How to interface with a 32x32 RGB LED matrix tile.

By Colin Funnell

Which panels are these?

Everyone loves a blinking LED – it's the first thing you try on any micro-controller just to show signs of life. With the advent of large LED signage and video walls, mass-produced LED tiles are now available to buy and blink instead in creative ways. It is great that they are generally available, normally from hobby electronics retailers. What is not-so great is the lack of information to use them. You'll occasionally find a software library or someone's personal project on-line, but very little on how the tiles work so you can make it work *your way.*

This tech tip won't go into the basics of multiplexing but will show the type of signal driving these panels require. In particular, we'll be looking at a common 32x32 matrix panel.

Figure 1 - A typical RGB LED matrix tile

NOTE: These tiles are expected to be driven in a multiplexed manner which reduces LED brightness. When experimenting, be careful of very bright LEDs.

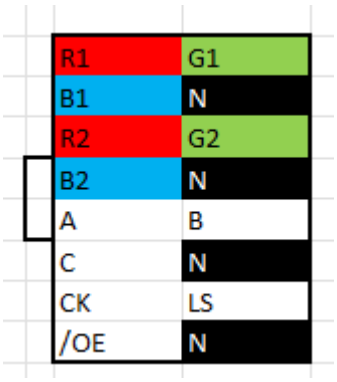
NOTE2: Some tiles use ICs with "anti-burn" features. LED output can be disabled if the A signal is not toggled within a timing window.

The basic connections

Let's start with what the tile allows us to connect to. Generally, there is a 4 pin power connector and a 2x8 way boxed header. This is sometimes referred to as a HUB75 connection.

Power is 5V and will require a couple of Amps. The LEDs are very bright and the larger panels require more power.

The boxed header has the control signals. There is one header for signal inputs and another for signal outputs. This allows tiles to be chained one after the other, increasing the signal 'string' length. If there are arrow markings on the PCB, they point showing the path from input to output.



R1	G1
B1	N
R2	G2
B2	N
A	B
C	N
CK	LS
/OE	N

Figure 2 - Typical tile connector pinout

Most LED tiles are similar but not necessarily identical to the above. The labelling may vary as well as pin-out. For the same size/type of tile they seem to be consistent. As a 5V system it is best to drive these signals with 5V logic to avoid problems, but experiments have seen 3.3V logic working fine.

N = Ground

RGB1 = Serial data for RGB LEDs on a string.

RGB2 = Second series RGB bus on a different string.

ABC = Block display selection. Selects a particular LED groups to drive.

LS = Latch Strobe. Copies the serial string data into a latch for LED driving output.

CK = Serial data clock (rising edge). Clocks the RGB data through the LED driver shift registers.

(/)OE = Output Enable (Active Low)

The basic tile architecture

There isn't control of individual LEDs, rather, it is the enabling of groups shared by a set of LED drivers that makes these tiles cost effective. The LED drivers load their data as a shift register. Data is shifted on the clock rising edge and when connected together form a long LED driver string. There is a data stream for the drivers that connect to the red elements of the LEDs, others for green and blue. The CK clock is common to them all. Many LEDs are connected to each LED driver line. A sub-set of these are enabled by the 3:8 decoder under control of the ABC inputs to control LEDs individually.

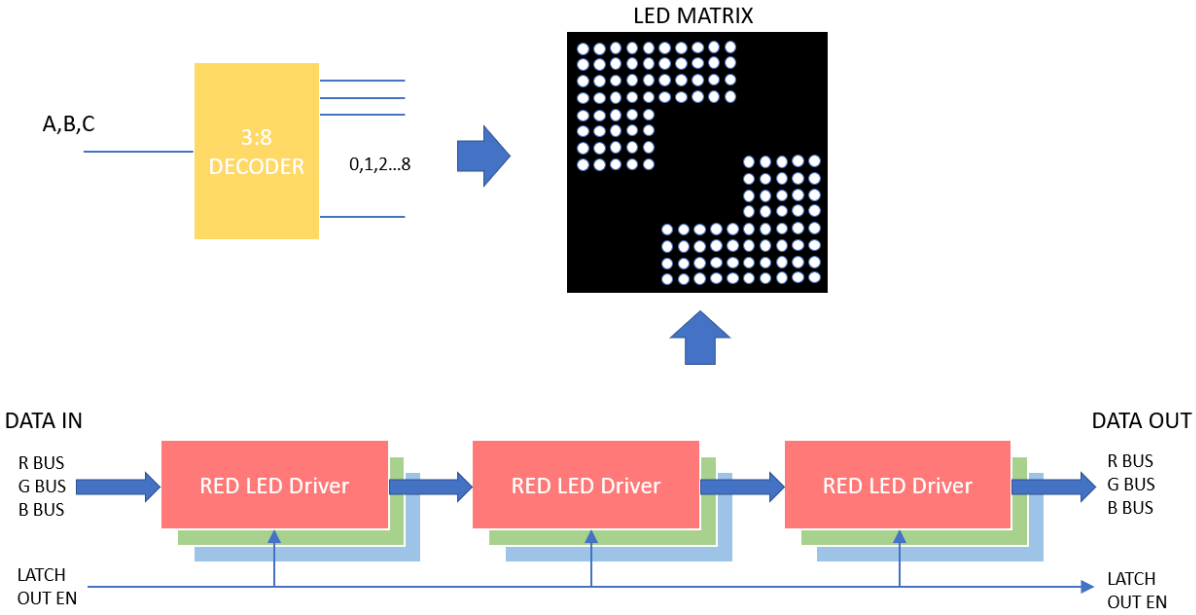


Figure 3 - Basic tile architecture

As we will see, the combination ABC selection and LED drivers is not going to be a simple relationship.

Datasheets can be found for the devices used on common tiles but may be unclear after translation.

Driving a single tile

The basic upshot of driving a tile is that we must conform to the approach that all the drivers of tiles seem to adopt – shifting our data along, capturing it with a latch, then enabling the LED drive output. The MBI5026 is one such device. The waveforms show how data is clocked but is not shown until latched and enabled.

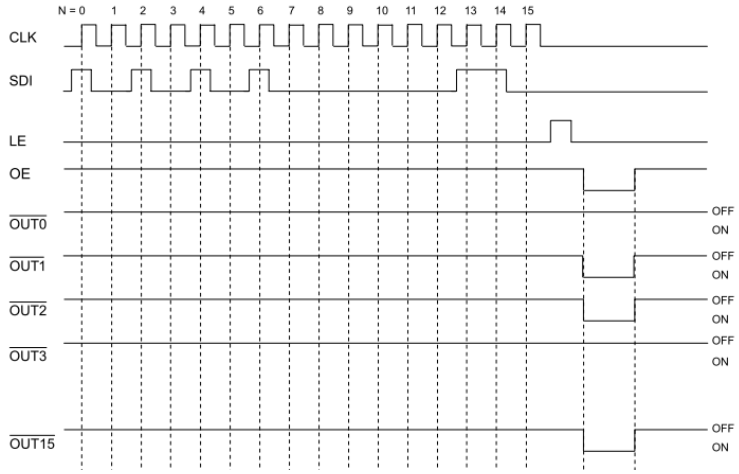


Figure 4 - Timing for an MBI5026 LED driver device

This, along with the general tile control activity can be summarised with the following waveform:

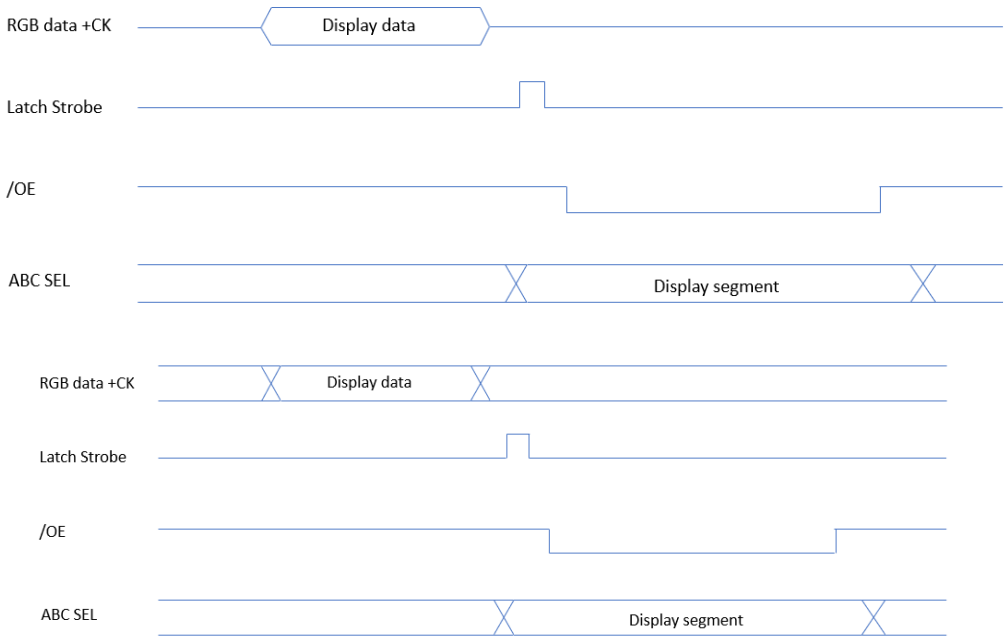


Figure 5 - Generalised waveform

RGB data and clock don't have to go tri-state but their general activity is shown. To make sure the LED patterns have enough 'on' time, /OE should be asserted as much as possible. The latch mechanism allows new data to be shift-loaded while this is occurring so we don't have to wait before we can show something. We can squeeze-up multiple cycles together. Provided the LED data loading time isn't too slow, it can fit in the /OE period.

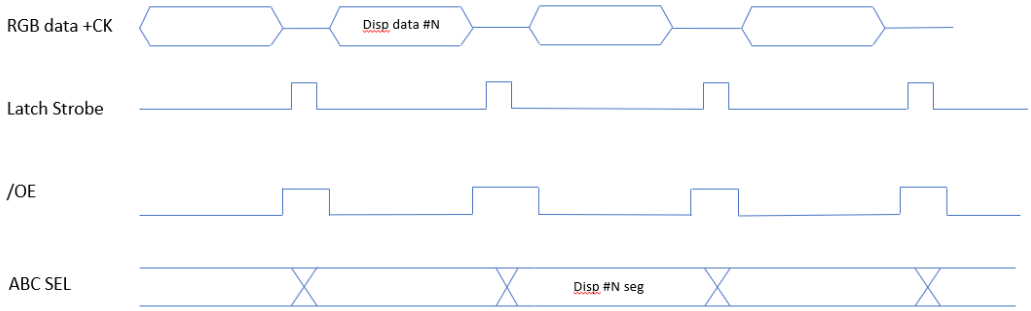


Figure 6 -Multiple segments waveform

Note that the ABC value for the display segment is for the LED data previously issued before the latch.

As a system where you are updating part of the display whilst showing another section, it needs continual driving to show the entire display.

Where is my pixel?

This is one of the harder aspects of controller design – matching the expected pixel position with the driver waveform. There are a number of connection and timing aspects which determine this:

Multiple tiles and busses

Chaining tiles extends the "length" of a serial string. Multiple busses can expand with another dimension. Complex chains can be made but they also make control more complicated.

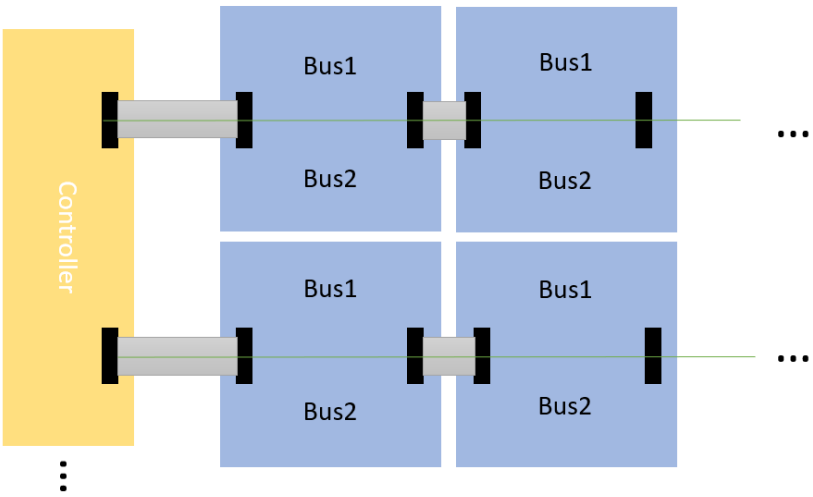
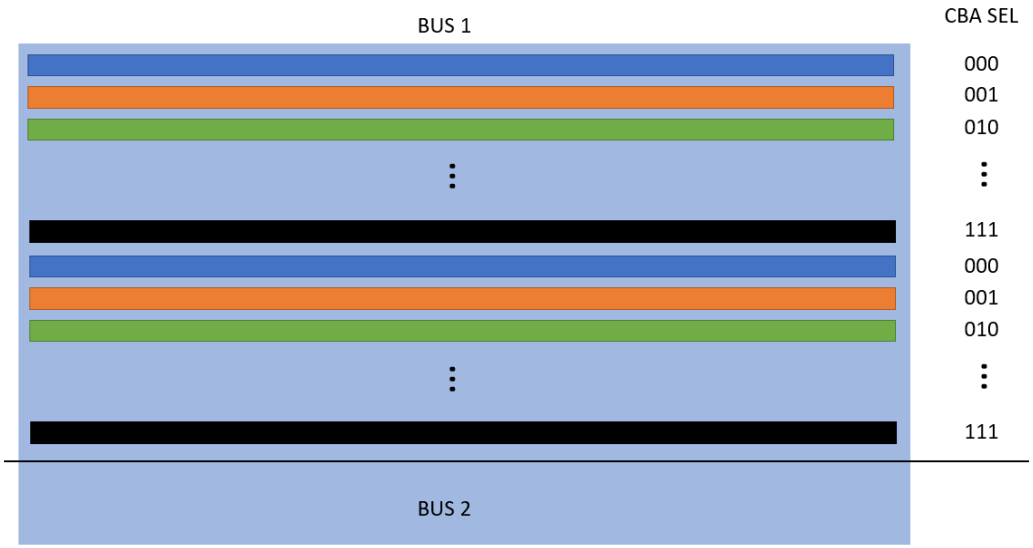


Figure 7 - Arrangement of multiple tiles for larger displays

Segment Multiplex

The ABC signal decoder helps direct the LED driver strings to certain sets of LEDs as part of the multiplexing matrix. For larger tiles, a segment may consist of more than one 'line' the string extends onto. For the 32x32 tile the LED string goes over two lines, giving a repeated pattern-as shown below:



With the ABC sel and /OE lines being cascaded along multiple tiles, the same segments will be illuminated on all tiles in the chain. The second RGB bus on a tile shares the ABC and /OE control lines but will have different data shifted onto it.

String position

Individual RGB data bit *positions* applies to the same LED. i.e. The data bit position for a particular red LED element applies to the green and blue elements on the same LED pixel, when driven on the other colour busses.

The position along the shift register string is not continuous. It performs a zig-zag pattern due to two lines being part of the segment.

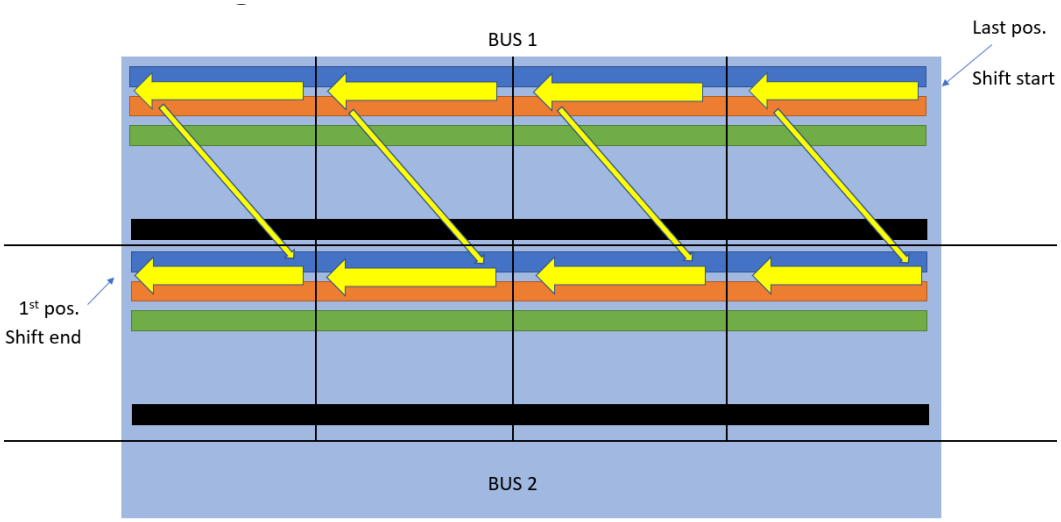


Figure 8 - Shift pattern along a string of LEDs

The yellow arrows show the order of the shift register. The entry point of the shift will be the last set of pixel data loaded. The RGB data will be shifted out of the last driver to the tile's output boxed header. Connecting multiple tiles allows for longer strings to be constructed.

Colour depth

It must be noted that the LED tile only deals with turning RGB LEDs purely on or off. This gives a range of only several colours. This may be fine for your application, but more colour capability wouldn't go amiss.

To add more colour resolution, we need more multiplexing. In a similar manner to Pulse Width Modulating (PWM) a single RGB LED to produce a range of colours, the same can be done here. The difference that each step of PWM control is now performed over a block of pixels instead of one. Whole or chunks of an image need RGB LEDs turned on or off quickly enough to give the illusion of PWM control without flicker. A simple analogy is a "flick book" of a static image where you only have red, green and blue pens. You can't shade with them but by drawing and *not drawing* with them on different pages you can achieve the effect of subtle colours.

Obviously, the greater the number of PWM steps, the greater the signal rate needed to update the display. Don't go overboard… for example, a PC display with 24-bit colour has 8-bits per

colour channel, giving 255 steps to each R, G, B element. That is a massive speed increase to handle.

…and all of this needs to be done repeatedly and quickly enough to avoid the display flickering.

In summary

The above should be enough to get some basic control signals running an LED tile.

The simple control of a single tile is well within the capabilities of a low-speed microcontroller. Driving groups of tiles together or for greater colour depth requires a proper architecture to be engineered. Meeting particular display rate timing might not be required in simple cases and the controller could just run "as fast as it can". More professional uses would need a better control system being designed.

Even if your system isn't the greatest, biggest, highest resolution display in the world, it's always more impressive when there are lots of LEDs to control. To be honest, making LEDs blink is always a bit of fun but now you can do it in *style*.

For more information visit our website: www.hitex.co.uk or get in touch: info@hitex.co.uk. You can also connect with us: LinkedIn