

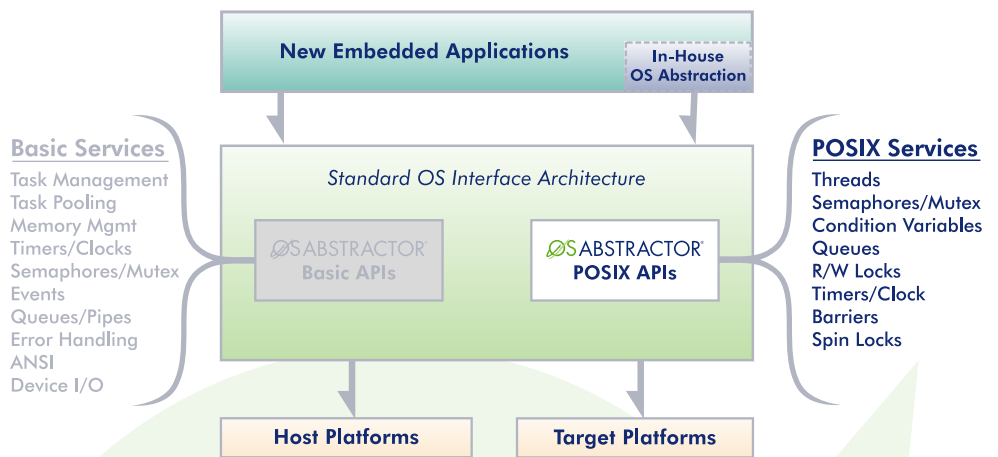
Write Portable Code – Protect Your Investment<sup>SM</sup>

## OS Abstractor with POSIX

OS Abstractor provides you a robust and standard OS interface architecture for flexible application development while eliminating the risks associated with selecting an OS and dependency on a single vendor. OS Abstractor makes your application adapt to multiple operating system platforms with a standard OS interface, thereby reducing cost associated with code maintenance and learning multiple operating systems. OS Abstractor with POSIX enhances the OS Abstractor Basic standard OS interface architecture with the addition of optimized non-proprietary and industry standard POSIX APIs to facilitate using open source POSIX/Linux in your design.

### OS Abstractor with POSIX Highlights

- Vendor independent and industry standards based solution protects your code investment and knowledge-base
- Leverage re-usable open source POSIX/Linux code to efficiently add feature rich services economically regardless of the underlying OS
- Get to market faster with compelling open-source applications and content in your design
- Tap into the large talent pool of engineers with POSIX/Linux experience
- Offers a high level of code re-usability across many supported POSIX and non-POSIX OS
- Scalability at component & individual feature levels to lower memory foot-print
- Offers POSIX OS functionality to enable complex applications to work on a single-memory address based real-time OS
- Set limit on individual application's heap memory
- Highly optimized for speed and memory footprint for each specific target OS
- Includes OS Abstractor Basic APIs for added flexibility in your development (refer to OS Abstractor Basic Datasheet)
- Easily connects to your in-house abstraction solution
- Easily extended to support your proprietary OS to enable POSIX compliance and/or to re-use open source solutions
- Offered royalty free and with source code



OS Abstractor with POSIX Product Overview

### Using OS Abstractor with POSIX

OS Abstractor with POSIX is designed for use as a fully scalable C library. Services used inside your application software are extracted from the OS Abstractor libraries and are combined with the other application objects to produce the complete image. This image may be downloaded to the target platform or placed in ROM on the target platform. Application developers need to specify the OS for the application and also include the required OS Abstractor libraries while building the application. Application developers can also select the individual OS Abstractor components that are needed and exclude the ones that are not required.

# OS Abtractor with POSIX API Support

The table below lists the POSIX API support offered across various OS platforms. Please note that there may be more POSIX API support available on some selected OS and also support for new APIs are constantly added in newer releases. Please contact MapuSoft to find out the latest POSIX API support for your target OS platform and refer to the corresponding reference manual for API compliance special notes.

POSIX API	VxWorks 5.x	VxWorks 6.x	Nucleus	Win XP	Win CE	ThreadX	MQX	T-Kernel	uTRON
<b>POSIX Threads</b>									
pthread_attr_getinheritsched	*	*	*	*	*	*	*	*	*
pthread_attr_getschedpolicy	*	*	*	*	*	*	*	*	*
pthread_attr_getscope	*	*	*	*	*	*	*	*	*
pthread_attr_setinheritsched	*	*	*	*	*	*	*	*	*
pthread_attr_setschedpolicy	*	*	*	*	*	*	*	*	*
pthread_attr_setscope	*	*	*	*	*	*	*	*	*
pthread_attr_getstackaddr	*	*	*	*	*	*	*	*	*
pthread_attr_setstackaddr	*	*	*	*	*	*	*	*	*
pthread_attr_getstacksize	*	*	*	*	*	*	*	*	*
pthread_attr_setstacksize	*	*	*	*	*	*	*	*	*
pthread_attr_destroy	*	*	*	*	*	*	*	*	*
pthread_attr_getdetachstate	*	*	*	*	*	*	*	*	*
pthread_attr_getschedparam	*	*	*	*	*	*	*	*	*
pthread_attr_init	*	*	*	*	*	*	*	*	*
pthread_attr_setdetachstate	*	*	*	*	*	*	*	*	*
pthread_attr_setschedparam	*	*	*	*	*	*	*	*	*
pthread_attr_getstack	*	*	*	*	*	*	*	*	*
pthread_attr_setstack	*	*	*	*	*	*	*	*	*
pthread_create	*	*	*	*	*	*	*	*	*
pthread_setcancelstate	*	*	*	*	*	*	*	*	*
pthread_cancel	*	*	*	*	*	*	*	*	*
pthread_testcancel	*	*	*	*	*	*	*	*	*
pthread_detach	*	*	*	*	*	*	*	*	*
pthread_equal	*	*	*	*	*	*	*	*	*
pthread_exit	*	*	*	*	*	*	*	*	*
pthread_getspecific	*	*	*	*	*	*	*	*	*
pthread_join	*	*	*	*	*	*	*	*	*
pthread_key_create	*	*	*	*	*	*	*	*	*
pthread_key_delete	*	*	*	*	*	*	*	*	*
pthread_once	*	*	*	*	*	*	*	*	*
pthread_self	*	*	*	*	*	*	*	*	*
pthread_setcanceltype	*	*	*	*	*	*	*	*	*
pthread_setspecific	*	*	*	*	*	*	*	*	*
pthread_sigmask	*	*	*	*	*	*	*	*	*
pthread_getschedparam	*	*	*	*	*	*	*	*	*
pthread_cleanup_pop	*	*	*	*	*	*	*	*	*
pthread_cleanup_push	*	*	*	*	*	*	*	*	*

POSIX API	VxWorks 5.x	VxWorks 6.x	Nucleus	Win XP	Win CE	ThreadX	MQX	T-Kernel	uITRON
<b>Mutex</b>									
pthread_mutex_timedlock	*	*	*	*	*	*	*	*	*
pthread_mutexattr_getprotocol	*	*	*	*	*	*	*	*	*
pthread_mutex_getprioceiling	*	*	*	*	*	*	*	*	*
pthread_mutex_setprioceiling	*	*	*	*	*	*	*	*	*
pthread_mutexattr_getprioceiling	*	*	*	*	*	*	*	*	*
pthread_mutexattr_setprioceiling	*	*	*	*	*	*	*	*	*
pthread_mutexattr_setprotocol	*	*	*	*	*	*	*	*	*
pthread_mutexattr_getpshared	*	*	*	*	*	*	*	*	*
pthread_mutexattr_setpshared	*	*	*	*	*	*	*	*	*
pthread_mutex_destroy	*	*	*	*	*	*	*	*	*
pthread_mutex_init	*	*	*	*	*	*	*	*	*
pthread_mutex_lock	*	*	*	*	*	*	*	*	*
pthread_mutex_trylock	*	*	*	*	*	*	*	*	*
pthread_mutex_unlock	*	*	*	*	*	*	*	*	*
pthread_mutexattr_destroy	*	*	*	*	*	*	*	*	*
pthread_mutexattr_init	*	*	*	*	*	*	*	*	*
pthread_mutexattr_gettype	*	*	*	*	*	*	*	*	*
pthread_mutexattr_settype	*	*	*	*	*	*	*	*	*
<b>Semaphores</b>									
sem_close	*	*	*	*	*	*	*	*	*
sem_destroy	*	*	*	*	*	*	*	*	*
sem_getvalue	*	*	*	*	*	*	*	*	*
sem_init	*	*	*	*	*	*	*	*	*
sem_open	*	*	*	*	*	*	*	*	*
sem_post	*	*	*	*	*	*	*	*	*
sem_trywait	*	*	*	*	*	*	*	*	*
sem_unlink	*	*	*	*	*	*	*	*	*
sem_wait	*	*	*	*	*	*	*	*	*
<b>Queues</b>									
mq_close	*	*	*	*	*	*	*	*	*
mq_getattr	*	*	*	*	*	*	*	*	*
mq_notify	*	*	*	*	*	*	*	*	*
mq_open	*	*	*	*	*	*	*	*	*
mq_receive	*	*	*	*	*	*	*	*	*
mq_send	*	*	*	*	*	*	*	*	*
mq_setattr	*	*	*	*	*	*	*	*	*
mq_unlink	*	*	*	*	*	*	*	*	*
mq_timedreceive	*	*	*	*	*	*	*	*	*
mq_timedsend	*	*	*	*	*	*	*	*	*

POSIX API	VxWorks 5.x	VxWorks 6.x	Nucleus	Win XP	Win CE	ThreadX	MQX	T-Kernel	uTRON
<b>R/W Locks</b>									
pthread_rwlockattr_getpshared	*	*	*	*	*	*	*	*	*
pthread_rwlockattr_setpshared	*	*	*	*	*	*	*	*	*
pthread_rwlock_destroy	*	*	*	*	*	*	*	*	*
pthread_rwlock_init	*	*	*	*	*	*	*	*	*
pthread_rwlock_rdlock	*	*	*	*	*	*	*	*	*
pthread_rwlock_tryrdlock	*	*	*	*	*	*	*	*	*
pthread_rwlock_trywrlock	*	*	*	*	*	*	*	*	*
pthread_rwlock_unlock	*	*	*	*	*	*	*	*	*
pthread_rwlock_wrlock	*	*	*	*	*	*	*	*	*
pthread_rwlockattr_destroy	*	*	*	*	*	*	*	*	*
pthread_rwlockattr_init	*	*	*	*	*	*	*	*	*
pthread_rwlock_timedrdlock	*	*	*	*	*	*	*	*	*
pthread_rwlock_timedwrlock	*	*	*	*	*	*	*	*	*
<b>Condition Variables</b>									
pthread_condattr_getclock	*	*	*	*	*	*	*	*	*
pthread_condattr_setclock	*	*	*	*	*	*	*	*	*
pthread_condattr_getpshared	*	*	*	*	*	*	*	*	*
pthread_condattr_setpshared	*	*	*	*	*	*	*	*	*
pthread_cond_broadcast	*	*	*	*	*	*	*	*	*
pthread_cond_destroy	*	*	*	*	*	*	*	*	*
pthread_cond_init	*	*	*	*	*	*	*	*	*
pthread_cond_signal	*	*	*	*	*	*	*	*	*
pthread_cond_timedwait	*	*	*	*	*	*	*	*	*
pthread_cond_wait	*	*	*	*	*	*	*	*	*
pthread_condattr_destroy	*	*	*	*	*	*	*	*	*
pthread_condattr_init	*	*	*	*	*	*	*	*	*
<b>Clock</b>									
clock	*	*	*	*	*	*	*	*	*
clock_getres	*	*	*	*	*	*	*	*	*
clock_gettime	*	*	*	*	*	*	*	*	*
clock_settime	*	*	*	*	*	*	*	*	*
clock_nanosleep	*	*	*	*	*	*	*	*	*
sleep	*	*	*	*	*	*	*	*	*
usleep	*	*	*	*	*	*	*	*	*
nanosleep	*	*	*	*	*	*	*	*	*
sched_yield	*	*	*	*	*	*	*	*	*
<b>Barriers</b>									
pthread_barrierattr_destroy	*	*	*	*	*	*	*	*	*
pthread_barrierattr_getpshared	*	*	*	*	*	*	*	*	*
pthread_barrierattr_init	*	*	*	*	*	*	*	*	*
pthread_barrierattr_setpshared	*	*	*	*	*	*	*	*	*

POSIX API	VxWorks 5.x	VxWorks 6.x	Nucleus	Win XP	Win CE	ThreadX	MQX	T-Kernel	uITRON
<b>Spin Locks</b>									
pthread_spin_destroy	*	*	*	*	*	*	*	*	*
pthread_spin_init	*	*	*	*	*	*	*	*	*
pthread_spin_lock	*	*	*	*	*	*	*	*	*
pthread_spin_trylock	*	*	*	*	*	*	*	*	*
pthread_spin_unlock	*	*	*	*	*	*	*	*	*
<b>Timers</b>									
timer_create	*	*	*	*	*	*	*	*	*
timer_delete	*	*	*	*	*	*	*	*	*
timer_gettime	*	*	*	*	*	*	*	*	*
timer_settime	*	*	*	*	*	*	*	*	*
<b>Miscellaneous</b>									
glob	*	*	*	*	*	*	*	*	*
globfree	*	*	*	*	*	*	*	*	*
getenv	*	*	*	*	*	*	*	*	*
setenv	*	*	*	*	*	*	*	*	*
clock_getcpuclockid	*	*	*	*	*	*	*	*	*
<b>Device I/O – ANSI</b>									
OS_Creat	*	*	*	*	*	*	*	*	*
OS_unlink	*	*	*	*	*	*	*	*	*
OS_remove	*	*	*	*	*	*	*	*	*
OS_open	*	*	*	*	*	*	*	*	*
OS_close	*	*	*	*	*	*	*	*	*
OS_rename	*	*	*	*	*	*	*	*	*
OS_read	*	*	*	*	*	*	*	*	*
OS_write	*	*	*	*	*	*	*	*	*
OS_ioctl	*	*	*	*	*	*	*	*	*
OS_lseek	*	*	*	*	*	*	*	*	*
OS_chdir	*	*	*	*	*	*	*	*	*
OS_getcwd	*	*	*	*	*	*	*	*	*
OS_getwd	*	*	*	*	*	*	*	*	*
OS_Printf	*	*	*	*	*	*	*	*	*
OS_Sprintf	*	*	*	*	*	*	*	*	*
<b>Memory Management - ANSI</b>									
OS_malloc	*	*	*	*	*	*	*	*	*
OS_free	*	*	*	*	*	*	*	*	*

## MapuSoft Technologies, Inc.

Porting embedded applications from one OS to another OS is often an underestimated, tedious and time-consuming task. It also requires expensive and skillful resources that take away the focus on building your product. Embedded applications demand more and more performance, scalability and development flexibility from the underlying OS. Developers are forced to change their OS or extend support for more than one OS quickly as the market demands. Developers find that they need to leverage the existing software and knowledge base when migrating to next generation platforms. This has brought a need for the development of highly re-usable software that can run across proprietary and multiple commercial operating systems as well as utilizes open source components or other low cost alternatives.

It's not easy for developers to adapt existing software to a new OS or enable it to support multiple operating systems without incurring high costs and increasing time to market entry. MapuSoft offers *OS PAL*, *OS Abstractor* and *OS Changer* products to help developers streamline development processes and re-use their embedded software on one or more operating systems. MT offers porting, integration, support and training services to help developers easily migrate from legacy platforms to the next generation.

## MapuSoft Custom Services

- Provide full porting, integration and validation services
- Extend OS Changer APIs
- Migrate in-house abstraction to OS Abstractor framework
- Add OS Abstractor support to your proprietary operating system
- Offer on-site and off-site training on operating systems and advanced porting techniques

# FREE Trial Software

Go to [www.mapusoft.com](http://www.mapusoft.com) to download a free evaluation of MapuSoft's products on host platforms.

## Contact

### Corporate Headquarters

MapuSoft Technologies, Inc.  
1301 Azalea Road  
Mobile, AL 36693  
Tel: 251-665-0280  
Fax: 251-665-0288  
[info@mapusoft.com](mailto:info@mapusoft.com)  
[www.mapusoft.com](http://www.mapusoft.com)

### India

MapuSoft Technologies (P) Ltd.  
#19, 17th Street  
Periyar Nagar  
Chennai 600082, India  
Tel: [91] 44-25504291  
Fax: [91] 44-25504291

Please contact MapuSoft corporate headquarters for international distributor information and locations.