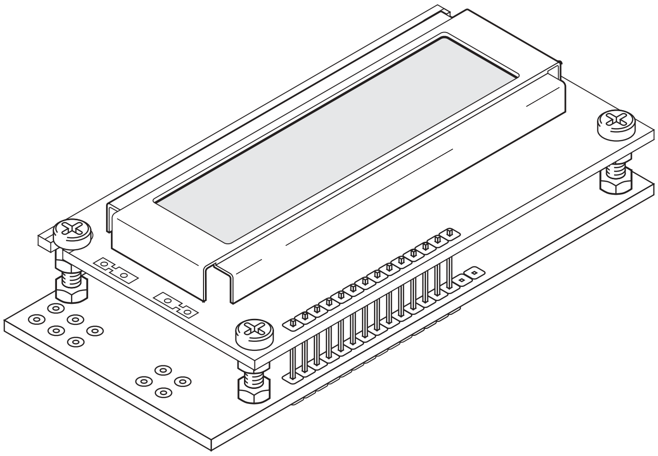
SERIAL/I2C LCD AND CLOCK (V2)



The serial LCD and clock module allows microcontroller systems (e.g. PICAXE) to visually output user instructions or readings, without the need for a computer. This is especially useful when working, for example, with analogue sensors, as the analogue reading can easily be displayed on the LCD module. All LCD commands are transmitted serially via a single microcontroller pin using the **serout** command. e.g.

to print the text 'Hello' the command is simply:

```
serout 7,N2400,("Hello")
```

The module can also store 7 programmable pre-defined messages to save memory space usage within the PICAXE system.

The optional low-cost clock upgrade provides a real-time clock and programmable alarm output. The LCD can show the current date and time on it's display, and the alarm output can be programmed to trigger at any period between 10 seconds and 1 year. The clock has a lithium coin cell backup that maintains the time for up to ten years when the main power supply is removed.

Key Features:

1. 16x2 LCD Alphanumeric Display
2. Simple serial (1 wire) connection to microcontroller (2400,N,8,1).
3. Optional i2c interface to PICAXE-X parts.
4. 7 Programmable pre-defined messages
5. Small footprint (almost same size as the LCD).
6. Optional low-cost clock upgrade, providing
   - Real Time Clock
   - Programmable Alarm Output
   - 1Hz pulse output
   - 10 year battery backup

Which Mode? (serial or i2c)

Most users will use the module in the default serial mode.
The only reason to use it in i2c mode is if:

1) You are using a PICAXE-X, X1 or X2 chip **and**
2) You wish to read the time/data from the DS1307 clock upgrade directly into the PICAXE chip.

In all othe cases the serial mode should be used.

In i2c mode the LCD module acts as a 'dumb' i2c slave device. The clock and alarm functions are not available - all clock and alarm functions must be carried out by the PICAXE X part itself.

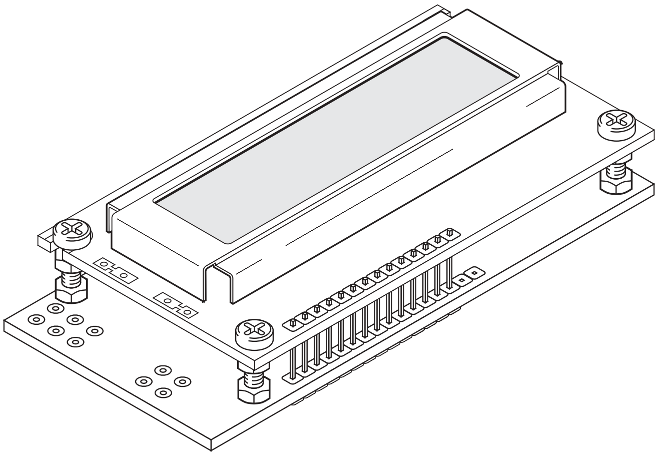
See Section 1 for construction and assembly details. A jumper link (J1) is required for i2c mode.

See Section 2 on pages 6-7 for i2c connection details and samples. Further information about i2c protocol and interfacing can be found in the 'i2c Tutorial' help file. It is assumed that the user has already read this help file.

See Section 3 on pages 8-15 for serial connection details and samples.

Note: Version 1 LCD modules

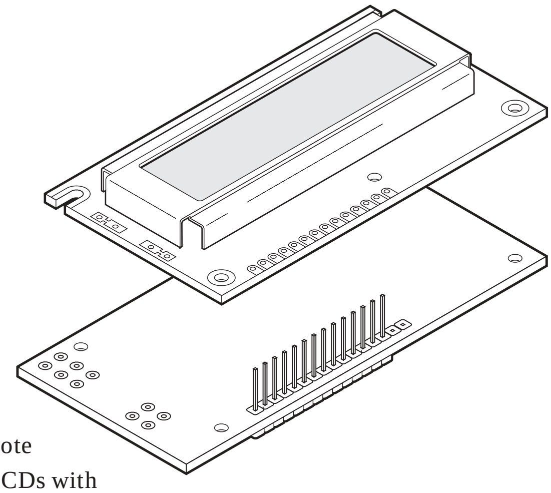
This datasheet is for version 2 (black colour pcb) modules. Version 1 (green colour) modules did not have the optional i2c mode, but this datasheet can still be used as the serial mode information in section 3 also applies to version 1 modules.



Section 1 - Construction and Kit Contents

- pre-populated PCB
- 16x2 alphanumeric display (brand may vary)
- bag of 4 support posts and 3x 10 pin headers

The LCD is supplied loose so that it can be either fitted directly to the board, or connected via a longer ribbon cable connection if desired. The following instructions explain how to fit the LCD directly to the board (track side) and presume the user is confident at soldering.



Note that the LCD is fitted above the TRACK side of the PCB. Ensure no solder bridges between pins on the header.

Connecting the LCD

1. Snap one header into a 4 and 6 way section. Place the short end of the 4 way section, with another 10 way header, through the holes labelled 1-14 on the pcb (from the track side of the PCB, so that the top of the short end shows on the component side). Note the extra holes marked A and K and resistor RB are only used on LCDs with LED backlights (the LCD in the kit does not have a backlight and so these connections are not required).

2. Carefully solder each header pin (on the component side of the pcb). Check each joint carefully for shorts between pins.

3. Solder a wire link in position J2 (power) if a 4.5V battery pack is to be used. This is not required for a 5V or 6V supply.

4. Solder a wire link in position J1 (mode) if the LCD is to be used in i2c mode. No link is required for the default serial mode.

5. Click the four support posts onto the PCB. They are designed to be a tight fit!

6. Carefully click the LCD over the header pins and onto the support posts;.

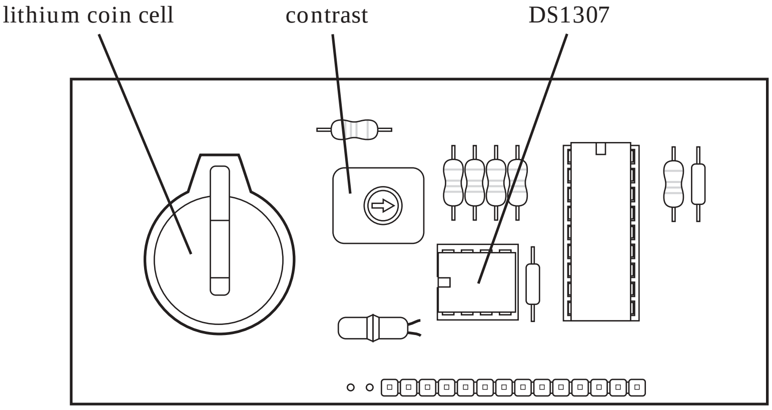
7. Solder the LCD to the pin headers.

8. Snap the 6 pin header into a 2 and 4 way section. Solder the two way section to the CLK contacts on the board.

9. Connect a power supply to the main connection header (red wire to V+, black wire to 0V). The LCD should display a time message **when the two CLK contacts are shorted** (e.g. with the jumper provided in the kit) and once **the contrast is adjusted** (via the variable resistor marked ‘contrast’). If the LCD does not display a message check the power, contrast and the 14 connector pins carefully. (Note that if the optional clock upgrade chip is not fitted, the time will always show as 00/00/00 00:00)

See page 8 for a sample PICAXE serial test program.

Installing the Optional Clock Update



Required:
- CR2032 lithium coin cell
- DS1307 Clock IC

Instructions:
1. Place the DS1307 into the 8 pin socket, ensuring pin 1 is facing the lithium cell holder.

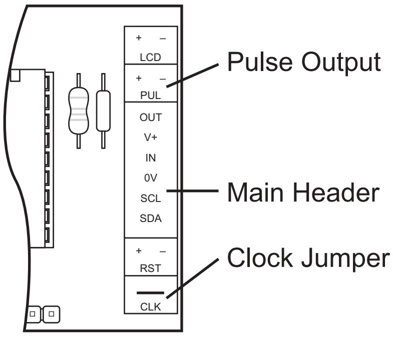
2. Place the CR2032 lithium coin cell in the holder, ensuring the positive (+) side is facing up.

Notes:
Note that the lithium coin cell keeps the DS1307 clock operating when the main power supply is not connected. This ensures accurate time is kept by the module. The coin cell **does not** power the LCD or the pulse output. The coin cell will last approximately 10 years.

Note that the clock and alarms (and pulse output) will not operate correctly until the initial time is programmed into the module (see the 'Programming Time into the Module' section below).

Users in Europe/USA
Please note that the date convention used in the module is the UK date format dd/mm/yy. The US date format mm/dd/yy is available by special order.

Input / Output / Power Connections



Main Header (V+,0V)
The main header provides connection for the power supply (5-6V DC on V+). If you wish to use 4.5V solder a wire link in position J2 (power). This shorts out the voltage protection diode D1, as this diode causes a 0.7V voltage drop, which can make the screen very dim at this lower 4.5V voltage.

Main header (IN)
These is the serial input (IN).

Main header (SDA and SCL)
These are the i2c mode connections. They must only be used when a wire link has been soldered in position J1 to put the module into i2c mode (see section 2).

Main header (OUT)
The alarm output triggers (goes high for 5 seconds) whenever a clock alarm occurs (in serial mode). The alarm output can sink or source 20mA.

Pulse Output (PLS)
The pulse output outputs a square wave of 1Hz (1 pulse per second) when the optional DS1307 clock IC is fitted. A 330R resistor is included on the board to allow a low current LED to be soldered directly to this connection to provide a flashing ‘second’ indicator. The pulse output can sink or source 20mA. The pulse output will not operate until the clock upgrade is fitted and the correct time is programmed into the unit.

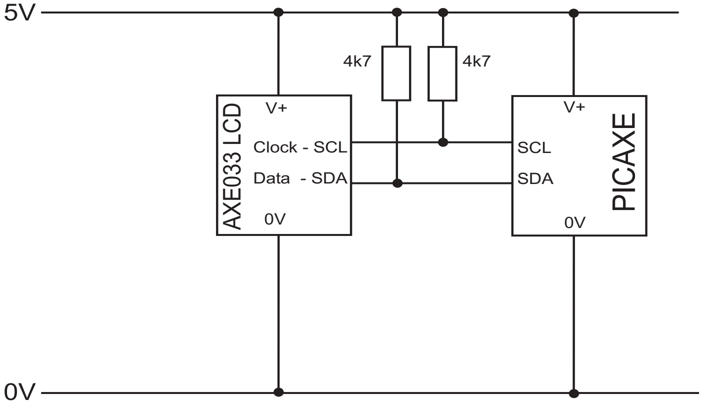
Clock Jumper (CLK)
When the clock jumper is fitted the module goes into clock mode. During this mode instructions cannot be sent via the serial connection, as the unit is acting as a standalone ‘alarm clock’. User defined message 1 is constantly shown on the top line of the LCD and the time is constantly shown on the bottom line of the LCD (when the module is powered). The pulse output and alarm output operate as normal.

LCD Backlight (LCD) (the LCD provided in the kit does not have a backlight)
When a backlight is fitted the LCD connections allows power to be applied to the backlight. Note that you must also have soldered the ‘A’ and ‘K’ connections on the LCD and added a resistor to position RB on the pcb to use this feature. See your LCD datasheet for suitable power and resistor values.

Resonator Tuning (RST)
The resonator tuning pin allows the internal resonator to be calibrated for use with various PICAXE chips when in serial mode. See appendix a, resonator tuning, for more details.

Section 2) Connecting the Module to a PICAXE Microcontroller (i2c mode)

The following diagram shows how to connect the LCD module to the PICAXE X i2c system.



Note that the 4k7 resistors pull up resistors are pre-soldered onto the AXE033 LCD module. Therefore additional external resistors are not required.

i2c programming details

The i2c communication protocol used with the LCD module is the same as popular eeprom's such as the 24C04. The SPE030 family code is $C6, operates at slow speed (i2cslow) and has a single byte (i2cbyte) address size. Therefeore the PICAXE i2c setup command (required before readi2c or writei2c is used) is
i2cslave $C6,i2cslow,i2cbyte

PICAXE (i2c) Test program

```
init: pause 500                    ‘ wait for display to initialise
      i2cslave $C6,i2cslow,i2cbyte ‘ set up i2cslave for LCD
main: writei2c 0,(254,128,255)     ‘ move to start of first line
      pause 10                     ‘ wait for LCD to process data
      writei2c 0,(“Hello!123”,255) ‘ output text
      end
```

The display is write only in i2c mode. Do not use the readi2c command at slave address $C6, as may cause unreliable behaviour which will require the module to be reset. Note that a 10ms delay (pause 10) should be placed between consequative writei2c commands to allow time for the data to be processed.

The LCD can display characters and can also accept certain control commands (e.g. clear display or move cursor to new position). Note that the LCD module takes approx half a second to initialise and so any data sent during this period will be lost. It is advisable to put a ‘pause 500’ command at the start of any program to ensure no data is lost when the system is powered up.

Characters are normal symbols that can be displayed on the LCD screen. See Appendix 1 for a table of the common ASCII characters.
All LCD data is written to the write buffer at address 0. This buffer stores the data, and then prints the data on the LCD screen at the current cursor position when the special byte ‘255’ is received. The buffer has a maximum size of 20 characters. Each write must terminate with the number 255, as this tells the module to start writing

the buffered characters to the LCD display itself. Allow 10ms for this processing.

Control Commands (254)

All LCD commands (move cursor etc) are preceeded by the number 254.
The most common control commands are

writei2c 0,(254,1,255)	Clear Display (must be followed by 'pause 30')
writei2c 0,(254,8,255)	Hide Display
writei2c 0,(254,12,255)	Restore Display
writei2c 0,(254,14,255)	Turn on Cursor
writei2c 0,(254,16,255)	Move Cursor Left
writei2c 0,(254,20,255)	Move Cursor Right
writei2c 0,(254,128,255)	Move to line 1, position 1
writei2c 0,(254,y,255)	Move to line 1, position x (where y = 128 + x)
writei2c 0,(254,192,255)	Move to line 2, position 1
writei2c 0,(254,y,255)	Move to line 2, position x (where y = 192 + x)

Using the Optional Clock Upgrade in i2c mode

When the clock upgrade is used the PICAXE must read the data directly from the DS1307 chip and then issue LCD write commands to display the data on the screen. The LCD module has no internal 'intelligent' clock routines when in i2c mode, as only the PICAXE can access the data (the LCD module is a slave, not master, device). Remember that when reading/sending the data to both LCD and DS1307 it is necessary to **keep re-issuing** the appropriate i2cslave command for each part.

Setting the Time / Date

To set the correct time after the circuit is first powered up, the current time must be written to the DS1307 registers. The following example PICAXE program will setup the time to 11:59:00 on Thursday 25/12/03.This is carried out by loading the registers in order from address 00 upwards i.e. seconds then minutes then hours etc.

```
i2cslave %11010000, i2cslow, i2cbyte
writei2c 0, ($00, $59, $11, $03, $25, $12, $03, $10)
end
```

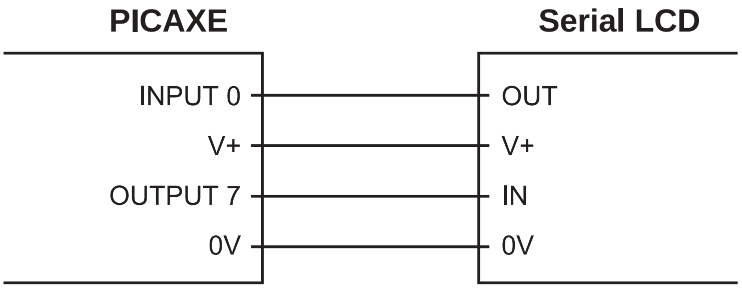
Reading the Time / Date

To read the current time you can use the following program to load variables within the PICAXE with the various register values from the DS1307. This example program acts as an alarm clock, checking the time every 30 seconds. If the time is exactly 07:00 then a buzzer, connected to output 7, will sound for 20 seconds.

```
i2cslave %11010000, i2cslow, i2cbyte     ‘ set slave details
loop:
      pause 30000                  ‘ wait 30 sec
      readi2c 0, (b0, b1, b2)      ‘ read sec, min, hour
      if b2  <> $07 then loop      ‘ if hour not 7 loop
      if b1 <> $00 then loop       ‘ if min not 00 loop
      high 7                       ‘ switch on buzzer
      pause 20000                  ‘ wait 20 sec
      low 7                        ‘ switch off buzzer
      pause 60000                  ‘ wait 60 sec
      goto loop                    ‘ loop
```

Section 3 - Connecting the LCD to a PICAXE Microcontroller (serial mode)

The following diagram shows how to connect the Serial LCD module to the PICAXE system. Output 7 is used to send signals to the LCD, and input 0 is used for the alarm signal from the clock alarm.



Note: The connections must be made **directly** to the PICAXE output pins (not via the darlington driver buffered outputs found on the PICAXE starter pack project boards)

PICAXE Test program

```
init: pause 500                     ‘ wait for display to initialise
main: serout 7,N2400,(254,128)      ‘ move to start of first line
      serout 7,N2400,(“Hello!123”)  ‘ output text
      end
```

Note the use of N2400 (not T2400) within PICAXE programs. Note the Serial LCD does not buffer bytes received, and so a small delay between bytes (to update the display) is required on non-PICAXE systems. This delay is applied automatically by the PICAXE system.

If the characters do not all appear as expected (e.g. as non standard symbols), see the ‘Resonator Tuning’ section of the datasheet.

Displaying Messages

The LCD can display characters, messages and the time, and can also accept certain control commands (e.g. clear display or move cursor to new position). Note that the serial LCD module takes approx half a second to initialise and so any data sent during this period will be lost. It is advisable to put a ‘pause 500’ command at the start of any program to ensure no data is lost when the system is powered up.

Characters

Characters are normal symbols that can be displayed on the LCD screen. See Appendix 1 for a table of the common ASCII characters. Note that 0-7 are special characters that actually print out the time and predefined messages. The numbers 253 and 254 are used to indicate a write memory or control command sequence follows.

0	Time
1-7	Predefined Messages
8-128	ASCII Characters (see Appendix 1)
129-252	Miscellaneous Characters (may vary dependant on LCD type)
253	Special Command – Write Memory
254	Special Command – Command Character
255	Reserved for future use

Characters can be output via two methods – either by using the ASCII number or the symbol enclosed in speech marks e.g. (65) and (“A”) both output the same symbol.

Control Commands (254)

Control commands are all prefixed by the number 254. They are used to send commands to the Serial LCD Module (e.g. move to line 2, switch cursor off etc.).

The most common control commands are

254,1	Clear Display (must be followed by a ‘pause 30’ command)
254,8	Hide Display
254,12	Restore Display
254,14	Turn on Cursor
254,16	Move Cursor Left
254,20	Move Cursor Right
254,128	Move to line 1, position 1
254, y	Move to line 1, position x (where y = 128 + x)
254,192	Move to line 2, position 1
254, y	Move to line 2, position x (where y = 192 + x)

Write Commands (253)

Write commands are all prefixed by the number 253. They are used to program the predefined messages, current time or alarm times into the Serial LCD module.

0	Set clock time
1-7	Set predefined messages 1-7
8	Set Alarm (date/time)
9	Set Alarm (interval)
10	Turn Alarm Off

All write commands must be followed by a 1000ms delay (pause 1000 command) to allow the internal save to be carried out. When a write command is used a brief ‘DATA SET’ mesaage will appear on the top line of the LCD to indicate the data has been saved. See the sections below for more details.

Programming a Predefined Message into the Module

The module can contain 7 user predefined messages, each message 16 characters long. These messages are stored on the LCD module and can be used to greatly reduce the ‘display text’ that must be stored within the PICAXE or Stamp (hence reducing the length of the program).
Messages 1,3,5,7 automatically appear on the top line of the display.
Messages 2,4,6 automatically appear on the second line of the display.

The messages must be programmed into the module using a small program running in a microcontroller such as the PICAXE. The following instructions presume the connection as shown in the example PICAXE circuit above.

To set message 1 to “Player 1 =” and message 2 to “Player 2 =“ program the PICAXE with the following program. This loads the message write instruction (253), followed by the message memory address (1 or 2) followed by the message.

```
init:       pause 500
main:       serout 7,N2400,(253,1,”Player 1=       ”)
            pause 1000
            serout 7,N2400,(253,2,”Player 2=       ”)
            pause 1000
            end
```

Note the messages must always be 16 characters long, so additional spaces **must be** added to the text to ensure the message is **exactly 16 characters long**. Note that a 1000 millisecond programming period must be added after every write instruction.

Displaying a Predefined Message

The predefined displayed messages are displayed in the same way as normal characters, using the character code 0 (time) or 1 to 7 (messages). Note that a 10ms delay (pause 10 command) must be added after each command to give the LCD enough time to display all the 16 characters in the message.

Therefore the following program will display message 1 on the top line of the display, and the time on the bottom line of the display.

```
init:       pause 500
main:       serout 7,N2400, (1)
            pause 10
            serout 7,N2400, (0)
            pause 500
            goto main
```

Combining Predefined Messages and Variables

It is often useful to combine predefined messages with variables e.g. displaying the score of a game. The following program shows how to show the two scores from two players, presuming message 1 and message 2 have been pre-programmed with the phrases “Player 1=” and “Player 2=” (see above).

```
init:       pause 500
main:       serout 7,N2400,(1)
            pause 10
            serout 7,N2400,(254,137,#b1,”  “)
            serout 7,N2400,(2)
            pause 10
            serout 7,N2400,(254,201,#b2,”  “)
            let b1 = b1 + 1
            let b2 = b2 + 2
            pause 500
            goto main
```

Note that the message code (1 or 2) is first output. A delay of 10ms is then added to allow the LCD module to display the message. The cursor is the moved 9 positions along the screen (to the position after the = sign by the 254,137 or 254,210 command) and then the variable value is displayed. Note that the # symbol tells the microcontroller to output the ASCII equivalent of the variable value, not the direct value (e.g. “6” “5” not the value 65, which would actually appear as the character “A”!) Two additional spaces are then also added to ensure variable value changes are overwritten correctly (e.g. to overwrite ‘234’ by ‘1’ you must output ‘1(space)(space)‘ to ensure the ‘34’ of the first number is overwritten by the spaces.)

Programming the Time into the Module

The current time must be programmed into the module using a small program running in a microcontroller such as the PICAXE. The following instructions presume the connection as shown in the example PICAXE circuit above. Note that once set, the lithium coin cell will maintain the clock time accurately for up to ten years.

To set the clock to 22:00 on 25/12/01 program the PICAXE with the following program. This program loads the write instruction (253), followed by the clock memory address (0), followed by the date and time (“25/12/01 22:00 ”)

```
init:       pause 500
main:       serout 7,N2400, (253,0,”25/12/01 22:00  ”)
            pause 1000
            serout 7,N2400, (0)
            end
```

Note the time and date must be presented exactly as shown, using the 24 hour clock format dd/mm/yy hh:mm. Note the write messages must always be 16 characters long, so 2 spaces are added to the text to ensure the message is exactly 16 characters long. Note that a 1000 millisecond programming period must be added after every write instruction. The last serout command shows the time to check it is correctly programmed.

To accurately enter a time, download the program (set with a time about 1 minute ahead of schedule) into the PICAXE. Then press the reset switch on the PICAXE (to re-run the program) at exactly the correct time. This will set the time accurately.

Displaying the Time

The time message is updated with the current date/time every time it is used. The time message is displayed in the same way as normal preset messages, using the special character code 0. The time always automatically appears on the second line of the display.

Therefore the following program will display message 1 on the top line of the display and the time on the bottom line of the display. The screen will update the time every 0.5 second.

```
init:       pause 500
main:       serout 7,N2400, (1)
            pause 10
            serout 7,N2400, (0)
            pause 490
            goto main
```

Programming the Alarm Time into the Module

The alarm output pin triggers (goes ‘high’ for 5 seconds) whenever the alarm time is reached. The alarm can be set to a specific date/time (write code 8), or can be set to repeat at certain time intervals (write code 9). Only one alarm type is active at any time – the last written alarm type is the one used within the module.

The alarm time or interval must be programmed into the module using a small program running in a microcontroller such as the PICAXE. The following instructions presume the connection as shown in the example PICAXE circuit above.

Setting an alarm at a specific time:
To set the alarm time clock to 07:30 every day (using write code 8), program the PICAXE with the following program. This program loads the write instruction (253), followed by the alarm address (8), followed by the time (“00/00/00 07:30”)

```
init:       pause 500
main:       serout 7,N2400, (253,8,”00/00/00 07:30  ”)
            pause 1000
            end
```

Note the time and date must be presented exactly as shown, using the 24 hour clock The ‘00’ characters can be used (within the date only) to indicate an ‘ignore’ condition, so in the example above the date is completely ignored, so the alarm will trigger every day at 07:30. Note the write messages must always be 16 characters long, so 2 spaces are added to the text to ensure the message is exactly 16 characters long. Note that a 1000 millisecond programming period must be added after every write instruction.

To set the alarm to trigger on the first of every month at midnight

```
init:       pause 500
main:       serout 7,N2400, (253,8,”01/00/00 00:00  ”)
            pause 500
            end
```

Setting an alarm at a specific time interval:
To set the alarm to trigger at an interval, instead of an exact time, use write code 9 instead of 8. For example, to trigger the alarm every ten minutes (using write code 9)

```
init:       pause 500
main:       serout 7,N2400, (253,9,”00:10:00        ”)
            pause 1000
            end
```

To set the alarm to trigger every 30 seconds

```
init:       pause 500
main:       serout 7,N2400, (253,9,”00:00:30        ”)
            pause 1000
            end
```

To set the alarm to trigger every two hours

```
init:       pause 500
main:       serout 7,N2400, (253,9,”02:00:00        ”)
            pause 1000
            end
```

Note the alarm trigger interval is denoted by a number of hours (00 to 23), minutes (00 to 59) and seconds (00 to 59) between alarms. The smallest practical alarm interval is 10 seconds, due to the five second ‘on time’ of the alarm output. Note the write messages must always be 16 characters long, so 8 spaces are added to the text to ensure the message is exactly 16 characters long. Trigger values longer than one day should be set using the time and date method shown above.

IMPORTANT NOTE
The interval timer operates as follows on power-up:
Upon power up the module reads the current time – and then adds the alarm interval to the current time to generate the next alarm time. When an alarm occurs the interval is once again added to the current time to create the next alarm time.

Therefore the interval timer is effectively reset every time the module is powered down – the first alarm will be activated the ‘interval time’ after power up. To keep the interval exactly consistent over a long period the module must be continuously powered.

Turning the alarm off:
To disable either type of alarm send the ‘10’ command (note that the 10 command does not require 16 characters to be sent as with all the other commands – it is just sent by itself)

```
init:       pause 500
main:       serout 7,N2400, (253,10)
            pause 500
            end
```

APPENDIX A

Internal Resonator Tuning

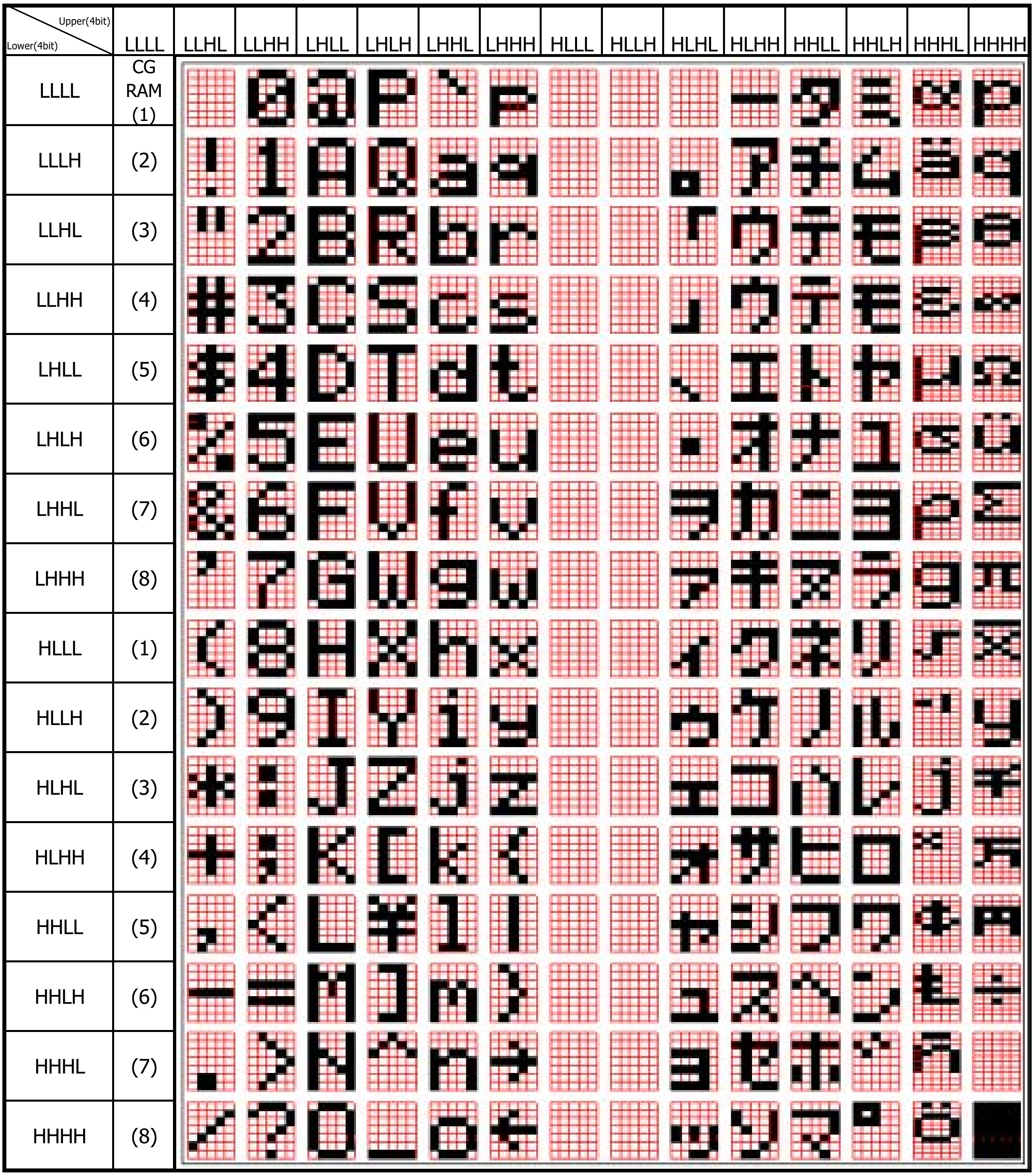
The microcontroller used as the controller on the serial LCD operates from an internal resonator. Many of the PICAXE chips also operate using the internal resonator.

Use of the internal resonator reduces cost of the product and simplifies PCB design. In most cases this causes no problems and the LCD will function correctly without any calibration.

However the internal resonator is not as accurate as external crystal devices, and it has been found on a very small number of PIC16F628 that the calibration of the internal resonator can drift slightly. If the PICAXE internal resonator frequency is also at one of the calibration extremes, you may at first experience some 'corrupt' characters being displayed on screen. Typically numeric charcters will work correctly, but text may appear as non-standard symbols.

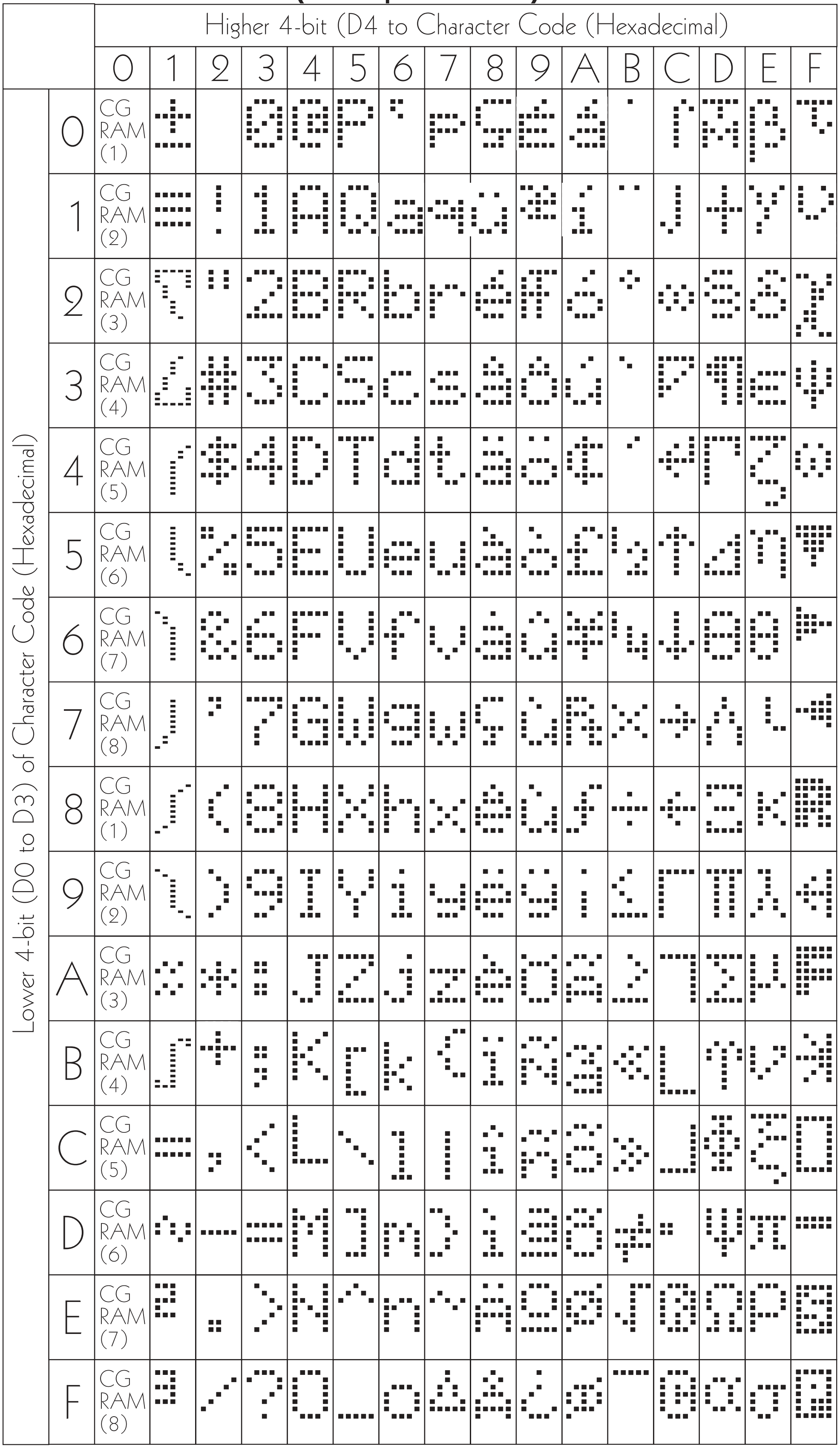
If you experience this issue, simply solder a wire link across the 'RST' pads. This will adjust the resonator frequency to allow correct operation.

APPENDIX B



Upper(4bit) Lower(4bit)	LLLL	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)		0	@	P	`	p				ー	タ	ミ	α	p
LLLH	(2)	!	1	A	Q	a	q			。	ア	チ	ム	ä	q
LLHL	(3)	"	2	B	R	b	r			「	ウ	テ	モ	β	θ
LLHH	(4)	#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
LHLL	(5)	$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
LHLH	(6)	%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
LHHL	(7)	&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
LHHH	(8)	'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
HLLL	(1)	(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
HLLH	(2)	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
HLHL	(3)	*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
HLHH	(4)	+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
HHLL	(5)	,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
HHLH	(6)	-	=	M	]	m	}			ュ	ス	ヘ	ン	Ł	÷
HHHL	(7)	.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
HHHH	(8)	/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█

Standard Character Pattern (Elec & Eltek LCD Module)
Standard Character Pattern (Powertip LCD Module)


Higher 4-bit (D4 to Character Code (Hexadecimal)
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
Lower 4-bit (D0 to D3) of Character Code (Hexadecimal)
CG RAM (1)
CG RAM (2)
CG RAM (3)
CG RAM (4)
CG RAM (5)
CG RAM (6)
CG RAM (7)
CG RAM (8)
CG RAM (1)
CG RAM (2)
CG RAM (3)
CG RAM (4)
CG RAM (5)
CG RAM (6)
CG RAM (7)
CG RAM (8)