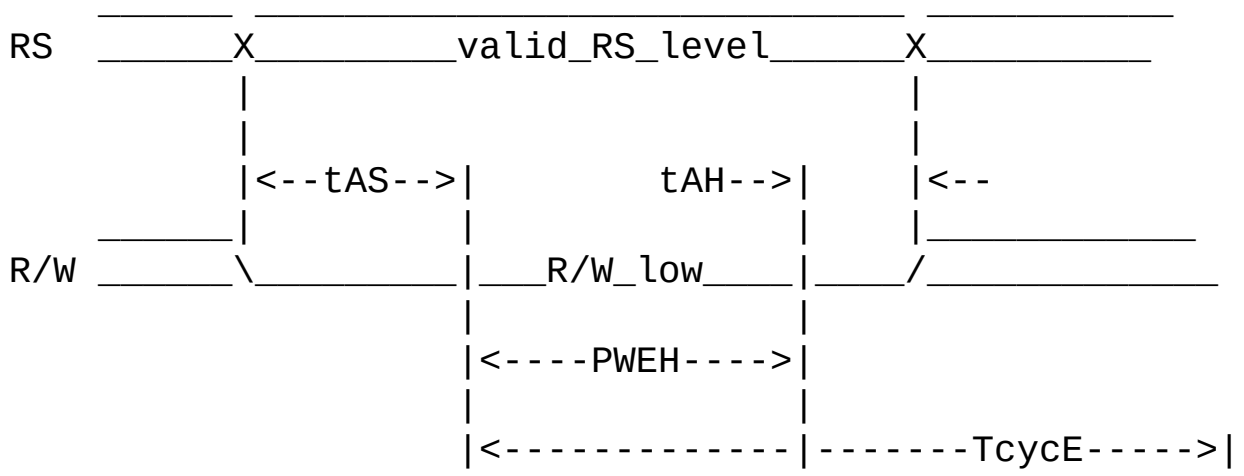
LCD Technical FAQ

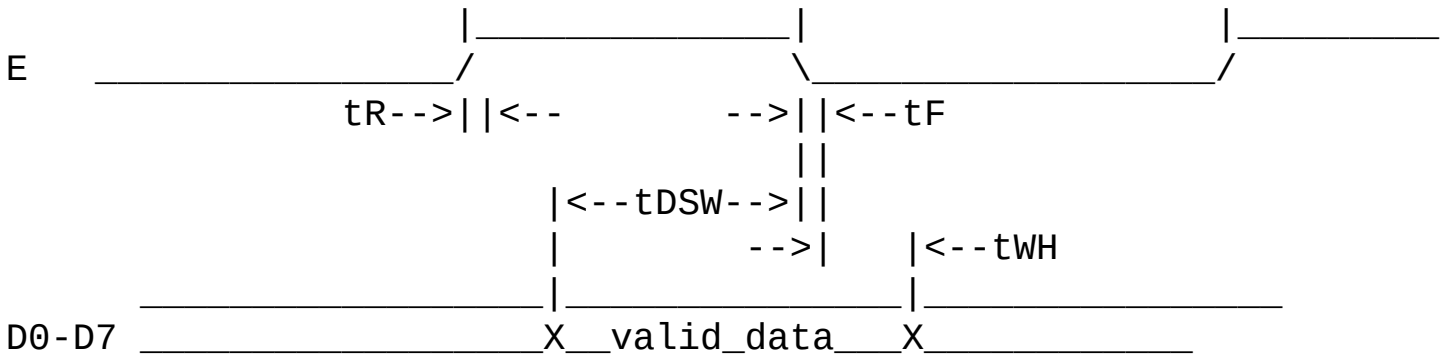
Contents:

3.4) Timing

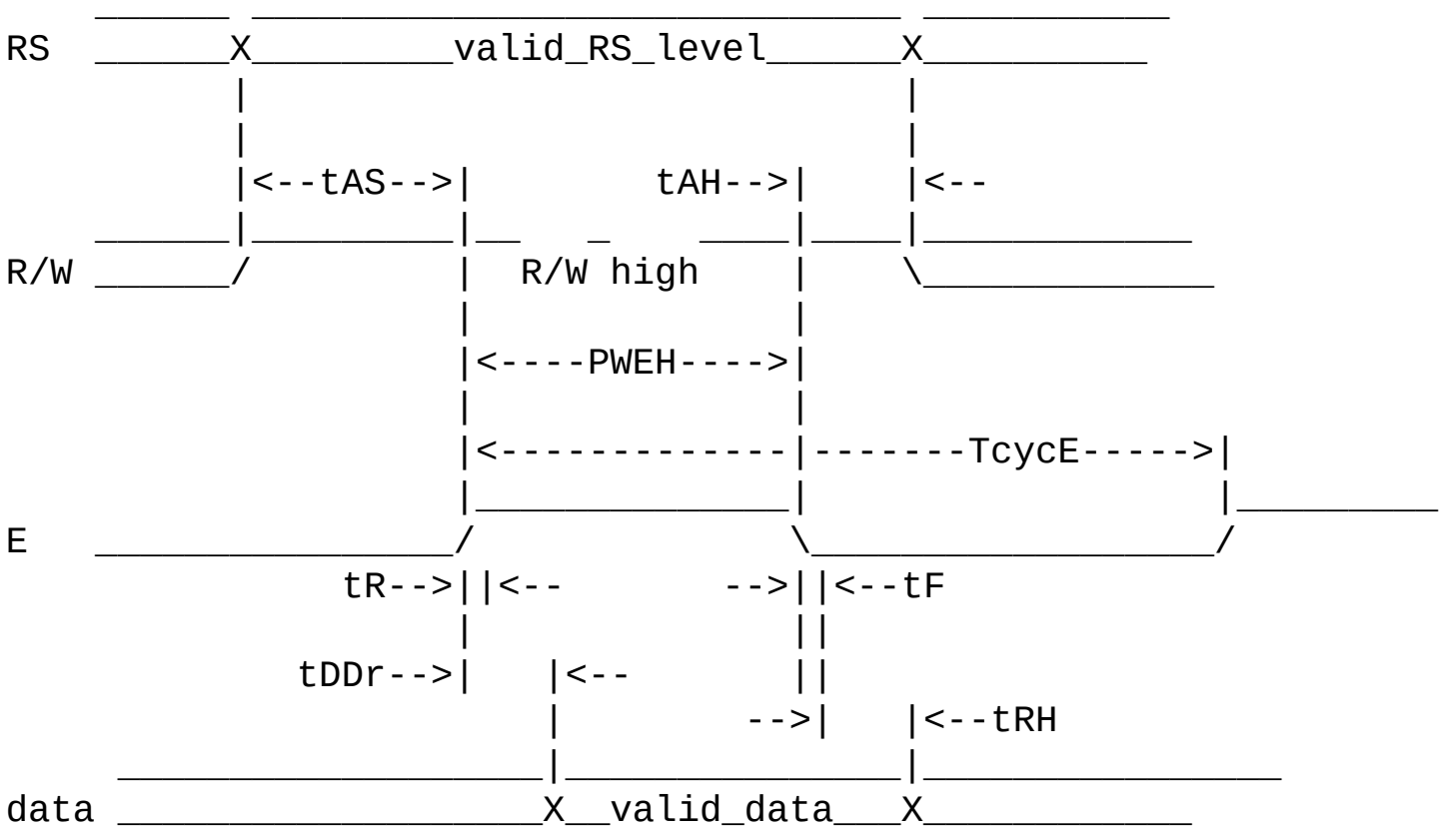
Execution times: Clear display 1.64ms; home cursor 40 us to 1.64ms depending on how far display is scrolled; all others 40us, except read busy flag which is complete in a single enable cycle (or two cycles in 4-bit mode), and character generator ram reads and writes which should be separated by 120us delays. These execution times mean that after an operation, the CPU must do Busy Flag checks until the BF (bit 7) is 0, or, when the connection to the module from the CPU is write-only, wait more than the execution time before the next operation. These times are usually strict, LCDs used in a write-only configuration should provide the specified delays. Some speed improvement might be realized by utilizing busy flag reads instead of delays.

WRITE:





READ:



TcycE

Enable cycle time (1000ns min)
Operation cycle time cannot be less than 1 microsecond

PWEH

Enable pulse width, high (450ns min)
Enable pulse must be at least 450 nanoseconds long, no maximum length

tRE

Enable rise time (25ns max)
Enable line must change state (L->H) in less than 25ns

tFE

Enable fall time (25ns max)

Enable line must change state (H->L) in less than 25ns

tAS

Address setup time (140ns min)
Register Select and R/W lines must be valid 140ns before enable pulse arrives

tAH

Address hold time (10ns min)
RS and R/W must be valid at least 10 ns after enable goes low

tDDR

Data delay time (320ns max)
When doing a read, the return data will be valid within 320ns of enable going high

tRH

Data hold time, read (20ns min)
When doing a read, the return data will be valid at least 20ns after enable goes low

tDSW

Data setup time (195ns min)
When doing a write, data on lines bit7-bit0 (or bit7-bit4 in 4-bit mode) must be valid at least 195 ns before enable goes low

tWH

Data hold time, write (10ns min)
When doing a write, data on lines must be valid for at least 10ns after enable goes low

Generally, there are no max time requirements on the user except Enable rise and fall times. An LCD module can be driven with just toggle switches for data, RS, and R/W, and a debounced pushbutton on the enable line.

3.5) Memory map

Character generator RAM appears to reside at locations 64-127 and display RAM seems to be at locations 128-255. This is an artifact of the control scheme. They are actually separate noncontiguous blocks of RAM.

3.5.1) Custom Characters

The display has 64 bytes of CG RAM, which supports 8 user-definable characters, each defined as a 5X8 character in an 8X8 block. The 5X8 region is right-justified. Note that in 5X10 matrix mode, only 4 user-defined characters are supported, using 11 bytes each.

```
CG Address      CG Data
==========      ===============
                D7-----------D0
x+0             x x x 1 1 1 1 1       *****
x+1             x x x 1 0 0 0 0       *
x+2             x x x 1 0 0 0 1       *   *
x+3             x x x 1 1 1 1 1   =   *****
x+4             x x x 1 0 0 0 1       *   *
x+5             x x x 1 0 0 0 0       *
x+6             x x x 1 0 0 0 0       *
x+7             x x x 0 0 0 0 0       (cursor line)
```

Where "x" is the base address for the character: CG0=0, CG1=8, CG2=16, CG3=24, CG4=32, CG5=40, CG6=48, CG7=56.

As you can see in the above figure, each byte of CG data represents 1 row of pixels in the character, with D0 being the rightmost pixel, and D4 being the leftmost (the upper 3 bits are not displayed). The first address for a character is the topmost row, while the 7th is the bottom. (The 8th is ORed with the cursor underbar). A '1' in any pixel position becomes a dark pixel on the display.

CG RAM occupies a separate address space from the data display (DD) RAM. One must set the (first) CG address before writing any CG data. Each write automatically increments/decrements the CG address pointer to the next location. Remember to set the display back to DD addressing mode before trying to write characters to be displayed.

As an example, we'll define character #3 to be the above symbol by first reading the DD address, writing the CG address to the start of the character's RAM, writing the successive rows of pixel data, then restoring the addressing mode back to DD addressing (which requires the DD address we saved at the beginning.

```
RS R/W D7----D0
== === ========
0  1   (baaaaaaa)      Read current DD address
0  0   01011000        Set CG RAM address to char #3
                                 (3x8=24=11000b)
1  0   00011111        Write top row of pixels
1  0   00010000                :
1  0   00010001                :
1  0   00011111                :
1  0   00010001                :
1  0   00010000                :
```

```
1  0   00010000          Write bottom row of pixels
1  0   00000000          Write cursor row (descender)
0  0   1aaaaaaa          Restore DD mode (& address)
```

Note: There should be a 120 uSec delay between successive reads or writes.

(Section 3.5.1 by Doug Girling)

See a clever example of using custom characters to create doubleheight characters in Ian Harries' tutorial at http://www.doc.ic.ac.uk/~ih/doc/lcd/double.html

---

3.5.2) Addressing Display RAM

In the list below, "line 1" is the topmost line, and "line n" is the bottommost line. On each line, the leftmost character has the lowest address, and addresses increase to the right. Also, DD RAM addresses are shown without the 80h mode bit set. Add 80h to the character address (set bit 7) to make the Display Data RAM Address Set command.

- 16x1 module is arranged as two 8-character lines side by side.

  ```
  "Line 1" (left)  addresses are 00h to 07h  (0 to 7)
  "Line 2" (right) addresses are 40h to 47h (64 to 71)
  ```

  As you write characters to the module, the cursor will automatically increment until you get to the 9th character--you have to move the cursor to address 40h before writing the 9th character on the 1x16 module.

  ```
  |00|01|02|03|04|05|06|07|40|41|42|43|44|45|46|47|
  ```

  Rarely, 16x1 modules are formatted as one line, 00h to 0Fh.

- 16x2 module is two lines by 16 chars

  ```
  Line 1 addresses are 00h to 0Fh  (0 to 15)
  Line 2 addresses are 40h to 4Fh  (64 to 79)
  ```

```
|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|
```

- 20x1 module

```
     Line 1 addresses are 00h to 13h  (0 to 19)
```

```
|00|01|02|03|04|05|06|07|08|09|0A|0B|0C|0D|0E|0F|10|11|12|13|
```

- 20x2 module

```
     Line 1 addresses are 00h to 13h  (0 to 19)
     Line 2 addresses are 40h to 53h  (64 to 83)
```

```
|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|
```

- 20x4 module

```
     Line 1 addresses are 00h to 13h  (0 to 19)
     Line 2 addresses are 40h to 53h  (64 to 83)
     Line 3 addresses are 14h to 27h  (20 to 39)
     Line 4 addresses are 54h to 67h  (84 to 103)
```

```
|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|
```

- 40x2 module

```
     Line 1 addresses are 00h to 27h  (0 to 39)
     Line 2 addresses are 40h to 67h  (64 to 103)
```

[table omitted, similar to above]

The full 80 bytes of display RAM exist no matter how many characters appear on the display. These extra bytes can be typed on when display window scrolling is enabled, or they can be used to store other information--external data RAM for the CPU, if you like. No RAM exists in the 28h-3Fh and 68h-7Fh blocks.

Chapter 4) Initialization

Modules with Hitachi controllers will properly self-initialize if Vcc rises from 0 to 4.5v in a period between .1mS and 10mS. I suppose an RC circuit would be needed to keep powerup rise time as slow as .1ms, so the manual initialization will be required in most applications. If you do use auto-initialization, it will come up in this mode: 8-bit interface, 1/8 duty cycle (1 line mode), 5x7 font, cursor increment right, no shift. On most displays, you want to switch to 1/16 duty cycle (2 line mode) because for all but the 20x1, there are two logical lines as the controller sees it. If you have a 5x11-size character dot matrix module, you'll want to switch to the 5x10 font as well (the 11th line is the cursor). See the Instruction Set section for details on the commands.

4.1) Initialization for 8-bit Operation

1. <POWER ON>

2. <Wait 15ms>

3.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   1   1   x   x   x   x

x=don't care
Set 8-bit mode
(attention!)
```

4. <Wait 4.1ms>

5.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   1   1   x   x   x   x
```

```
Set 8-bit mode (attention!)
```

6. <Wait 100us>

7.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   1   1   x   x   x   x

Set 8-bit mode (attention!)
```

8. <Wait 4.1ms>

9. <Busy Flag cannot be checked before this point>

10.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   1   1   1   F   x   x

8-bit operation
1/16 duty cycle
F=font,
1 for 5x11 dot matrix
0 for 5x8 dot matrix
```

11. <Wait 40us>

12.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   0   0   1   0   0   0

Display off,
cursor off,
blink off
```

13. <Wait 40us>

14.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   0   0   1   1   C   B

Display on,
C=1 for cursor on
B=1 for blinking cursor
```

15. <Wait 1.64ms>

16.
```
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0   0   0   0   0   0   0   1   I   S

Set entry mode
I=1 for cursor move to the right
I=0 for cursor move to the left,
S=1 for shift display,
S=0 for no shift
```

17. <Wait 40us>

18. <INITIALIZATION COMPLETE>

Please check attribution section for Author of this document! This article was written by **filipg@repairfaq.org** [Feedback Form] [mailto]. The most recent version is available on the WWW server http://www.repairfaq.org/filipg/ [Copyright] [Disclaimer]