USING LCD DISPLAY MODULES

Ian Jackson March 2011

BASIC LCD DISPLAY MODULES

One of the difficulties with small Microprocessor projects is the Human Interface. Letting an operator know what the project is thinking becomes difficult if it must go beyond a few LED's and electronic 'beep' sounds. For many years there have been a range of Liquid Crystal Display (LCD) modules available that can show plain numbers and text on a small screen.
The cost of such modules have become inexpensive and are a sound method of getting the message across.

Main screen in a solar Air Heater system

The problem is that if you have not had a reasonable introduction first, they can be hard to work with. There are pitfalls for the unwary. A special 'chipset' has been developed called the **HD4478U** and pretty much all 'non-graphic' displays loosely work to that standard. The problem is that this standard has been varied in subtle ways depending upon who's brand of display you purchase.

Electrical Connections

These modules work from a 5V D.C. supply. The display part only draws 2-3 milliamps, but the LED backlight can draw up to 120ma if it is pushed hard enough. Normally this is current limited with say a 22 Ohm resistor down to a dull glow so that it can still be read ok at night.

There are usually 16 electrical connections to the display modules. Sometimes they are **2-rows-of-8** suitable for a ribbon cable plug, but more often they are **1-row-of-16**. To make it interesting Pins 15 and 16 for the LED backlight are usually next to pins 1 and 2. Be warned, on some modules pins 1 and 2 (5V power rail) are swapped and Pins 15 and 16 (Backlight) may also be swapped.

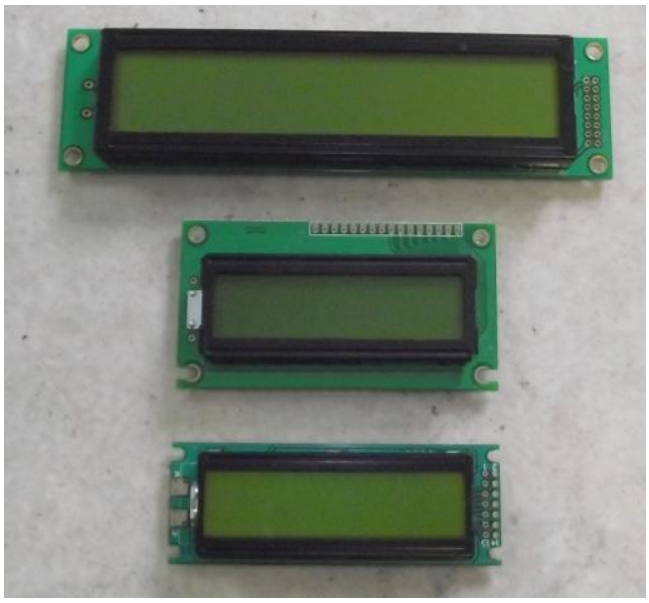
With all that in mind, here is a table of common electrical connections:

Pin	Symbol	Function	
1	Vss	Negative rail of display (GND)	Sometimes Swapped In some module brands
2	Vdd	Positive rail of display (+ 5V)	
3	Vo	Adjust display contrast	
4	RS	Register Select signal	
5	R/W	Read / Write signal	
6	E	Operation Enable signal	
7	DB0	Data Bus line	
8	DB1	Data Bus line	
9	DB2	Data Bus line	
10	DB3	Data Bus line	
11	DB4	Data Bus line	
12	DB5	Data Bus line	
13	DB6	Data Bus line	
14	DB7	Data Bus line	
15	LED +	Power to LED backlight	Sometimes Swapped In some module brands
16	LED -	Power to LED backlight	

As may be seen, it consumes quite a lot of resources from the microprocessor managing the display, using up 11 CPU ports in one go!. (DB0-7 plus RS, R/W and E)
(The 11 ports can be reduced to 7 ports if a fiddly addressing mode is used on the Data Bus.)

The adjacent examples of LCD displays show:

- A large 1 line x 16 character module,
- A 2 line 16 char module with the 16 pins in a row.
- And a smaller 2 line x 16 character module with the 2 x 8 pin IDC header format.



To the left of each module two terminal pads may be observed, which are alternative terminations for accessing the screen backlight feature

Many more sizes and types are available in the family of displays, mostly priced between $20 and $50 each

Hardware Control Lines

This is where it gets a little complex as communicating with these displays is very programmer 'unfriendly'. Should you ever meet the person who designed this system, do not offer to buy them a beer. In fact, they should be given a 'lecture' in the car park. On the bright side once routines have been put together for these displays, they are reasonably universal may be applied to most of the modules available.

The **EN** line is called "**Enable**." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring **EN** **high** (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD, but is usually around 250 nanoseconds), and end by bringing it **low** (0) again.

The **RS** line is the "**Register Select**" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When **RS** is high (1), the data being sent is text data which should be displayed on the screen. For example, to Display the letter "G" on the screen you would set **RS** **high**.

The **RW** line is the "**Read/Write**" control line. When **RW** is **low** (0), the information on the data bus is being written to the LCD. When **RW** is **high** (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Retrieve LCD status") is a read command. All others are write commands…so **RW** will almost always be **low**.

The **Data Bus** consists of **4** or **8** lines (depending on the mode of operation selected by the user). In the case of an **8-bit** data bus, the lines are referred to as **DB0, DB1, DB2, DB3, DB4, DB5, DB6,** and **DB7**

At this point the process becomes more complex, as to get the display going various initialisation 'commands' must be issued by presenting a command code to the **Data Bus, RS, R/W** lines and driving **E** line into the correct state.

There are about a dozen Command Codes in the display data sheets, but typically only a few of these are needed to present text to the display. (values of **x** mean don't care)

Command	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear disp.	0	0	0	0	0	0	0	0	0	1
Turn the Cursor OFF	0	0	0	0	0	0	1	1	0	0
Set Data Bus to 4 bit	0	0	0	0	1	0	1	1	x	x
Set Data Bus to 8 bit	0	0	0	0	1	1	1	1	x	x
Set a char address to: line1, position 0 on a 2-line screen (addr 00h)	0	0	1	0 (A6)	0 (A5)	0 (A4)	0 (A3)	0 (A2)	0 (A1)	0 (A0)
Set a char address to: line2, position 0 on a 2-line screen (addr 40h)	0	0	1	1 (A6)	0 (A5)	0 (A4)	0 (A3)	0 (A2)	0 (A1)	0 (A0)
Write char to that address	1	0	(D7)	(D6)	(D5)	(D4)	(D3)	(D2)	(D1)	(D0)

In practise, to write a program to present character information to a screen, a long and carefully timed sequence of values must be presented to the screen module over many steps.

Screen Addressing

Screen Addressing is important. If you wish to put a number or letter up on a screen in a particular position, you must write an ASCII character value into an appropriate screen address. The screen addresses are separated into Rows and Columns, with the rows starting with '**Row 1**' and the columns starting with '**Column 0**'

The screen addresses within data sheets are normally in **Hexidecimal**, but for this exercise, they are shown here in their **decimal** equivalents. Addresses are set by writing values to 7 of the 8 lines within the Data Bus, designated as **A0** to **A6**. The mapping is a bit of a dog's breakfast as the locations are not as consecutive as perhaps they could have been:

A 16 character x 1-line display

Char. Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1 Address	00	01	02	03	04	05	06	07	64	65	66	67	68	69	70	71

A 16 character x 2-line display

Char. Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1 Address	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Line 2 Address	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79

A 16 character x 4-line display

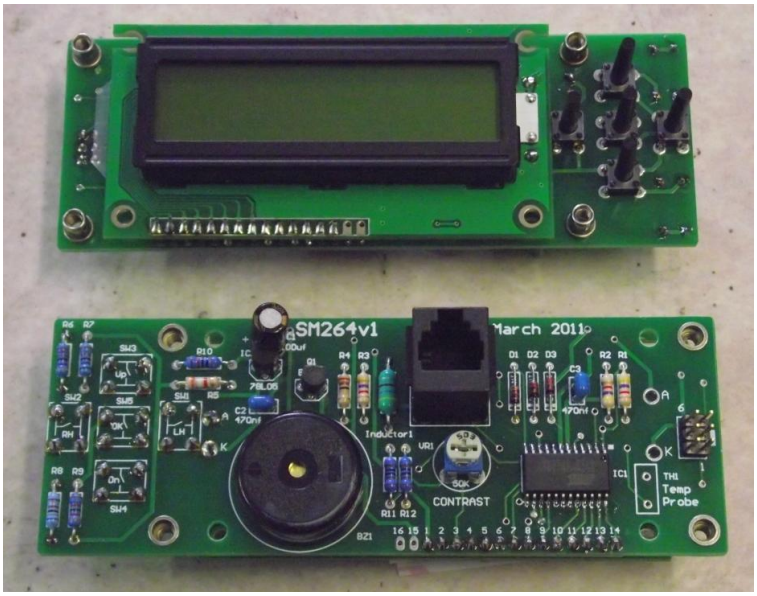
Char. Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1 Address	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Line 2 Address	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Line 3 Address	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Line 4 Address	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95

These three screen types are just a few of the more common sizes available. They may be purchased with and without LED backlight and with a High or Low viewing angle.

An external trimpot must be applied to all displays to set the screen contrast, otherwise it will just appear as a 'blank' screen.

USING THE SM264 DISPLAY ADAPTER MODULE

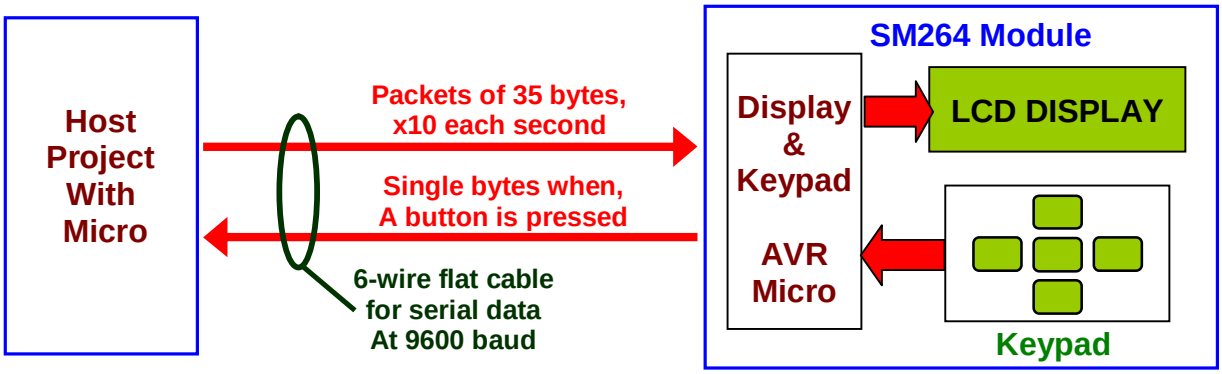
When working with these LCD screens it quickly becomes evident that using a display on a small project may use up about half of the available ports on a small microprocessor and perhaps half of the available program space to communicate with them. This can create hardware resource problems.



Also, A display usually needs to go on the front of an enclosure or cabinet where it may not be convenient to have the entire project mounted, particularly if there are AC mains and other heavier wiring that may need to wire to a panel.

Lastly such displays often need some push button controls adjacent to them so that an operator may be able to change the contents and format of what is being displayed.

For this reason it became expedient to create a dedicated adaptor module to integrate these needs. This module has its own micro that can: Communicate with a common 2 x 16 character display, Monitor push button switches and Drive a Beeper for confirmation of the button presses. This module presents all the signals needed to the LCD module, but it only requires *one* microprocessor port from any host project as it receives its text as a serial data string at 9600 baud. An outgoing data path is used to signal the host when any button or combination of buttons have been pressed. Along with 12V power rails, the module can be placed up to about 5 metres from a main board via a flat RJ12 telephone type patch cable.



The **SM264** module (shown above with front and rear views) can be seen interfacing with a standard Backlit **2 x 16 character** display module using an Atmel AVR microprocessor.
It includes an adjustable trimpot for setting the contrast of the LCD screen. It also has provision for a thermistor type temperature probe wired to a spare conductor within the 6 wire cable, so that a host system can measure the room temperature at the location of the screen.

3mm thread have been inset into the module to permit the mounting of the module to an enclosure or cabinet. The module also has its own 5V voltage regulator.

The 5-button keypad is connected to a single analog input of the AVR micro. Each button has a different resistor value wired to it, so that when any button (or combination of buttons) are pressed a different voltage is presented to the analog input. Pressing a button causes the micro to send a single letter of the alphabet in ASCII at 9600 baud to the host system so that it may read and respond to the button presses

Outgoing characters generated by button presses

Button Function	ASCII text character	Decimal value
No key press	0	48
Left button	1	49
Right button	2	50
Up button	3	51
Down Button	4	52
Central 'OK' (pressed once)	5	53
Left plus Right together	6	54
Up plus Down together	7	55
'OK' after 4 second press	A	65
'OK' after 8 second press	B	66

Common ASCII characters were used to make it possible to monitor a keypad with a PC running a dumb terminal program or Visual Basic application.

Receiving text from the host system

The **SM264v2** program expects to receive a string of **35 characters** at **9600** baud at any intervals anywhere between ten times per second and several minutes.

The first two characters are always Decimal **13** and Decimal **10**. This corresponds with the older style '**Carriage Return**' and '**Line Feed**' characters used with printers and terminals. In this context they are used a synchronisation string. Whenever these two characters are received by the display program, they know that the next 32 characters must be directed to the screen.

The next 16 characters received represent the characters that will appear on the **top line of the display**. The following 16 characters will appear on **line 2**. These 32 characters are saved in the memory of the display module as 32 variables which are automatically updated to the screen ten times per second.

The last byte (Character 35) is a Control Byte.

Normally this would be character 64, the '@' character as a dummy character when no control functions are being performed.

- If Control code is ASCII '**1**' to '**5**' it will cause a rapid beep of a corresponding 1 to 5 times.
- If Control code is ASCII '**A**' it will override the rest of the display string and clear the screen.

Often ASCII codes '**B**' through to '**Z**' are used to trigger custom strings of text to appear on the display. This is useful for cueing fixed messages where the host microprocessor is running out of memory space to hold all of the messages needed.

The Schematic diagram

The schematic circuit of this module has been included with this document to both to help with an understanding of the sample software which supports it and as an example of what is required electrically to connect a display module to a microprocessor.

Note that the data stream between this module and host micro is simple 0-5V logic and not the +/-9V of the RS232 standard. This limits the cabling range to the display, but simplifies the electrical connections between the two Display and Host microprocessors.

(This layout is very similar to an earlier version of the display module designated as SM217, which used the same software, but had no provision for a temperature sensor.)

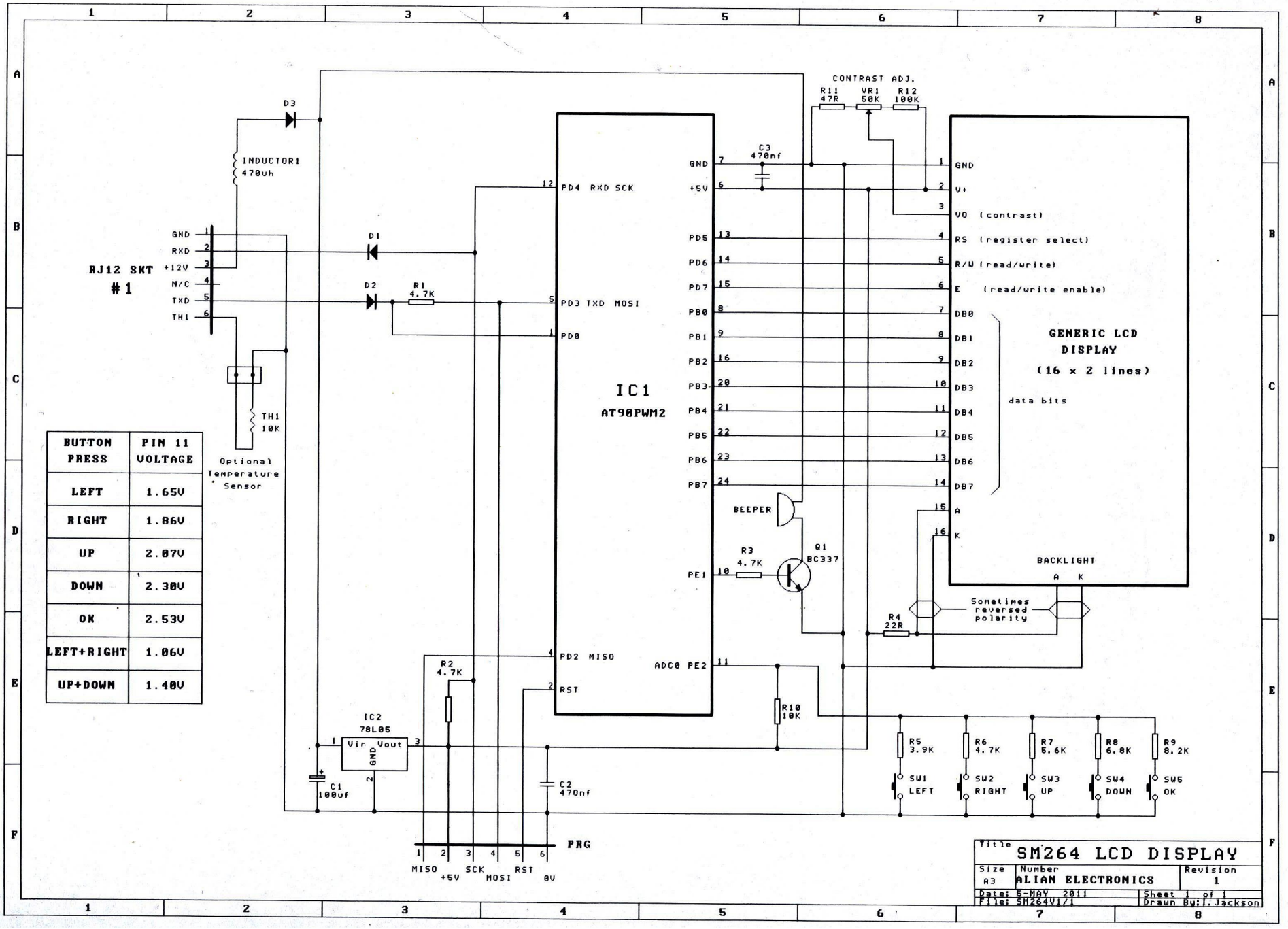
The schematic diagram also shows a table of D.C. voltages that are presented to port the Analog Input port **PE2** when buttons are pressed.

The Piezo beeper operates via the +12V rail and an NPN transistor, as this increases the volume of the beeper compared to a beeper only operating on a +5V rail

SM264 RJ12 connections

The 6 wire cable between the Display Module and the host project uses a standard 6 wire RJ12 plug similar to those used by telephone cables. Note that both ends of the cable should be crimped identically so that there is no 180º twist in the cable.

Wire 1	**Negative ground.**
Wire 2	**Receive Data (from the host)**
Wire 3	**+12V supply,**
Wire 4	**Unused (so no damaged is incurred if cable is reversed in error)**
Wire 5	**Transmit Data (to the host)**
Wire 6	**is reserved for an NTC resistive temperature sensor.**



BUTTON PRESS	PIN 11 VOLTAGE
LEFT	1.65V
RIGHT	1.86V
UP	2.07V
DOWN	2.38V
OK	2.53V
LEFT+RIGHT	1.06V
UP+DOWN	1.48V

Title: SM264 LCD DISPLAY		
Size: A3	Number: ALIAN ELECTRONICS	Revision: 1
Date: 5-MAY 2011		Sheet 1 of 1
File: SM264V1/1		Drawn By: I. Jackson