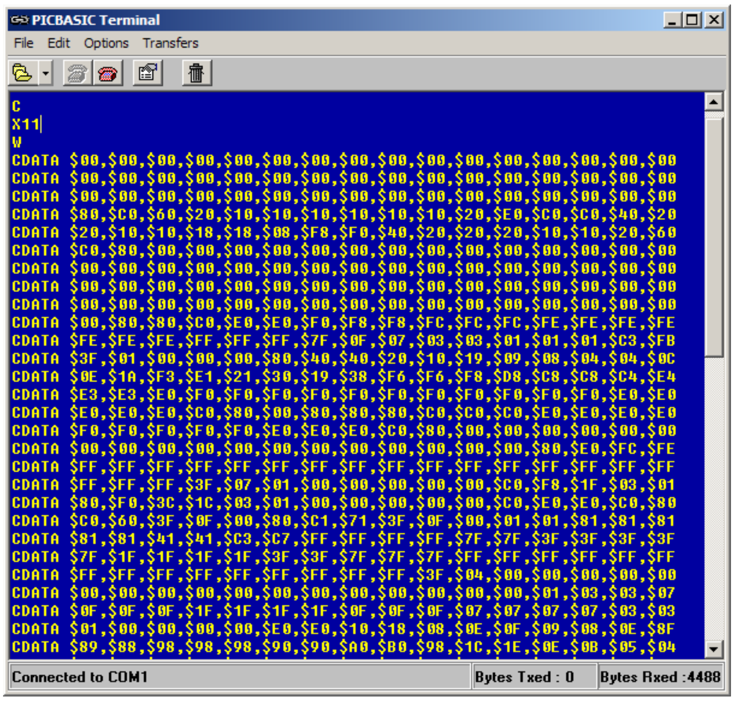
Example: -

In order to demonstrate downloading an image to the PC, you must first have created or loaded one. For this example, we'll use the previously uploaded image of the dog (see UPLOAD a SCREEN).

Open the Serial Terminal and enter the following commands: -

C
X11
W

The above commands clear the LCD (command C), and load the previously saved image from location 11. The W command then transfers the image to the Serial terminal (shown below).



```
C
X11
W
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $80,$C0,$60,$20,$10,$10,$10,$10,$10,$10,$20,$E0,$C0,$C0,$40,$20
CDATA $20,$10,$10,$18,$18,$08,$F8,$F0,$40,$20,$20,$20,$10,$10,$20,$60
CDATA $C0,$80,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
CDATA $00,$80,$80,$C0,$E0,$E0,$F0,$F8,$F8,$FC,$FC,$FC,$FE,$FE,$FE,$FE
CDATA $FE,$FE,$FE,$FF,$FF,$FF,$7F,$0F,$07,$03,$03,$01,$01,$01,$C3,$FB
CDATA $3F,$01,$00,$00,$00,$80,$40,$40,$20,$10,$19,$09,$08,$04,$04,$0C
CDATA $0E,$1A,$F3,$E1,$21,$30,$19,$38,$F6,$F6,$F8,$D8,$C8,$C8,$C4,$E4
CDATA $E3,$E3,$E0,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$E0,$E0
CDATA $E0,$E0,$E0,$C0,$80,$00,$80,$80,$80,$C0,$C0,$C0,$E0,$E0,$E0,$E0
CDATA $F0,$F0,$F0,$F0,$F0,$E0,$E0,$E0,$C0,$80,$00,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$80,$E0,$FC,$FE
CDATA $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
CDATA $FF,$FF,$FF,$3F,$07,$01,$00,$00,$00,$00,$00,$C0,$F8,$1F,$03,$01
CDATA $80,$F0,$3C,$1C,$03,$01,$00,$00,$00,$00,$00,$C0,$E0,$E0,$C0,$80
CDATA $C0,$60,$3F,$0F,$00,$80,$C1,$71,$3F,$0F,$00,$01,$01,$81,$81,$81
CDATA $81,$81,$41,$41,$C3,$C7,$FF,$FF,$FF,$FF,$7F,$7F,$3F,$3F,$3F,$3F
CDATA $7F,$1F,$1F,$1F,$1F,$3F,$3F,$7F,$7F,$7F,$FF,$FF,$FF,$FF,$FF,$FF
CDATA $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$3F,$04,$00,$00,$00,$00,$00
CDATA $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$01,$03,$03,$07
CDATA $0F,$0F,$0F,$1F,$1F,$1F,$1F,$0F,$0F,$0F,$07,$07,$07,$07,$03,$03
CDATA $01,$00,$00,$00,$00,$E0,$E0,$10,$18,$08,$0E,$0F,$09,$08,$0E,$8F
CDATA $89,$88,$98,$98,$98,$90,$90,$A0,$B0,$98,$1C,$1E,$0E,$0B,$05,$04
```

Connected to COM1 | Bytes Txed : 0 | Bytes Rxed :4488

LCD BACKLIGHT CONTROL

Explanation.

The graphic LCD supplied with the PROTON SGLCD has a backlight facility, but the default for this is OFF in order to conserve power. The brightness of the backlight can be controlled using the BACKLIGHT CONTROL command, and uses the microcontroller's hardware PWM (Pulse Width Modulation) feature to further save power usage when the backlight is in use.

Command: - **G**

Syntax: -
G BRIGHTNESS LEVEL (*7F to FF*)

Example: -
GFF

The above example will illuminate the LCD's backlight to full brightness level.

Example 2: -
G00

Example 2 will extinguish the LCD's backlight.

Example 3: -
GF0

Example 3 will dimly illuminate the LCD's backlight.

Generally, a value between E0 to FF is all that's required for the full illumination swing of the backlight.

ENABLE/DISABLE RESPONSE

Explanation.

Each command available in the PROTON SGLCD takes a finite amount of time to complete, and while this is unimportant when using the serial terminal as an interface, it is not the case if a program is automatically using the interface (such as the image uploader). What is then required is a method of knowing that the PROTON SGLCD has finished its current task and is ready for the next. This is the job of the ENABLE/DISABLE RESPONSE command. When enabled, each command will send an acknowledgement of characters <CR> 'O' <CR> if all was OK, and other ACKS if something went wrong will performing the specific command.

Note. <CR> represents Carriage Return which has a value of 13.

Command: - **Q**

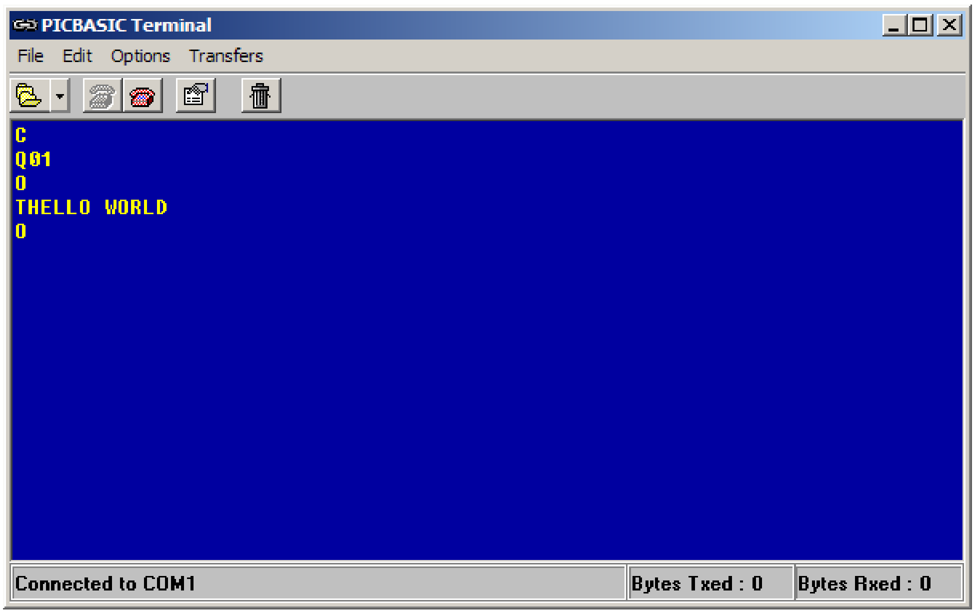
Syntax: -

Q DISABLE or ENABLE RESPONSED (*00 or 01*)

Example: -

C
Q01
THELLO WORLD

The example first clears the LCD (command C), then enables RESPONSES (command Q). The PROTON SGLCD will immediately respond with the ack character 'O'. Text is then displayed (command T) and the PROTON SGLCD responds with the ack character 'O' when the text has finished being displayed. You should see the same results as shown in the screen shot below.



```
C
Q01
O
THELLO WORLD
O
```

Connected to COM1 | Bytes Txed : 0 | Bytes Rxed : 0

To disable the responses, then issue the command Q00. The PROTON SGLCD will give a response of ‘O’ to signify OK.

Other responses are: -

N… Command not recognised. This will also be produced if the RETURN key is pressed, as the serial interface does not recognise this as a command.
T… Timeout occurred. If the information is not entered in a reasonable time period, a timeout will occur to allow the serial interface to look for another command.

ADJUST BAUD RATE

Explanation.
By default, the PROTON SGLCD’s serial interface works at a speed of 9600 Bits Per Second (Baud). However, it is capable of working at much slower or higher baud rates. Up to 115200 Baud in fact.

Command: - **H**

Syntax: -
H BAUD RATE VALUE (Decimal value 150 to 115200) <CR>

Example: -
C
THELLO WORLD
H19200 <CR>

< Adjust baud rate of serial terminal to 19200 before typing the following text >

THELLO AGAIN

The example above will first clear the LCD (command C), then display some text at the current baud rate (9600 by default). The baud rate will then be changed to 19200 by the H command. A carriage return must be typed after the decimal baud rate is entered. The PROTON SGLCD will then respond with an ack of ‘O’.

The baud rate of the PROTON SGLCD is now set to 19200, so change the baud rate of the Serial Terminal, and type in the second text message.

The new baud rate is not retained by the PROTON SGLCD after power down, therefore baud rate negotiation must be accomplished upon use.

Microcontroller Interface Firmware

The PROTON SGLCD will most often be accessed by another microcontroller, therefore HEX parameter values can become cumbersome to use.

By programming the PROTON SGLCD with the microcontroller firmware, named **MICRO_INTERFACE.BAS**, the command values stay the same, but the parameters are not required to be in ASCII HEX digits.

For example to draw a circle on the LCD, the **RSOUT** command could be used.

```
RSOUT "R" , 63 , 32 , 10 , 1
```

Notice how the actual command is still an ASCII ‘R’, but the parameters are now standard decimal values.

Negotiating a Response.

One thing that is profoundly different is the way the PROTON SGLCD responds to a command.

Each command takes a predetermined time to performs its task, and the controlling microcontroller needs to know when the command is finished and able to accept another.

The PROTON SGLCD always send an acknowledge after a command, but does so with a minimal delay of 1ms. This may be too quick for the controlling microcontroller to re-adjust itself from transmitting the command to accepting the ack from the serial port, so a method of calibrating the ack response is always a wise precaution if the controlling microcontroller is using a clock speed of less that 8MHz. Code that could be used to calibrate the ack response is listed below.

```
Dim RESPONSE_TIME as Byte
Dim BLANK as Byte
RESPONSE_TIME = 0         ' Set initial response to 0
Repeat                    ' Create a loop
 NEGOTIATE_RESPONSE:      ' Timeout label
 Inc RESPONSE_TIME        ' Increment the RESPONSE time every cycle
 HRSOUT "Q",RESPONSE_TIME      ' Send RESPONSE delay
' Exit loop if response received
Until HRSIN ,{100,NEGOTIATE_RESPONSE} = "O"
```

The code above starts at the smallest response time which is 1, then sends the RESPONSE command of ‘Q’, and waits 100ms for a response of character ‘O’.

If a response was not received (signified by a timeout) , then the response time is incremented until a response of character ‘O’ is received correctly.

Once the response time its established, the PROTON SGLCD will continue using the time delay until power is removed, or another response calibration is carried out.

In tests, a response time of 2ms was found to be fairly standard using a controller operating with a 4MHz crystal.

The syntax for the RESPONSE command is: -

Q DELAY BETWEEN RESPONSES (0 to 255)

Once the response time has been negotiated and calibrated, it’s a simple process for sending commands: -

Send Command
Wait for acknowledge

Which relates to: -

```
RSOUT "R" , 63 , 32 , 10 , 1  ' Draw a circle
BLANK = RSIN                   ' Wait for acknowledge
```

The **RSIN** command used for receiving the ack from the serial interface could also have a timeout value to prevent any lock-ups if the ack byte is missed. Of course, not only **RSIN/RSOUT** can be used. Any of the compiler’s serial commands can be used to access the PROTON SGLCD’s serial interface.

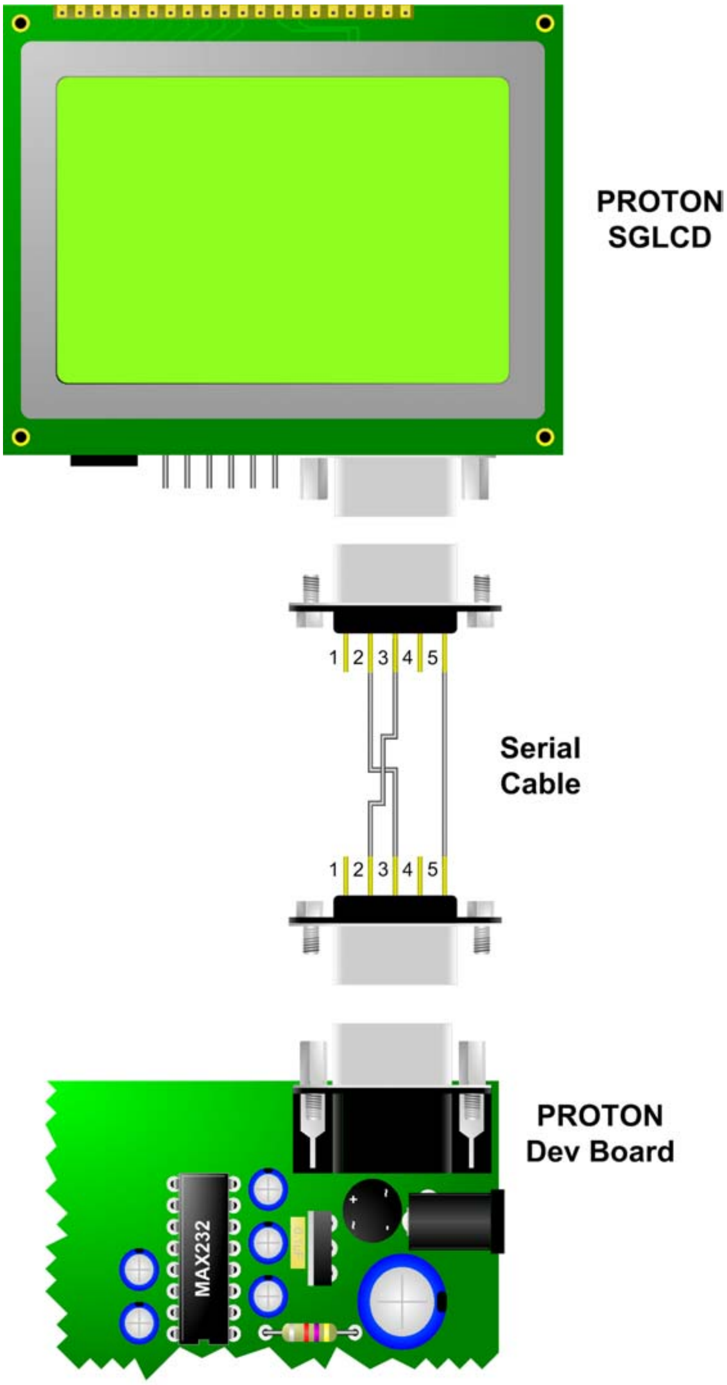
DOWNLOADING a SCREEN

Another difference with the microcontroller firmware is the data received by the DOWNLOAD SCREEN command. Instead of the data being formatted for **CDATA** statements, it is simply transmitted as a series of values.

Shown overleaf is a demonstration program for use with the PROTON Development board, controlling the PROTON SGLCD. Each device is controlled via its serial socket.

The serial cable used must be a type that has pin 3 of one socket connected to pin 2 of the other, and vice-versa. This is commonly known as a NULL modem cable.

The illustration below shows the connections of the serial cable and how it attaches the PROTON SGLCD to the PROTON Development board.



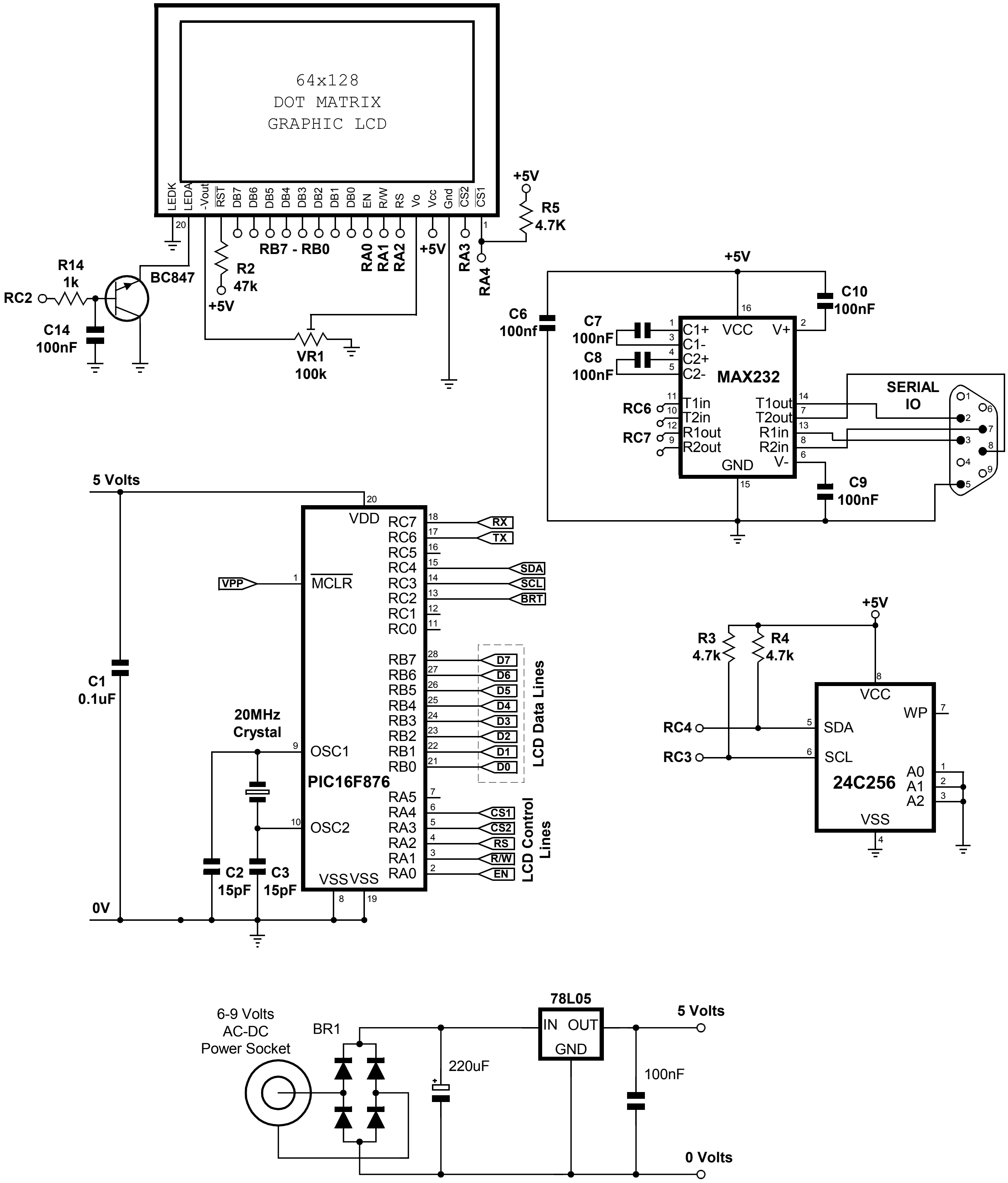
Shown overleaf is the demonstration program mentioned earlier. The program animates the 5 sample images stored in the eeprom when it's shipped.

I hope you agree that even with only 5 images, the effect is rather impressive.

PROTON+ Compiler Demonstration

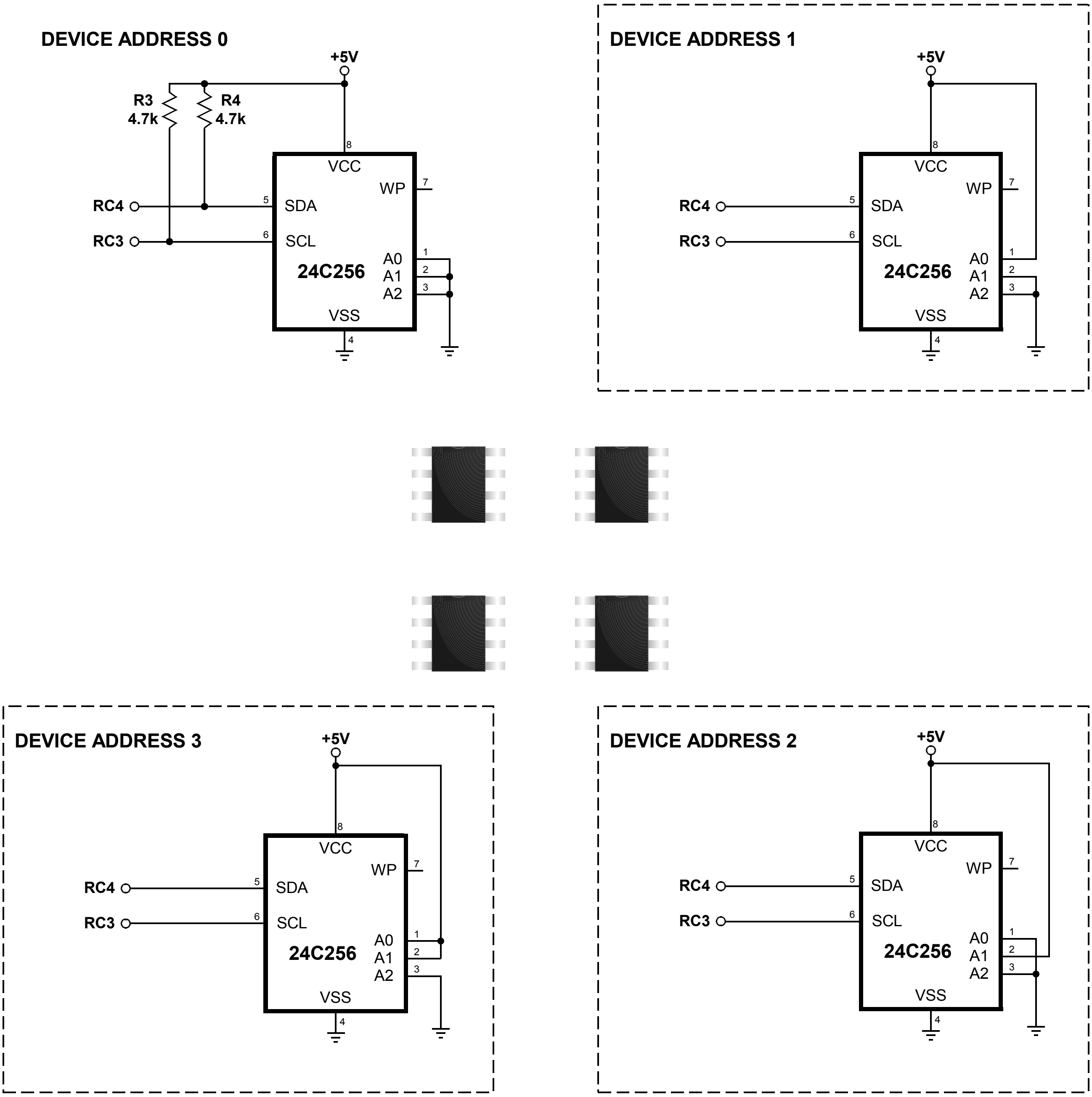
```
' Interface test for microcontroller firmware version
' of PROTON SGLCD
'
' For use on the PROTON PICmicro Development Board
'
' Displays the 5 demo images stored in serial eeprom

Include "PROTON_4.INC"        ' Use the PROTON Dev board at 4MHz
Dim BLANK as Byte
Dim RESPONSE_TIME as Byte
Dim DISPLAY_LOOP as Byte
Delayms 500                   ' Wait for PICmicro to stabilise

' Calibrate the response time
RESPONSE_TIME = 0             ' Set initial response to 0
Repeat                        ' Create a loop
NEGOTIATE_RESPONSE:           ' Timeout label
Inc RESPONSE_TIME        ' Increment the RESPONSE time every cycle
HRSOUT "Q",RESPONSE_TIME           ' Send RESPONSE delay
' Exit loop if response received
Until HRSIN ,{100,NEGOTIATE_RESPONSE} = "O"

HRSOUT "G" , 255              ' Backlight to full
BLANK = HRSIN                 ' Wait for a response

' Display the moving pictures
While 1 = 1                   ' Create an infinite loop
DISPLAY_LOOP = 1              ' Clear the display loop counter
Repeat                        ' Form a loop for the frames
HSEROUT ["X", DISPLAY_LOOP]   ' Load an image from eeprom memory
BLANK = HRSIN                 ' Wait for a response
Inc DISPLAY_LOOP              ' Advance a frame
Until DISPLAY_LOOP > 5        ' Exit after 5 frames
Wend                          ' Do it forever
```

PROTON SGLCD Circuit.



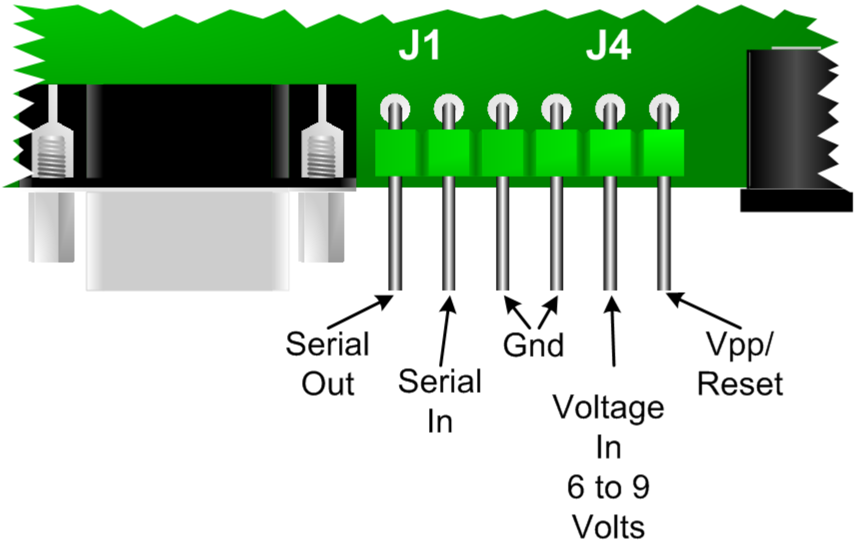
Eeprom Layout.

The PROTON SGLCD is supplied with one 24LC256 serial eeprom addressed as device 0, however, there is room for three more eeproms, each having a different address. The diagram below shows the correlation between the physical eeprom layout on the PCB and their individual circuits.



Direct Connection to the PROTON SGLCD.

The SIL header pins between the serial and power sockets, are for connections to the board using a ribbon cable. The pinouts for the header are shown below.



SERIAL IN connects to pin 13 (R1 In) of the MAX232 RS232 transceiver, and carries serial data to the on-board PICmicro.

SERIAL OUT connects to pin 14 (T1 Out) of the MAX232 RS232 transceiver, and carries serial data from the on-board PICmicro.

GND is the common ground (0v) connection.

VOLTAGE IN connects directly to the input of the 7805 voltage regulator, and can be any DC voltage from 6 to 9 Volts.

VPP/RESET connects to the on-board PICmicro's MCLR pin, and can be used to reset the PROTON SGLCD.

Programming the PROTON SGLCD.

The firmware sources supplied with the PROTON SGLCD, are actually intended as large demonstrations as to what can be achieved with some simple code. The true flexibility of the PROTON SGLCD comes from the fact that it is fully re-programmable with code of your own design.

There are two methods for programming the on-board microcontroller. The simplest method is by the use of a bootloader.

Bootloading a Program.

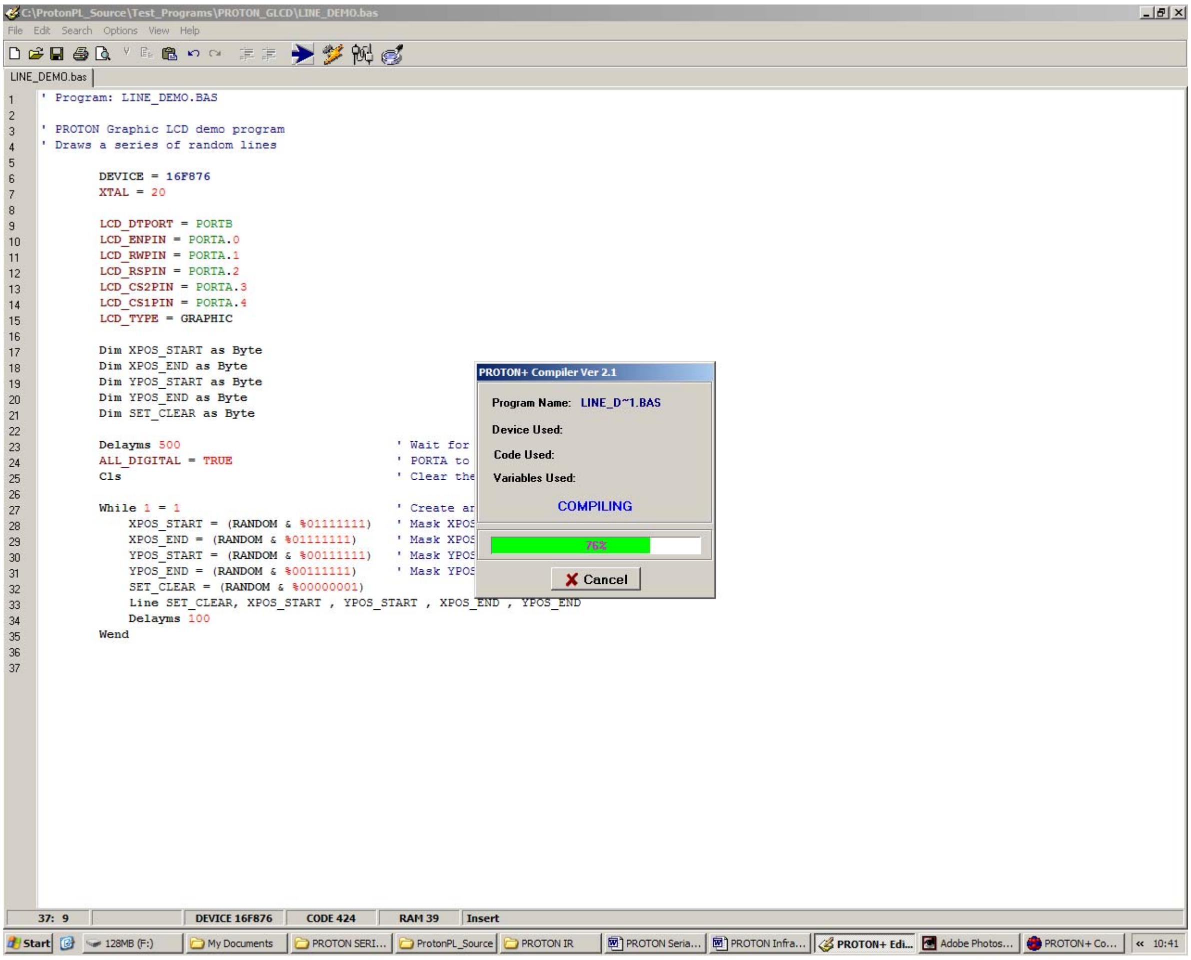
The PROTON SGLCD's microcontroller has the bootloading code already contained in its memory, so all that is required is a cable from the PCs serial port to the PROTON SGLCD's serial socket.

As a demonstration showing how incredibly simple it is to program the PROTON SGLCD, we'll create a simple BASIC program, and download it to the on-board PICmicro microcontroller.

Load the program **LINE_DEMO.BAS** from the samples folder, or type the program in to the compiler's IDE from the listing below.

```
' Program: LINE_DEMO.BAS
' PROTON Graphic LCD demo program
' Draws a series of random lines
DEVICE = 16F876
XTAL = 20
LCD_DTPORT = PORTB
LCD_ENPIN = PORTA.0
LCD_RWPIN = PORTA.1
LCD_RSPIN = PORTA.2
LCD_CS2PIN = PORTA.3
LCD_CS1PIN = PORTA.4
LCD_TYPE = GRAPHIC

Dim XPOS_START as Byte
Dim XPOS_END as Byte
Dim YPOS_START as Byte
Dim YPOS_END as Byte
Dim SET_CLEAR as Byte

Delayms 500                                 ' Wait for PICmicro to stabilise
ALL_DIGITAL = TRUE                          ' PORTA to all digital
Cls                                         ' Clear the LCD
While 1 = 1                                 ' Create an infinite loop
 XPOS_START = (RANDOM & %01111111)          ' Mask XPOS_START (0 to 127)
 XPOS_END = (RANDOM & %01111111)            ' Mask XPOS_END (0 to 127)
 YPOS_START = (RANDOM & %00111111)          ' Mask YPOS_START (0 to 63)
 YPOS_END = (RANDOM & %00111111)            ' Mask YPOS_END (0 to 63)
 SET_CLEAR = (RANDOM & %00000001)           ' Mask SET_CLEAR (0 or 1)
 Line SET_CLEAR, XPOS_START , YPOS_START , XPOS_END , YPOS_END
 Delayms 100
Wend
```

Compile the program, by clicking on the **COMPILE** icon ➔ located on the toolbar, and you should see the screen shown below.



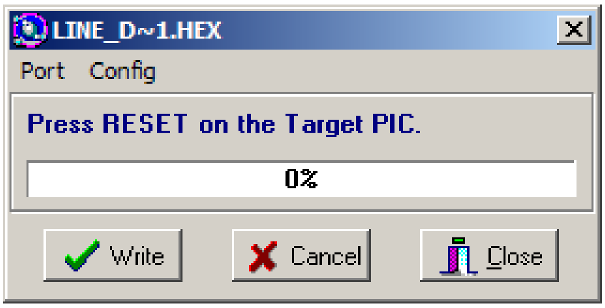
```
' Program: LINE_DEMO.BAS

' PROTON Graphic LCD demo program
' Draws a series of random lines

        DEVICE = 16F876
        XTAL = 20

        LCD_DTPORT = PORTB
        LCD_ENPIN = PORTA.0
        LCD_RWPIN = PORTA.1
        LCD_RSPIN = PORTA.2
        LCD_CS2PIN = PORTA.3
        LCD_CS1PIN = PORTA.4
        LCD_TYPE = GRAPHIC

        Dim XPOS_START as Byte
        Dim XPOS_END as Byte
        Dim YPOS_START as Byte
        Dim YPOS_END as Byte
        Dim SET_CLEAR as Byte

        Delayms 500                             ' Wait for
        ALL_DIGITAL = TRUE                      ' PORTA to
        Cls                                     ' Clear the

        While 1 = 1                             ' Create ar
            XPOS_START = (RANDOM & %01111111)   ' Mask XPOS
            XPOS_END = (RANDOM & %01111111)     ' Mask XPOS
            YPOS_START = (RANDOM & %00111111)   ' Mask YPOS
            YPOS_END = (RANDOM & %00111111)     ' Mask YPOS
            SET_CLEAR = (RANDOM & %00000001)
            Line SET_CLEAR, XPOS_START , YPOS_START , XPOS_END , YPOS_END
            Delayms 100
        Wend
```

Program Name: LINE_D~1.BAS
Device Used:
Code Used:
Variables Used:
COMPILING
76%

37: 9 | DEVICE 16F876 | CODE 424 | RAM 39 | Insert

If no errors were produced while compiling, the program is ready for downloading to the PROTON SGLCD board.

Connect the PROTON SGLCD to the PC using the serial cable and connect the power. Now click on the **DOWNLOAD** icon. You will be greeted with the window shown below.



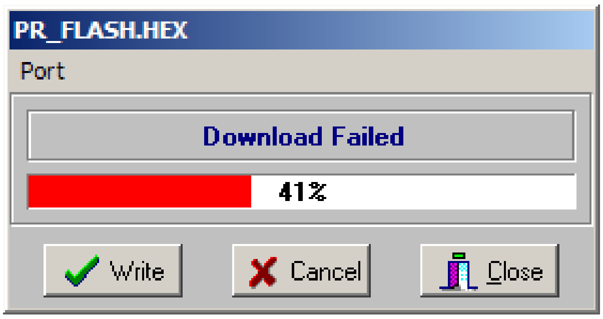
Press RESET on the Target PIC.

0%

On some occasions, you will not be required to press the RESET button, and in this case, the program will be downloaded immediately.

If the program was successfully downloaded to the PROTON SGLCD board, then the LCD should now be displaying random length lines at random locations. You've now successfully programmed the PROTON SGLCD board, easy wasn't it?.

However, if the program failed to download and shows the below window: -

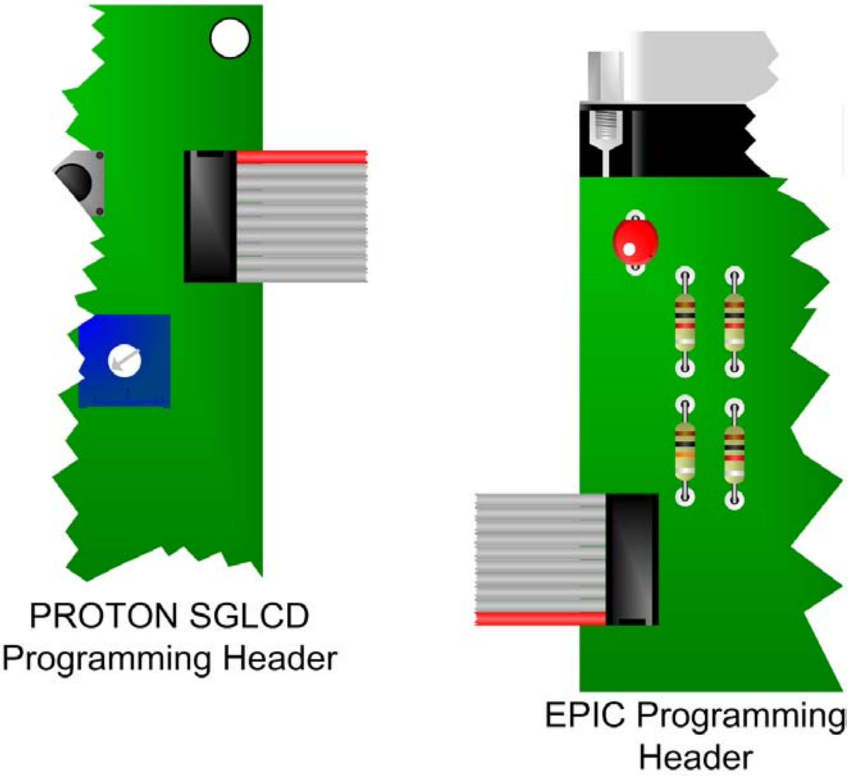


then don't panic, simply click on the WRITE button, and start the process again.

Conventional Programming.

In order to place code into the PROTON SGLCD's microcontroller more securely, a conventional DEVICE programmer (such as the EPIC^{tm}) can be used to In-Circuit-Serial-Program (ICSP) the PICmicro.

The illustration below shows the connections from an EPIC^{tm} programmer to the PROTON SGLCD's 10-way ICSP header.



ASCII Controlled Firmware Source.

```
' PROTON Graphic LCD RS232 serial interface
' For use with an ASCII command set
'
' Uses the low-level LCD interfacing subroutines for intensive commands
' Such as HORIZONTAL SCROLLS

DEVICE = 16F876
XTAL = 20

LCD_DTPORT = PORTB
LCD_ENPIN = PORTA.0
LCD_RWPIN = PORTA.1
LCD_RSPIN = PORTA.2
LCD_CS2PIN = PORTA.3
LCD_CS1PIN = PORTA.4
LCD_TYPE = GRAPHIC
INTERNAL_FONT = ON                  ' Use an internal font for the LCD text

SDA_PIN = PORTC.4                   ' Set serial eeprom
SCL_PIN = PORTC.3                   ' data/clock lines

HSERIAL_BAUD = 9600                 ' Set baud rate to 9600
HSERIAL_RCSTA = %10010000           ' Enable serial port and continuous receive
HSERIAL_TXSTA = %00100100           ' Enable transmit and asynchronous mode
HSERIAL_CLEAR = ON                  ' Enable Error clearing on received characters

CCP1_PIN = PORTC.2
CCP2_PIN = PORTC.1

HBUS_BITRATE = 400                  ' Set I2C coms to 400 Kbits

' Define the USER pinouts for the graphic LCD
Symbol GLCD_DTPORT = PORTB          ' LCD's DT port
Symbol GLCD_RSPIN = PORTA.2         ' LCD's RS pin
Symbol GLCD_ENPIN = PORTA.0         ' LCD's EN pin
Symbol GLCD_RWPIN = PORTA.1         ' LCD's RW pin
Symbol GLCD_CS1PIN = PORTA.4        ' LCD's CS1 pin
Symbol GLCD_CS2PIN = PORTA.3        ' LCD's CS2 pin

Symbol TIMEOUT = 1000               ' A 1 second timeout for serial reception
Symbol RESPONSE_DELAY = 10          ' Wait 10ms before sending the response

Dim BYTE_SAVE[8] as Byte            ' Holds byte info for LCD ROTATES and SCROLLS
Dim XPOS as Byte SYSTEM             ' \
Dim YPOS as Byte SYSTEM             '  \
Dim XPOS_START as Byte SYSTEM       '   | For use with low-level LCD commands
Dim YPOS_START as Byte SYSTEM       '   | Must stay as SYSTEM to be in BANK0
Dim XPOS_END as Byte SYSTEM         '  /
Dim YPOS_END as Byte SYSTEM         ' /
Dim X_POSITION as Byte
Dim Y_POSITION as Byte
Dim XPOS_LOOP as Byte               ' Used in HORIZONTAL SCROLL
Dim AMOUNT_OF_SCROLLS as Byte       ' Used in HORIZONTAL SCROLL
Dim SERIAL_COMMAND as Byte          ' Holds the serial command instruction
Dim DATA_LOOP as Byte               ' Loop used when inputting, or outputting data
Dim DATA_BYTE as Byte
Dim RESPONSE as Byte                ' Response required after each instruction
Dim RESPONSE_ENABLE as Bit          ' ENABLE/DISABLE a RESPONSE after each command
Dim SET_CLEAR as Byte               ' Used by SET or CLEAR PIXEL
Dim RADIUS as Byte                  ' Used by CIRCLE and SQUARE
Dim LOCATION as Byte
Dim SLAVE_ADDR as Byte
Dim ADDR as Word
Dim BAUDRATE as Word                ' Save a copy of the baudrate
Dim TEMP_DWORD as Dword
Dim TEXT[201] as Byte               ' Text buffer array to hold 200 characters

' Define some RESPONSES
Symbol OK = "O"                     ' Standard RESPONSE
Symbol COMMAND_NOT_RECOGNISED = "N"
Symbol TIMEOUT_ERROR = "T"

'---------------------------------------------------------------------------------
Delayms 200                         ' Wait for the PICmicro to stabilise
Goto OVER_LOW_LEVEL_SUBS            ' Jump over the low level ASM subroutines

'---------------------------------------------------------------------------------
' LOW LEVEL assembler LCD Access Subroutines

Include "LL_INTERFACE.INC"    ' Load the low level LCD routines into memory
Include "LL_SCROLLS.INC"      ' Load the low level scroll routines into memory

'----[SETUP SOME VARIABLES BEFORE THE MAIN PROGRAM LOOP]--------------------------
OVER_LOW_LEVEL_SUBS:
Clear                               ' Clear all RAM before we start
ALL_DIGITAL = TRUE                  ' PORTA to all digital
Cls                                 ' Clear the LCD
X_POSITION = 0                      ' Reset the GLOBAL XPOS of the LCD
Y_POSITION = 0                      ' Reset the GLOBAL YPOS of the LCD
RESPONSE_ENABLE = 0                 ' Default to NO responses given
Goto MAIN_LOOP                      ' Jump over the subroutines

'----[SEND AN ACKNOWLEDGE]--------------------------------------------------------
' If enabled, a response is sent to the calling devices.
' The ACKNOWLEDGE is held in variable RESPONSE
SEND_RESPONSE:
If RESPONSE_ENABLE = 1 Then         ' Are RESPONSES enabled ?
 Delayms RESPONSE_DELAY             ' Yes. So wait appropriate time
 Hserout [13,HEX2 RESPONSE,13]      ' Send the RESPONSE value
Endif
Goto WAIT_FOR_INSTRUCTION           ' Go and wait for another command

'----[ACT UPON A TIMEOUT ERROR]---------------------------------------------------
' A TIMEOUT error was encountered, while reading the instructions from the PC
TIMEOUT_ERR:
If RESPONSE_ENABLE = 1 Then         ' Are RESPONSES enabled ?
 Delayms RESPONSE_DELAY             ' Yes. So wait appropriate time
 Hserout [13,"T",13]                ' Send a TIMEOUT error
Endif
Goto WAIT_FOR_INSTRUCTION           ' Go and wait for another command

'----[SET/CLEAR A PIXEL]----------------------------------------------------------
' Command expects XPOS (00 - 7F), YPOS (00 - 3F), SET or CLEAR (00 - 01)
DO_PLOT:
' Wait for XPOS, YPOS, and SET or CLEAR commands
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS, HEX2 YPOS, HEX2 SET_CLEAR]

XPOS = XPOS & %01111111             ' Ensure XPOS is within limits
YPOS = YPOS & %00111111             ' Ensure YPOS is within limits
If SET_CLEAR = 0 Then               ' Do we require the pixel cleared ?
 UnPlot YPOS,XPOS                   ' Yes. So CLEAR the pixel
Else                                ' Otherwise SET the pixel
```

```
 Plot YPOS,XPOS
Endif
Goto SEND_RESPONSE                        ' Send a response if enabled

'----[SCROLL THE SCREEN UP ONE LINE]----------------------------------------------
SCROLL:
YPOS = 1                                  ' Start at line 0
Repeat
 XPOS = 0                                 ' and position 0 on the line
 Repeat
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read from a line
  LCDWRITE YPOS - 1 , XPOS , [DATA_BYTE]  ' And write to the line above
  Inc XPOS
 Until XPOS.7 = 1                         ' Until a count less than 127
 Inc YPOS
Until YPOS.3 = 1                          ' Until a count less than 8
Print at 7,0,REP " "\21                   ' Blank the bottom line
Return

'----[SCROLL THE SCREEN DOWN ONE LINE]--------------------------------------------
SCROLL_DOWN:
YPOS = 6                                  ' Start at line 6
Repeat
 XPOS = 0                                 ' and position 0 on the line
 Repeat
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read from a line
  LCDWRITE YPOS + 1 , XPOS , [DATA_BYTE]  ' And write to the line below
  Inc XPOS
 Until XPOS.7 = 1                         ' Until a count less than 127
 Dec YPOS
Until YPOS = 255                          ' Until LINE reaches 0
Print at 0,0,REP " "\21                   ' Blank the top line
Return

'----[DISPLAY TEXT ON THE LCD]----------------------------------------------------
' Command expects text terminated by a NUL (0)
DO_TEXT:
' Receive the DATA to display, and load the array TEXT with it.
DATA_LOOP = 0
Repeat
 Hserin 500,TEXT_TIMEOUT,[DATA_BYTE] ' Get the text serially. Display if timed out
 TEXT[DATA_LOOP] = DATA_BYTE              ' Load the array with the incoming data
 If DATA_BYTE = 0 Then SKIP_OVER          ' Escape the loop if NUL found
 Inc DATA_LOOP                            ' Increment the loop
Until DATA_LOOP > 200               ' Allow a maximum of 200 characters printable

TEXT_TIMEOUT:
TEXT[DATA_LOOP] = 0                       ' Add a NUL if the loop timed out

SKIP_OVER:
DATA_LOOP = 0
Repeat
 DATA_BYTE = TEXT[DATA_LOOP]
 If DATA_BYTE = 0 Then SEND_RESPONSE      ' Escape the loop if  NUL found
 If DATA_BYTE = 13 Then                   ' CR found ?
  X_POSITION = 0
  Inc Y_POSITION
  Goto CHECK_FOR_SCROLL                   ' Time to scroll ?
 Endif
 If DATA_BYTE = 10 Then                   ' NEW LINE found ?
  Inc Y_POSITION
  Goto CHECK_FOR_SCROLL                   ' Time to scroll ?
 Endif
 Print at Y_POSITION , X_POSITION , DATA_BYTE
 Inc X_POSITION
 If X_POSITION > 20 Then X_POSITION = 0: Inc Y_POSITION
 CHECK_FOR_SCROLL:
 If Y_POSITION > 7 Then                   ' Time to scroll ?
  Y_POSITION = 7
  Gosub SCROLL
 Endif
 Inc DATA_LOOP
Until DATA_LOOP > 200               ' Allow a maximum of 200 characters printable
Goto SEND_RESPONSE                        ' Send a response if enabled

'----[POSITION THE CURSOR]--------------------------------------------------------
' Command expects XPOS (00 - 14), YPOS (00 - 07)
DO_POS:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS, HEX2 YPOS]

If XPOS > 20 Then Goto WAIT_FOR_INSTRUCTION     ' Is the XPOS range OK ?
If YPOS > 7 Then Goto WAIT_FOR_INSTRUCTION      ' Is the YPOS range OK ?

X_POSITION = XPOS                         ' Load GLOBAL XPOS with new XPOS
Y_POSITION = YPOS                         ' Load GLOBAL YPOS with new YPOS
Cursor Y_POSITION , X_POSITION            ' Move the cursor to position
Goto SEND_RESPONSE                        ' Send a response if enabled

'----[DRAW A LINE]----------------------------------------------------------------
' Command expects XPOS-START (00 - 7F), YPOS-START (00 - 3F), XPOS-END (00 - 7F),
' YPOS-END (00 - 3F), SET or CLEAR (00 - 01)
' Any parameters out of range will be reduced in size in order to fit
DO_LINE:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS_START, HEX2 YPOS_START,_
                            HEX2 XPOS_END, HEX2 YPOS_END, HEX2 SET_CLEAR]

XPOS_START = XPOS_START & %01111111
XPOS_END = XPOS_END & %01111111
YPOS_START = YPOS_START & %00111111
YPOS_END = YPOS_END & %00111111
Line SET_CLEAR , XPOS_START , YPOS_START , XPOS_END , YPOS_END
Goto SEND_RESPONSE                              ' Send a response if enabled

'----[DRAW A CIRCLE]--------------------------------------------------------------
' Command expects XPOS (00 - 7F), YPOS (00 - 3F), RADIUS (00 - 7F) ,
' SET or CLEAR (00 - 01)
' Any parameters out of range will be reduced in size in order to fit
DO_CIRCLE:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS, HEX2 YPOS, HEX2 RADIUS, HEX2 SET_CLEAR]

XPOS = XPOS & %01111111
YPOS = YPOS & %00111111
Circle SET_CLEAR , XPOS , YPOS , RADIUS   ' Draw a circle
Goto SEND_RESPONSE                        ' Send a response if enabled
```

```
'----[DRAW A SQUARE]--------------------------------------------------------------
' Command expects XPOS (00 - 7F), YPOS (00 - 3F), RADIUS (00 - FF) ,
' SET or CLEAR (00 - 01)
' Any parameters out of range will be reduced in size in order to fit
DO_BOX:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS, HEX2 YPOS, HEX2 RADIUS, HEX2 SET_CLEAR]

XPOS = XPOS & %01111111
YPOS = YPOS & %00111111
Box SET_CLEAR , XPOS , YPOS , RADIUS      ' Draw a box
Goto SEND_RESPONSE                        ' Send a response if enabled

'-----[SAVE A SCREEN TO EEPROM]---------------------------------------------------
' Command expects the eeprom position to save screen too.
' This corresponds to a 1K (1024) byte block of I2C eeprom memory
' Starting at eeprom 1. Which is address 0
' The first 1k of this eeprom is reserved for FONT data
' With a 24LC256 eeprom, 32 screens can be saved
DO_SAVE_SCREEN:

' Receive the eeprom location data
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 LOCATION]

If LOCATION = 0 Then                      ' Make sure we do not overwrite
 Delayms 10                               ' Font data
 Hserout [13,"E",13]                      ' Stored at eeprom location 0
 Goto WAIT_FOR_INSTRUCTION                ' Go and look for another command
Endif
If LOCATION > 128 Then Goto WAIT_FOR_INSTRUCTION ' Ignore if out of range

SLAVE_ADDR = %10100000                    ' Point to an eeprom
' Set the SLAVE address of the particular eeprom
' according to the LOCATION data received
If LOCATION > 32 Then SLAVE_ADDR = %10100010: LOCATION = LOCATION - 32
If LOCATION > 64 Then SLAVE_ADDR = %10100110: LOCATION = LOCATION - 64
If LOCATION > 96 Then SLAVE_ADDR = %10101000: LOCATION = LOCATION - 96
' A Screen of data is 1024 bytes in length
' So add the offset to the eeprom address
ADDR = LOCATION * 1024

YPOS = 0                                  ' Reset the outer loop counter
Repeat                                    ' Start an outer loop for the YPOS
 XPOS = 0                                 ' Reset the inner loop counter
 Repeat                                   ' Start an inner loop for the XPOS
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read the graphic LCD
  HBusout SLAVE_ADDR, ADDR, [DATA_BYTE]   ' Save the byte to eeprom
  Delayms 5                               ' Wait for the write to complete
  Inc ADDR                                ' Point to next address in eeprom memory
  Inc XPOS                                ' Point to the next XPOS location
 Until XPOS.7 = 1                         ' Loop until a count less than 127
 Inc YPOS                                 ' Point to next YPOS (line) location
Until YPOS.3 = 1                          ' Loop until a count less than 8
Hserout [13,"O",13]                       ' Acknowledge a screen saved

Goto WAIT_FOR_INSTRUCTION                 ' Go and look for another command
```

```
'-----[LOAD A SCREEN FROM EEPROM]-------------------------------------------------
' Command expects the eeprom position to load screen from.
' This corresponds to a 1K (1024) byte block of I2C eeprom memory
' Starting at eeprom 1. Which is address 0
' The first 1k of this eeprom is reserved for FONT data
' With a 24LC256 eeprom, 32 screens can be saved
DO_LOAD_SCREEN:

' Receive the eeprom location data
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 LOCATION]

If LOCATION > 128 Then Goto WAIT_FOR_INSTRUCTION      ' Ignore if out of range

SLAVE_ADDR = %10100000                                ' Point to an eeprom
' Set the SLAVE address of the particular eeprom
' according to the LOCATION data received
If LOCATION > 32 Then SLAVE_ADDR = %10100010: LOCATION = LOCATION - 32
If LOCATION > 64 Then SLAVE_ADDR = %10100110: LOCATION = LOCATION - 64
If LOCATION > 96 Then SLAVE_ADDR = %10101000: LOCATION = LOCATION - 96
' A Screen of data is 1024 bytes in length
' So add the offset to the eeprom address
ADDR = LOCATION * 1024

YPOS = 0                                  ' Reset the outer loop counter
Repeat                                    ' Start an outer loop for the YPOS
 XPOS = 0                                 ' Reset the inner loop counter
 Repeat                                   ' Start an inner loop for the XPOS
  Hbusin SLAVE_ADDR, ADDR, [DATA_BYTE]    ' Read the eeprom
  Call PBYTE                              ' Send byte to LCD
  'LCDWRITE YPOS, XPOS,[DATA_BYTE]        ' Same code as ASM but in BASIC
  Inc ADDR                                ' Point to next address in eeprom memory
  Inc XPOS                                ' Point to the next XPOS location
 Until XPOS.7 = 1                         ' Loop until a count less than 127
 Inc YPOS                                 ' Point to next YPOS (line) location
Until YPOS.3 = 1                          ' Until a count less than 8
Goto SEND_RESPONSE                        ' Send a response if enabled

'----[UPLOAD A SCREEN FROM THE SERIAL INTERFACE TO THE LCD]-----------------------
' Displays bytes read from the serial port

DO_UPLOAD_SCREEN:
YPOS = 0                                  ' Reset the outer loop counter
Repeat                                    ' Start an outer loop for the YPOS
 XPOS = 0                                 ' Reset the inner loop counter
 Repeat                                   ' Start an inner loop for the XPOS
  Hserin 100,WAIT_FOR_INSTRUCTION,[HEX2 DATA_BYTE] ' Get serial data
  Call PBYTE                              ' Send byte to LCD
  'LCDWRITE YPOS, XPOS,[DATA_BYTE]        ' Same code as ASM but in BASIC
  Inc XPOS                                ' Point to the next XPOS location
 Until XPOS.7 = 1                         ' Loop until a count less than 127
 Inc YPOS                                 ' Point to next YPOS (line) location
Until YPOS.3 = 1                          ' Loop until a count less than 8
Goto SEND_RESPONSE                        ' Send a response if enabled
```

```
'----[DOWNLOAD A SCREEN FROM THE LCD TO THE SERIAL INTERFACE]---------------------
' Sends the contents of the LCD via the serial interface
' Data is formatted as CDATA commands for easy insertion into
' a BASIC program
DO_DOWNLOAD_SCREEN:
Hserout [13,"CDATA "]                         ' Transmit first CDATA command
YPOS = 0                                      ' Reset the outer loop counter
Repeat                                        ' Start an outer loop for the YPOS
 XPOS = 0                                     ' Reset the inner loop counter
 Repeat                                       ' Start an inner loop for the XPOS
  DATA_BYTE = LCDREAD YPOS, XPOS              ' Read a byte from the LCD
  Hserout [IHEX2 DATA_BYTE]                   ' Send it to the serial port
  ' Format data into CDATA or COMMA
  If XPOS & %00001111 <> 15 Then Hserout[","] : Else : Hserout [13,"CDATA "]
  Inc XPOS                                    ' Point to the next XPOS location
 Until XPOS.7 = 1                             ' Loop until a count less than 127
 Inc YPOS                               ' Point to next YPOS (line) location
Until YPOS.3 = 1                              ' Loop until a count less than 8
Goto SEND_RESPONSE                            ' Send a response if enabled

'----[INCREASE or DECREASE THE LCD BACKLIGHT BRIGHTNESS]--------------------------
DO_BACKLIGHT:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 DATA_BYTE]   ' Receive the brightness level

Hpwm 1,DATA_BYTE,15000                        ' PWM at 15KHz
Goto SEND_RESPONSE                            ' Send a response if enabled

'----[SET THE BAUD RATE OF THE SERIAL INTERFACE]--------------------------------
DO_SET_BAUD:
' Sets the baud rate used by the USART serial interface
'SPBRG = ((((20000000) / (BAUDRATE)) + 8) / 16) - 1   ;     // For BRGH = 1

' Receive the baudrate as a decimal value
Hserin TIMEOUT,TIMEOUT_ERR,[DEC BAUDRATE]

TEMP_DWORD = 20000000 / BAUDRATE     ' Calculate the value to place into SPBRG
BAUDRATE = TEMP_DWORD + 8
DATA_BYTE = BAUDRATE / 16
DATA_BYTE = DATA_BYTE - 1                     ' DATA_BYTE now holds new baudrate
Delayms 100                                   ' Wait 100ms
HSerout [13,"O",13]                           ' Send acknowledge at old baudrate
Delayms 10                                    ' Wait another 10ms
SPBRG = DATA_BYTE                             ' Set the baudrate of the USART
Goto WAIT_FOR_INSTRUCTION                     ' Go and look for another command

'----[SET THE RESPONSE REPLY]-----------------------------------------------------
' ENABLES/DISABLES a response after a command
' 00 Disables the response
' 01 Enables the response
DO_RESPONSE_SET:
Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 RESPONSE_ENABLE]
Goto SEND_RESPONSE                            ' Send a response if enabled
```

```
'----[DISPLAY THE SPLASH SCREEN FOR THE LCD STARTUP]-------------------------------
' Displays a short splash screen
' Using some of the low-level scroll subroutines

SPLASH_SCREEN:
Print at 0,2,"SERIAL GRAPHIC LCD"
For XPOS_LOOP = 0 To 7
 Gosub SMOOTH_SCROLL_DOWN
 Delayms  27 - XPOS_LOOP
Next
Print at 0,6,"CROWNHILL"
For XPOS_LOOP = 0 To 7
 Gosub SMOOTH_SCROLL_DOWN
 Delayms 20 - XPOS_LOOP
Next
Print at 0,4,"WELCOME TO THE"
For XPOS_LOOP = 0 To 15
 Gosub SMOOTH_SCROLL_DOWN
 Delayms 16 - XPOS_LOOP
Next

Line 1,0,0,127,0                          ' Top Line
Line 1,0,63,127,63                        ' Bottom Line
Line 1,0,0,0,63                           ' Left Line
Line 1,127,0,127,63                       ' Right Line
Delayms 200
For XPOS_LOOP = 0 to 63
 Gosub SCROLL_LEFT
 Gosub SCROLL_LEFT
 Gosub SMOOTH_SCROLL_DOWN
Next
Cls
Cursor 0,0
Return

'----[MAIN PROGRAM LOOP STARTS HERE]-----------------------------------------------
MAIN_LOOP:
Gosub SPLASH_SCREEN                       ' Display the splash screen at start-up
WAIT_FOR_INSTRUCTION:
RESPONSE = OK                             ' Default to an OK response
XPOS_LOOP = 0                       ' Reset the counting loop before we enter
Hserin [SERIAL_COMMAND]                   ' Get the instruction
If SERIAL_COMMAND = "T" Then DO_TEXT      ' Goto TEXT routine ?
If SERIAL_COMMAND = "P" Then DO_POS       ' Goto POSITION CURSOR routine ?
If SERIAL_COMMAND = "A" Then DO_PLOT      ' Goto PLOT routine ?
If SERIAL_COMMAND = "L" Then DO_LINE      ' Goto LINE routine ?
If SERIAL_COMMAND = "B" Then DO_BOX       ' Goto BOX routine ?
If SERIAL_COMMAND = "R" Then DO_CIRCLE    ' Goto CIRCLE routine ?
' Goto SCROLL UP routine ?
If SERIAL_COMMAND = "S" Then Gosub SCROLL: X_POSITION = 0: Goto SEND_RESPONSE
' Goto SCROLL DOWN routine ?
If SERIAL_COMMAND = "D" Then Gosub SCROLL_DOWN: X_POSITION = 0: Goto SEND_RESPONSE
If SERIAL_COMMAND = "E" Then                  ' Do SCROLL LEFT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS] ' How many times ?
 Repeat                                       ' Create a loop for the amount
  Gosub SCROLL_LEFT                           ' SCROLL the display LEFT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "F" Then                  ' Do SCROLL RIGHT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub SCROLL_RIGHT                          ' SCROLL the display RIGHT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "I" Then                  ' Do ROTATE RIGHT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_RIGHT                          ' ROTATE the display RIGHT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "K" Then                  ' Do ROTATE LEFT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_LEFT                           ' ROTATE the display LEFT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "J" Then                  ' Do ROTATE PART RIGHT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS_START,HEX2 YPOS_START,HEX2 XPOS_END,_
 HEX2 YPOS_END,HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_PART_RIGHT                     ' ROTATE PART of the display RIGHT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "M" Then                  ' Goto ROTATE PART LEFT routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 XPOS_START,HEX2 YPOS_START,HEX2 XPOS_END,_
 HEX2 YPOS_END,HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_PART_LEFT                      ' ROTATE PART of the display LEFT
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "N" Then                  ' Do SMOOTH SCROLL UP routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub SMOOTH_SCROLL_UP                      ' SMOOTH SCROLL the display UP
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS         ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                     ' Go wait for another command via a RESPONSE
Endif
```

```
If SERIAL_COMMAND = "O" Then              ' Do SMOOTH SCROLL DOWN routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub SMOOTH_SCROLL_DOWN                ' SMOOTH SCROLL the display DOWN
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS     ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                 ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "V" Then              ' Goto SMOOTH ROTATE DOWN routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_DOWN                       ' SMOOTH ROTATE the display DOWN
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS     ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                 ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "Y" Then              ' Do SMOOTH ROTATE UP routine ?
 Hserin TIMEOUT,TIMEOUT_ERR,[HEX2 AMOUNT_OF_SCROLLS]
 Repeat
  Gosub ROTATE_UP                         ' SMOOTH SCROLL the display UP
  Inc XPOS_LOOP
 Until XPOS_LOOP >= AMOUNT_OF_SCROLLS     ' Have we reached the end of the loop ?
 Goto SEND_RESPONSE                 ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "C" Then
 Cls
 X_POSITION = 0
 Y_POSITION = 0
 Goto SEND_RESPONSE                 ' Go wait for another command via a RESPONSE
Endif
If SERIAL_COMMAND = "Z" Then DO_SAVE_SCREEN     ' Goto SAVE SCREEN routine ?
If SERIAL_COMMAND = "X" Then DO_LOAD_SCREEN     ' Goto LOAD SCREEN routine ?
If SERIAL_COMMAND = "U" Then DO_UPLOAD_SCREEN   ' Goto UPLOAD SCREEN routine ?
If SERIAL_COMMAND = "W" Then DO_DOWNLOAD_SCREEN ' Goto DOWNLOAD SCREEN routine ?
If SERIAL_COMMAND = "H" Then DO_SET_BAUD        ' Goto BAUDRATE setup routine ?
If SERIAL_COMMAND = "G" Then DO_BACKLIGHT ' Goto BACKLIGHT BRIGHTNESS routine ?
If SERIAL_COMMAND = "Q" Then DO_RESPONSE_SET    ' Goto RESPONSE SET routine ?

' Command must not be recognised, so look for another
RESPONSE = COMMAND_NOT_RECOGNISED
Goto SEND_RESPONSE

Include "FONT.INC"                        ' Load the LCD font table
```

Microcontroller Controlled Firmware Source

```
' PROTON Graphic LCD RS232 serial interface
' For use with a microcontroller
'
' Uses the low-level LCD interfacing subroutines for intensive commands
' Such as HORIZONTAL SCROLLS

DEVICE = 16F876
XTAL = 20

LCD_DTPORT = PORTB
LCD_ENPIN = PORTA.0
LCD_RWPIN = PORTA.1
LCD_RSPIN = PORTA.2
LCD_CS2PIN = PORTA.3
LCD_CS1PIN = PORTA.4
LCD_TYPE = GRAPHIC
INTERNAL_FONT = ON                    ' Use an internal font for the LCD text

SDA_PIN = PORTC.4                     ' Set serial eeprom
SCL_PIN = PORTC.3                     ' data/clock lines

HSERIAL_BAUD = 9600                   ' Set baud rate to 9600
HSERIAL_RCSTA = %10010000             ' Enable serial port and continuous receive
HSERIAL_TXSTA = %00100100             ' Enable transmit and asynchronous mode
HSERIAL_CLEAR = ON                    ' Enable Error clearing on received characters

CCP1_PIN = PORTC.2
CCP2_PIN = PORTC.1

HBUS_BITRATE = 400                    ' Set I2C coms to 400 Kbits

' Define the USER pinouts for the graphic LCD
Symbol GLCD_DTPORT = PORTB            ' LCD's DT port
Symbol GLCD_RSPIN = PORTA.2           ' LCD's RS pin
Symbol GLCD_ENPIN = PORTA.0           ' LCD's EN pin
Symbol GLCD_RWPIN = PORTA.1           ' LCD's RW pin
Symbol GLCD_CS1PIN = PORTA.4          ' LCD's CS1 pin
Symbol GLCD_CS2PIN = PORTA.3          ' LCD's CS2 pin

Symbol TIMEOUT = 1000                 ' A 1 second timeout for serial reception

Dim BYTE_SAVE[8] as Byte              ' Holds byte info for LCD ROTATES and SCROLLS
Dim XPOS as Byte SYSTEM               ' \
Dim YPOS as Byte SYSTEM               '  \
Dim XPOS_START as Byte SYSTEM         '   | For use with low-level LCD commands
Dim YPOS_START as Byte SYSTEM         '   | Must stay as SYSTEM to be in BANK0
Dim XPOS_END as Byte SYSTEM           '  /
Dim YPOS_END as Byte SYSTEM           ' /
Dim X_POSITION as Byte
Dim Y_POSITION as Byte
Dim XPOS_LOOP as Byte                 ' Used in HORIZONTAL SCROLL
Dim AMOUNT_OF_SCROLLS as Byte         ' Used in HORIZONTAL SCROLL
Dim SERIAL_COMMAND as Byte            ' Holds the serial command instruction
Dim DATA_LOOP as Byte                 ' Loop used when inputting, or outputting data
Dim DATA_BYTE as Byte
Dim SET_CLEAR as Byte                 ' Used by SET or CLEAR PIXEL
Dim RADIUS as Byte                    ' Used by CIRCLE and SQUARE
Dim RESPONSE_DELAY as Byte
Dim LOCATION as Byte
Dim SLAVE_ADDR as Byte
Dim ADDR as Word
Dim BAUDRATE as Word                    ' Save a copy of the baudrate
Dim TEMP_DWORD as Dword
Dim TEXT[201] as Byte                   ' Text buffer array to hold 200 characters

'--------------------------------------------------------------------------------
Delayms 200                             ' Wait for the PICmicro to stabilise
Goto OVER_LOW_LEVEL_SUBS                ' Jump over the low level ASM subroutines

'--------------------------------------------------------------------------------
' LOW LEVEL assembler LCD Access Subroutines

Include "LL_INTERFACE.INC"      ' Load the low level LCD routines into memory
Include "LL_SCROLLS.INC"        ' Load the low level scroll routines into memory

'----[SETUP SOME VARIABLES BEFORE THE MAIN PROGRAM LOOP]--------------------------
OVER_LOW_LEVEL_SUBS:
Clear                                   ' Clear all RAM before we start
ALL_DIGITAL = TRUE                      ' PORTA to all digital
Cls                                     ' Clear the LCD
X_POSITION = 0                          ' Reset the GLOBAL XPOS of the LCD
Y_POSITION = 0                          ' Reset the GLOBAL YPOS of the LCD
RESPONSE_DELAY = 1

'----[MAIN PROGRAM LOOP STARTS HERE]----------------------------------------------

Gosub SPLASH_SCREEN                     ' Display the splash screen at start-up

WAIT_FOR_INSTRUCTION:
Hserin [SERIAL_COMMAND]                       ' Get the instruction
SERIAL_COMMAND = SERIAL_COMMAND - 65          ' Strip off the ASCII code

BRANCHL SERIAL_COMMAND,[DO_PLOT,DO_BOX,DO_CLS,DO_SCROLL_DOWN,DO_SCROLL_LEFT,_
DO_SCROLL_RIGHT,DO_BACKLIGHT,DO_SET_BAUD,DO_ROTATE_RIGHT,DO_ROTATE_PART_RIGHT,_
DO_ROTATE_LEFT,DO_LINE,DO_ROTATE_PART_LEFT,DO_SMOOTH_SCROLL_UP,_
DO_SMOOTH_SCROLL_DOWN,DO_POS,DO_RESPONSE_SET,DO_CIRCLE,DO_SCROLL_UP,_
DO_TEXT,DO_UPLOAD_SCREEN,DO_ROTATE_DOWN,DO_DOWNLOAD_SCREEN,_
DO_LOAD_SCREEN,DO_ROTATE_UP,DO_SAVE_SCREEN]

Goto WAIT_FOR_INSTRUCTION                     ' Go and wait for another instruction

'----[ACT UPON A TIMEOUT ERROR]---------------------------------------------------
' A TIMEOUT error was encountered, while reading the instructions from the PC
TIMEOUT_ERR:
Delayms RESPONSE_DELAY                  ' Yes. So wait appropriate time
Hserout ["T"]                           ' Send a TIMEOUT error
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command

'----[SET/CLEAR A PIXEL]----------------------------------------------------------
' Command expects XPOS (0 - 127), YPOS (0 - 63), SET or CLEAR (0 - 1)
DO_PLOT:
' Wait for XPOS, YPOS, and SET or CLEAR commands
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS, YPOS, SET_CLEAR]

XPOS = XPOS & %01111111                 ' Ensure XPOS is within limits
YPOS = YPOS & %00111111                 ' Ensure YPOS is within limits
If SET_CLEAR = 0 Then                   ' Do we require the pixel cleared ?
 UnPlot YPOS,XPOS                       ' Yes. So CLEAR the pixel
Else                                    ' Otherwise SET the pixel
```

```
 Plot YPOS,XPOS
Endif
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[SCROLL THE SCREEN UP ONE LINE]----------------------------------------------
SCROLL:
YPOS = 1                                  ' Start at line 0
Repeat
 XPOS = 0                                 ' and position 0 on the line
 Repeat
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read from a line
  LCDWRITE YPOS - 1 , XPOS , [DATA_BYTE]  ' And write to the line above
  Inc XPOS
 Until XPOS.7 = 1                         ' Until a count less than 127
 Inc YPOS
Until YPOS.3 = 1                          ' Until a count less than 8
Print at 7,0,REP " "\21                   ' Blank the bottom line
Return

'----[SCROLL THE SCREEN DOWN ONE LINE]--------------------------------------------
SCROLL_DOWN:
YPOS = 6                                  ' Start at line 6
Repeat
 XPOS = 0                                 ' and position 0 on the line
 Repeat
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read from a line
  LCDWRITE YPOS + 1 , XPOS , [DATA_BYTE]  ' And write to the line below
  Inc XPOS
 Until XPOS.7 = 1                         ' Until a count less than 127
 Dec YPOS
Until YPOS = 255                          ' Until LINE reaches 0
Print at 0,0,REP " "\21                   ' Blank the top line
Return

'----[DISPLAY TEXT ON THE LCD]----------------------------------------------------
' Command expects text terminated by a NUL (0)
DO_TEXT:
' Receive the DATA to display, and load the array TEXT with it.
DATA_LOOP = 0
Repeat
 Hserin 500,TEXT_TIMEOUT,[DATA_BYTE] ' Get the text serially. Display if timed out
 TEXT[DATA_LOOP] = DATA_BYTE              ' Load the array with the incoming data
 If DATA_BYTE = 0 Then SKIP_OVER          ' Escape the loop if NUL found
 Inc DATA_LOOP                            ' Increment the loop
Until DATA_LOOP > 200               ' Allow a maximum of 200 characters printable

TEXT_TIMEOUT:
TEXT[DATA_LOOP] = 0                       ' Add a NUL if the loop timed out

SKIP_OVER:
DATA_LOOP = 0
Repeat
 DATA_BYTE = TEXT[DATA_LOOP]
 If DATA_BYTE = 0 Then                    ' Escape the loop if  NUL found
  Delayms RESPONSE_DELAY
  Hserout ["O"]                           ' Send an ACK
  Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command
 Endif
 If DATA_BYTE = 13 Then                   ' CR found ?
  X_POSITION = 0
  Inc Y_POSITION
  Goto CHECK_FOR_SCROLL                   ' Time to scroll ?
 Endif
 If DATA_BYTE = 10 Then                   ' NEW LINE found ?
  Inc Y_POSITION
  Goto CHECK_FOR_SCROLL                   ' Time to scroll ?
 Endif
 Print at Y_POSITION , X_POSITION , DATA_BYTE
 Inc X_POSITION
 If X_POSITION > 20 Then X_POSITION = 0: Inc Y_POSITION
 CHECK_FOR_SCROLL:
 If Y_POSITION > 7 Then                   ' Time to scroll ?
  Y_POSITION = 7
  Gosub SCROLL
 Endif
 Inc DATA_LOOP
Until DATA_LOOP > 200               ' Allow a maximum of 200 characters printable
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[POSITION THE CURSOR]--------------------------------------------------------
' Command expects XPOS (0 - 20), YPOS (0 - 7)
DO_POS:
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS, YPOS]

If XPOS > 20 Then Goto WAIT_FOR_INSTRUCTION     ' Is the XPOS range OK ?
If YPOS > 7 Then Goto WAIT_FOR_INSTRUCTION      ' Is the YPOS range OK ?

X_POSITION = XPOS                         ' Load GLOBAL XPOS with new XPOS
Y_POSITION = YPOS                         ' Load GLOBAL YPOS with new YPOS
Cursor Y_POSITION , X_POSITION            ' Move the cursor to position
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[CLEAR THE LCD]--------------------------------------------------------------
DO_CLS:
Cls
X_POSITION = 0
Y_POSITION = 0
Delayms RESPONSE_DELAY                    ' Wait appropriate time
Hserout ["O"]                             ' Send the RESPONSE value
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another instruction

'----[DRAW A LINE]----------------------------------------------------------------
' Command expects XPOS-START (0 - 127), YPOS-START (0 - 63), XPOS-END (0 - 127),
' YPOS-END (0 - 127), SET or CLEAR (0 - 1)
' Any parameters out of range will be reduced in size in order to fit
DO_LINE:
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS_START, YPOS_START, _END, YPOS_END, SET_CLEAR]

XPOS_START = XPOS_START & %01111111
XPOS_END = XPOS_END & %01111111
YPOS_START = YPOS_START & %00111111
YPOS_END = YPOS_END & %00111111
Line SET_CLEAR , XPOS_START , YPOS_START , XPOS_END , YPOS_END
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command
```

```
'----[DRAW A CIRCLE]--------------------------------------------------------------
' Command expects XPOS (0 - 127), YPOS (0 - 63), RADIUS (0 - 255) ,
' SET or CLEAR (0 - 1)
' Any parameters out of range will be reduced in size in order to fit
DO_CIRCLE:
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS, YPOS, RADIUS, SET_CLEAR]

XPOS = XPOS & %01111111
YPOS = YPOS & %00111111
Circle SET_CLEAR , XPOS , YPOS , RADIUS   ' Draw a circle
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[DRAW A SQUARE]--------------------------------------------------------------
' Command expects XPOS (0 - 127), YPOS (0 - 63), RADIUS (0 - 255) ,
' SET or CLEAR (00 - 01)
' Any parameters out of range will be reduced in size in order to fit
DO_BOX:
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS, YPOS, RADIUS, SET_CLEAR]

XPOS = XPOS & %01111111
YPOS = YPOS & %00111111
Box SET_CLEAR , XPOS , YPOS , RADIUS      ' Draw a box
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'-----[SAVE A SCREEN TO EEPROM]---------------------------------------------------
' Command expects the eeprom position to save screen too.
' This corresponds to a 1K (1024) byte block of I2C eeprom memory
' Starting at eeprom 1. Which is address 0
' The first 1k of this eeprom is reserved for FONT data
' With a 24LC256 eeprom, 32 screens can be saved
DO_SAVE_SCREEN:

' Receive the eeprom location data
Hserin TIMEOUT,TIMEOUT_ERR,[LOCATION]

If LOCATION = 0 Then                      ' Make sure we do not overwrite
 Delayms 10                               ' Font data
 Hserout ["E"]                      ' Stored at eeprom location 0
 Goto WAIT_FOR_INSTRUCTION                ' Go and look for another command
Endif

SLAVE_ADDR = %10100000                    ' Point to an eeprom
' Set the SLAVE address of the particular eeprom
' according to the LOCATION data received
If LOCATION > 32 Then SLAVE_ADDR = %10100010: LOCATION = LOCATION - 32
If LOCATION > 64 Then SLAVE_ADDR = %10100110: LOCATION = LOCATION - 64
If LOCATION > 96 Then SLAVE_ADDR = %10101000: LOCATION = LOCATION - 96
' A Screen of data is 1024 bytes in length
' So add the offset to the eeprom address
ADDR = LOCATION * 1024

YPOS = 0                                  ' Reset the outer loop counter
Repeat                                    ' Start an outer loop for the YPOS
 XPOS = 0                                 ' Reset the inner loop counter
 Repeat                                   ' Start an inner loop for the XPOS
  DATA_BYTE = LCDREAD YPOS, XPOS          ' Read the graphic LCD
  HBusout SLAVE_ADDR, ADDR, [DATA_BYTE]   ' Save the byte to eeprom
  Delayms 5                               ' Wait for the write to complete
  Inc ADDR                                 ' Point to next address in eeprom memory
  Inc XPOS                                 ' Point to the next XPOS location
 Until XPOS.7 = 1                          ' Loop until a count less than 127
 Inc YPOS                                  ' Point to next YPOS (line) location
Until YPOS.3 = 1                           ' Loop until a count less than 8
Hserout ["O"]                              ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                  ' Go and wait for another command

'-----[LOAD A SCREEN FROM EEPROM]----------------------------------------------------
' Command expects the eeprom position to load screen from.
' This corresponds to a 1K (1024) byte block of I2C eeprom memory
' Starting at eeprom 1. Which is address 0
' The first 1k of this eeprom is reserved for FONT data
' With a 24LC256 eeprom, 32 screens can be saved
DO_LOAD_SCREEN:

' Receive the eeprom location data
Hserin TIMEOUT,TIMEOUT_ERR,[LOCATION]
SLAVE_ADDR = %10100000                                 ' Point to an eeprom
' Set the SLAVE address of the particular eeprom
' according to the LOCATION data received
If LOCATION > 32 Then SLAVE_ADDR = %10100010: LOCATION = LOCATION - 32
If LOCATION > 64 Then SLAVE_ADDR = %10100110: LOCATION = LOCATION - 64
If LOCATION > 96 Then SLAVE_ADDR = %10101000: LOCATION = LOCATION - 96
' A Screen of data is 1024 bytes in length
' So add the offset to the eeprom address
ADDR = LOCATION * 1024
YPOS = 0                                   ' Reset the outer loop counter
Repeat                                     ' Start an outer loop for the YPOS
 XPOS = 0                                  ' Reset the inner loop counter
 Repeat                                    ' Start an inner loop for the XPOS
  Hbusin SLAVE_ADDR, ADDR, [DATA_BYTE]     ' Read the eeprom
  Call PBYTE                               ' Send byte to LCD
  'LCDWRITE YPOS, XPOS,[DATA_BYTE]         ' Same code as ASM but in BASIC
  Inc ADDR                                 ' Point to next address in eeprom memory
  Inc XPOS                                 ' Point to the next XPOS location
 Until XPOS.7 = 1                          ' Loop until a count less than 127
 Inc YPOS                                  ' Point to next YPOS (line) location
Until YPOS.3 = 1                           ' Until a count less than 8
Delayms RESPONSE_DELAY
Hserout ["O"]                              ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                  ' Go and wait for another command

'----[UPLOAD A SCREEN FROM THE SERIAL INTERFACE TO THE LCD]-----------------------
' Displays bytes read from the serial port
DO_UPLOAD_SCREEN:
YPOS = 0                                   ' Reset the outer loop counter
Repeat                                     ' Start an outer loop for the YPOS
 XPOS = 0                                  ' Reset the inner loop counter
 Repeat                                    ' Start an inner loop for the XPOS
  Hserin 100,WAIT_FOR_INSTRUCTION,[DATA_BYTE] ' Get serial data
  Call PBYTE                               ' Send byte to LCD
  'LCDWRITE YPOS, XPOS,[DATA_BYTE]         ' Same code as ASM but in BASIC
  Inc XPOS                                 ' Point to the next XPOS location
 Until XPOS.7 = 1                          ' Loop until a count less than 127
 Inc YPOS                                  ' Point to next YPOS (line) location
Until YPOS.3 = 1                           ' Loop until a count less than 8
Delayms RESPONSE_DELAY
Hserout ["O"]                              ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                  ' Go and wait for another command
```

```
'----[DOWNLOAD A SCREEN FROM THE LCD TO THE SERIAL INTERFACE]---------------------
' Sends the contents of the LCD via the serial interface
DO_DOWNLOAD_SCREEN:
YPOS = 0                                        ' Reset the outer loop counter
Repeat                                          ' Start an outer loop for the YPOS
 XPOS = 0                                       ' Reset the inner loop counter
 Repeat                                         ' Start an inner loop for the XPOS
  DATA_BYTE = LCDREAD YPOS, XPOS                ' Read a byte from the LCD
  Hserout [DATA_BYTE]                           ' Send it to the serial port
  Inc XPOS                                      ' Point to the next XPOS location
 Until XPOS.7 = 1                               ' Loop until a count less than 127
 Inc YPOS                                 ' Point to next YPOS (line) location
Until YPOS.3 = 1                                ' Loop until a count less than 8
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[INCREASE or DECREASE THE LCD BACKLIGHT BRIGHTNESS]--------------------------
DO_BACKLIGHT:
Hserin TIMEOUT,TIMEOUT_ERR,[DATA_BYTE]          ' Receive the brightness level

Hpwm 1,DATA_BYTE,15000                          ' PWM at 15KHz
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[SET THE BAUD RATE OF THE SERIAL INTERFACE]--------------------------------
DO_SET_BAUD:
' Sets the baud rate used by the USART serial interface
'SPBRG = ((((20000000) / (BAUDRATE)) + 8) / 16) - 1   ;     // For BRGH = 1

' Receive the baudrate as a decimal value
Hserin TIMEOUT,TIMEOUT_ERR,[DEC BAUDRATE]

TEMP_DWORD = 20000000 / BAUDRATE    ' Calculate the value to place into SPBRG
BAUDRATE = TEMP_DWORD + 8
DATA_BYTE = BAUDRATE / 16
DATA_BYTE = DATA_BYTE - 1                       ' DATA_BYTE now holds new baudrate
Delayms 100                                     ' Wait 100ms
HSerout ["O"]                                   ' Send acknowledge at old baudrate
Delayms 10                                      ' Wait another 10ms
SPBRG = DATA_BYTE                               ' Set the baudrate of the USART
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[SET THE RESPONSE REPLY]-----------------------------------------------------
' ENABLES/DISABLES a delay before response
' 0 Disables delay
' 1 Enables delay
DO_RESPONSE_SET:
Hserin TIMEOUT,TIMEOUT_ERR,[RESPONSE_ENABLE]
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command
```

```
'----[SCROLL SUBROUTINE LAUNCHERS]------------------------------------------------
'
'----[DO SCROLL UP ROUTINE]-------------------------------------------------------
DO_SCROLL_UP:
Gosub SCROLL
X_POSITION = 0
Delayms RESPONSE_DELAY                          ' Wait appropriate time
Hserout ["O"]                                   ' Send the RESPONSE value
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another instruction

'----[DO SCROLL DOWN ROUTINE]-----------------------------------------------------
DO_SCROLL_DOWN:
Gosub SCROLL_DOWN
X_POSITION = 0
Delayms RESPONSE_DELAY                          ' Wait appropriate time
Hserout ["O"]                                   ' Send the RESPONSE value
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another instruction

'----[DO SCROLL LEFT ROUTINE]-----------------------------------------------------
DO_SCROLL_LEFT:
XPOS_LOOP = 0                                   ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                          ' Create a loop for the amount
 Gosub SCROLL_LEFT                              ' SCROLL the display LEFT
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS      ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command


'----[DO SCROLL RIGHT ROUTINE]----------------------------------------------------
DO_SCROLL_RIGHT:
XPOS_LOOP = 0                                   ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                          ' Create a loop for the amount
 Gosub SCROLL_RIGHT                             ' SCROLL the display RIGHT
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS      ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[DO SMOOTH SCROLL UP ROUTINE]------------------------------------------------
DO_SMOOTH_SCROLL_UP:
XPOS_LOOP = 0                                   ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                          ' Create a loop for the amount
 Gosub SMOOTH_SCROLL_UP                         ' SMOOTH SCROLL the display UP
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS      ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command
'----[DO SMOOTH SCROLL DOWN ROUTINE]----------------------------------------------
DO_SMOOTH_SCROLL_DOWN:
XPOS_LOOP = 0                                   ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                          ' Create a loop for the amount
 Gosub SMOOTH_SCROLL_DOWN                       ' SMOOTH SCROLL the display DOWN
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS      ' Have we reached the end of the loop ?
```

```
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command
'----[DO ROTATE RIGHT ROUTINE]----------------------------------------------------
DO_ROTATE_RIGHT:
XPOS_LOOP = 0                                 ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                        ' Create a loop for the amount
 Gosub ROTATE_RIGHT                           ' ROTATE the display RIGHT
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS    ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command

'----[DO ROTATE LEFT ROUTINE]-----------------------------------------------------
DO_ROTATE_LEFT:
XPOS_LOOP = 0                                 ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                        ' Create a loop for the amount
 Gosub ROTATE_LEFT                            ' ROTATE the display LEFT
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS    ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command

'----[DO ROTATE UP ROUTINE]-------------------------------------------------------
DO_ROTATE_UP:
XPOS_LOOP = 0                                 ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                        ' Create a loop for the amount
 Gosub ROTATE_UP                              ' SMOOTH SCROLL the display UP
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS    ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command

'----[DO ROTATE DOWN ROUTINE]-----------------------------------------------------
DO_ROTATE_DOWN:
XPOS_LOOP = 0                                 ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[AMOUNT_OF_SCROLLS]  ' How many times ?
Repeat                                        ' Create a loop for the amount
 Gosub ROTATE_DOWN                            ' SMOOTH SCROLL the display DOWN
 Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS    ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
Goto WAIT_FOR_INSTRUCTION               ' Go and wait for another command

'----[DO ROTATE PART RIGHT ROUTINE]-----------------------------------------------
DO_ROTATE_PART_RIGHT:
XPOS_LOOP = 0                                 ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS_START,YPOS_START,_
                            XPOS_END, YPOS_END,AMOUNT_OF_SCROLLS]
Repeat
Gosub ROTATE_PART_RIGHT                       ' ROTATE PART of the display RIGHT
Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS    ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                           ' Send an ACK
```

```
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command
'----[DO ROTATE PART LEFT ROUTINE]-----------------------------------------------
DO_ROTATE_PART_LEFT:
XPOS_LOOP = 0                                   ' Reset the counting loop
Hserin TIMEOUT,TIMEOUT_ERR,[XPOS_START,YPOS_START,_
                            XPOS_END, YPOS_END,AMOUNT_OF_SCROLLS]
Repeat
Gosub ROTATE_PART_LEFT                          ' ROTATE PART of the display LEFT
Inc XPOS_LOOP
Until XPOS_LOOP >= AMOUNT_OF_SCROLLS      ' Have we reached the end of the loop ?
Delayms RESPONSE_DELAY
Hserout ["O"]                             ' Send an ACK
Goto WAIT_FOR_INSTRUCTION                 ' Go and wait for another command

'----[DISPLAY THE SPLASH SCREEN FOR THE LCD STARTUP]------------------------------
' Displays a short splash screen
' Using some of the low-level scroll subroutines
SPLASH_SCREEN:
Print at 0,2,"SERIAL GRAPHIC LCD"
For XPOS_LOOP = 0 To 7
 Gosub SMOOTH_SCROLL_DOWN
 Delayms  27 - XPOS_LOOP
Next
Print at 0,6,"CROWNHILL"
For XPOS_LOOP = 0 To 7
 Gosub SMOOTH_SCROLL_DOWN
 Delayms 20 - XPOS_LOOP
Next
Print at 0,4,"WELCOME TO THE"
For XPOS_LOOP = 0 To 15
 Gosub SMOOTH_SCROLL_DOWN
 Delayms 16 - XPOS_LOOP
Next

Line 1,0,0,127,0                          ' Top Line
Line 1,0,63,127,63                        ' Bottom Line
Line 1,0,0,0,63                           ' Left Line
Line 1,127,0,127,63                       ' Right Line
Delayms 200
For XPOS_LOOP = 0 to 63
 Gosub SCROLL_LEFT
 Gosub SCROLL_LEFT
 Gosub SMOOTH_SCROLL_DOWN
Next


Cls
Cursor 0,0
Return

Include "FONT.INC"                        ' Load the LCD font table
```