Protect Your Software Investment

OSABSTRACTOR®

Design Better.
Reduce Risks.
Ease Upgrades.

Protect Your Software Investment

The Difficulty with Embedded Software Development

Developing embedded software is complicated. A software project has several critical elements contributing to its success. Issues with budgets, software tools, project schedules, hardware choices, and application requirements are just a few of the many elements involved in software projects. All of these factors are necessary, but not sufficient, for your project to finish successfully. Coordinating these project aspects and moving forward can be overwhelming, and one minor piece can make or break your product launch.

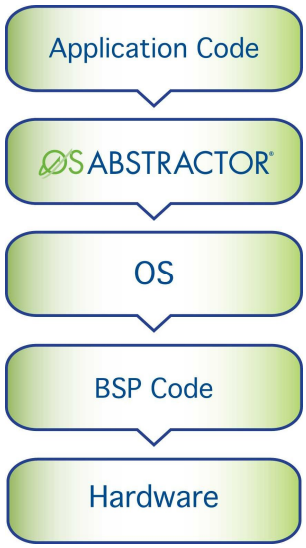
When you are choosing specific software tools for the project, there are several variables to consider. You must spend precious time evaluating the capabilities, verifying availability, confirming pricing, and making an overall decision for the fit of the tool. Significant time is invested in this process. You want time invested in evaluation to lead to an efficient and practical solution. A poor choice in tools could lead to project delays or a project cancellation.

One of the major software decisions you must make is the type of operating system (OS) that will be used in the project. The OS is a core component of the application design, and the OS capabilities must adequately fit the needs of the project. The choices surrounding the OS are numerous. You have over 100 different commercial OS vendors to evaluate. You must take into consideration many variables: What is the OS functionality? Has the OS been certified? What is the OS code size? How fast is the kernel? What is the interrupt latency? Are the price and business model right for my project? If any one of these variables goes awry, then your project has the potential for getting derailed.

By using OS Abstractor you gain a solid architecture for application development and eliminate the risk associated with the OS choice. By acting as a safety zone between your code and the application, the abstraction technology creates a seamless separator from the application code to the operating system, minimizing any application switching or upgrading costs with the operating system. OS Abstractor alleviates key issues surrounding your OS choice, including the support of new hardware. If you need to move to another OS or potentially change hardware vendors during the life of your project, OS Abstractor should be a key component of your software design.

OS Abstractor - Overview

In software programming practices, defining the software architecture is a vital element for proper application design. A strong architecture simplifies future activities like adding new features, maintenance, bug fixing, and optimization. In a software project with OS Abstractor, your architecture will look like this:



Your application has a clear and distinct separation according to solid programming practices. Your application code still makes calls to the underlying operating system, but the untidy coupling between your application and the OS is eliminated.

Should you need to move your code to another operating system or upgrade to a new system, the old target OS can simply be swapped out with the new one:

OS Abstractor allows you to use the native APIs in your application, if necessary. The library is designed to be fully portable with other commercial or proprietary real-time operating systems. It's independent of the target hardware and development tools (such as compilers and debuggers), which allows it to work with all the target hardware and toolsets supported by the underlying operating system.

Your application code is completely unaffected and you move your code to a new OS without modifying your application. Potential delays from unnecessary porting work are eliminated with OS Abstractor.

Using OS Abstractor in Your Application

OS Abstractor provides an important level of abstraction to your application. It also provides performance enhancement features like re-using kernel resources and allowing optimization of the abstraction code that is specific to your application. OS Abstractor was designed for embedded application development. It provides software design abstraction without affecting your application's performance.

Many roll-your-own abstraction solutions add significant overhead to the application code. OS Abstractor's code size is minimal, with overhead as little as 10K depending on the architecture, tools, and operating system. Since OS Abstractor is written in C, it easily integrates into your existing C/C++ and Ada software projects.

Home-grown products perform the abstraction by using a wrapper implementation. Wrappers are easy to implement and straightforward to understand, however they suffer from several problems. First, in-house wrapper methods do not always cover the majority of the operating system function calls. OS Abstractor provides strong coverage for the most popular functions used in modern commercial real time operating systems.

Another drawback of an in-house wrapper is that a proprietary solution only abstracts the function calls. OS Abstractor abstracts all aspects of the operating system, not just the function calls. The control blocks, APIs, header files and data types are all abstracted, providing you with complete separation from your operating system.

When possible, OS Abstractor uses the compiler preprocessor to parse your application code for the MapuSoft function calls, and then translates them to the appropriate operating system call. By leveraging the compiler preprocessor, we are able to eliminate the overhead associated with the wrapper solution. The preprocessor performs the necessary mapping and translation between the application code and operating system calls. As an application programmer, you write your code using C/C++ and let OS Abstractor perform all the necessary translation for your specific operating system.

OS Abstractor uses several techniques to improve your application's run-time performance. For instance, OS Abstractor can be configured to create a pool of resources statically during initialization and transparently manage them during run-time. OS Abstractor includes profiling tools to identify the APIs heavily used by the application and automatically optimize them by eliminating their function wrappers. You can also quickly and easily disable error checking and debug data for additional performance gains.

OS Abstractor eliminates dynamic memory allocation by pre-allocating memory during the operating system initialization. It also ensures that the application will not use more memory than the allocated amount to prevent the system from running out of memory and impacting other applications. The OS Abstractor design minimizes the occurrences of task switching and user/kernel mode switching to improve your applications run-time performance.

Changing Hardware

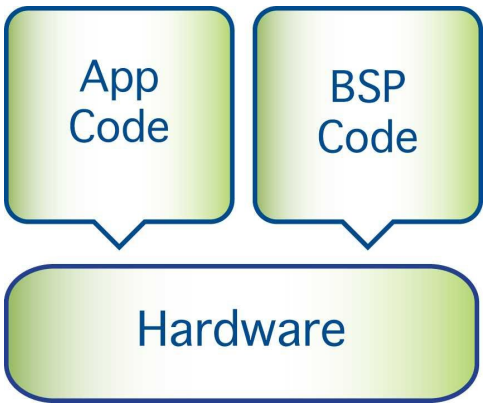
One major factor that disrupts embedded software development is changing hardware. Bringing in new hardware, whether it's a new CPU or development board, can wreak havoc with your application development. Hardware changes can mean hours of re-work and code updates. OS Abstractor alleviates these problems and reduces your future work.

Most embedded software development is divided into two groups: low-level programming and application development. The low-level programmers deal with hardware issues like writing software drivers, start-up configuration, hardware boot, soft / hard device resets, and device initialization.

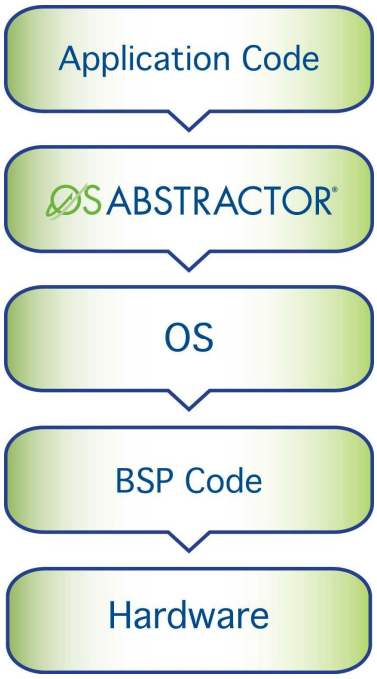
The application development team creates the high-level functionality of the product. These are the features that will be used by the end user of the

application. Application developers focus on the core functionality of the application. They make several assumptions about the state of the hardware – they assume it's working properly and has been initialized. They don't worry about memory integrity checks, start-up sequences, initialization, or other power issues with the hardware. Those issues normally fall into the hands of the programming team handling the BSP and device driver development. Unfortunately, there isn't a clear division of the software layers.

When the hardware changes, there are going to be some software changes. It really doesn't make much sense to re-work high level application code. With OS Abstractor changes can be made to the low-level code without impacting the application. Unfortunately most software designs don't provide this separation which protects the original application code.



Poorly designed architecture will create future coding issues



Well-designed architecture create stable base for future coding and maintenance

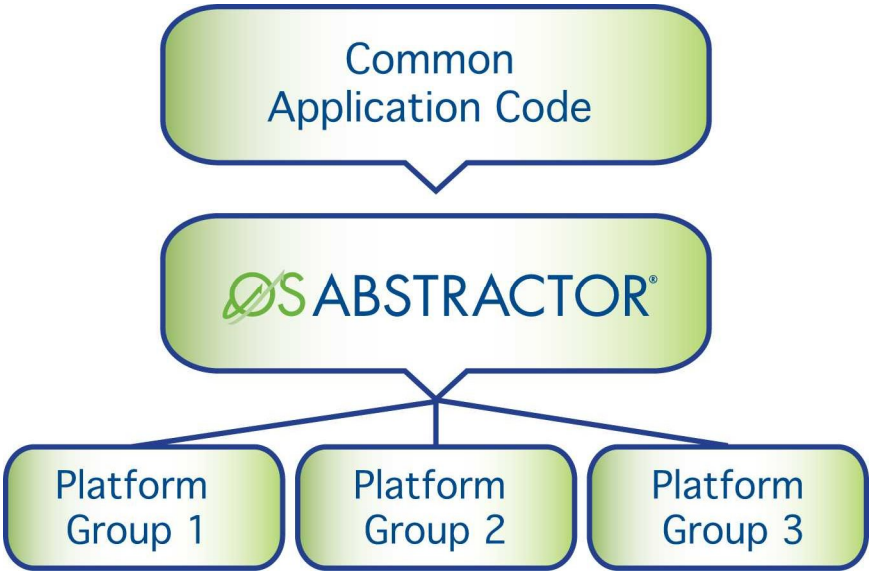
OS Abstractor provides a blanket of security for these situations. It allows you to continue your application development while the BSP team deals with the hardware change. It provides a layer of separation between your BSP / driver development and your application development. So when you do change hardware, you can continue with your application development safely and on schedule. Using OS Abstractor, you create a software layer that eliminates the need to modify the application code.

Many times, this hardware change is unexpected. Either a board or CPU is discontinued, a new peripheral is added, or the performance is improved so drastically that there's no alternative but to jump to the next version of the hardware.

Additional reasons for changing hardware are:

- A new development board is released
- A new set of peripherals are added
- A CPU upgrade for speed, functionality, or manufacturing
- A marketing mandate to support multiple processors
- A CPU has become popular with customers

There is one additional scenario for supporting multiple hardware platforms. Sometimes application code needs to be shared across multiple groups or divisions. The individual groups may be targeting different markets or could be working on product enhancements. Sharing the common application code across these groups creates significant work for supporting each individual platform. By using OS Abstractor, this sharing of code is made easier, since all groups are using a common API.



Hardware Changes from Industry Consolidation

A fragmented market is characterized as having many vendors and products. In the embedded world, there are hundreds of options for operating systems and hardware platforms. Among other things, the options vary greatly on performance, business model, and product features. These variations make for a textbook example of a fragmented market.

In a fragmented market, there is one thing that always occurs - consolidation. Industry consolidation is typified by mergers and acquisitions (M&A) between related as well as rival firms. You can watch the business news each week and see announcements in various markets. The embedded and semiconductor markets have seen consolidation over the years. Wind River acquired ISI back in the late 1990s. Renesas came from Hitachi and Mitsubishi. AMD acquired ATI. Mentor Graphics acquired Accelerated Technology, and recently, Intel acquired Wind River.

Industry consolidation through M&A is more important than a simple partnership or alliance, because there is a transfer of ownership between two companies. This allows a single management team to drive the direction of the newly acquired company. From a management point of view, this provides more control and, in theory, produces better results when compared with a simple partnership.

Despite the positives, there is also a dark side to mergers and acquisitions. The dark side occurs when existing operating systems support fewer hardware platforms. If the hardware you manufacture is no longer supported, then your customers may be forced to change their platform, potentially moving to your competitor's hardware architecture.

OS Abstractor provides a migration path away from the discontinued operating systems. It allows your hardware to immediately support up to twenty commercial operating systems from major OS vendors.

CPU vendors / silicon vendors

Silicon and CPU vendors are in an interesting position. When they develop a new CPU, there is the task of getting customers to use their new platform. These new processors offer the potential for new and innovative devices because of their improved performance, but there is limited software support.

From a customer's perspective, there may be resistances to these upgrades from the old CPU because of one thing – their application software is written to one individual platform. Moving to a new platform means weeks, if not months, of updating and re-writing software. OS Abstractor eliminates the changes needed to the application code, making the change to a new CPU virtually seamless.

When new hardware platforms are introduced, OS Abstractor provides a software migration path to the new platform. An example of this would be the introduction of the Atom architecture from Intel. This new architecture represents a future path for Intel and its customers. The Atom is designed for low power devices and targets mobile applications, Netbooks, and entry-level PCs. If you, as a developer, want to move your software from the x86 architecture, there will be significant work associated with the transition. However, OS Abstractor provides you with the capability to quickly move your application code to this new architecture with little rework. The low-level drivers and BSP will require modification, but OS Abstractor eliminates changes to your application code.

There is another issue when changing CPUs or hardware platforms - How can you make sure your application is performing at the same level?

By using the profiler with OS Abstractor, you're able to monitor your application performance as you make the transition. Using the OS Abstractor profiling options, you can record multiple sessions and track the metrics for performance. Then, as you make the change to the new hardware system, you can keep a running record of the application's performance. As you are progressing, you generate a Timing Comparison Report to compare two different sets of measurements and reports. You make changes as needed to maintain your particular performance metrics. This will help guarantee that your application is performing efficiently and accurately.

Middleware and Platform Vendors

Some embedded companies need to support multiple operating systems. These software vendors provide application middleware or software platforms to developers. For these companies, the more operating systems they support, the larger their customer base. As these vendors grow and look for new customers, they spend more time and money adding support for new operating systems. OS Abstractor provides a tool to quickly add support for multiple operating systems without incurring the costs of years of engineering software effort.

Let's briefly define some terms. A middleware vendor is a company that provides software modules to developers that perform specific functions. These modules are operating system independent, meaning they run on any operating system. For example, a networking stack or a graphics engine would be considered middleware. A platform vendor provides a software tool or set of services that provides an embedded developer a 'platform' to build an application. Examples of platform providers include Rhapsody (UML) and the military's System of Systems Common Operating Environment (SoSCOE).

Some software products (for example, an optimized graphics rendering engine), work and run with multiple operating systems. The graphics engine is designed to do only one thing – image rendering. This creates a Catch-22 situation. For the vendor, they can focus on their area of expertise and create a high-performance product. But when it comes to deploying it into a real world application, they need the support of the operating system for target initialization, memory management, process/task creation, and scheduling. And this means supporting an operating system.

The work of supporting an operating system involves the tedious tasks of determining a suitable hardware platform, configuring the operating system demo code, and incorporating the appropriate API calls into their graphics engine. This engineering porting work is done each time the vendor increases their list of 'supported' operating systems. Eventually, over many months and many years, they increase their supported operating systems list to a modest number of operating systems, maybe 5 or 6. This is a very modest number considering the amount of effort and time that was put into the effort.

By using OS Abstractor in their projects, these vendors could add support for multiple operating systems and instantly widen their potential customer base. Since OS Abstractor supports a large number of the major operating systems, a middleware vendor can save months of effort and a considerable amount of money in porting work. For the middleware and platform vendor, their success depends on the number of operating systems they support. Supporting more operating systems translates into more customers for your product. By using OS Abstractor, the work of supporting multiple operating systems is minimized, saving precious time, resources, and money.

Linux Optimization and OS Abstractor

Many embedded developers use Linux in their projects. Today, Linux is running on many devices, from cell phones to medical devices to netbooks. However some developers are feeling additional pressure during their application development. As their application grows and expands, they need to get an additional performance squeeze from the embedded Linux operating system. To achieve this additional performance means that they will need to dig into the existing Linux operating system code, which could be disastrous. OS Abstractor provides several options for Linux users to realize higher performance.

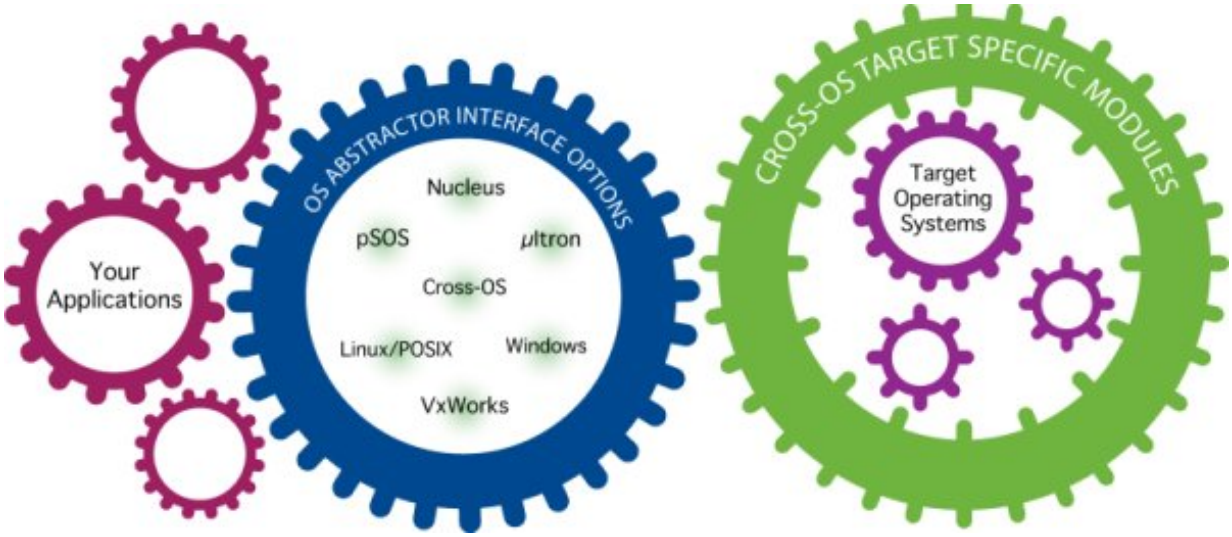
OS Abstractor, in addition to providing an abstraction, was built to provide additional performance for all embedded applications. For Linux, this means you don't need to keep and maintain multiple 'variants' of your application in order to keep track of the changes in the POSIX API. OS Abstractor allows you to easily incorporate those new APIs into your application, which eliminates having multiple copies of your application code around. There are also multiple optimization switches that are available for increased performance, including eliminating the need to switch between kernel and user mode, adding mission critical features, limiting memory allocation settings. These features are easily configured using the OS Abstractor Target Code Generator.

OS Abstractor also provides API optimization, which is a way to get additional performance by optimizing the most used functions in your application. OS Abstractor reads the application source code, determines the services used by your application, and produces operating system interface code optimized for your application and target platform. Even with these optimizations, if you need additional performance tuning, you can use the profiling feature of OS Abstractor.

For those sections of the application needing extra attention, use the profiling options to fix your application bottlenecks. Profiling can be done to accurately pinpoint the areas of the application that are performing poorly and adjust them.

OS Abstractor - The Solution

OS Abstractor provides an excellent choice for your software project and offers several benefits for your software application.



Reduce your the learning curve

One of the most under-estimated tasks in a software project is the ramp-up time developers must go through to become productive on a new operating system. The cost of learning a new operating system is significant. OS Abstractor offers an easy-to-learn interface that can be re-used across projects, reducing your team's learning curve and increasing their productivity. Alternatively, you can also continue developing using your existing API and let OS Abstractor handle the work of supporting the new operating system.

Good programming techniques reduce un-necessary bugs and help maintenance

Having a solid architecture brings many benefits to a software project. A solid architecture allows for better testing during both unit testing and system testing. Project activities of adding new features and maintenance are significantly easier when an application has a solid architecture.

Reduce your risk and dependency on a single OS vendor

Coupling too closely to the OS can lead to dependency on the future of the operating system vendor. As vendors drop support for your hardware platform or prices change, your project could be jeopardized. Using OS Abstractor helps reduce dependency on the OS vendors.

Provide additional functionality that your OS may be missing

OS Abstractor also provides additional functionality to your development platform and OS. Many of these features are available through OS Abstractor even if your operating system doesn't support them. These features include application profiling capabilities, application optimization features, and API optimization. You can also use additional features in your application like software based processes, shared memory functionality, setting memory allocation limits, thread pooling, mission critical features, and zero-copy message queues. All of these advanced features are available to your application through OS Abstractor, even if the underlying operating system does not support them.

Reduces work when changing hardware

Hardware changes can cause major re-writes of your application code. Using the OS Abstractor helps isolate your application code from the CPU which significantly reduces the work required for future hardware changes. By providing a separation between your application and the hardware, only low-level hardware changes are needed. There is no need to modify the application code when you move to a new hardware platform.

About Mapusoft

MapuSoft Technologies (MT) is the number one provider of embedded software re-usability solutions and services that are designed to protect software investment by providing customers a greater level of flexibility and control with product development. In addition to off-the-shelf tools, MT offers porting, integration, support and training services to help developers easily migrate from legacy platforms to the next generation. We believe that our advanced software and vision will revolutionize the embedded software industry. We are working hard to provide software that is practical, familiar, financially reasonable, and easily operable. We provide full source code with no royalty fees. Our licensing strategy makes it extremely affordable for you to incorporate our products into your embedded applications. In addition, our attention to engineering detail provides you with robust software and requires minimal technical maintenance.

About OS Abstractor

OS Abstractor is a C/C++ source-level virtualization technology that provides a flexible and robust real-time application development framework, which prevents your software from being locked to a specific operating system (OS) or version. This negates future porting issues because your software will support multiple operating systems and versions from the start of your project. It also eliminates the risk associated with the OS selection process, since the same application can be tested on multiple platforms for comparison and won't be locked-in to the chosen OS.

For more information

To download MapuSoft's free software evaluation visit:
http://mapusoft.com/downloads/

To learn more about our licenses and request a quote visit:
http://mapusoft.com/downloads/request-a-quote/

OS Abstractor Development Kit Overview Datasheet

OS Abstractor Development Kit Technical Datasheet

Contact Information

For more information about OS Abstractor or Mapusoft, please contact us at:

US Headquarters

MapuSoft Technologies, Inc.
1301 Azalea Road
Mobile, AL 36693

Tel: (251) 665-0280
Toll Free: 1-877-MAPUSOFT (1-877-627-8763)
Fax: (251) 665-0288

www.mapusoft.com

E-mail: info@mapusoft.com or sales@mapusoft.com

For a complete listing of International Offices, please click here.

Mapusoft's OS PAL: