Working across Multiple Embedded Platforms

Embedded systems are those computer systems that do not look like computer systems to the everyday user. They are the hidden computer systems that form a part of a larger system or product, part of anything from toys to trucks, from mobile phones to medical devices.

In fact more microprocessors around the globe are used in embedded systems rather than in PCs. Those already large numbers are increasing at a phenomenal rate as the devices that surround us in our everyday lives become smarter. A consequence of an insatiable drive towards having control over devices and access to data anywhere, anytime. Needless to say we prefer them connected - wired or wireless.

Traditionally the requirements placed upon embedded systems are quite different to those applied for desktop computing. Because embedded systems are in general designed to accomplish a very specific task or group of tasks there is no single characterization that applies to the whole gamut of embedded systems. However some combination of the variables of robustness, small size and weight, real-time requirements, long life cycle and low price could be expected to figure in the design criteria for most embedded systems.

Less tolerance for malfunctions in some cases may be simply a convenience and cost issue, such as the lack of permanent I/O connections which makes debugging more difficult, or it can far more serious such as the failure a mission critical component which could have extreme consequences.

Real time requirements combine the constraint of time and correctness - not only does the computation need to be correct but it also needs to be at the correct time. This requires an estimate of the worst case performance, which on complicated architectures can be difficult, and leads to overly conservative estimates. Mission Critical systems as a class have a significant requirement for real time operation in order to meet external I/O and control stability requirements.

Low price translates to reduced resources such as processor speed and memory size which in turn constrains software development and execution. Often embedded devices are very sensitive to cost. A variation of even a few cents per device can be significant due to the huge multiplier of production quantity combined with the higher percentage of total system cost it represents.

These constraints though difficult, were manageable, if the application is simple and small enough to and run without the need of an underlying operating system.

This changes once there is a need to manage many variables such as serial, USB, TCP/IP, Bluetooth, Wireles LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on.

Continuing to use the traditional approach to the modern complex designs can become chaotic and ad hoc – and require very experienced personnel. Complexity can increase to the point where it becomes inefficient not to have an operating system to handle various tasks on behalf of the application. This brings in its own issues: Embedded-system developers must select an operating system platform prior to starting to their application development process. Which one do you chose? The timing of this decision forces developers to choose the embedded operating system for their device based upon current requirements and so locking in their future options to a large extent. What happens if your requirements change and the chosen platform does not support the new requirement? What happens if a major client requests you change to another platform?

One way to deal with a requirement for multiple platforms is to deploy multiple teams of personnel experiences on a particular platform. Often developers are very familiar and experienced in one environment or one group of related environments only. If there is a need to offer the application embedded Linux / WindowsCe and a proprietary RTOS then three teams and three parallel developments may be required. Alternatively the development may occur in series with an experienced team being required to port to the various platforms in series.

Unfortunately there is no one-size-fits-all solution to such issues; however there are some basic principles and general solutions that can assist the process of building reliable portable embedded systems applications on time and on budget; guidelines and products that help the design of well integrated code and tools to help deal with debugging the inevitable issues that occur.

Some of these solutions can be obtained by various books and courses and other from development tools and environments. Several framework environment tools exist but usually are specific to a very limited platform set such as Linux and embedded Linux and so on. SoftFrame, from Clarinox Technologies however, provides an infrastructure that reduces the cost and improves the efficiency of embedded systems application development across many platforms. Develop and debug on Windows, and run, for example, on Intel Bulverde processor with embedded Linux or StongArm processor with WindowsCE. A visual representation of SoftFrame is in Figure 1.

SoftFrame provides the tools and the environment to get embedded systems applications design and development under control. There are no detailed formal methodologies to master; and most designers are up and running within hours.

The product provides the entire necessary infrastructure for the embedded software engineer to develop without the need to know the real-time operating system (using Microsoft Visual Studio C++, embedded Visual C++ or GNU tools as appropriate). Prototyping and simulations can be done, without the need for hardware, on a PC.

The OS Wrapper includes functions such as:

- Threading
- Timers
- Semaphores

- Mutexes
- Dynamic memory management without fragmentation
- Inter-process message passing
- Event/Message handling
- Finite state machine
- Serial device driver encapsulation
- USB device driver encapsulation
- TCP/UDP Socket encapsulation

SoftFrame is an extension to the debugging tools and Board Support Package or Hardware Adaptation Layer provided by RTOS manufactures for example the Intel SA-110, SA-1100, SA-1110, SA-120, SA-1500 evaluation boards, as SoftFrame presents debugging tools that can handle complex multi threaded applications that are not specific to only one environment.

The SoftFrame does not claim to solve all the issues involved in the design of embedded systems – but it does claim to relieve the application programmers from some of the distractions that take the focus away from the application through, for example:.

- **dynamic memory management.** SoftFrame provide a simple and effective memory management module to replace C style (malloc/free) or C++ style (new/delete) calls. Inside these calls are the smart, and adaptive memory and pool management that does not result with memory fragmentation, yet fast and efficient.
- **standard libraries** and stream libraries guarantee are provided that the code works the same way on each platform.
- Debug mode **function profiling**, function entry/exit **tracing** and indented display of threads and functions with timestamp would prove the value of presenting information. Especially, while trying to find a problem that happens in a system with a large number of threads and protocols running concurrently

These benefits, plus the ability to run across multiple platforms, makes SoftFrame a product that is extremely complementary to makers of embedded development boards or embedded products. SoftFrame speeds up development and eliminates the need for porting the application from platform to platform by encapsulating the major functions of an operation within a standardized API call system. It offers an ever increasing number of platforms that the developer can change between combined with a fast and easy application development infrastructure.

To address today’s demand for short range wireless applications, Clarinox has available add-on options for Bluetooth, RFID rand WiFi for any of the supported platforms.

The Clarinox embedded software development framework enables the development of standardised applications with reduced errors, reduced development time and reduced complexity. So whether you wish to run on an Intel Centrino, an Intel XScale or an Intel StrongARM, whether using embedded XP or Windows CE or an RTOS such as eCos;

you can develop one application on Windows and then run on either or both of these platforms without a lengthy porting process and with better debugging tools.

About Clarinox Technologies

Clarinox Technologies specializes in embedded systems and short range wireless technologies, in particular the design and implementation of efficient. Products include SoftFrame, ClarinoxBlue, WiFi and RFID modules. Together these products provide a "plug and play" type approach to constructing an embedded short range wireless product.

References

http://nicta.com.au/director/research/programs/ertos.cfm
http://ptolemy.eecs.berkeley.edu/~kienhuis/dacSlides/intro.pdf
http://www.ece.cmu.edu/~koopman/iccd96/iccd96.html