



Getting Performance with Memory Protection in Real-time Windows Systems

by Rick Knowles, Vice President of Engineering, Ardence, Inc.

When designing Windows real-time embedded systems, maintaining deterministic performance is as critical as ensuring that the system itself runs error-free. Applications can be developed and deployed in Ring 3 or Ring 0. In fact, they can be developed in one mode yet deployed in the other. Development in the application mode (Ring 3) provides memory protection and application stability. However, to deploy with high system predictability and hard real-time determinism, the application must reside in the kernel mode (Ring 0). The following note outlines the proper technique for integrating the benefits of Ring 3 and Ring 0.

Development using Ring 3 provides the benefit of memory protection from the application, which aids in debugging the application by letting the hardware catch common programming errors. This enables rapid development, because in many cases, only the application will crash upon hitting a bug. With the operating system running, the application can be restarted and the debugging can be continued – resulting in a quick test, debug and fix cycle.

However, this does not hold true during deployment. A real-time system, by its very nature, requires the application and the operating system to be fully functioning and stable. The risk of damage to the operating system when an application faults is too large to rely on merely restarting the application – therefore the entire system should be restarted. An application is typically controlling and monitoring other systems. If it crashes, what happens to the other systems?

Most real-time systems require access to external sensors or other systems. During deployment, this leads to applications running in Ring 3 indirectly controlling Ring 0 software in order to access hardware. Because this use of Ring 0 software happens outside of the core application development, it is open to programming errors due to misunderstanding the details of the software

interfaces used. An example of this problem is commonly seen in the many security holes reported in the press about software containing the “buffer overflow” problem. However, security holes are not the only issue with the Ring 3 deployment model. With no direct access to hardware, deployment in Ring 3 leaves the system susceptible to a number of outcomes that can cause it to fall short of its requirements as a reliable real-time application, including:

- A performance penalty caused by the memory protection
- Additional time to tune for performance with no guaranteed outcome
- Unexpected loss of memory protection for any low-level calls
- Lack of scalability within a single system

When an application is designed and debugged in Ring 3, it still requires real-time performance for deployment. Keeping the application in the Windows application space (Ring 3) to maintain memory protection, often results in losing a significant level of real-time performance and predictability, which are critical features of any real-time system. Furthermore, if an application is already deployed in Ring 3 and the developer tries to increase performance, the memory protection assumed by the software will be lost – instantly negating the benefits of Ring 3 development. Few software developers would be happy to see their product fail or be late to market because of poor coding methods.

One of the few certainties of software based products is that features will continue to be added over time. With the Ring 3 approach, each new feature may require retuning the whole application to keep the minimal performance. And in most cases, the added cost of distributed processing with additional hardware dictates that scalability cannot be achieved in Ring 3 deployments. Support for complex applications requiring servicing of interrupts in the 30KHz range is not possible in a single platform.

But, by using the proper development and deployment techniques, system designers can leverage the benefits of Ring 3 and Ring 0 and deploy with high system predictability and hard real-time determinism. So while the degraded performance of deployment in Ring 3 might fall within a developer’s minimum performance requirements in some cases, it is easily possible to maintain optimized performance by deploying the debugged application in the Ring 0 kernel mode.

The benefits of kernel mode deployment are clear. This architecture provides the flexibility and performance to guarantee that the deployed application meets product requirements while enabling system scalability. Simply put, taking advantage of the protection mechanisms in Ring 3 during development

and then moving to Ring 0 to achieve optimal performance for deployment, is a superior design practice.

The Ring 0 deployment architecture technique allows the developer to:

- Develop in Ring 3 and deploy in Ring 0
- Ensure deterministic and predictable performance
- Integrate application and system scalability
- Support highly demanding applications with 30KHz sustained sample rates
- Rapidly move from prototype to production

Conclusions

- Memory protection leveraged by running in Ring 3 in the development stage is unnecessary for deployment. The reason for this is that memory protection functions by trapping a fault, halting the application. But at deployment, a down system is a down system.
- Keeping the application in the Windows application space (Ring 3) to obtain memory protection often results in losing a significant level of real-time performance and scalability.
- The ability to take advantage of the protection mechanisms in Ring 3 during development and then move to Ring 0 when appropriate to achieve optimal performance and scalability for deployment and the RTX Ring 0 architecture is the superior design solution.

by Rick Knowles, Vice President of Engineering, Ardenne, Inc.

January 24, 2006