http://web.alfredstate.edu/faculty/weimandn/lcd/lcd_addressing/lcd_addressing_index.html

LCD Addressing

Character based LCD modules

The information in this section relates to Character based LCD modules, specifically those controlled by an HD44780 or equivalent.

Disclaimer

This is not official manufacturer's information. It is distilled from information on many different data sheets and from my own experience. I have never used some of these displays. I have extrapolated what I know about the displays that I have used to the ones that I have just read about. I welcome any suggestions and corrections. Contact information is at the bottom of the page.

Introduction

The HD44780 type controller chip is used with a wide variety of Liquid Crystal Displays. These LCDs come in many configurations each with between 8 and 80 viewable characters arranged in 1, 2, or 4 rows.

The problem is that there is no way to inform the controller of the configuration of the display that it is driving. The controller operates exactly the same way for all displays and it is up to the programmer of the device that is controlling the LCD controller (usually a host microcontroller) to deal with this situation.

The controller contains 80 bytes of *Display Data Random Access Memory* which is usually referred to as DDRAM. When the controller is used with a 40 x 2 display (forty characters on each of two rows) the operation is quite straightforward and that operation will be explained first. Each of the other configurations introduces one or more quirks so it is best to understand the operation of the 40 x 2 before proceeding to the description of the operation of any of the others.

LCD Controller Memory

This is the most common *Memory Map* for the 80 bytes of DDRAM in the HD44780 controller. There is another rarely encountered configuration that will be presented later.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

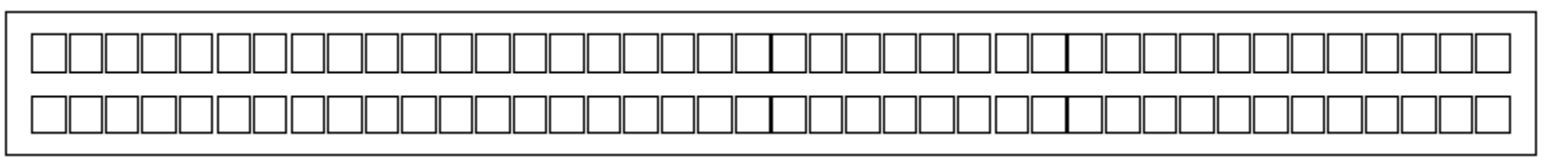
As you can see the DDRAM consists of two *lines* of memory with a somewhat mysterious gap (see note 1) in the addressing when going from the first line of memory to the second. The first line has 40 storage locations identified by the addresses 00h through 27h. The second line has another 40 storage locations identified by the addresses 40h through 67h.

Each of these DDRAM memory addresses corresponds to a character position on an attached display, but the specific position varies depending on the configuration of that display. As part of the initialization sequence the display is cleared by storing the ASCII code for a *space* in each of the 80 memory locations. Subsequently if a different ASCII code is stored in any of those memory locations then the character corresponding to that ASCII code is automatically displayed at a specific location on the display.

You can tell the controller where you want the first ASCII character that you send it to be stored, this is usually address 00h. After receiving that character it will automatically update its address pointer and put the next ASCII character you send into an adjacent memory location with no more addressing work on your part. You can specify whether to increment or decrement the address counter but normally it is incremented, so the next character will be put into address 01h. The LCD controller automatically accounts for the gap in addresses and after storing an ASCII code in address 27h it puts the next code in address 40h. Similarly it increments from address 67h back to 00h.

40 x 2 LCD

Here is a simplified diagram of the display on a 40 x 2 LCD Module. Each of the boxes in the diagram represents a location where a character can be displayed.

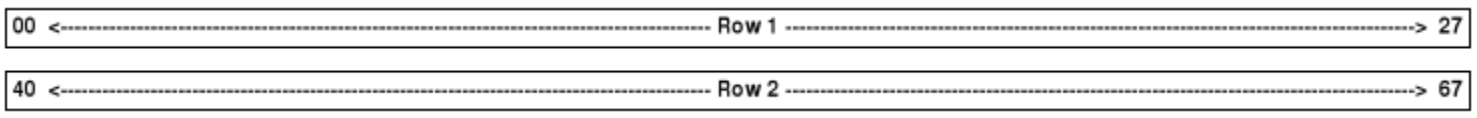


Here is a copy of the memory map of the controller. Remember, each of the memory locations in the controller chip is directly associated with one of the character locations on the display.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

By some miracle of modern technology there is actually a one for one relationship between these two diagrams. If an ASCII code is stored at address 00h in memory the corresponding character will appear at the left end of the top row of the display. If an ASCII code is stored at address 63h in memory the corresponding character will appear five locations in from the right end of the second row of the display.

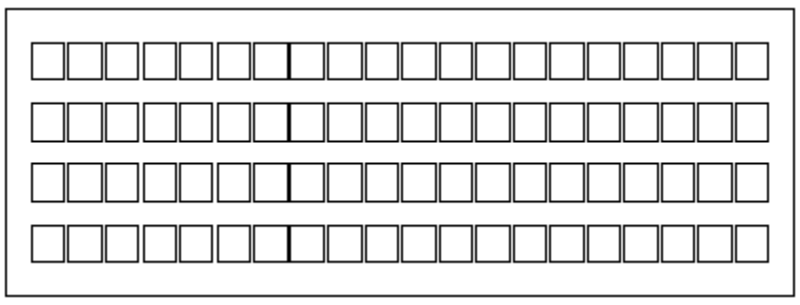
Here is a diagram showing how the two **rows** of the display are mapped into the two **lines** of memory. It is basically a combination of the two diagrams just above.



When the host controller wants to display a string of characters on the display all it has to do is specify a starting DDRAM address and then start sending the string of ASCII codes corresponding to the desired characters to the LCD controller, one after another. The LCD controller takes the first code that it receives, stores it at the specified address, and simultaneously displays the corresponding character on the display. It then increments it's internal address counter and stores the next ASCII code that it receives in the next DDRAM location which causes the corresponding character to appear in the next location on the display. As mentioned before the LCD controller automatically accounts for the gap in addresses and after storing an ASCII code in address 27h it puts the next code in address 40h. Similarly it increments from address 67h back to 00h.

20 x 4 LCD

This display also has 80 characters, but the relationship between the DDRAM addresses and the character locations on the LCD is not quite as straightforward as the LCD with two rows of 40 characters. Here is a diagram of the device.



The memory map is always the same regardless of the display configuration, but in this drawing I have shown a small space between addresses 13h and 14h on the first line and another between addresses 53h and 54h on the second line.

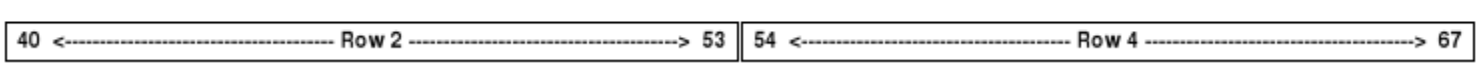
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Here is the same memory map, rearranged this time to show how the memory addresses relate to the character positions on a 20 x 4 LCD. Note how the right half of the previous diagram is now below the left half and note the resulting sequence of starting addresses for each display row (00h, 40h, 14h, 54h).

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Remember that the LCD controller still considers this to be two lines of RAM. It is important to understand that this way of picturing the DDRAM addresses helps relate the memory addresses to the character locations but does not change the fact that as far as the controller is concerned there are only two lines of memory. In other words, although this diagram shows the DDRAM differently than before, the actual DDRAM configuration and operation is **exactly** the same as described above for the 40 x 2 display since there is no way of telling the LCD controller that there are now 4 rows of 20 characters instead of 2 rows of 40 characters.

Here is a diagram showing how the four rows of the display are mapped into the two lines of memory.



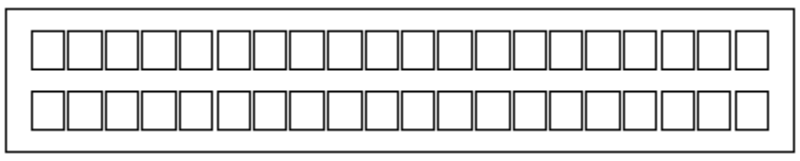
When a long string of ASCII codes is sent to this LCD controller the action is not quite as simple as for the 40 x 2 display. After the first row is full the characters will continue on to the third row. The normal automatic incrementing from 27h to 40h will then cause the display to continue on the second row, and from there it will continue to the fourth row. After that the following characters will appear back on the first row, and so on.

In order to get a coherent display on sequential rows it is necessary to compensate for the design of the LCD controller when programming the host microcontroller. Basically the program on the host microcontroller can keep track of the DDRAM addresses, and when appropriate it can set up a new starting DDRAM address.

For each of the above displays there are 80 addresses in memory and there are 80 character locations on the display so it should be obvious that any time you send an ASCII code to the controller the corresponding character will show up *somewhere* on the display. If you mess up the address the character may not show up where you expected it, but it will be visible somewhere. If you work back from where it actually appears you can usually figure out where you made your mistake. All of the displays that follow have fewer character locations on the display than memory addresses in the controller. This makes the operation somewhat more complicated and troubleshooting more difficult.

20 x 2 LCD

This can be thought of as a truncated 40 x 2 display, but there are some ramifications of this truncation that may not be readily apparent. Here is a drawing of the device.



Here is the part of the DDRAM memory that is normally used to display characters on the LCD.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53

It is important to understand that, although this diagram shows only the part of the DDRAM that is normally used to display information on the 20 x 2 LCD, the actual memory map and controller operation is **exactly** the same as described above for the previous displays. Again that is because there is no way of telling the LCD controller that there are only 40 characters on the attached display.

Here is a drawing of the complete memory map. Note that this drawing is the same as the one for the 20 x 4 display except that the addresses on the right side are greyed out.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Although this display has only 40 characters there are still 80 bytes of DDRAM and they are still configured the same as they were before. The greyed out addresses are the locations in DDRAM that do not have corrresponding locations on the display. Any ASCII codes that are written to those locations are not lost, and it is possible to display them by 'shifting' the display window, but in normal use as described here they are simply not displayed.

Here is a diagram showing how the two **rows** of the display are mapped into the two **lines** of memory.

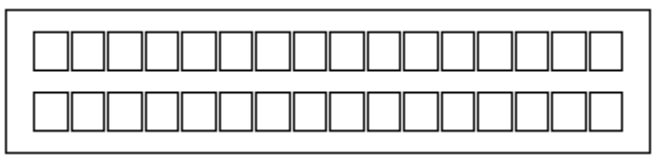
00 <---------- Row 1 ----------> 13	14 <---------- not displayed ----------> 27
40 <---------- Row 2 ----------> 53	54 <---------- not displayed ----------> 67

When a long string of ASCII codes is sent to this LCD controller the action is not quite as simple as for either of the 80 character displays. Assume that the host controller is sending a string of characters as described above. Consider what happens after the LCD controller stores an ASCII code in address 13h and displays the corresponding character at the right end of the top row on the LCD. It then stores the next ASCII code in address 14h, which has no corresponding location on this 20x2 display. As more ASCII codes are sent to the LCD controller they are stored in the DDRAM but do not appear on the display until the LCD controller finally increments it's address counter from 27h to 40h at which time subsequent characters start to appear on the second row of the display. As far as a viewer of the display is concerned there is a gap of 20 missing characters. The same thing will happen on the second row when ASCII codes are stored in addresses 54h - 67h.

In order to prevent any missing characters the program on the host microcontroller can keep track of the DDRAM addresses, and when appropriate it can set up a new starting DDRAM address. On the other hand the display can be *shifted* to display those missing characters, but the techniques to do that will not be covered here.

16 x 2 LCD

This is a commonly found configuration and its operation is almost identical to that of the 20 x 2. The relationship between the DDRAM addresses and the character locations on the LCD is a subset of the example shown above. Here is a drawing of the device.



Here is the part of the DDRAM memory that is normally used to display characters on the LCD:

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Once again it is important to understand that although this diagram shows only the part of the DDRAM that is normally used to display information on the 16 x 2 LCD the actual DDRAM configuration and operation is **exactly** the same as described above for the 40 x 2 display. This is because there is no way of telling the LCD controller that there are only 32 characters on the attached display.

Here is a drawing of the complete memory map. Note that this drawing is the same as the one for the 20 x 2 display except that a different range of addresses on the right side are greyed out.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Here is a diagram showing how the two **rows** of the display are mapped into the two **lines** of memory.

The operation of this display when a long string of characters is sent to it is that same as described for the 20 x 2 display except that there is a gap of 24 missing characters at the end of each line (instead of 20).

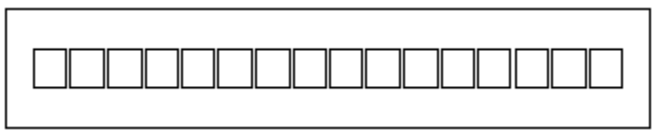
16 x 1 LCD

There are actually two different varieties of 16 x 1 LCD displays and the initialization and operation of each is different so it is important to determine which one you have.

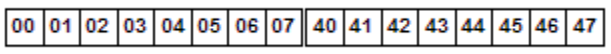
When power is first applied to any of the multi-row LCD modules and before the controller is initialized you will see that the character locations corresponding to the first line of memory are dark and the others are light (assuming that the contrast adjustment is properly set). If you apply power to a 16 x 1 LCD module and only the left 8 character locations are dark you have what I will call a type 1 module. If only the right 8 character locations are dark this is also a type 1 module but it is upside down! If all 16 character locations are dark then it is what I will call a type 2 module. This is my own terminology used only for the purpose of keeping them differentiated while describing their operation. The type 1 modules will have only one IC on the back of the pcb while the type 2 will have 2 (I guess this tidbit gives away the source of my 'type' designations). This distinction *may* apply to newer devices with epoxy blobs instead of ICs, but I believe that sometimes one blob may cover more than one equivalent IC function.

16 x 1 LCD (Type 1)

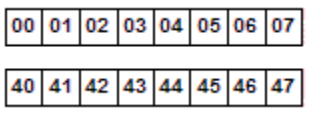
This is the most common configuration, probably because it is less expensive to produce than the other. Here is a drawing of the device.



Here is the part of the DDRAM memory that is normally used to display characters on the LCD.

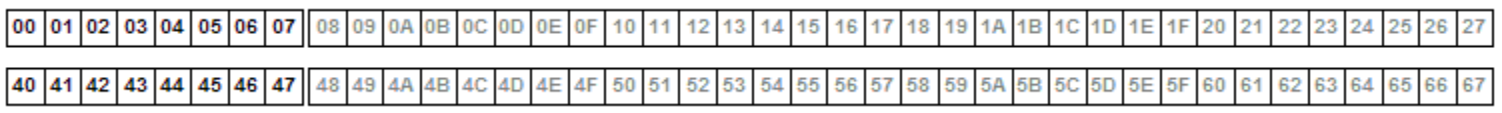


But here is the the way that part of the DDRAM memory is actually configured.



From this you can see that although the display has only a single row of characters, as far as the LCD controller is concerned it is using two lines of memory and it must be considered to be a 2 line device when initializing the controller.

Here is a drawing of the complete memory map. Note that this drawing is the same as the one for the 20 x 2 and 16 x 2 displays except that a different range of addresses on the right side are greyed out.



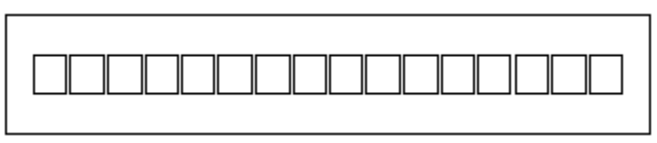
Here is a diagram showing how the single **row** of the display is mapped into the two **lines** of memory.

Here you can see that if the host controller sends a long string of characters without periodically adjusting the DDRAM starting address then after each 8 characters are displayed the next 32 will "disappear".

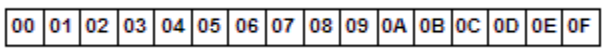
Also, to display a message of more than 8 characters on the 16 character display the host controller will have to readjust the DDRAM address after displaying the first 8 characters.

16 x 1 LCD (Type 2)

This is a less common configuration but if you can find one they are easier to deal with. The device looks just like the Type 1.

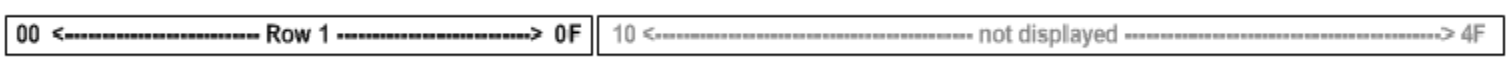


Here is the part of the DDRAM memory that is normally used to display characters on the LCD.



Here is a drawing of the complete memory map. Note that this memory map is different than all of the previous ones. This is the only device described here that is a true 'one-line' display (see note 2) and as such it has a different memory map.

Here is a diagram showing how the single **row** of the display is mapped into the single **line** of memory.



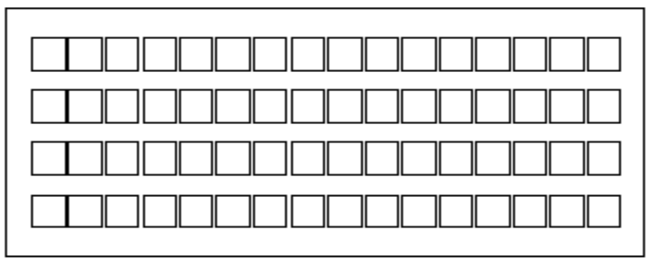
Here you can see that if the host controller sends a long string of characters after the first 16 characters are displayed the next 64 will "disappear".

At the expense of an extra IC you get some slightly easier programming since, in order to display a message of more than 8 characters on the 16 character display, the host microcontroller does not have to readjust the DDRAM address after displaying the first 8 characters.

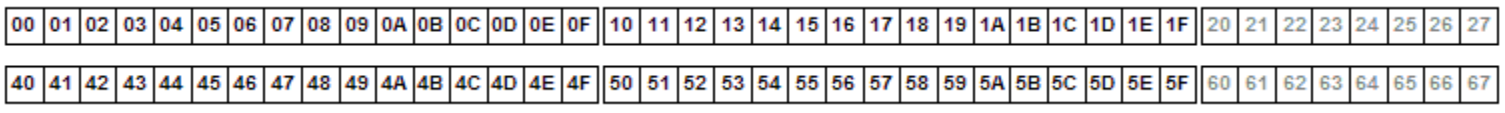
Since only one line of memory is in use this LCD module should be configured as a 1-line device. As far as I can determine, this changes the multiplex frequency which changes the display brightness and/or contrast. Also, there are some single row LCDs that are capable of displaying a larger 5x10 font instead of the more common 5x7 font.

16 x 4 LCD

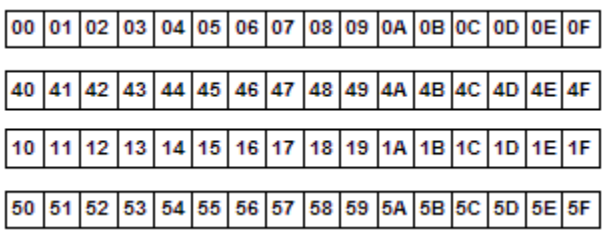
I've stuck this description toward the end because, as you will see, it has some of the characteristics of the 40 x 4 and some of the characteristics of the 16 x 2. Here is a drawing of the device.



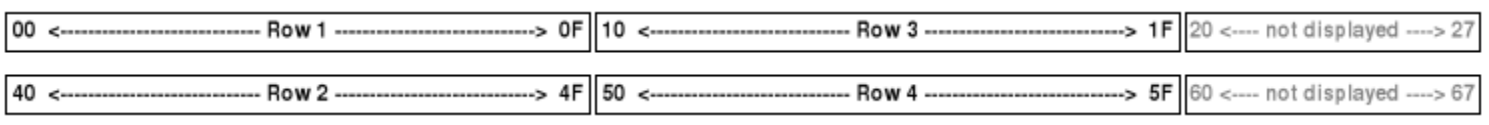
Here is a drawing of the complete memory map. Note that this drawing is different than all of the previous ones.



Here is the same memory map, rearranged this time to show how the memory addresses relate to the character positions on a 16 x 4 LCD. Note how the center of the previous diagram is now below the left part, the right part is missing, and the resulting sequence of starting addresses for each display row is different than for the 20 x 4 (00h, 40h, 10h, 50h).



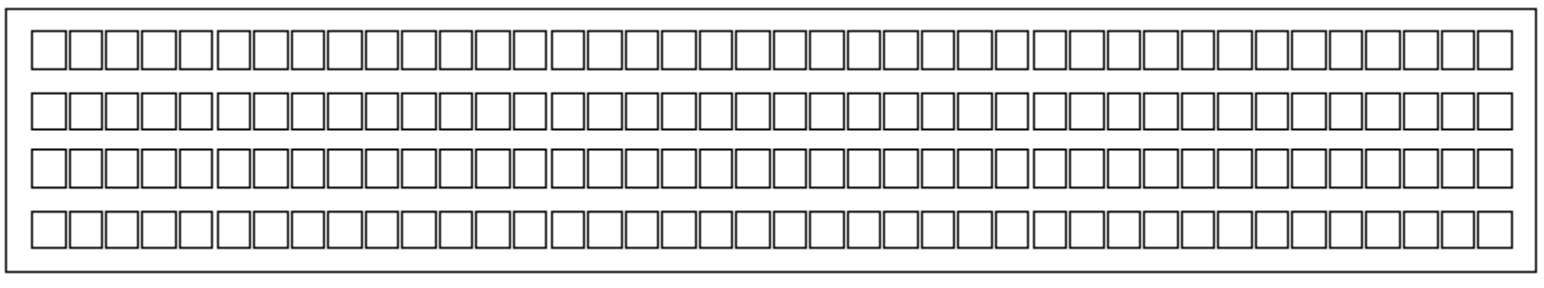
Here is a diagram showing how the four **rows** of the display are mapped into the two **lines** of memory.



Here you can see that if the host controller sends a long string of characters without periodically adjusting the DDRAM starting address then after the first row is full the characters will continue on to the third row. After the third row is full the next eight characters "disappear". The normal automatic incrementing from 27h to 40h will then cause the display to continue on the second row, and from there it will continue to the fourth row. After the fourth row is full the next eight characters will "disappear" and then it's back to the first row.

40 x 4 LCD

The 40 x 4 LCD is treated essentially as two 40 x 2 devices stacked one on top of another in the same glass enclosure. Electrically it uses what amounts to two HD44780 controller chips and it therefore has two separate memory maps each with the same address range. One is used for the top two lines and the other is used for the bottom two lines. The memories are accessed individually by strobing the desired *Enable* pin of which there are now two. Here is a diagram of the device.



Here is the memory map of the controller.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

Here is a diagram showing how the four **rows** of the display are mapped into the four **lines** of memory.

00	<---------- Row 1 ---------->	27
40	<---------- Row 2 ---------->	67
00	<---------- Row 3 ---------->	27
40	<---------- Row 4 ---------->	67

To display a really long string of characters on this display the host controller would start out just like it did for the 40 x 2 display. It would specify a starting DDRAM address (typically 00h) and then start sending the string of ASCII codes corresponding to the desired characters to the LCD controller, one after another, making sure to strobe the enable pin associated with the upper memory bank. After storing an ASCII code in address 67h the LCD controller will automatically increment to address 00h as before *and at this time the host controller must stop strobing the enable pin for the upper bank and start strobing the one for the lower bank*.

Other Configurations

There are other LCD configurations available but I believe that any of them can be handled by a technique similar to one of the examples above. If you have a display that seems to be considerably different from any of these I would like to hear from you.

Notes

(1) The mysterious gap is due to considerations resulting from the multiplexing of the display. The DDRAM addressing uses seven bit addressing and the highest bit signifies which row of memory is involved. If you compare the addresses in the first row with those just below in the second row you will see that the only difference is in that one bit.

(2) As implied above the number of rows of characters that can be displayed on the LCD is not the same as the number of lines of memory used by its controller. Only some of the 16x1 displays use 'one line' of memory, all of the other displays including most 16x1 displays, use 'two lines' of memory.

Back to the index page.
Last modified on 29 September 2012
Copyright © 2009, 2011, 2012 Donald Weiman (weimandn@alfredstate.edu)
This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/