White Paper

Migrating to Advanced Displays

Renesas Electronics America Inc.

July 2015

Steve Jobs did a great job at Apple, but he also changed the world. A simple 7-segment display is not enough for even the most simple of applications today; customers are asking for bigger displays with more impressive graphics. Marketing teams are therefore asking engineering teams to implement this on the next generation of products. This is true for many applications, such as coffee machines, ovens, factory automation and household boilers, and is naturally presenting a new and interesting challenge for engineering teams. So the simple question is how to add a display into your application?

Let us first have a look at what different options you have in picking a display. There are lots of TLA's (three letter acronyms) used in the industry which are actually very simple but can lead to a bit of confusion.

Everyone in the engineering community is familiar with the Liquid Crystal Display (LCD) concept; even if they have subsequently moved to the dark side and are now working in "sales" they will likely remember the calculator that they once had to use for real work, that had an LCD screen on it.

The basic concept is very simple. A layer of liquid crystals is arranged between two polarized layers (at 90 degrees to each other), such that without the liquid crystal, no light would pass through at all, but the crystal is arranged in a kind of helix pattern which rotates the polarization of the light between the two polarized screens, meaning all the light can pass though. However, when an electric charge is applied to the liquid crystals they unroll or straighten out, thus no longer rotating the light and will appear black.

This "twisted" orientation is the most common configuration inside and LCD, and this is where the TN-LCD (Twisted Nematic Liquid Crystal Display) comes from. "Nematic" is just a way of describing the physical state of the liquid crystal.

A color display can be built up using an LCD by using three separate LCD cells per pixel and applying a red, green or blue filter to each one. The pixels are then lined up in columns and rows, and by applying a voltage to the column and grounding the row, a specific pixel can be turned on or off. The problem with this approach is, however, that for larger screens, when several pixels in one column and in another row are switched concurrently, there is a long delay as the charge propagates through the matrix, as well as a poor contrast as the charge is distributed across this matrix.

This problem is somewhat overcome by using the STNLCD module. The STN in this case stands for Super Twisted Nematic screen. In an STN screen, the 90 degree rotated liquid crystal is replaced by a "super-twisted"— or typically 270 degree twisted— liquid crystal.

This is, however, still building up a matrix of charged columns and grounded rows, also known as a passive, which has inherent limits as described before. Moving to an active control gives a much better contrast and a much faster response time. In an active LCD system, there is a dedicated transistor for every single LCD cell, and as such, the transistor controls the switching of that cell or pixel. This gives rise to the phrase "TFT-LCD" which is then an LCD screen controlled by a Thin Filmed Transistor, and is the most common screen used in most display applications today.

The alternative, though not as popular a display technology in this market, is the OLED display. This is essentially a matrix of LEDs, with one OLED per pixel. They are lower power, lighter and can be put on flexible surfaces when compared to their LCD-based counterparts, however, the expensive manufacturing process and the somewhat limited lifetime of the blue component in the displays has delayed their full-scale introduction into the market.

Of course, there are a few other alternatives that are less popular but should nonetheless be understood, and these are listed below:

- EPD – the so-called e-paper used in e-readers and in some watches, but not yet suitable for the embedded, non-consumer space.
- LCOS – Liquid Crystal on Silicon, used for "near eye" or projectors but have not taken off in the industrial market as yet.
- PDP or plasma – used in older, larger screens but are quickly being replaced by LCD or LED screens.

Now that we know how the picture is actually displayed on the screen and what the technology looks like, so let us have a look at how the screens are actually driven. For smaller screens, often using a passive control system, it is common to see new COG (Chip on Glass) technology coming through, where the driver IC for the LCD is actually integrated onto the glass of the panel. This can provide a good cost reduction for smaller screens but is yet to make the breakthrough into larger color screens, the likes of which the aforementioned Steve Jobs would have put onto his smartphones or other gadgets. These screens for the most part include an additional PCB, which contains the control mechanism for the screen and can additionally contain a controller for the touch interface.

There are then two further options available in terms of displays. There are so-called display modules, and there are standard displays. A module is just as you'd expect— a complete module with all the memory required to save the picture data which is being displayed on the screen and normally a simple SPI interface. We will not go into these in much more detail, but suffice to say that they are slightly more expensive than a standard display but offer a much simpler solution for the novice user.

There are two standard interfaces to a standard display— RGB signals or LVDS signals. It tends to be that displays over a certain size (about WVGA) will offer an LVDS interface and smaller ones will offer an RGB interface. There is of course no hard and fast rule here but the bigger the screen the more likely it will be to have only and LVDS interface. RGB is essentially a parallel interface whereby each color (Red, Green and Blue) is represented by a parallel bus.

Thus for a 24-bit colour display there will be 24 "data" bits. This is the most simple of interfaces as there is a standard one to one transfer of every pixel data on the bus to the way it is stored in RAM. There are several different notations of RGB standards, such as RGB565 of RGB666, which simply denotes the number of bits taken for each color. In RGB666, there are 6 bits reserved for each colour, and it is therefore an 18 bpp colour. In addition to these data signals there are also the clock signals to synchronize the panel.

A panel clock (or pixel clock) sets the pace for the whole interface and subsequently the data transfer occurs. There is then an Hsync clock, or horizontal synchronisation clock, which indicates after a number of pixel clocks when to jump to the next line.

Then at an even slower frequency, there is the Vsync signal (vertical synchronisation) which in turn indicates when all the rows have been written to, and it is possible to then start the next picture or frame.

Of course, the bigger the display, the higher the speed required for the pixel clock in order to meet the refresh rate of the screen. As external signal frequencies get higher, the risk of signal corruption also increases. For this reason, larger screens now tend to use an LVDS interface instead of the standard parallel RGB. LVDS stands for Low Voltage Differential Signal. LVDS technology is used in many applications where signal integrity is very important, especially at high frequencies. The LVDS signal uses a two-wire interface (per channel) and has a common voltage (normally 1.2 V). Then to create a “high” signal, the voltage on one line is raised by 100 mV and the signal on the second line is lowered by 100m V. This allows for low power, high frequency, high-reliability signals to be transmitted. In a display, there are typically four LVDS channels. These channels are used for the red, green, blue and clock signals in turn, and then the data is transferred serially rather than in parallel.

Now we know how the screens are set up, let us move to the other side of the application and see how they should be driven. Let’s look first at how the images are stored in memory. We are all now familiar with our holiday photos being stored as a JPEG on our PCs at home. Sadly, this is not the format that is used; the image is saved as a raster image or a bitmap. This is, of course, significantly larger than the JPEG that you use for your holiday photos, so let’s have a look at the way that these images are actually saved in RAM and how much of it you actually need.

As we said, the basic picture is stored as a bitmap, whereby every pixel in the picture is stored and represented by unique data. There is no data compression, like in other formats. It could be a 16-bit or 24-bit color depth; for a 24-bit color, that each pixel would be represented by 3 bytes. You can then immediately see that this means a lot of memory will be used. For example, a VGA screen of 640 by 480 pixels would have 307 K pixels, and as such need about 900 KB of data per image on the screen.

Sadly, however, the RAM usage story is not over yet. A typical GUI application will be made up of several picture layers. These layers would be then displayed on top of each other. For example, one layer could be the corporate background image, and the next layer might be a frame around the outside of the picture with some data displayed on it, such as the temperature and the time. A third layer could then be a graph showing real-time measured data in your application. The reason that you store these pictures in different layers, and therefore in different areas of the RAM, is so that you only need to change one small picture rather than re-calculate the whole image. If you needed to re-work the entire GUI every time that the graph was updated or the temperature changed, it would just take too much CPU power.

These layers are then combined together, either by hardware acceleration or by software, using a number of different mechanisms. The two key concepts here are alpha blending and chroma keying. Alpha blending defines what is known as an alpha channel, which is an additional 8-bit value added to the 24-bit color signal for every pixel. This alpha value defines the transparency of each pixel such that the layers can be placed as semi-transparent on top of the background layers. Chroma keying is slightly less memory intensive, and is again very useful for combining pictures. Chroma keying is the special effect which we are all familiar with from the movies, which is also known as “green-screen” whereby the actor stands in front of a green curtain and the green is
replaced by a film showing the angry dinosaurs that are chasing him. In the movies, green or blue is used most often as the human skin tones are not affected, but actually, you could use red as well. This effect is really useful for creating different shaped objects, as you can simply use a square and then color the area around the object with the color to be removed.

In the previous example, we used a 24-bit color depth VGA screen and needed 900kB for the image. Now let us suppose that this image is just the background of the image and that there are two other pictures to be displayed on top of it. There is a graph and the frame that will be the other layers. These don’t have to be a full screen so in this case we could just use a smaller size image, perhaps QVGA size, needing then 225 KB each, and now the total RAM needed now comes in at about 1.3 MB.

Therefore, you have your 1.3 MB of data for the current data that you are showing on the screen, but sadly, you still need a bit more than just 1.3MB. The problem here is quite simple. If you change a picture, for example, because the background in one screen is different to the one on the next screen, you would have to get the CPU to change the data in the RAM whilst it is still being written to the screen. This causes a visible flicker on the screen and can cause a situation where the screen shows half of one image and half of the next one. This clearly looks poor; especially when we consider that the screen is being added to create a higher value proposition for the end customer.

This issue can be overcome very simply by using a concept such as double buffering. In double buffering, you actually double-up all of the RAM in the system so that the next picture can be set up in the back buffer and then you simply switch from one area of RAM to the next once the picture is ready. This doubles the RAM use and now we end up with a RAM requirement of 2.6 MB.

Now we have RAM filled with 2.6 MB of picture data for our VGA screen with 24-bit color depth. There are, of course, ways to both scale this up and scale this down. It is possible to take a smaller screen and this will reduce your memory requirement. For every less pixel wide your screen is, you save the 480 other pixels in the column, but the same is true in the other direction too. The XGA screen, which is 720 x 1024 pixels, needs 2.1 MB of RAM for the background image using 24 bpp color depth, rather than the 900 KB of the VGA screen. The other thing that can been changed is how many bits need to be used for the color of each pixel. We have used 24-bits per pixel in this example, as it is the most popular choice at the moment. It gives the most flexibility and ensures that the display does actually show what the graphical team would like to be on there. It is, of course, possible to use an 8-bit color set up, however this brings with it some other issues. With only 255 colors, there are many issues that will be encountered. For example, something as simple as displaying text is difficult. Modern fonts are not simply made up of white background and black letters with a width of an exact number of pixels. Even simple letters are made up of an array of blacks and greys so they are easily recognizable to the human eye. It is then possible to use a small screen and an 8-bit color to display some data, but it is questionable as to whether Steve Jobs would have put the Apple name on your end product!

With this in mind, there are a number of options in terms of processor or controller to drive the screen. The general rule of thumb is that a microcontroller is a good low cost option to drive smaller screens with a lower color depth; this makes sense up to about QVGA (320 x 240) size, which needs only about 150 KB (using 16 bpp) of RAM for the background image. However, above this, very often the bandwidth of the MCU and the performance of the core in the MCU is insufficient to create a truly rich user experience. Thus, above the QVGA size, the standard choice tends to be either an MPU, or a new embedded-MPU, now available from a number of suppliers and is essentially an MPU with the memory embedded in the device already.

So in summary, the embedded electronics world is being swept along by the consumer trend of adding a screen to many systems. This simple step allows OEM's to differentiate and add value to their end products, but creates a new challenge for design engineers taking their first step into this world. There are many different options of screen in terms of technology, color and size. It is also an area where there are many new changes happening on a near daily basis, as the mobile and consumer world continues to drive the display technology in new and interesting directions. For the engineer newly moving into the arena, the key design criteria to be fixed early in the process is what the screen size will be (measured in pixels), and from there most of the other decisions are relatively simple.
If the screen is going to be large, then you will need a lot of RAM and you will need a controller/processor that supports that much RAM, supports that interface and also has sufficient performance to drive it.