

Thank you for purchasing this *Hansen Hobbies Mini Scrolling LED Sign Kit!* This is a great kit to help you learn about electronics and build your soldering skills. An explanation of the circuit is included but you don't have to read or understand it to build this kit. First, please make sure you have everything:

- 2.4x1.7" PCB	(1)	- PIC Microcontroller	(1)	- 18pic IC socket	(1)
- 74AC164 shift register	(3)	- tact switch	(4)	- 1.3x2.5mm plug	(1)
- 8x8 LED matrix	(3)	- 2222A transistor	(9)	- 7805 regulator	(1)
- LM317 regulator	(1)	- 1.3x2.5mm jack	(1)	- 1x2 header	(1)
- jumper	(1)	- 1x2 connector*	(1)	- gold terminals*	(2)
- .1uF ceramic cap	(2)	- 22uF electrolytic	(1)	- 10K $\Omega$ resistor	(4)
- 470 $\Omega$ resistor	(8)	- 1K $\Omega$ resistor	(1)	- 680 $\Omega$ resistor	(1)
- 560 $\Omega$ resistor*	(1)	- 430 $\Omega$ resistor*	(1)	- solder 48"	(1)

\*parts are optional or may not be used in finished circuit

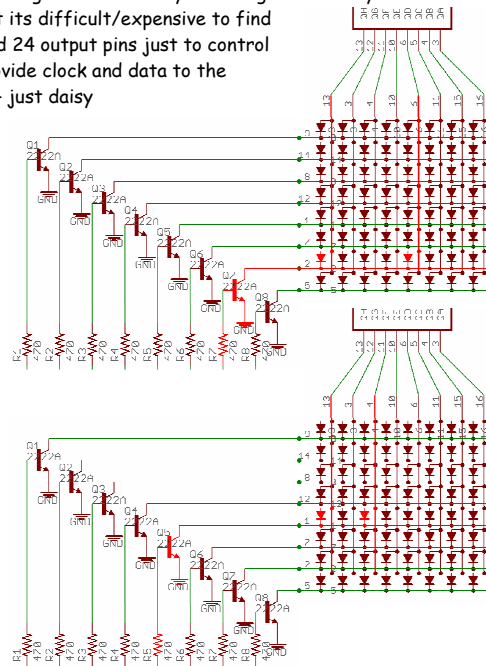
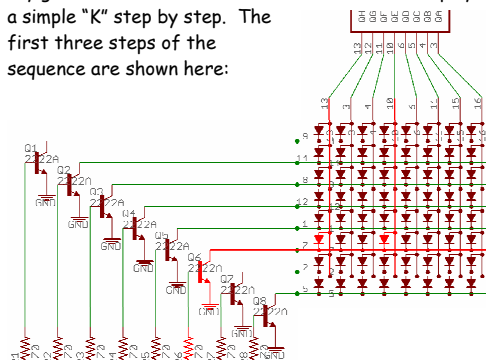
#### Circuit explanation:

Displaying characters on a sign is pretty easy, right? You just power on the lights you need and void! Well, it's not quite that simple. If we just made a big array of LED's and controlled each one separately then we'd need a chip with about 200 pins for this small sign, and hundreds or thousands more for a larger sign. By using a few handy control and power techniques we've managed to build a scrolling LED sign with a lot of functionality in a small package and everything runs on a very simple 8-bit microcontroller.

So how do we control 192 pixels with an 18pin microcontroller? A process called multiplexing...it's a lot like how your TV or computer screen works. Instead of trying to control each pixel all the time we scan through them much faster than the eye can see. Take a look at the schematic - you can see how all of the LED's are connected. An individual LED has a cathode (negative) and anode (positive) lead. To make it light we just give it power and current flows from anode to cathode. Every LED in a given row has a common cathode (most of these connections are made inside the LED matrix) and they're connected to a transistor to ground. The microcontroller can turn on any of the 8 row transistors and provide that row with a path to ground.

OK, now what about the anode/positive terminal? On any given column all of the anodes are common and connect to the output pins of shift registers. That pin can either be low/0/negative in which case it will not supply current to flow through the LED, or it can be high/1/positive. So why use shift registers here, can't we just connect directly to the microcontroller? There are two good reasons why it's designed this way and one has to do with power (explained below). The other is that its difficult/expensive to find a microcontroller with enough output pins. We would need 24 output pins just to control all the columns, but in this setup we just use 2 pins to provide clock and data to the shift registers. Also, this setup is infinitely expandable - just daisy chain more shift registers together to make a longer display.

So now we can control the current path to any given column and any given row - it's basically a big matrix of switches, and we can use the switch matrix to display things. We can shift out a string of bits to define which columns are on, and then we can turn on any given row. Let's look at an 8x8 matrix and display a simple "K" step by step. The first three steps of the sequence are shown here:



Power is a significant driving force in this circuit design. You may wonder why we need all those transistors to provide current paths to ground for the rows - can't we just connect them directly to the microcontroller? Well, if all the LED's in a row were on then over 100mA of current would be surging through the transistors which are rated for 600mA, while the microcontroller pins are rated for 20mA.

Here's where we explain the second reason for using shift registers in our circuit. Take a look at the far right of the schematic - that's the power stuff. You have the power jack on top where we give the circuit 5-12V and then there are two voltage regulators. The 7805 is a 5V regulator and provides 5V to the microcontroller. The LM317 is an adjustable regulator and it supplies about 2V to the shift registers. The output voltage can be adjusted using two resistors (R11 & R12). This is a little trick used to reduce the number of components in the circuit. Normally, if you wanted to power an LED from 5V, you'd have to put a resistor in series with it to reduce the current to normal levels (red LED's run at around 2V). Instead of adding 24 resistors between each of the shift register output pins and the columns we just power that entire portion of the circuit from 2V. In order to do that we use special chips rated for a lower voltage - a little more expensive, but a good tradeoff. Using the LM317 to create 2V has a very handy side effect; we can adjust the output voltage, and therefore the display brightness, by changing just one of the resistors. Two extra resistors are provided for this. Using the 680 $\Omega$  resistor for R11 will yield the brightest display and the 560 $\Omega$  or 430 $\Omega$  can be used to make it less bright and draw less power (a plus if you plan to make this battery powered). If your power supply voltage is on the high end (9V and up) you may also consider using a lower R11 resistor to reduce the load on the LM317 (which has to dissipate all that excess voltage as heat).

#### Assembly instructions:

**Note: We highly recommend that you read the instructions in their entirety before you start assembly.**

We'll assume you've soldered before, but if you haven't there are plenty of good tutorials on the internet. This kit includes solder that has a water-soluble flux in it. Flux is the stuff in solder that helps make a good joint and also leaves an ugly mess around your solder joints. We prefer to use water-soluble flux because it can easily be cleaned off with hot water and a brush (a hog-hair brush is best, but a toothbrush will work too). Cleaning the water-soluble flux off the board is not optional - it must be cleaned off or the circuit could malfunction. Running hot water over your circuit will not damage it in any way. If you want to use the plain rosin flux you've got sitting on your workbench that's fine, and you don't have to clean it if you don't want to. Pretty much any flux can also be cleaned with isopropyl alcohol.

A separate page has several photos to help you with your build. Begin by placing and soldering the three shift registers on the back of the circuit board. DO NOT solder these onto the wrong side of the board. They should go on the side that doesn't have the white silkscreen text. Notice the notches in the end of the chip denote orientation. These solder joints will be covered by the LED matrices so be sure to make good joints and no shorts; you won't be able to go back and fix them later. After soldering clean the board of any flux and clip the leads flush.

From here on you can solder in any order you choose. A good suggestion is to place and solder the smaller parts first (resistors and capacitors) then do the larger parts. A recommended sequence is: resistors (14), capacitors (3), transistors (9), voltage regulators (2), tact switches (4), IC socket (1), header (1), jack (1), LED matrices (3). Here are a few tips and photos:

- Resistors and ceramic capacitors have no special orientation. Inspect resistor colors carefully.
- The electrolytic capacitor is oriented so the longer lead is positive (see the "+" mark) and the white stripe is negative
- Do not confuse the transistors with the 2 voltage regulators - read the part numbers printed on them and place them correctly
- The text on the LED matrices must face down as pictured
- Clip the leads as you go so they don't interfere with soldering
- Solder the IC socket in place, NOT the microcontroller itself

If you plan to use the power supply that we sell then you're good to go. If you plan to use your own supply or battery then you need to solder the included plug onto the leads. The center contact is positive and the outer contact is negative. Be sure to thread the wires through the plug housing before soldering.

After you have all the parts soldered in place clip off any remaining leads and clean the board with hot water and a brush (if you used the water-soluble solder included). After washing with tap water we highly recommend doing a final rinse with distilled water or isopropyl alcohol if you have it (this will wash away any

impurities in your tap water). Dry the board as best you can with a towel (you may have to wing it around or smack it against your thigh to get the water out from under the parts).

Make a final check of all the parts and solder joints - make sure everything is in the right spot and there aren't any shorts. For the first power-up, it is best to leave the microcontroller out. Plug in power and check to make sure nothing gets hot (especially the voltage regulators). Measure the voltage on the regulators with a meter if you have one: the 7805 should output 5V, and the LM317 should be about 2V. Unplug power and insert the microcontroller (watch the orientation) and also plug the jumper onto the header. Plug power back in and a message should be displayed. Note that the MLS (Mini LED Sign) will stop displaying this message after it's been powered up 50 times. From then on it will load your customized message on startup.








### Debugging problems:

You may power on your circuit and find that one of the rows or columns isn't working. It's easy to track down the problem using the schematic - just trace the row/column back to the transistor/shiftreg that controls it and check for bad solder joints.









### Button Operations:

We've worked hard to make an intuitive and easy to use button interface for this sign. With a little bit of practice you can enter messages quite fast. The sign is operated in two different modes; a Scrolling Mode where the sign is displaying your message, and a Text Input mode where you can modify the message.

#### While the sign is in Scrolling Mode:

	Manually scroll the display window to the right.
	Decrease the scroll rate (there are 8 different speeds).
	Increase the scroll rate (there are 8 different speeds).
	Manually scroll the display window to the left.
	Send string over serial port (microcontroller pin 8) to another sign.
	Restart the message from the beginning.
	Stop Scrolling Mode and enter Text Input Mode.




#### While the sign is in Text Input Mode:

	Move cursor to the right.
	Decrement the character to the right of the cursor; if no character exists, insert a space.
	Increment the character to the right of the cursor; if no character exists, insert an "A".
	Insert a character (an "A") where the cursor is.
	Delete the character to the right of the cursor.
	Delete the character to the left of the cursor (backspace).
	Move cursor to the left
	Stop Text Input Mode and enter Scrolling Mode. The message you entered is stored in EEPROM.

For any of the keystrokes shown you can hold down the button(s) to make the operation execute over and over at a fast rate (just like with your computer keyboard).

There is one last set of button presses that provide some very interesting options. In order to better show the multiplexing technique described in the circuit explanation above, the MLS is equipped with a "Teaching Mode" which allows the user to slow down the display refresh rate and see how the entire display is drawn one row at a time. To enter Teaching Mode, press the Up and Down buttons while powering up the MLS. In Teaching Mode the MLS will be completely operational but two additional button strokes will be available.

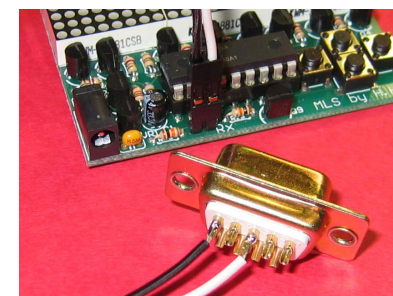
#### When Teaching Mode is active:

	Hold these buttons down while turning on power to allow the Teaching Mode options below to work.
	Decrease the display refresh rate (there are 8 different speeds).
	Increase the display refresh rate (there are 8 different speeds).

Starting from the normal refresh rate, each decrement will reduce the refresh rate by a factor of four, and you will be able to see how the MLS turns on just one row at a time to draw the whole display. On the three lowest speeds the rate at which the data is shifted across the shift registers is slowed down and displayed so you can see that part of the control. It's impressive to see how much is being done at speeds that our eyes can't detect! Take note that when the MLS is operating the shift registers it cannot detect button presses, so in the very slow speed settings you will have to press the buttons in between row operations - this isn't a problem during normal operation because the MLS usually only spends less than 0.0001s shifting out one row, and there is plenty of time to check the buttons. The MLS also won't be able to process serial port commands while loitering in the row shifting function.

### Connecting the MLS to Your Computer:

The MLS has the ability to take input via your computer's serial port. You could also communicate with it using another MLS or microcontroller. To connect the sign to your computer you must make a cable (or modify a serial cable) using the supplied connector. The connector is a simple 0.1" female connector with crimp terminals (we have crimping docs and videos on our website). You need to ground the circuit to the computer ground, and connect to the transmit pin on the serial port. You can use pretty much any 2 conductor wire, or as mentioned, take a serial cable, cutoff one end, and attached the supplied connector to the ground and transmit wires. An example of a cable and how it connects to the MLS is shown to the left. You can control the MLS by opening hyperterminal - a serial port terminal included with all versions of Windows (except Windows 7, but you can download and install it). It can be found under Programs>Accessories>Communications. Make a new connection under whichever serial port you're using, and set the "bits per second" to 19200 and "flow control" to none. Leave the rest of the options alone. Once connected you can begin typing things. Here's a basic chart of commands:



#### While the sign is in Scrolling Mode (hex values shown for reference):

Keystroke	Description	Hex Transmission
Enter	Switch to text input mode	0x0D
Right/Left Arrow Keys	Manually scroll right/left	0x1B, 0x5B, A 'C'/'D'
Up/Down Arrow Keys	Change scrolling speed	0x1B, 0x5B, A 'A'/'B'
Home	Restart the message from the beginning	0x1B, 0x5B, A 'H'

#### While the sign is in Text Input Mode (hex values shown for reference):

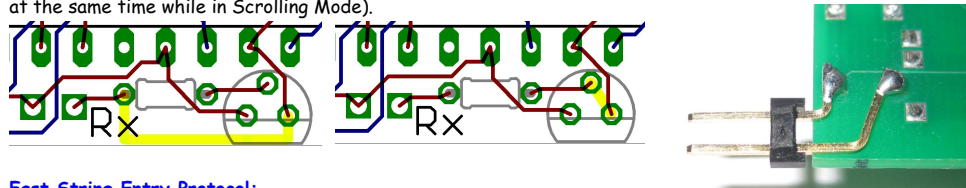
Keystroke	Description	Hex Transmission
Standard Keyboard Characters	Inserts that character	0x20 through 0x7E
Backspace	Backspace	0x08
Home	Bring cursor to beginning of message	0x1B, 0x5B, A 'H'
End	Bring cursor to end of message	0x1B, 0x5B, A 'K'
Delete (doesn't work in hyperterminal)	Delete	0x7F
Enter	Start Scrolling Mode	0x0D
Right/Left Arrow Keys	Move cursor right/left	0x1B, 0x5B, A 'C'/'D'



You can view a table of ASCII characters at [www.asciitable.com](http://www.asciitable.com). Note that the ASCII character 0x7F (the delete character) is actually a smiley face, but it can only be directly entered with the fast string entry mentioned below since 0x7F is otherwise reserved for the delete function. If you're running your own application then you should consider using the fast string entry command detailed below.

**Connecting the MLS to another MLS or Microcontroller:**

If your microcontroller is equipped with a RS232 level converter than connect to the MLS as above. If you want to connect the MLS directly to another microcontroller then you have to bypass the RS232 level converter so that your signal won't be inverted. The best way to do this is by removing the resistor R13 and transistor Q9 (or don't solder them there in the first place) and putting a jumper wire where the yellow line is in the image below (left). You could also just solder the 2-pin header in where Q9 is supposed to go and get identical results. Notice that this puts your microcontroller in direct connection with the microcontroller on the MLS. If the circuit transmitting data has a higher supply voltage, then this type of connection is not recommended. If this is a concern then you can try going through the resistor R13 (still remove Q9) as shown below (center) - this will allow higher signal levels to bleed through the clamping diode in the MLS's microcontroller. The third image (right) shows how you can tap in to ground (bottom pin) and pin 8 (the pin used for both the right button and serial transmission) of the MLS microcontroller by soldering to the legs of the right tact switch on the bottom of the board. When the MLS in the right image is connected to the MLS in the left/center image it can copy it's message over to the receiving MLS (by pressing the up/down buttons at the same time while in Scrolling Mode).



**Fast String Entry Protocol:**

When using a microcontroller or custom user application, communication with the MLS could be done exactly as with hyperterminal above, but that could cause problems since the application may not know where the cursor is at a given time, and MLS takes a while to process such commands. If you plan to do it this way then you must include reasonable delays (10ms) between bytes for processing time. For most applications you'll want to use the fast string entry command developed for instant data entry. The command is sent at 19.2kbaud and is shown here:

Transmit byte or range (hexadecimal)	Description
0x02	This is the start byte
0x01 through 0x80	Send any of these values to assign the scroll rate, 0x01 being the fastest and 0x80 being standstill
0x20 through 0x7F	Here is where you send your message - any ASCII character between 0x20 and 0x7F is valid. You can have up to 232 characters here - any extra will be truncated.
0x20 through 0x7F	
0x20 through 0x7F	
0x20 through 0x7F	
0x0D	End of message indicator - MLS will begin displaying your message

While the string is being received/processed the sign will display nothing and wait perpetually for the end of message indicator. If you don't know what state the sign is in, a safe way to send this command is to send 0x20, 0x0D, then 0x02.

Have fun!

