ID 415: Drive a Color TFT-LCD panel with Low-cost Flash MCUs

Sridhar Lingam
Product Marketing Manager
12 October 2010
Version: 06
Sridhar Lingam

- **Product Marketing Manager**
  - M16C/R32C, H8S/H8SX Product Families
  - TFT-LCD solution for H8S and H8SX
  - Renesas MCU CAN Solutions

- **Education**
  - MSEE from the Clemson University, Clemson, SC

- **Work Experience**
  - 16 years experience with semiconductor Industry
  - Varied experience as Product Engineer, FAE and Product Marketing
  - Responsible for definition and Marketing of Memory & MCU product families
  - Previously worked at National Semiconductor, STMicroelectronics & Atmel
In the session 110C, Renesas Next Generation Microcontroller and Microprocessor Technology Roadmap, Ritesh Tyagi introduces this high level image of where the Renesas Products fit. The big picture.
This is where our session, “ID415: Low-Cost Solutions for Driving TFT-LCD panels”, is focused within the ‘Big picture of Renesas Products’
Here are the MCU and MPU Product Lines, I am not going to cover any specific information on these families, but rather I want to show you where this session is focused
Notes for Devcon Positioning Slide: There’s a lot of vital information on this slide, which spotlights the Renesas MCU/MPU product lines recommended for new designs. Perhaps the best way to discuss this material is to cover it from a very high level.
The one product feature that helps differentiate itself from its competition is the user interface. Using a color LCD for an user interface helps the product in several ways.

For e.g., A color LCD has the effect of raising the perceived quality of the end product.

For example, the stainless steel cook range which is seen here is a mid-range in terms of burners, features, and capacity.... but definitely draws a consumer attention with its color TFT display giving it a better perception of quality.
Likewise, the putter with a graphical display gives an user the perception of better capability and helps command a premium in pricing.

Once you add a TFT panel, a touch screen is an obvious addition to make your product interactive both ways. In a very natural and intuitive way, customers can touch icons to activate functions, or slide a finger for variable input. Even traditional buttons and knobs can be replaced to reduce product cost and increase the sleekness of the end product

Wouldn’t you agree that the cost of adding the TFT could command a much higher retail price on the appliance, driving a higher profit margin and product differentiation?
Renesas’ LCD Direct Drive solutions provide you with a simple, low-risk and low-cost implementation to drive a graphical display that will enhance your product.

The myth about driving color LCDs...

They’re expensive, difficult, and a risk to the development schedule.

Today I’m here to tell you about solutions from Renesas which break this myth.

<Click>
Today I’ll cover some basic information about LCDs, in particular TFT LCDs, I’ll discuss some basics of TFT LCD starting with a very high level overview of TFT LCD panel technologies and why they are gaining popularity in embedded applications. The factors that are usually considered when picking a graphical display solution.

The MCU solutions from Renesas and our 3rd party partners have to offer, followed by a question and answer session.
Key Takeaways

- TFT LCD Basics
- A low cost Direct Drive LCD solution implementation
- Hardware and Software tools to guide the design effort
- Renesas MCU solutions for Direct Drive solution
Let’s begin by taking a look at 3 main LCD technologies that are popular within embedded applications.

First is STN. It’s a mature technology that’s very cheap, but unfortunately also looks very cheap. Passive STN is very slow with an animation rate limited to about 10 frames per second. STN has poor contrast ratio, and it has a very narrow viewing angle. STN is available in both color and monochrome, and quite often an STN module is driven by a serial connection to an MCU.

Next is TFT. Here, the response time is very good and it supports motion well. The contrast ratio and viewing angle is also very good. There are hundreds of sources of TFT panels, and cost is dropping rapidly.

Finally there is OLED. This is a relatively new technology with some very compelling characteristics like ultra-fast response time, excellent contrast, a near perfect viewing angle range, and ultra low power consumption compared to TFT. On top of this, OLED panels need no backlight, and they have the potential to be manufactured very inexpensively and in unusual shapes (like flexible display panels). However, OLED technology in today’s form has a limited life-span compared to TFT (some say 1/3 the life of TFT), and is very expensive (at least 2x that of TFT, more for large panels). As OLED technology matures, it’s cost is expected to drop, and it’s short lifetime issues solved.

So today, the best all around technology for embedded solutions in terms of price, visual impact, performance, and availability is TFT. You can see that last year TFT accounted for 80% of the total addressable market for LCD panels.
And finally, to add to the argument of adding a TFT display to your product, here’s what an industry analyst at iSuppli has to say...

"The overall small/medium display market, which also includes monochrome character-segmented displays used in digital clocks and other such applications, is changing to color and is adopting better-performing, higher-end Thin-Film Transistor-Liquid Crystal Display (TFT-LCD) technology."

Vinita Jakhanwal,
principal analyst, mobile displays, for iSuppli

Target customers moving up from user interface with simple segment, character, or dot matrix LCD
Target applications need color TFT-LCD with simple buttons, slide bars, text, informational graphics
There is no need for heavy animation or motion video in many of these apps.
**200MHz MPU solutions are overkill**, but customers still look to many existing MPU solutions in market
But today we’re talking about small TFT panels. Looking at the latest marketing data released by iSuppli not too long ago, we see that there is a considerable increase in the quantity of the small/medium displays going forward through 2014. However, despite this strong growth, the revenue is expected to fall for the same time period.
Pricing Trends of Small/Medium TFT-LCD Panels

A Continuing Price Decline !!!

Source: iSuppli Corporation. June 2010

Now, let us look at the next chart which helps us pinpoint the reason for the declining revenues. As you can see the ASP of the panels is forecasted to decline across all the application segments.
TFT panels have a digital interface which consist of parallel Red, Green, and Blue data information, plus control strobes to clock the info into the panel.

Color TFT panels are composed of individual pixels, with each pixel being able to display 3 colors of varying intensity red, green, and blue. To the human eye, the 3 colors in one pixel blend to appear as one color.

*click* If the MCU has a 32-bit data bus available, as shown here, then it’s possible to drive 8 bits of each color red, green, and blue. This is known as 24 bits per pixel color depth, or 24bpp. Each of the 3 color components can have 256 levels of intensity, making it possible for one pixel to appear as one of 16 million different colors (256 x 256 x 256). TFT panels can be driven with as little as 3 bpp, such as a red logo on a white background, as well as other color depth settings like 6, 12, and 18 bpp (262K possible colors per pixel).

*click* However, many MCUs physically have only 16 bits available on their external data bus.

*click* In this case, it’s very common in the industry to use 18 bpp color depth on a 16 bit bus. How?! You may be asking, how can you get 18 bits on a 16 bit bus? The answer is by reducing each of the red and blue components from 6 bits to 5 bits, and leaving green at 6 bits. This is done by tying the MSB and LSB of the red color together. Same thing is done to blue. This combination of 5,6,5 works quite well, with little loss of color clarity because green has the most impact to the eye relative to red and blue.

*click* There are vertical and horizontal strobes to synchronize each frame, and each line within the a frame. Each frame is rasterized progressively, no interleaving is done.

*click* There is a pixel clock which strobes in the RGB data for each pixel within each line, as well as a data strobe to gate the data that is clocked into the panel.

*click* And finally, there are optional signals which control the orientation of the image, meaning firmware can flip the image on either it’s vertical or horizontal axis, or both. If these signals are not used, they can be grounded or pulled up to the correct state.

That’s it, as little as 20 signals.
Here’s an example of a very common size TFT panel. This is QVGA, which has 320 pixels per line in the horizontal axis, and 240 lines in the vertical axis. To understand the relationship between pixel placement relative to the vertical and horizontal strobes, let’s look at this in more detail.

Here is an individual pixel (also referred to as a dot), and there are almost 80,000 (320 x 240) pixels on this QVGA panel. Each pixel has associated with it a 16 bit word of data, the familiar 5 red, 6 green, 5 blue bits. These pixels are painted on the panel in a progressive raster scan, just like an old typewriter, starting at upper left and ending at lower right. The pixels must be refreshed at a constant rate.

Painting of individual dots on the panel is controlled by vertical and horizontal strobe signals. Here you see the Vsync pulse, which goes active one time at the beginning of each entire frame. A “frame” refers to one complete screen with all ~80,000 dots. The Vsync pulse rate represents the refresh rate of the frame, typically ranging from 25 to 60 frames-per-second, or fps.

For each Vsync pulse, there are 240 horizontal strobes, also known as Hsync pulses. Each Hsync pulse indicates the start of a new line. Here you see the first Hsync pulse for line #1. Notice that within line #1, 320 pixels are clocked into the panel, pixels 1 through 320. It’s not shown here, but there is a pixel clock signal that pulses 320 times per line, each time latching one 16-bit RGB value into the TFT panel. We’ll see more of this on the next slide.

The second Hsync pulse begins line #2

and so on, with 320 pixels being clocked into the panel for each line
and finally the last line, number 240. So at the lower right we have line #240, and pixel #320. The frame is now painted, and the process will repeat with the next Vsync pulse, over and over again.
Zooming in a little further, now let’s look at what happens within a single line on the display, just after an Hsync pulse.

Here you see the beginning of a line on the rising edge of Hsync. Next you see what is known as the Horizontal back porch, which is a period just behind the rising edge of Hsync where the pixel clock (or dot clock) is active but there is no RGB data being latched into the panel. The number of dot clocks for the horizontal back porch varies from panel to panel, and is specified by the panel manufacturer.

Now the 1st pixel is clocked into the panel, also indicated by the rising edge of the Data Enable signal. At 16 bit RGB value is latched into the panel at each falling edge of the dot clock.

There’s the second pixel and so on until the final, or 320th pixel, is latched in for the line.

Now we have a horizontal front porch where the dot clock is pulsing but there is no RGB data being latched into the panel. You see also that the Data Enable signal goes inactive. The duration of the horizontal front porch is also specified by the panel manufacturer. It’s called the “front” porch because it comes before the new Hsync pulse.

And finally, the next horizontal line begins with the next rising edge of Hsync. This process repeats until all 240 lines are painted on the frame.

What about Vsync?

Here we see the falling edge of Vsync kicking off a new frame. There is a vertical back porch following Vsync where you have Hsync pulsing a number of times, but lines are not being painted on the display. The duration of vertical back porch is specified by the panel manufacturer in terms of the number of horizontal lines in the porch.

Now the Hsync signal indicates the first line is painted with a burst of 320 pixels being clocked into the panel.

The next line is painted and so on until the last line, number 240 is painted.

After the last line is complete, there is the vertical front porch, where again there are a number of pulses on Hsync, but no lines are being painted on the panel. The minimum duration of the vertical front porch is specified by the panel manufacturer. This period is extremely important, because you will see that it’s during this period that the H8S/SX MCU gets a chance to write the frame buffer memory with a new image prior to painting the next frame.
Factors that influence the Direct Drive Solution

- Interface
- CPU Loading
- Loading on the MCU internal and external busses
- System Cost without sacrificing performance
- Software development effort for LCD graphics
- Migration Path to higher resolutions

Here’s a laundry list of myths about using a TFT panel in an embedded design.

Let’s take them on one at a time and I’ll show you the solution that Renesas has to offer...
Let’s look at the positioning of LCD panels with respect to their interface

These are typically greater than 10” panels and have high-end features like heavy animation, video, and MP3 decoding need an MPU or a dedicated Graphic controller. However, many of these GUI applications do not require this and are cost sensitive.

Small TFT-LCD panels of 3.2” and smaller are typically display modules having a chip-on-glass (COG) controller to interface to an MCU with SPI or 8-bit parallel, and include an SRAM frame buffer. These COG display do not need Direct Drive LCD, just a standard MCU. MPUs and Direct Drive LCD are overkill for these applications. However, the customer is typically limited to small size and low animation levels with these COG displays.

This is the sweet spot. Applications that take advantage of larger TFT-LCDs (prices are continuously falling) larger than 3.5” with decent animation levels. In this case, and MPU solution is overkill for these simple GUI applications and Direct Drive LCD is a perfect fit.

What we target with our DDLCD solution are panels between 3.5” and 10”, which are commonly used in embedded control applications such as white goods, POS devices, climate control, and security control panels.
Question

- For which of the following requirements can DDLCD solution be applicable?
  - A: Light to medium animation is required
  - B: Need a screen measuring more than 3.5”
  - C: Need to have motion video

- Which ONE is a target application for DDLCD solution?
  - A: Thermostats
  - B: Medical Patient Monitors
  - C: White Goods User Interface
  - D: Home Security Keypad
  - E: All of the above

Look in the custom animation!

By putting a picture on top, we will print this page as a handout and it will not contain the answer!

The picture is the page being presented, but only the question is showing. I used Snag-It to capture the picture.

The picture goes away with previous! (using custom animation)
Direct drive removes the burden from the CPU. How does it work? Let’s take a look.

First there are the connections this WQVGA example.... In the box you see a Renesas MCU which runs from 35 MHz to 100 MHz.

The yellow box is the external frame buffer, where there are several choices of memory size and type:
- In terms of size you can choose to have a single-buffered frame which requires a 2Mbit, 128K x 16 memory, or a double buffered frame which requires a 4Mbit, 256 x 16 memory. For heavy animation, double-buffering reduces the effect of image “tearing” on the screen because the CPU builds one display buffer “A” for example while the other buffer “B” is being painted to the screen, then the CPU just flips the pointer to the buffer “A” before the next frame is painted. In this way, you’ll never see on the screen the effect of the frame buffer being updated while it’s being painted.
- In terms of memory type, SRAM, PSRAM, or SDRAM can be used. SRAM tends to be most expensive but supports the highest speed, PSRAM is very cost effective for moderate speed, and SDRAM is most cost effective for larger densities like 1M bytes that VGA requires. H8S and H8SX support all 3 memory types (except the synchronous mode of PSRAM).

The gray box is the QVGA TFT panel

Notice that the MCU, the frame buffer, and the TFT panel are all sharing the 16-bit external data bus of the MCU. Address lines and read/write data strobes to the SRAM are generated by the MCU. The MCU’s timer unit (TPU) synchronizes the timing control to the TFT panel with vertical and horizontal strobes, pixel clock, and data enable.

Now you see that internal to the MCU, the CPU can access it’s internal Flash, SRAM, and peripherals, at the same time that the external DMA unit (ExDMA) is continuously transferring the 16-bit RGB data from the frame buffer to the TFT panel. Neither the activity of CPU, nor the ExDMA unit interfere with each other with any significance (only 5% loading on CPU). Note that the ExDMA unit does not care what’s in the frame buffer, it just moves the data to the LCD panel at a pre-programmed repetition rate.

To recap, there’s a “functional barrier”, if you will, between the CPU internal operation,
Here we have a simple base-line case where a static image is constantly refreshed on the display. As you saw before, the SRAM frame buffer is connected to the MCU external address/data bus, and the LCD panel is connected to the same external MCU data bus as the SRAM. Synchronized LCD panel timing is controlled by the MCU’s TPU timer unit.

First the MCU must build the static graphic image and load it into the external memory frame buffer.

The CPU drives the destination address on the external bus which is connected to the external memory frame buffer.

Then the image data that is stored in internal Flash moves over the data lines to the external frame buffer memory. Each 16-bit word of RGB pixel data is transferred into the frame buffer.

Alternatively, the CPU can retrieve the image data from an external serial Flash memory and send it to the frame buffer.

The static image is now completely stored into external SRAM frame buffer. As stated earlier, a single-buffered or double-buffered approach can be taken depending on your system animation level and cost requirements.

Now let’s look at the ExDMA unit. After initialization, the ExDMA unit runs freely, automatically reading whatever is in the frame buffer and transferring it into the LCD panel.

The ExDMA unit takes control of the external bus, and generates an address to the SRAM frame buffer and a read strobe.

The SRAM produces a 16-bit word of RGB data for a single pixel, and the TPU generates the clocking signals to latch the RGB data into the LCD panel. These timing signals are synchronized with the ExDMA activity.

This process automatically repeats again and again for each pixel on each line on each frame, constantly refreshing the LCD panel at a pre-programmed 50 Hz refresh rate.
In this case we have “medium” graphic animation by adding a sun which a rises and sets over our thermostat image. The CPU must perform more calculations to reposition the sun within each and every frame, and update the some of the content of the external frame buffer memory each frame. Since the entire image does not change each time when the sun moves, the entire frame buffer does not have to be re-written each frame, saving bus bandwidth.

The CPU must jump onto the external data bus during the vertical blanking period that we discussed. I’m only showing activity on the data lines for simplicity, but the same automatic generation of address, write strobes, read strobes, and pixel clocking signals is still going on.

But you can see the external data bus now has 2 masters, one is the CPU transferring data from it’s internal memory to the external frame buffer memory, and the other is the ExDMA unit transferring data from the external frame buffer to the LCD panel. Since the image is no longer static, traffic on the buses increases. How much?

In this example at 50 fps for QVGA, internal bus loading climbed to 35% which consists of CPU 5% baseline overhead, plus 5% for the CPU to calculate new image data based on the moving sun, and 25% for the CPU to get the data from internal MCU memory outside to the external bus.

External bus loading rises to 67% which consists of the original 42% baseline loading of the ExDMA traffic, plus the new 25% loading from the CPU sharing the external bus to move new “sun” data to the frame buffer between each frame.

What about heavy graphic animation?
Here are a couple of system examples with suggested resale pricing at 50Ku.

For QVGA you can have the MCU and frame buffer for only $4.55 to control the display and run the application. This is $2 to $3 less than a minimum microprocessor (MPU) system by the time you add up the MPU, the SDRAM, and the external flash memory. **Here is where it is apparent that an MPU system is overkill for a typical simple color GUI that needs only light to medium animation.** This system solution is so inexpensive, you can use it as a TFT-LCD “co-processor” to easily add a color TFT-LCD to an existing design as an add-on which requires only minimal changes to the base system.

For VGA you also have very competitive system solution pricing at just $5.96 to control the display and to run the application.
Renesas Graphics API and Library

- Create your own GUI with Free API, Library, and Demos
- Use buttons, sliders, shapes, and manipulate bitmap images
- Import Standard Vector Font Files, proportion and display fonts
- Place a text string within a bitmap button or other object
- Supports transparency, coloring, and direction of characters

Graphics Application Programming Interface: GAPI
“A set of routines that allow for the simple creation and manipulation of raster based images in RAM memory frames”

**Graphics library with an API (GAPI)** based on bit map manipulation. A framework is built on top of the GAPI providing functions that:
- Draw and manage objects such as buttons and slide bars
- Copy bit-map images to and from frame buffer in 1bpp, 4bpp, and 8bpp indexed and RLE compressed formats
- Create and fill simple shapes
- Place characters (optional rotation) on the display and on other objects
- Create proportional fonts
- Bitmap transparency

You’ll also find support for proportional fonts and text manipulation with anti-aliasing
When more advanced animation is required, such as windows, widgets, and alpha-blending, these three 3rd party graphic software vendors have products supporting the Renesas DDLCD solution.
Question

What % of internal CPU bandwidth is required of DDLCD to refresh a TFT panel at 60Hz?

- A: 50%
- B: 5%
The Renesas Development kit for Direct Drive LCD comes with a 4.3” WQVGA TFT LCD panel and a RSK for either the 2456 or the 1668X device. An E-10A debugger is also included with the Development kit. The kit contains a CD with the HEW IDE, C/C++ compiler and ready to start evaluating the system and developing graphics with the Free Renesas Graphics API, library and examples.

The CD also contains exercises that will help the user with self-training.

Please do attend the lab sessions on our Direct Drive LCD solution offered by our application engineers.
Renesas offers multiple Flash MCU solutions for the Direct Drive LCD solution which uses a simple, low-cost Flash MCU to drive the TFT-LCD and touch-screen, as well as running the remainder of the embedded application.

I will start with the legacy 16 & 32 bit H8S and H8SX product families that run at 35 & 50 MHz respectively. These two families come with up to 512 Kb to 1 MB of Flash and are capable of driving displays with resolutions up to QVGA and VGA respectively.

The next generation high performance low cost RX600 family comes with up to 1 MB Flash. The RX family runs at a frequency of 100 MHz and delivers 165 DMIPS of performance. The RX can support WQVGA resolutions with even less than 5% of its bandwidth. Besides the Direct Drive capability, RX also comes with abundant peripherals like High-speed USB host & device, Ethernet, CAN that extend the connectivity solutions.

For applications requiring higher resolutions with motion video like requirements, Renesas offers the SH-2A and SH-4A solutions.
Question

- List at least 4 target applications for DDLCD?
  1. Thermostats
  2. Security Panels
  3. Medical Patient Monitors
  4. Climate Control
  5. Exercise Equipment
  6. White Goods User Interface
  7. Industrial Process Control User Interface
  8. Instrumentation
The one product feature that helps differentiate itself from its competition is the user interface. Using a color LCD for an user interface helps the product in several ways.

For e.g., A color LCD has the effect of raising the perceived quality of the end product.

For example, the stainless steel cook range which is seen here is a mid-range in terms of burners, features, and capacity.... but definitely draws a consumer attention with its color TFT display giving it a better perception of quality.
Likewise, the putter with a graphical display gives an user the perception of better capability and helps command a premium in pricing.

Once you add a TFT panel, a touch screen is an obvious addition to make your product interactive both ways. In a very natural and intuitive way, customers can touch icons to activate functions, or slide a finger for variable input. Even traditional buttons and knobs can be replaced to reduce product cost and increase the sleekness of the end product

Wouldn’t you agree that the cost of adding the TFT could command a much higher retail price on the appliance, driving a higher profit margin and product differentiation?
Thank You!
TFT Interface Throughput

Examples of system resources needed for some very typical LCD sizes and color depths

- **QVGA (320 x 240)**, 16 bpp, 50 frames per second (fps)
  - *Frame buffer size = 154 Kbytes* ............... [ 320 pixels x 240 lines x 2 bytes per pixel ]
  - Actual qty of pixel (or dot) clock periods with front and back porch per frame:
    - *Dot Clocks per Frame = 95882 clocks* ....... (2HS+52HB+320+8HF) x (2VS+5VB+240+4VF)
  - We artificially extend the Vertical Front Porch so CPU can update frame buffer:
    - *Dot Clocks per Frame = 225,762 clocks* (2HS+52HB+320+8HF) x (2VS+5VB+240+345VF)
  - *Final Dot Clock Rate = 11.3 MHz* ......................... [ 225,762 dot periods x 50 fps ]

- **VGA (640 x 480)**, 16 bpp, 35 fps
  - *Frame buffer size = 614 Kbytes*
  - *Extended Dot Clocks per Frame = 498,960 clocks*
  - *Dot Clock Rate = 17.5 MHz*

Let’s take examples of 2 popular TFT panel sizes, and examine what kind of system performance and capacity is needed.

First is the familiar QVGA panel that we’ve been discussing. In this example, will use 18bpp color depth (with the trick of 16 data bits), and we’ll refresh the frame at 50 frames-per-second, or 50 fps.

Somewhere in the system there must be a fast volatile memory capable of holding the entire graphic image, and it must be updated between each frame if there is any animation on the screen. This memory is typically SRAM for QVGA panels and smaller, but SDRAM for panels larger than QVGA. The size of this frame buffer memory is determined by multiplying the number of pixels per line (320), times the number of lines per frame (240). Since each pixel consists of 16 bits of data, each pixel is 2 bytes. So the frame buffer size in bytes is 320 x 240 x 2, or 154Kbytes.

But because there is a back and front porch in the horizontal and vertical directions, there are more pixel clocks (or dot clocks) than there are pixels. To calculate the number of dot clocks that occur in one frame, we must add the number of “dummy” dot clocks of the front and back horizontal porches, plus the Hsync pulse width itself, to the number of actual pixel clocks. Then we and add the number of “dummy” lines of front and back vertical porches, plus the pulse width of Vsync itself, to the actual number of lines. Multiply these horizontal and vertical components as shown. The result is almost 96,000 dot clocks per frame.

Remember from the previous slide that it’s during the vertical front porch that the CPU can jump onto the external bus and write to the frame buffer before the next frame gets painted. So we artificially extend the duration of the vertical front porch to allow the CPU to jump on the bus. The more we extend the vertical front porch duration, the less time the ExDMA has to use the external bus within the fixed frame refresh period.
Thanks for spending this time with us, we hope you enjoyed it!