GUI Design - The Intersection of Art and Engineering

Author:
Ken Klask – CEO
Amulet Technologies

Synopsis:
Industrial design and a positive user experience is the key to a successful product today. Making a product worthwhile to work with is now just as important as making it work. This article will illustrate how collaboration between marketing, graphic artists, usability experts and embedded systems programmers throughout all phases of the product design cycle is essential if the goal is to create a compelling interface that can differentiate a product and a brand. GUI solutions are now available for the ARM7™ processor architecture that not only foster this collaboration, but support user-friendly embedded systems with an LCD user interface that would otherwise require a more powerful processor.

Graphic User Interfaces (GUI's) are not just for personal computers anymore. Consumers have come to expect visually appealing and easy to use Graphic User Interfaces on all new electronic devices. They feel at ease navigating through screens to reach different features of their new device. At the same time, product designers and marketers have discovered the secret to differentiating their products and luring consumers to their brand. The secret lies in the marriage of digital electronics and industrial design, and a well-designed GUI is often a central component in the implementation of this strategy.

Changing Trends

This consumer trend is not unique to iPods® and cell phones. It is also having a serious impact on many other Embedded products. New technological advances are increasing the capabilities and complexities of our devices. In many cases, there just isn't enough space to add more buttons and LED's to make the added functionality accessible. Likewise, engineers are being pressured to reduce the cost and increase the reliability. For this reason, we are nearing the end of an era where the human interface for an embedded system holds a close resemblance to a 1970's digital alarm clock with push buttons, knobs, sliders, switches, and LED's.

The Right Fit

The difficult part is finding a GUI solution that is right for your application. Embedded systems cover a diverse product range. For some embedded products, such as those in the automobile navigation and portable media markets, there are already a variety of solutions supported by operating systems derived from the desktop and powerful applications processors. But, what about the embedded control engineer who runs his code on bare metal with no operating system or maybe just a simple RTOS kernel? He might be tasked with designing a new washing machine, coffee maker, copy machine, spa or other consumer product that is often driven by one or more small processors.

In this embedded control space, 8-bitters continue to dominate in unit sales because performance is still adequate for many applications. They are also cheaper and simpler to use than 32-bitters. But, when the 8-bit microcontroller reaches its limit in performance or memory addressing capability it is time to consider an upgrade to a 16- or 32-bit microcontroller. A full color interactive GUI is one of those applications that exceeds the memory addressing of 8-bit processors and is therefore an opportunity for migration to ARM® processors-based solutions.

Because of their low prices and high peripheral integration, microcontrollers based on the ARM7TDMI™ are positioned very well as the perfect stepping-stone from the 8-bit world to the 32-bit world. But, obstacles still exist if the ARM7 processor is to fill the gap between the 8-bit user interface and the 400 MHz multimedia processor. First, the memory bandwidth of the ARM7 needs to be boosted to accommodate the display refresh requirements of full color LCD displays. Second, desktop style operating systems don't run on the ARM7 processor. This leaves a gap in both the firmware and development tools required to deliver an attractive GUI on the ARM7.

Let's try to understand the first concern – memory bandwidth. In most embedded GUI applications, the required amount of frame buffer memory is so huge that it is not cost-effective to put it on the controller. For example, a 24-bit color VGA (640x480 pixels) LCD requires a frame buffer of 1.2 MBytes. A microcontroller with this much internal SRAM would be prohibitively expensive, so the frame-buffer must be

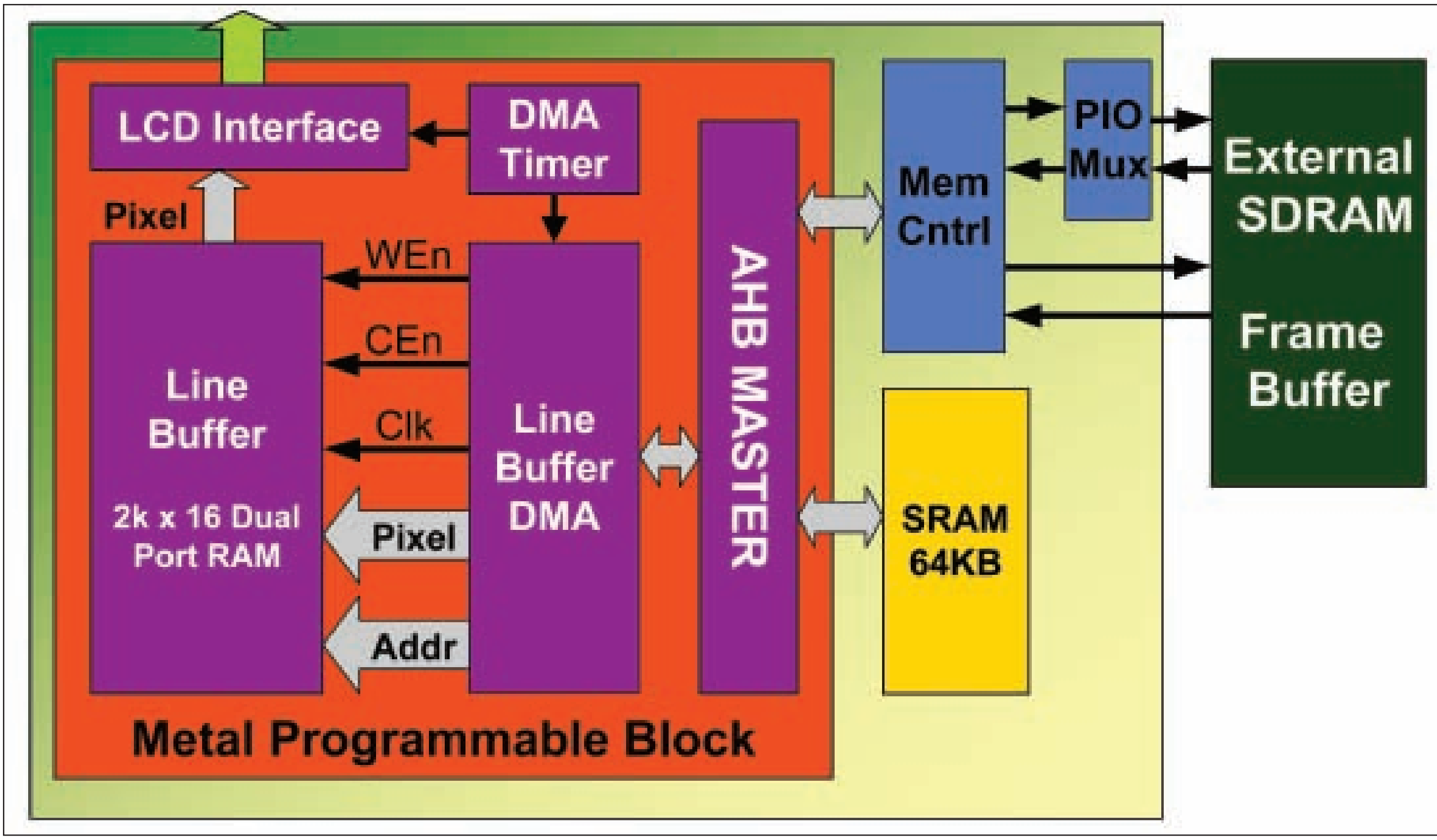

Figure 1: Amulet's GUI Engine DMA from the Frame Buffer to the LCD

stored in external RAM. In addition, consider that the minimum refresh rate for an LCD display is typically 60 frames per second. This means that the CPU needs to fetch 1.2 MBytes of data 60 times per second, or 73 Megabytes per second. Even at 80 MHz, a conventional ARM7 processor cannot possibly achieve this level of throughput.

But, with the combination of a multi-layer bus matrix and two independent memory systems, the memory bandwidth problem can be overcome. An example of this solution is the Amulet GUI Engine IP available for Atmel® Corporation's CAP7™ customizable microcontroller. This intriguing design allows an ARM7-based CAP microcontoller to control user-friendly embedded systems that otherwise would require a more powerful microcontroller. Amulet Technologies is a fabless semiconductor company that currently builds its patented Graphical OS in Silicon™ technology into its own chips. Porting the technology to Atmel's CAP and licensing the macro as hard IP is a new venture for the company. Because the demand for screen refresh exceeds the ARM7TDMI processor's bus bandwidth, Amulet's custom logic offloads that task from the CPU. This frees the ARM7 core to run the application code without interruptions from the screen refresh. The Amulet LCD controller can drive passive monochrome or color LCDs with resolutions up to 640 x 480 pixels, or color TFT LCDs with resolutions up to 800 x 600 pixels. It handles all refresh tasks and has an internal frame buffer sufficient for small displays. Larger screens require an external frame buffer in SDRAM.

As shown in the block diagram of Figure 1, Amulet's GUI Engine accesses the frame buffer directly via DMA for GUI rendering and LCD refreshing, leaving CAP7's internal memory and bus free for code execution via the ARM7 processor core. On smaller displays, the internal SRAM can be partitioned into two independent blocks – one connected to the Amulet GUI engine and the other connected to the ARM. This also yields concurrent processor execution and LCD refresh, but with no external memory required – this is a real plus in applications constrained by cost or board real estate.

Besides controlling the LCD, Amulet's custom logic has a GUI engine that manages a graphical user interface, offloading more work from the CPU. Graphic designers can visually design the GUI using standard web-design tools, and then use Amulet's proprietary development tools to compile the HTML into a special language called microHTMLTM that drives Amulet's GUI engine. The embedded systems programmer takes it from there by connecting the application data to the microHTML GUI through a simple API that allows read and write access to the arrays of shared data. A connection of the shared variable data to the visual GUI objects is illustrated by the GUI example shown in Figure 2 on next page.

Collaboration

This brings us to our second obstacle in the migration to an ARM7 core-based GUI platform for embedded control systems. Most GUI solutions are designed to run on powerful operating systems derived from the desktop. This means that they require considerable computing power and memory that is far beyond the capabilities of the ARM7, and have a price point that is

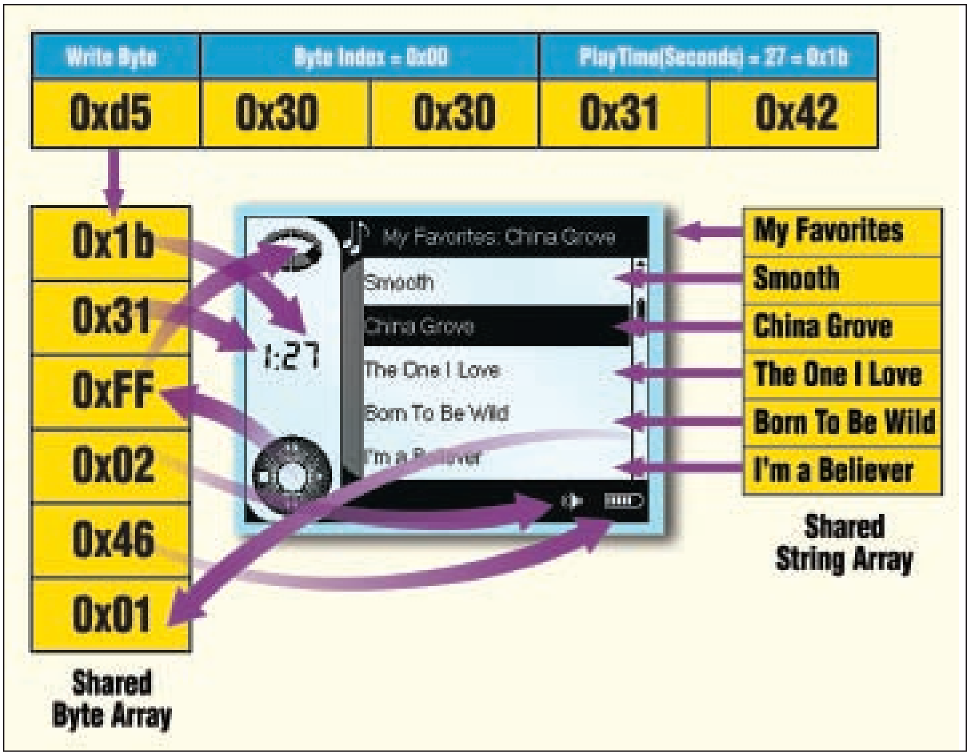

Figure 2: Connecting the application data to the GUI objects

unacceptably high for many embedded applications. If the embedded system falls into this category, the embedded engineer is often forced to code his GUI operating system and tools from scratch. This is not an easy task, but he will eventually make the GUI work. The major challenge with this approach is making the GUI look good.

So, why is this a challenge for many embedded programmers? Well, just ask Apple Inc.® In his book, "TOG On Interface," usability guru, Bruce Tognazzini, tells of an informal study that he conducted while employed at Apple in the mid 1980's. The results of this study helped Bruce and Apple confirm why software engineers are not the best candidates for designing user interfaces that need to appeal to the general population. In his study, Bruce conducted a Meyers-Briggs personality type test on all of Apple's software and hardware engineers. When he compared the personality types to that of the general public, he learned that "Engineers are from Mars and Consumers are from Venus."[1]

As indicated in Figure 3, Bruce found that engineers interact with the world "intuitively." They are hardwired to see things differently than the average consumer. This gives them outstanding visualization skills that enable them to see deeply into the inner workings of products. Unfortunately this personality trait makes it difficult for them to understand why "sensory" types cannot see the same product inner workings. This is also why they have difficulty understanding what things attract consumers to one brand over another when both products work the same.

On the other hand, the "sensory" types in the organization are more like the average

Preference	Males	Females	Total	Apple Engineers
Extrovert	45.9%	52.5%	49.3%	58%
Introvert	54.1%	47.5%	50.7%	42%
Sensing	71.7%	74.9%	73.3%	25%
iNtuitive	28.3%	25.1%	26.7%	75%
Thinking	56.5%	24.5%	40.2%	X
Feeling	43.5%	75.5%	59.8%	X
Judging	52.0%	56.2%	54.1%	X
Perceiving	48.0%	43.8%	45.9%	X

Figure 3: Meyers-Briggs Personality Types -- Engineers Versus The General Population

consumer. These are the marketing people, industrial designers, and graphics artists. They rely heavily on sensory information to interact with people and products. This gives them the ability to understand what the average consumer wants. But, it also means that they lack the visualization skills required to assemble the 1's and 0's in a way that makes the digital device work.

What Apple discovered is that the only way to make a product work well, and be a thrill to work with, is through multidisciplinary team collaboration. The key is to find a solution that maximizes the collaboration between the embedded engineers that are concerned with making the product work, and the industrial designers that are concerned with making the product worthwhile to work with. If that isn't enough, this collaboration must be enforced throughout the entire product design cycle.

The Amulet design methodology was designed to enforce this concept of collaboration. Since the look and feel of the GUI is best authored by graphic designers and usability experts, Amulet's tool chain embraces tools that these professionals are already familiar with – web authoring tools and Adobe® Photoshop®. Thus, Art directors are not forced to become C-programmers.

Likewise, the programmer's interface to the GUI is very simple and doesn't force the embedded programmer to get involved with the GUI if he doesn't want to. He can simply control the device and manage the parametric data that is generated or presented by the GUI. The most important part of this design flow is that it enables collaboration between the marketing, graphic artists, usability experts and the embedded systems programmers. Collaboration throughout all phases of the product design cycle is essential if the goal is to create a compelling interface that can differentiate a product and a brand.

Another side effect of this design partitioning is that it simplifies testing. The embedded device can be easily tested independently of the GUI by manipulating the same variables that would otherwise come from the GUI. Likewise, user testing of the GUI can be conducted in parallel with the development of the embedded control code. More importantly, embedded programmers no longer need to worry about those last minute GUI changes breaking the control system code.

Simply sophisticated

Things should be made as simple as possible, but not any simpler. That is the philosophy behind Amulet's patented GUI architecture. GUI design and implementation is made as simple as possible for the designers and engineers, but not so simple as to compromise the usability and appeal for the end consumer. Using the Amulet approach, embedded systems designers can now build sophisticated, yet easy to use devices without fear of compromising the rock-solid reliability that they have worked so hard to achieve.

Need a low-cost, low-risk direction for your next ARM Powered® Chip?

Selecting the right path to silicon is like producing a movie: it requires a team of specialists – a Value Chain – to provide a smooth development process.

Instead of coping with writers, camera crews and directors, you have to deal with chip complexity, multiple suppliers, time-to-market pressures and more.

At eSilicon, we offer a flexible, low-cost, lower-risk path to silicon tailored specifically for your needs. We produce an optimized Value Chain for your ARM Powered® chip design, enabling you to get your ideas to market faster.

eSilicon provides solutions that enable our customers to get their chips to market faster without sacrificing performance, power or cost. You too can benefit from eSilicon's wide range of optimized solutions for your next chip design:

- Pre-hardened ARM7™ and ARM9™ cores in process geometries ranging from 0.18μ to 65nM
- Processor hardening services for custom implementations
- Direct access to ARM PrimeCell™ Peripherals
- Flexible peripheral subsystems for quick and easy system development

eSilicon–Enabling Your Silicon Success™

Contact us today at www.esilicon.com and let us start you in the right direction!

eSilicon®