

# **PIC-LCD**

---

PIC based LCD Display Interface  
for the Circuit Cellar<sup>®</sup> Home  
Control System (HCS)

**March 1999**

**v1.04**

**©1998-1999 Creative Control Concepts**

---

---

## PIC-LCD Overview



The PIC-LCD provides users with a small (18-pin) and cost-effective LCD Display interface for their HCS II Home Automation System. Up to 8 can be connected to an HCS II RS-485 network. It supports a number of different size displays and provides control capabilities for the LCD backlight or a beeper. It also has EEPROM memory capability so that settings are permanently saved and users can save and use up to 8 custom characters on their LCD's. Note that the PIC-LCD does **NOT** support a keypad. It should be used where you need a display, but do not require a keypad. The PIC-LCD+ which is due to be released later this year will have support for a keypad, screen switching, and 4x40 LCD Displays.

Numerous ANSI and non-ANSI control commands are supported for functions such as cursor control, backlight control, and normal text controls like line-feed, tabs, and carriage return.

The PIC-LCD can also be used with any system that has serial data capabilities. Need a remote LCD Display for your next project? Just connect your controller's serial port to a PIC-LCD using RS-232 or RS-485 transceivers. The PIC-LCD allows you to connect a LCD to your system with just 2 wires. You can even add checksums to validate the command packet data.

The PIC-LCD packs a lot of functionality in an 18-pin package. It allows the user to completely control an LCD display and take advantage of all of its features using simple ASCII commands.

---

## PIC-LCD Protocol Overview

---

The PIC-LCD recognizes all Circuit Cellar<sup>®</sup> LCD-Link LCD Display control commands used in a Circuit Cellar<sup>®</sup> HCS II Home Automation system plus some additional control commands specific to the PIC-LCD. PIC-LCD's are configured in XPRESS<sup>®</sup> as LCD-Links. When used on a multi-drop RS-485 network, up to eight PIC-LCD's may be connected on the same network.

Commands are sent to the PIC-LCD via a standard serial port. When connected to a Circuit Cellar<sup>®</sup> HCS II, they send and receive commands over the HCS II RS-485 multi-drop serial network. To connect a PIC-LCD to a PC serial port, use a voltage shifter such as a MAX232 from Maxim. When connecting PIC-LCD's to an RS-485 network, use a converter such as a 75176.

PIC-LCD's communicate with their host using the following serial settings:

```
9600 Baud
8 Bits
No Parity
1 Stop Bit
```

### Command/Packet Format

When commands are sent to a PIC-LCD, they must be in one of the following formats:

```
! TERM# command\n
#XX TERM# command\n
```

Packets starting with ! do not use checksums. If you are using checksums (see below) all packets sent to the PIC-LCD must start with #. The computed checksum goes in place of the XX. Replace the # after TERM with the address number of the node which must be between 0 and 7. All packets must end with a carriage return (ASCII 13) and/or line-feed (ASCII 10).

Replies from the PIC-LCD will always start with \$. If the original command packet used a checksum, the response will also have a checksum, otherwise it will not:

```
$ TERM# reply\n
$XX TERM# reply\n
```

## Checksums

To ensure reliable communications between the host computer and the PIC-LCD, checksums can be used in packets to help ensure the validity of data. If the PIC-LCD receives a packet with an invalid checksum, the packet is simply ignored and no response is sent. Your controller must recognize the lack of a response and resend the packet after a given timeout period. When connected to an HCS II controller, a received packet with a bad checksum is indicated by an E in the node status window.

Checksums for command packets are generated as follows:

1. Construct the command packet with zeros in the checksum location. Do **NOT** include the trailing carriage return:

```
#00 TERM3 Q
```

2. Add up all ASCII values of all characters in the packet, except for the `\n`:

#	23	(these are HEX values)
0	30	
0	30	
space	20	
T	54	
E	45	
R	52	
M	4D	
3	33	
space	20	
Q	51	
<b>Total</b>	<b>7F</b>	(only the low byte is kept)

3. Take the 2's complement of the checksum byte computed above:

```
-(7F) = 81
```

4. Convert the checksum byte into 2 ASCII chars and replace the zeros in the checksum location:

```
#81 TERM3 Q
```

To validate the checksum of a received packet from a PIC-LCD, do the following:

1. Discard the trailing carriage return (and linefeed if present):

```
$73 TERM1 00
```

2. Convert the ASCII checksum into a single hex byte.

3. Replace the checksum characters with zeros in the packet:

```
$00 TERM1 00
```

5. Add all ASCII values of the characters in the packet, minus any trailing \n or \r:

\$	24	(these are HEX values)
0	30	
0	30	
space	20	
T	54	
E	45	
R	52	
M	4D	
1	31	
space	20	
0	30	
0	30	
<b>Total</b>	<b>8D</b>	(only the low byte is kept)

4. Add the checksum byte to the packet total. If the result is 00, the packet is probably valid:

```
73 + 8D = 00 (The packet is okay)
```

That's all there is to it! Here is a great hint from the original Circuit Cellar® DIO-Link User Manual:

“After you think about it for a while you’ll write a single subroutine to calculate or verify the checksum. If you hand the routine a line with a 00 checksum, it’ll replace the zeros with the new checksum so you can send the line. If you hand it a (presumably) valid received line it will do the same, but also return the calculated checksum to the caller as an integer.”

---

## PIC-LCD Command Overview

---

The PIC-LCD recognizes all the LCD-Link network commands used in a Circuit Cellar® HCS II System for displaying text on a LCD Display. It also understands the keypad query command and will respond to it, but since a keypad cannot be connected, it will always return 00. This is necessary because the HCS II uses the keypad query command to keep track of the LCD-Links and their existence on the network. If any stop responding, the HCS II will flag an error in the HOST console window.

All command packets must end with a carriage return (indicated by <CR>).

- **Query Keypad**

This command allows the HCS II to query a LCD terminal to see if keys have been pressed. Since the PIC-LCD does not support a keypad, it will return a default response so the HCS II can monitor its existence on the network.

```
! TERM2 Q<CR>
#cc TERM2 Q<CR>      (cc is computed checksum byte)
```

This will return 00's for the keypad value:

```
$ TERM2 00<CR>
```

If a checksum was included in the original packet, one will be included in the response.

- **Send Text to LCD Display**

To send text or control commands to the LCD Display, the S= command is used. The PIC-LCD has a 62-character buffer for incoming packets so you can easily send two lines worth of data to a 20x4 LCD Display in one command. Remember that the 62-character limit **includes** the checksum (if used) and network address, etc. To send text or commands to the LCD, format a packet as follows:

```
! TERM3 S=\e[2JHello World<CR>
#cc TERM3 S=\e[2JHello World<CR> (cc is computed checksum byte)
```

The above packet would clear the LCD Display and print "Hello World" on the display.

- **LCD Control Commands**

In addition to normal text, the PIC-LCD understands many ANSI and non-ANSI control commands which can be included in a S= packet (like the above `\e[ 2J` command).

Note that the upper left character of any display is row,col 1,1.

### **ANSI Command Set**

`\e[#A` Move cursor up # rows  
`\e[#B` Move cursor down # rows  
`\e[#C` Move cursor right # columns  
`\e[#D` Move cursor left # columns  
`\e[#;#H` 'Home' - Move cursor to row,col 1,1 (`\e[H`)  
`\e[#;#f` Move cursor to row,col #,# (i.e. move to any location)  
`\e[#;#j` Move cursor to row,col #,1 (`\e[3j` moves to row 3, column 1)  
**NOTE:** For the above 3 commands, the #;# defaults to 1,1 if a number is not included so, technically, they all function the same way.  
`\e[s` Save current cursor position  
`\e[u` Jump to saved cursor position  
`\e[ 2J` Clear LCD display and home cursor to 1,1  
`\e[K` Clear to end of row. **NOTE** the cursor will not move to end of line  
`\e[7h` Set PIC-LCD to wrap mode.  
Lines > max # of columns will wrap onto the same line.  
`\e[7l` Set PIC-LCD to auto CR/LF mode  
Lines > max # of columns will wrap onto the next line.

The PIC-LCD has an auto-scroll feature where the entire display will scroll upward when a line wraps down (or a `\n` is output) on the bottom line.

### **Non-ANSI Command Set**

`\e[b` Turn line cursor on  
`\e[c` Turn line cursor off  
`\e[o` Turn backlight on (if configured)  
`\e[p` Turn backlight off (if configured)  
`\e[x` Turn blinking box cursor on  
`\e[y` Turn blinking box cursor off

### **C-Style Escape Commands**

`\b` Sound Beeper (if configured)  
`\e` Escape Character (ASCII 27) for above commands  
`\t` Tab (4, 8, 12, 16, 20, etc.)  
`\n` Carriage Return/Line Feed  
`\r` Carriage Return Only  
`\x##` Output ASCII char ## in hex (`\xDF` for degree symbol, etc.)  
`\\` Print backslash ```

Remember that the total number of characters, including the network start char, address, checksum, and command data cannot exceed 62 characters. Packets that overflow the buffer are ignored.

Note that letters in packet commands are case sensitive!

---

## PIC-LCD Configuration

---

Before your PIC-LCD can be used, it must be configured for proper operation. PIC-LCD configuration is done using a serial terminal or PC running a serial communications package such as HyperTerminal. **NOTE!** You must turn on local echo on whatever serial device you use. I used up almost every byte of ROM in the PIC to provide usable functionality, but adding the echo back put it over the top. You will need an RS-232 voltage shifter like a Maxim MAX232. If you have an HCS, you likely have a few lying around since all HCS network nodes come with an extra MAX232.

Before you can configure your PIC-LCD, it must be constructed according to the schematic provided or at the very least, wire up the PIC, EEPROM with pull-ups, crystal, 20pF caps, and MAX232 on a breadboard. All configuration settings are stored in the EEPROM chip and will be usable if you construct your PIC-LCD later.

### Entering Configuration Mode

Disconnect any power to your PIC-LCD! If you have constructed your PIC-LCD completely, remove the 75176 chip. Now, connect a MAX232 RS-232 interface chip to your PIC-LCD's serial in and out lines. To connect it to a PC, you will need a DB9 or DB25 connector. A simple schematic can be found on our web page.

One hint – you probably will use a breadboard for connecting the MAX232, so an easy way to connect your MAX232 to your PIC-LIC serial lines is to take small gauge wire and insert it into the 75176 IC socket Pins 1 (Serial In) and 4 (Serial Out). Make sure you use small wire like 24 gauge to avoid damaging the IC socket. Doing this will help you easily enter configuration mode as outlined below.

One of my main goals with the PIC-LCD was to design a very small, 18-Pin interface. By doing this, it cut down on the number of I/O ports available for use and preventing me from using an I/O port just for entering configuration mode. Therefore, I had to get creative. Thus, I had to find an INPUT that I could manipulate in a "non-standard" way to indicate to the PIC-LCD that I wanted to enter configuration mode. However, I also had to be careful that normal operations did not accidentally trigger configuration mode.

Since the serial input is pulled high when no data is coming in and it is never pulled low for more than 936 $\mu$ s (One start bit plus 8 low bits), this made it easy for me to identify a configuration request. On power up, if the serial in line is low for more than 2ms, configuration mode is entered. However, if you power-up the PIC-LCD and data is actively coming into the PIC-LCD, it will recognize this and NOT enter configuration mode.

To enter configuration mode, ground the PIC-LCD's serial in pin (75176 IC socket pin 1) and power up the PIC-LCD. Now connect the serial in wire from your MAX232 to the PIC-LCD's Serial In line.

Remember, your serial device must have local echo turned on! Hit the return key once or twice and you should see the following on your serial terminal:

```
Creative Control Concepts PIC-LCD v#.##  
TERM#>
```

This means the PIC-LCD has successfully entered configuration mode. If you already have your LCD display connected to your PIC-LCD, the text `LCD OK` should show up on Line 1 to indicate the LCD is functioning and the PIC-LCD has control of it.

If you see these messages, you are now in configuration mode.

## **Configuring Your PIC-LCD**

### **Setting the Node Address**

The PIC-LCD can be used in a multi-drop configuration where each unit has a unique address. Addresses are 5 characters: `TERM#` where `#` is the unique ID of the node which must be between 0 and 7. Up to eight PIC-LCD's can be used on the same RS-485 network. If you use a PIC-LCD connected to an RS-232 serial port, you can only have one connected at a time so you can simply use the default node address of `TERM0`.

To set the address of your PIC-LCD, use the `A` command:

```
TERM#> A 3  
A:3  
TERM3>
```

If the address set was successful, the node number will be returned next to the command letter. If the value entered is out of range, "Error" is returned. The configuration prompt will also change to reflect the current node address.

### **Setting the Output Mode**

The PIC-LCD has an open drain output that can be used in three different ways. It is usually used to control a beeper or the LCD backlight. However, you can use it for anything you want since escape commands exist allowing you to turn the output on and off at will.

The following modes are supported:

- 0 – Beeper Mode. The pin is pulsed low for approx. 100ms when a `\b` command is sent in a `S=` packet.
- 1 – Manual Backlight Control. The pin is controlled by the `\e[o` (on) and `\e[p` (off) commands.
- 2 – Automatic Backlight Control. The pin is pulsed low for approx. 15 seconds after an `S=` packet is received. If the pin is still low when a new packet arrives, the 15-second timer is reset.

Like the address command, setting the beeper mode is done with a simple command letter and the appropriate mode number:

```
TERM3> B 1  
B:1  
TERM3>
```

If the command was successful, the command letter and setting are returned. If not, an error message is returned. The above example puts the output under manual control.

### Setting the LCD Display Type

The PIC-LCD supports 5 different sizes of LCD displays. In order for line wrapping and the auto scroll function to work, the PIC-LCD must know how big the display is. The PIC-LCD can be interfaced with any LCD display using the Hitachi HD44780 or equivalent controller chipset. The following display sizes are supported:

- 0 – 20x4
- 1 – 24x2
- 2 – 40x2
- 3 – Not Valid
- 4 – 16x2
- 5 – 20x2

Setting the display type is just like the previous settings. Use a simple command letter and display type number:

```
TERM3> L 2
L: 2
TERM3>
```

If the command was successful, the command letter and setting are returned. If not, an error message is returned. The above example sets the display size to 40x2.

### Configuring and Using Custom Characters

Most LCD displays have the capability to store and display up to 8 custom characters so users can design characters they need that are not part of the standard LCD character set. The PIC-LCD makes it very easy to create and store these custom characters in EEPROM. When the PIC-LCD is power cycled, the custom characters are loaded into the LCD Display from the EEPROM.

Custom Characters are created using the lower 5 bits of 8 binary bytes. For example, to create an up arrow, the character is created like this:

Byte	D7	-----	D0	Hex					
B0	x	x	x	0	0	1	0	0	= 04
B1	x	x	x	0	1	1	1	0	= 0E
B2	x	x	x	1	1	1	1	1	= 1F
B3	x	x	x	0	0	1	0	0	= 04
B4	x	x	x	0	0	1	0	0	= 04
B5	x	x	x	0	0	1	0	0	= 04
B6	x	x	x	0	0	1	0	0	= 04
B7	x	x	x	0	0	1	0	0	= 04

The bytes are saved using the S configuration command:

```
TERM3> S # B0 B1 B2 B3 B4 B5 B6 B7
S: #
TERM3>
```

Where # is the character number (0-7) and B0-7 are the binary pixel bytes. If the character is saved correctly, the command letter (S) and the character number are returned. If not, an error message is returned. **NOTE!** The PIC-LCD checks the character number for validity [0-7], but the pixel bytes are NOT checked in anyway so typos will be accepted and its anyone's guess what the character will look like ☺.

To save the above example as custom character 3, format the command as follows:

```
TERM3> S 3 04 0E 1F 04 04 04 04 04
S:3
TERM3>
```

The custom characters you create are saved in the LCD as ASCII characters 00 through 07. To display the above example character on your LCD, send a packet like this:

```
! TERM3 S=Temp Trend: \x03<CR>
```

### **Exiting Configuration Mode**

To exit configuration mode, power down the PIC-LCD, install the 75176 IC, and connect it to the network. This will ensure the serial in pin is high (or not low for more than 2ms) when the PIC-LCD is powered back up.

## PIC-LCD Pinout

---

Below is the pinout for the PIC-LCD chip:

### **PIC-LCD**

Pin 1	EEPROM/I <sup>2</sup> C SDA Line
Pin 2	EEPROM/I <sup>2</sup> C SCL Line
Pin 3	Backlight/Beeper Output (Open Drain)
Pin 4	/RESET
Pin 5	GND
Pin 6	Serial In
Pin 7	LCD RS
Pin 8	LCD R/W
Pin 9	LCD Enable
Pin 10	LCD Data Bit 4
Pin 11	LCD Data Bit 5
Pin 12	LCD Data Bit 6
Pin 13	LCD Data Bit 7
Pin 14	Vcc (+5V)
Pin 15	OSC2
Pin 16	OSC1
Pin 17	Serial Out
Pin 18	Serial Enable

Note the Backlight/Beeper pin is an open drain pin, which must be pulled-up externally. This pin can only sink 25mA! You must use a PNP transistor for higher currents like an LCD backlight.

---

## PIC-LCD Miscellaneous

---

The PIC-LCD chip has a 62-character buffer to store incoming commands. This is more than enough to handle one 40 character line or 2 20-character lines, both with control commands. Remember that the buffer is taken up by everything in a packet, including the start character, node address, checksum, and S=. Since the serial input is interrupt driven, the PIC-LCD will handle an incoming packet while processing the current packet in the buffer.

If you use the auto scroll feature to scroll the display, understand that each scroll can take almost 3ms to complete since most LCD Displays take up to 40 $\mu$ s per character. The PIC-LCD can receive a new packet while the scroll is happening, but if the new buffer catches up to the old, it will be dropped. This normally is not a problem, but if you send a packet like the one below (the cursor is on Line 4) and then immediately send another long packet, the second packet may be dropped because it caught up to the first (usually at the \n):

```
#cc TERM3 S=\nThis \n causes a scroll
```

```
#cc TERM3 S=\nAnother line and scroll
```

Since the \n that causes a scroll is very early in the packet, if the second packet is sent right after the first, the second packet may catch up to the first and be dropped. Normal HCS network traffic does not send S= packets one after the other so this isn't really a problem. The HCS II will often send a query packet right after an S= packet, but the PIC-LCD can handle that easily since the Q packet is too short to catch up with the long packet (since when it is received the PIC-LCD is already at the \n).

So, if you are using the PIC-LCD in a system other than the HCS II, be sure that packets that cause a display scroll are sent no closer than 5ms apart or so. If you use all cursor manipulation commands and don't auto scroll, the PIC-LCD & LCD Display can process the packets faster than they will arrive over the serial port.

---

## Constructing your PIC-LCD

---

If you purchased a kit from CCC, check the contents to ensure you have all the necessary parts:

### PIC-LCD

Qty	Description
1	18-Pin PIC-LCD IC
1	75176 RS-485 IC
1	24C02 Serial EEPROM IC
1	PN2907 Transistor (500mA MAX)
1	100 $\Omega$ ¼ watt Resistor
2	4.7K ¼ watt Resistors
1	1K ¼ watt Resistor
1	10K Potentiometer
1	2 Pin Jumper Block
1	Shorting Block
1	18-Pin IC Socket
2	8-Pin IC Sockets
1	4MHz Low Profile Crystal
2	20pF Capacitors
1	7805 Voltage Regulator
1	TO-220 Heatsink
2	0.1 $\mu$ F Capacitors
1	220 $\mu$ F Capacitor
2	2 Position Terminal Blocks
1	2x8 Pin Header Connector
1	2x8 IDC Connector
1ft	16 Conductor Ribbon Cable

Here are some hints when constructing your PIC-LCD:

- When building the circuit, mark off each connection on the schematic after it is soldered into place.
- Use ESD protection when handling the ICs! If you do not have a strap, touch something grounded before handling the chips.
- Construct the circuit using the IC sockets, leaving the chips alone. When you are done, check that +5V and GND appear on the proper pins in the IC sockets. Then install the chips.
- Locate the Crystal and 20pf capacitors as close to the OSC pins as possible.
- Install the shorting block to terminate the RS-485 network if necessary (i.e. if this module is on the extreme end of the network run). Only 2 modules can have active terminators!
- If you drive your LCD's backlight with the 7805, make sure it has a good size heatsink on it since it will get quite hot driving the 250mA or so with a 12V input supply.

## Need Help?

If you get stuck trying to get your system to work, drop us a line at [support@cc-concepts.com](mailto:support@cc-concepts.com) and we will do our best to help you get your PIC-LCD working.

Check out our web site at <http://www.cc-concepts.com/> for updates, bug reports, etc.

Here are some useful links for additional info referenced in this manual:

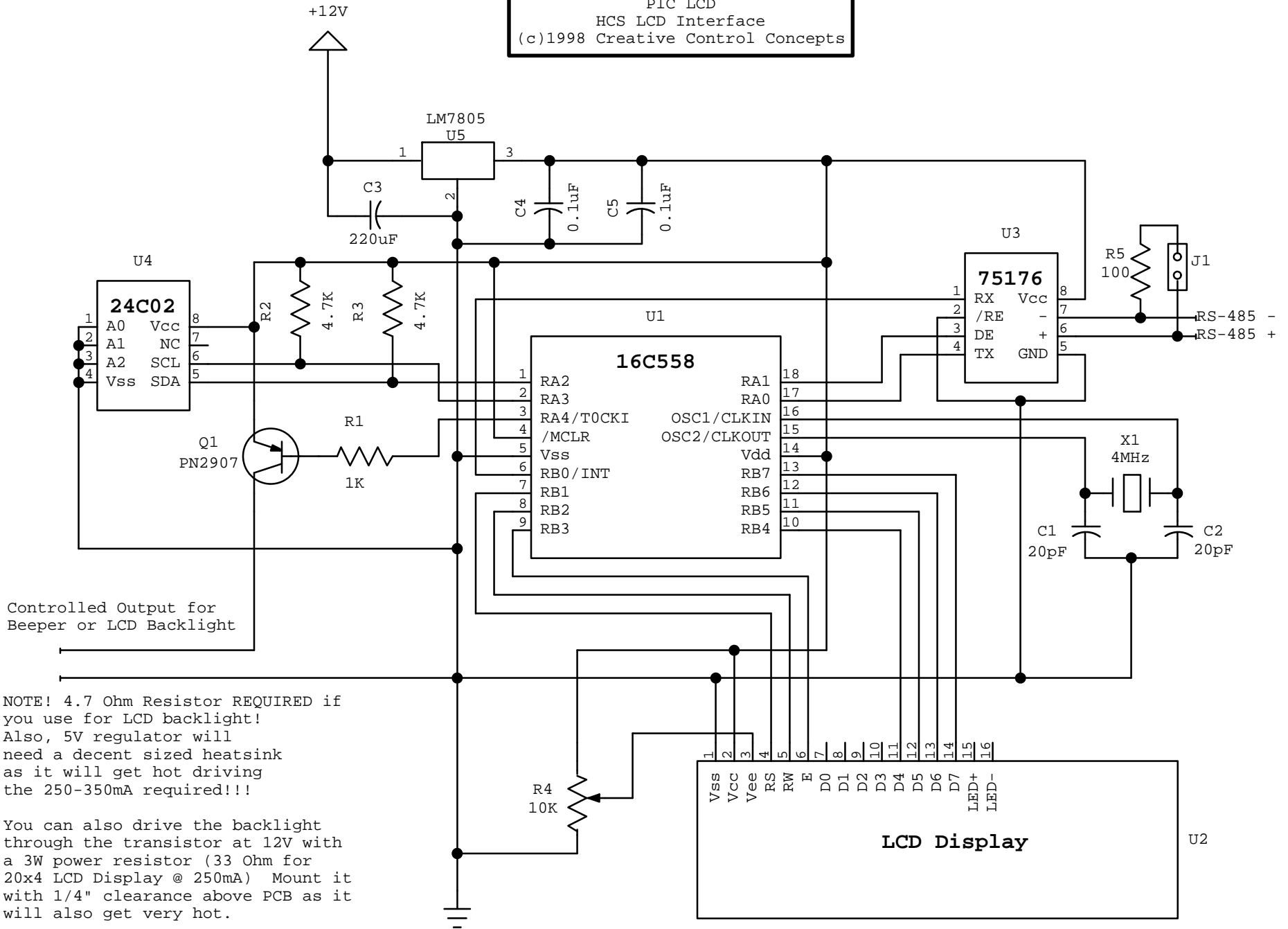
Microchip PICs: <http://www.microchip.com>

Circuit Cellar® HCS: <http://www.circuitcellar.com/hcs.html>

Optrex LCD Displays: <http://www.optrex.com/>



PIC LCD  
HCS LCD Interface  
(c)1998 Creative Control Concepts



Controlled Output for  
Beeper or LCD Backlight

NOTE! 4.7 Ohm Resistor REQUIRED if  
you use for LCD backlight!  
Also, 5V regulator will  
need a decent sized heatsink  
as it will get hot driving  
the 250-350mA required!!!

You can also drive the backlight  
through the transistor at 12V with  
a 3W power resistor (33 Ohm for  
20x4 LCD Display @ 250mA) Mount it  
with 1/4" clearance above PCB as it  
will also get very hot.