

Rapid Application Development for Embedded Systems

The article discusses a methodology to accelerate project schedules by implementing parallel software and hardware development. It defines the RTOS characteristics required for this strategy and explains how OS-9 meets the requirements. Graphics are included to clarify the effects of RAD on project schedules, and to explain the OS-9 architecture. The article also discusses other architectural elements of OS-9 that increase the efficiency of application software and system software development for embedded systems.

Rapid Application Development (RAD) is a well-accepted concept with end user software development, but it applies equally well to real-time embedded systems. By combining an intelligent methodology with the right RTOS solution and smart people, you can significantly decrease your total development time.

RTOS REQUIREMENTS FOR RAD METHODOLOGY

Successful utilisation of RAD for embedded systems requires specific characteristics from your RTOS:

- Ability to produce hardware-independent applications
- Comprehensive support of development board I/O
- Minimised time to set up development board
- Efficient development tools

INTELLIGENT METHODOLOGY

Traditional Development Cycle

Projects based on custom hardware consist of two separate engineering efforts for hardware and software development. In the traditional cycle, these two phases cannot be done in parallel because of the software dependencies on hardware. After the initial design phase, software engineers wait for the hardware to be developed. Then the hardware engineers wait for the software to be completed. Finally, the new software and hardware are evaluated together.

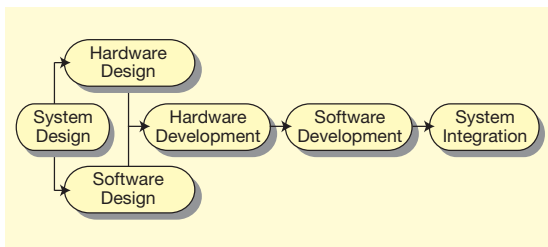


Figure 1. Traditional Development Project Cycle

Streamlined Development

It is possible to eliminate the unproductive waiting periods of the traditional development cycle. You can write and test your software application without the

final product hardware by using off-the-shelf boards for software development while your custom hardware is being developed. With an effective RTOS architecture, your applications will contain no hardware-specific code, so the software can later be transferred to the final hardware.

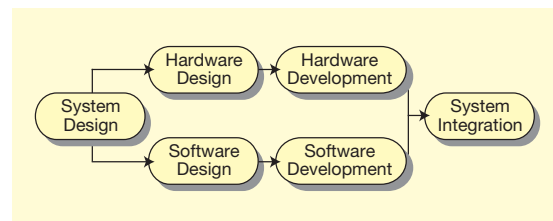


Figure 2: OS-9 RAD Development Project with Custom Hardware

Decreased System Integration Time

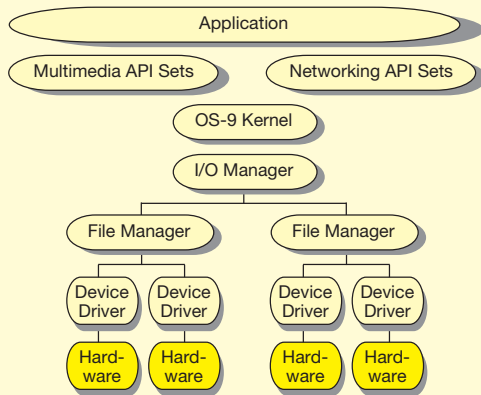
The "System Integration" portion in Figures 1 and 2 is usually the most critical and most stressful phase of the project. Often, a project progresses smoothly up until this point, and suddenly the unexpected happens, causing your schedule to slip.

If you use the traditional model, you will be using new, untested software on new, untested hardware. This environment will make it difficult and time consuming to determine if an exception is based in hardware or software. You might waste weeks tracking down a software bug, only to discover that faulty hardware is the actual cause.

If you use the RAD method, you reduce your risk and save time during system integration. Your development hardware is stable, so you can concentrate on software debugging. Once the software is debugged, you have a stable software package to test on the new hardware. If you don't know if the source of a problem is in software or hardware, you can compare the software's behaviour on the final product hardware with results on the development board.

MICROWARE'S OS-9 RTOS ENABLES RAD

In order to understand how OS-9 enables the smart RAD methodology, it is necessary to examine the basic OS-9 RTOS architecture.



- **API Sets:** abstract from hardware specifics; modular multimedia and communications libraries
- **OS-9 Kernel:** manages all system services, memory, I/O and process execution
- **I/O Manager:** passes I/O requests to the appropriate file managers
- **File Managers:** contain generic support for each class of I/O device, such as character oriented, block oriented, tape devices, etc.
- **Device Drivers:** perform the basic low-level physical input/output functions of specific hardware controllers

Figure 3. OS-9 Architecture

Hardware Independent Applications

The ability to write hardware independent applications is essential to successfully streamline your project cycle. The unique architecture of Microware's OS-9 RTOS shields applications from hardware-specific code, enabling you to transfer software applications from the development board to the custom hardware platform.

Applications are written to full-featured multimedia and networking Application Programming Interface (API) sets that abstract the hardware. The programming interface used by the application is identical whether the application is using a hard drive, serial device, or network interface. OS-9 APIs are written in ANSI-C language, so they can be ported to any CPU architecture supported by OS-9.

OS-9 implements a unified Input/Output (I/O) system. When an application initiates the opening of a path OS-9 creates a link between the application, file manager, and device driver. The application uses the resulting path to access services provided by the I/O system. All modules in the system are fully re-entrant and position independent.

Let's look at an example of how the API abstraction works. The multimedia input process is responsible for insulating the API layers from differences in pointer and keycode devices. This process reads raw input from Serial Character File manager (SCF) devices and uses protocol modules to translate the data into standard-

ised messages. These messages are then inserted into the application's mailbox so that the applications may act on the user input.

Solid Hardware Support

Another key to success with RAD is solid and comprehensive support of development boards. OS-9 supports a wide range of cost-effective boards with fully tested object and source-code drivers. Effective binary drivers are preferred for application development because it requires minimal effort to get them to run. Along with the featured CPU chip, OS-9 includes extensive drivers for the I/O featured on the board. OS-9 supports x86, 68K, PowerPC, SuperH, StrongARM, MIPS, and Sparc development boards.

DECREASED SOFTWARE DEVELOPMENT TIME

In addition to streamlining project tasks, consider ways to increase productivity of the "Software Development" portion of Figures 1 and 2. Efficiency in software is maximised with the superior tools and the premium OS-9 architecture.

Development Board Set-up

It is important to minimise the time spent setting up your development board since this effort does not directly apply to the final product. OS-9 accomplishes this with a built-in configuration wizard, so you can immediately focus on your product development effort. Select the name of the board in the configuration wizard, and OS-9 will include the appropriate serial, Ethernet, PCMCIA, IrDA, etc. drivers for that board. It's intuitive and quick.

Development Tools

Microware's Hawk development tool is designed to increase the efficiency of development under OS-9. Hawk's highly integrated tool set simplifies and automates the tasks of creating, debugging, analysing, and managing complex real-time software development projects. The HawkEye visual analysis tool enables you to find the source of bugs even more quickly.

PersonalJava™ Solution

PersonalJava™ technology, running on OS-9, reduces high-level application development time and enables you to take advantage of advanced development tools and the Java™ developer community. Tools like the Java code generator eliminate the learning curve, because any developer can start work right away. Java code, well known for portability, can be transferred across hardware and operating system platforms.

Start Development Sooner

You can begin development, even if there are some unknowns in the system design. You do not need to know the exact hardware configuration prior to development.

Take an example of a Point of Sale (POS) terminal based on OS-9 and Hitachi's SH-3 7709 microcontroller. Hitachi's EBX7709 development board features extensive I/O in an attempt to meet the needs of numerous applications. You immediately begin application development on this platform.

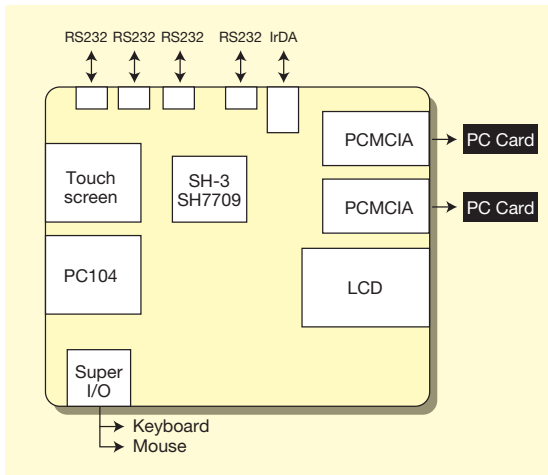


Figure 4. EBX7709 Diagram

Your final hardware design could resemble either of the following prototypes:

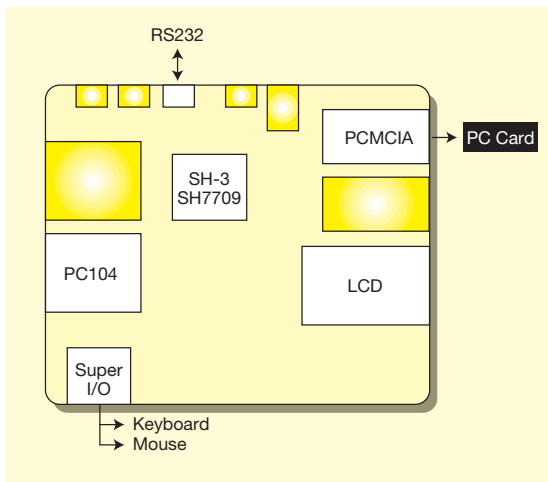


Figure 5. Example Prototype 1

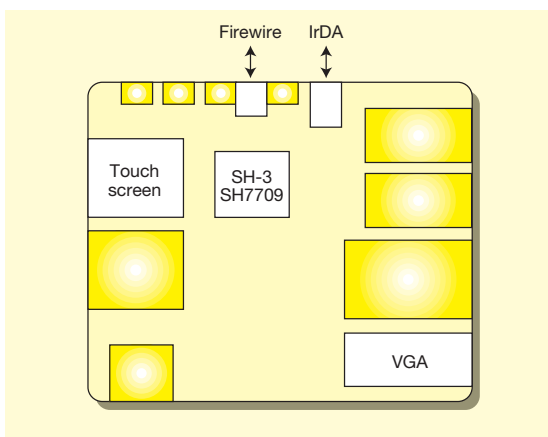


Figure 6. Example Prototype 2

Prototype 1 is the same as the EBX7709 development board, minus some of the I/O. Since OS-9 is modular with one device driver for each hardware controller, select the OS-9 drivers that are necessary for the devices on your chosen prototype. The other drivers

can be omitted to minimise your footprint, and the subtraction of drivers will not affect the application you have written.

The Example Prototype 2 includes the addition of VGA and firewire (IEEE1394). Even with the addition of new I/O, your application software will not be affected. For instance, when the application opens a socket for networking, it behaves no differently if that networking happens to be serial, like on the development hardware, or firewire, like on the example prototype.

SYSTEM SOFTWARE DEVELOPMENT

This article has discussed application software development, but many projects also involve some level of system software development. You must write a new device driver if you select a hardware controller that is not currently supported by OS-9. For example, if you prefer an IrDA controller that is neither on the development board, nor supported by OS-9, you can place that controller on your custom board and write a simple OS-9 driver yourself. System software development cannot begin until hardware is available, so the streamlined development cycle does not apply.

However, OS-9 minimises the effort of writing device drivers by utilising file managers (see Figure 3). File managers contain generic support for each class of I/O device, such as character oriented, block oriented, tape devices, etc. This leaves less code to be included in the device drivers, which handle the basic physical functions for specific hardware controllers. Since there is less to write, they take less time to develop, debug, and test. Furthermore, you can use the provided source code drivers as examples or templates.

QUALITY

It is challenging to find a balance between development speed and quality results. A project initially done in record time can end up costing more time in the long run if quality has been sacrificed. The RAD methodology with OS-9 accentuates both speed and quality, making it the preferred solution.

FOR MORE INFORMATION

To request free OS-9 technical documentation and evaluation software, contact Microware Systems Corporation in the U.K. at +44 (0) 1628 667578 or info@microware.co.uk.

Ann C. Schaffer is the product manager for the SuperH product team at Microware Systems Corporation. She earned a B.S. degree in International Management from Central College, Pella, Iowa, U.S.A. and has five years experience in the embedded systems industry. Ann has been involved in the development of products for digital television, communications, and embedded market segments.