

## Hughes 0488 LCD Driver

Written by Paul Robson ([paul@robsons.org.uk](mailto:paul@robsons.org.uk)) 4<sup>th</sup> January 2014.

This document should be read in conjunction with the 4 page Datasheet available from <http://mbmicrovision.blogspot.com>. Microvision Circuit Diagrams were traced and created by Dan Boris. Thanks to Jeff Salzman for explaining the Polarity issue.

### Introduction

The Hughes 0488 LCD Driver is a single chip 40 pin driver which is designed to drive a 16 x 16 pixel LCD Display, such as the Microvision. I don't know if it has any other uses than the Microvision.

It is unusual in that unlike modern LCD drivers it has to be continually refreshed at about 30-50Hz. You cannot just set a pixel on, you have to do it over and over again, in code.

### External Interface

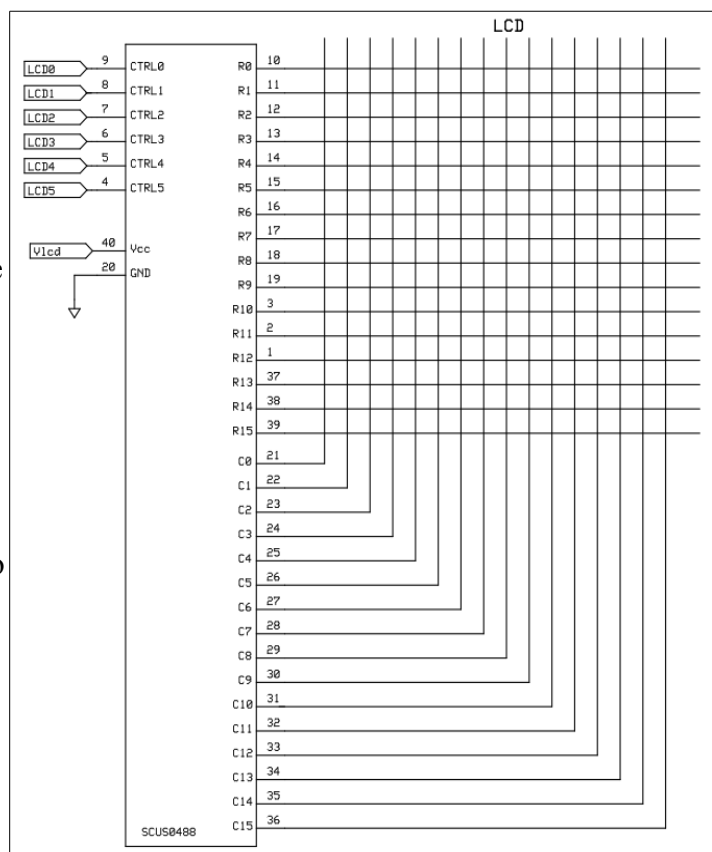
This picture shows the basic connections of the Hughes 0488 (thanks to Dan Boris for tracing this circuit).

The wiring is actually very simple. There are 16 row connections (R0-R15) and 16 column connections (C0-C15). Anywhere there is a logic '1' on both columns, the LCD screen is turned on (which on an LCD means darkened, e.g. black rather than light grey)

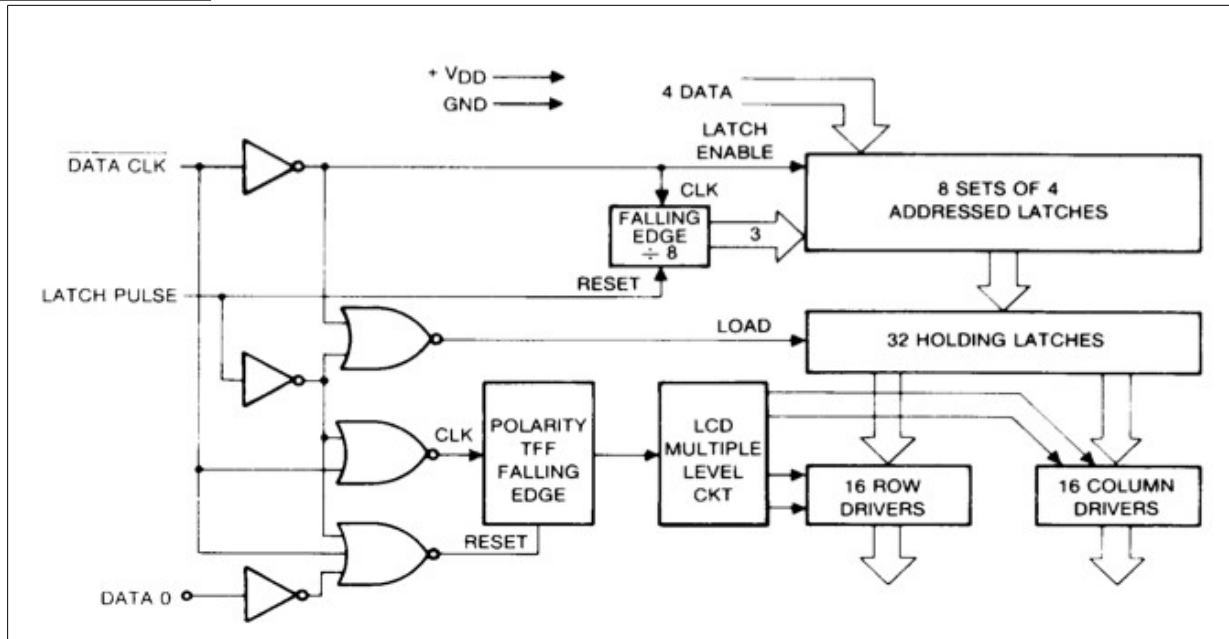
The other pins (besides power) include four data pins (CTRL0-CTRL3 on here, D3-D0 on the Hughes Datasheet) and two control pins. Data Clock is an active low pin which is used to clock data into the LCD Driver, and Latch Pulse which is used to 'copy' the latches to the screen (see later)

The most obvious consequence of this wiring is that LCD display generation has to be done a row or a column at a time. So to put a pattern on row 0, R0 is set to 1 and C0-C15 are set to 1 to turn the pixel on. If a row is empty (e.g. all pixels off) there appears to be no need to do anything on it e.g. it is off by default.

You can only do multiple rows if they have the same pattern (e.g. if R1 is set to 1 it will display the same pattern as R0 in the above example). It is possible, though probably not advisable because it's conceptually more complicated to do it in columns.



## Internal Hardware



Understanding this diagram isn't too hard. Forget most of it. The only things that really matter are the right hand side (the latches and the row driver) and the box marked "Falling Edge".

The basic idea is straightforward. The '16 row drivers' and '16 column drivers' correspond to the connections on the previous page (e.g. C0-C15, R0-R15).

These are fed from a set of 32 'holding latches', which are in turn set from the 32 'addressed latches' – these correspond to the 32 row/column drivers on a one to one basis.

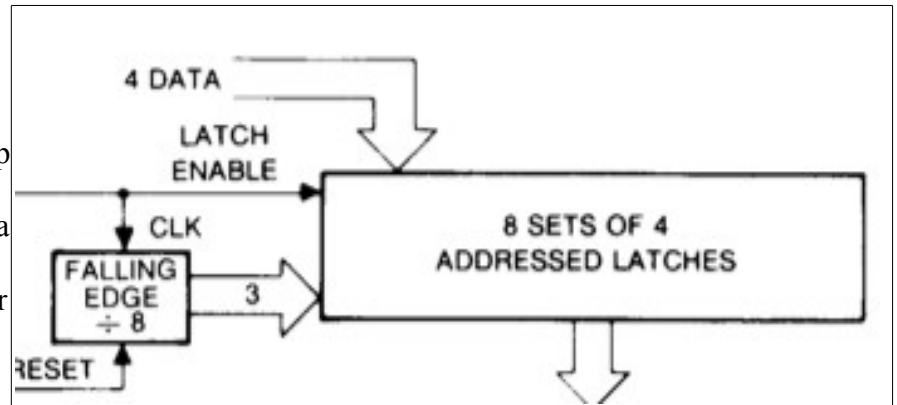
What happens in brief is that the 'addressed latches' at the top are loaded with the row and column bits you want to turn on. Suppose say you wanted to turn on Row 2 Column 3, then your latches would contain

```
ROW 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000
COL 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

for example (you can turn on multiple LCD pixels in the same row or the same column). Your program loads this into the addressed latches (using the !DATA CLK line). When the address latches are loaded with the pattern, then you transfer these to the holding latches (using the LATCH PULSE line) whereupon that pattern will appear on the C0-C15 and R0-R15 lines.

## Loading the Addressed Latches

Loading the addressed latches is relatively simple. There is a 3 bit counter (the "Falling Edge" box) that determines which latch group is to be written to – there are 32 latches (16 row, 16 column) and a 4 bit bus so that is 8 writes ( $8 \times 4 = 32$ ) – the "falling edge" counter determines which group is to be written to (1 of 8, hence the '3' in the arrow leading to the latches)



The groups are specified on the last page of the datasheet (section 10) but it is basically all the rows then all the columns in 4 bit right to left format (e.g. the obvious one !).

(Note that the datasheet uses R1-R16 and C1-C16 and Dan's circuit uses 0-15)

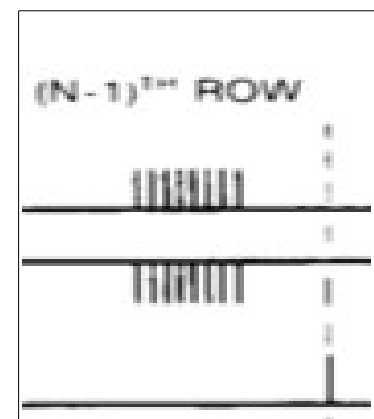
Writing to the latch is done whenever the !Data Clock line is logic '0', and the clock itself is incremented on a falling edge on the Data Clock line *e.g. on a rising edge on the !Data Clock line – there is an inverter to the left of this circuit you can see on the previous page – so to write to a latch, you put the data on the data lines (TDO on a TMS1100), set the !DataClock line to logic '0' and reset it to '1' (it's normal state)*

In the above diagram this data is copied in the normal state (e.g. !DC = 1, Latch Enable/Clk above = 0, see Note 1) and when you set it to logic '1' this sets this internal line to logic '0' thus clocking the 'falling edge' counter.

The Falling Edge counter is reset when the Latch Pulse input is high. This latch pulse also transfers the contents of the addressed latches to the holding latches when Data Clock is high. This explains this sequence from the waveforms in the datasheet.

This is clocking in 8 data bits. The top row is the data line D0, the second row is the !Data Clock line. There are 8 1->0->1 transitions on the !Data Clock line which clock in 8 lots of 4 bit data into the addressed latches.

The bottom row is the latch pulse, which is normally zero – setting this to logic '1' does two things – firstly, it resets the falling edge counter, moving the addressed latch back to the first one, secondly it copies the addressed latches to the holding latches.



## The Polarity Stuff

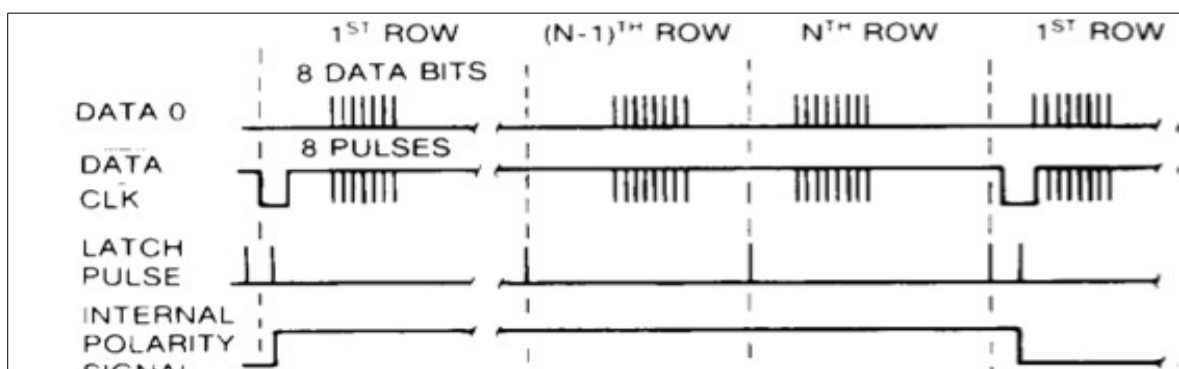
The remainder of the diagrams and comments are taken up with the polarity issue. 7B on the last page says “the polarity signal should be a square wave of exactly 50% to keep DC off the display”. It's not clear what is happening, Jeff Salzman described it to me thus:

*“The DC build up is most likely an electrostatic charge present in or within the glass and liquid crystal of the display. This will tend to cause the liquid crystal to "permanently" polarize.*

*Since LCD components work best with a +/- voltage swing at a 50% duty cycle on each swing, DC build up is normally not an issue in most applications. The constant polarity reversal collapses any potential DC build-up that might occur.*

*I'm guessing that there were a lot of "shortcuts" taken when designing the LCD system (and physical display) of the Microvision in that the display itself is prone to DC build up if the polarity reversal isn't done fast enough”*

So it stuffs the display up if you don't do this 'polarity switch'. The good news is that the controller chip pretty much does the hard work for you. All that is required is that at the start of each 'frame' - e.g. a complete display refresh, all of it, not just one column or one row, you instruct the display controller to 'invert the polarity'. This is done by setting !Data Clock to Low, then clocking the latch pulse (e.g. 0->1->0) , then setting !Data Clock to High, as can be seen in the waveforms.



At the start of each drawing frame (in this example, the first row) the polarity switch signal is applied – this is the 'wide' dropping of the !Data Clock line, which you will note, has a corresponding 'Latch Pulse' pulse, which can be switching the internal polarity below.

### The odd timing Issue and associated possible error

Unfortunately, combining both of these – the polarity reversal and the data write is not quite as simple as it sounds.

The alert reader will have noticed that the patterns on the two '1<sup>st</sup> Rows' are different. This is because the Latch Pulse resets the 'Falling Edge' counter to zero, but when the Data Clock line goes from '0' to '1' this creates a falling edge on the Clock input to the falling edge counter, so it increments by one.

Hence in the second first row (lower picture), the first data bit is actually clocked in on Data Clock 0→1 transition.

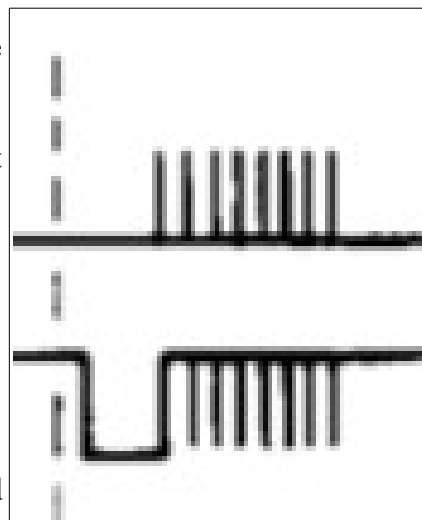
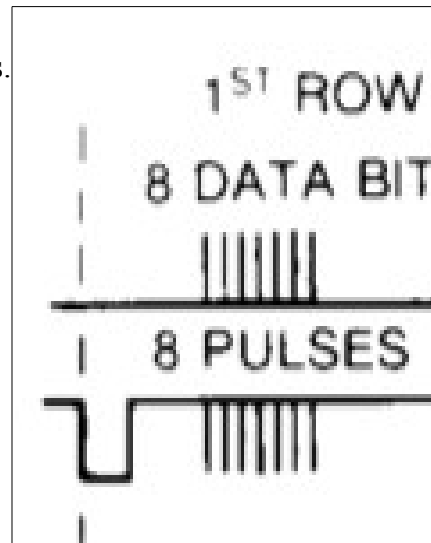
This is an error in the datasheet, IMO. If you look closely at the first “first row” (upper picture) there are actually only *seven* data bits where it says “8 data bits”, so this first data transition is missing – it should be like the other “first row”. There are 8 pulses if you include the long one.

This will have to be dealt with programatically. I will write some code showing how to do this at a later date.

In outline though, there needs to be two similar but different routines 'write first row' and 'write other rows'.

The write first row routine will drop DataClock low, do the latch pulse to reverse polarity and the drop through to the 'write other row' code.

This would load the data onto the data bus, drop data clock to low (which if called via 'write first row' will have no effect because it already is), then set it to high (which would cause the falling edge counter to increment), repeat this 7 further times, and then do the final latch pulse.



### Timing Pulses

A slight concern is that the latch pulse width is given as a maximum of 500 nanoseconds.

The TMS1100 and Intel 8021 are pretty slow, so that to generate any pulse at all takes 40 us (a SETR and RSTR instruction) because the TMS1100 only operates at 50Khz Instruction Rate (an 8021 is about twice as fast). It actually isn't possible to generate a pulse of 500 nanoseconds. I am guessing that this is another error and that the Latch Pulse Width (tPW) should be a minimum of 500 nanoseconds – I cannot see, from the circuit diagram why it should matter whether its 0.5Mhz or 0.5Khz as the signal appears to be connected to counters and flip flops.