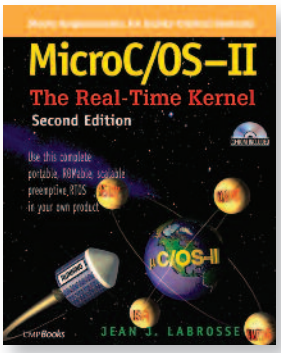
FEATURES AND BENEFITS

µC/OS-II is a portable, ROMable, scalable, preemptive real-time, deterministic, multitasking kernel for microprocessors, microcontrollers and DSPs. µC/OS-II can manage up to 250 application tasks and provides the following services:

Semaphores | Event Flags | Mutual Exclusion Semaphores (to reduce priority inversions) | Message Mailboxes | Message Queues | Task Management | Time Management | Fixed Sized Memory Block Management



µC/OS-II comes with ALL the source code. In fact, the source code is 100% portable ANSI C and is probably the cleanest and most consistent code of any kernel. The internals of µC/OS-II are described in the book Micro µC/OS-II, The Real-Time Kernel (ISBN 1-57820-103-9) by Jean J. Labrosse. All services provided by µC/OS-II start with the prefix 'OS'. This makes it easier to know that the functions refer to kernel services in your application. Also, the services are neatly grouped by categories: OSTask???() relate to task management functions, OSQ???() relate to message queue management, OSSem???()relate to semaphore management etc.

A validation suite has been developed for µC/OS-II and provides all the documentation necessary to prove that µC/OS-II is suitable for Safety Critical Systems common to Aviation and Medical products. Although this feature may not be applicable to your needs, it does prove that µC/OS-II is a very robust Kernel.

µC/OS-II is now 99% compliant with the Motor Industry Software Reliability Association (MISRA) C Coding Standards. These standards were created by MISRA to improve the reliability and predictability of C programs in critical automotive systems. A detailed µC/OS-II compliance matrix describing all of MISRA's 127 C Coding Rules is available from Micrium's website.

µC/OS-II runs on a large number of processor architectures and ports are available (FREE download) from our web site. The vast number of ports should convince you that µC/OS-II is truly very portable and thus will most likely be ported to new processors as they become available. Architectures supported by µC/OS-II include ARM7, ARM9, Cortex-M1, Cortex-M3, AVR, AVR32, M16C, M32C, MicroBlaze, Nios II, PIC24, dsPIC33, PIC32, and PowerPC. For a comprehensive listing of supported devices, please consult Micrium's Web site at **www.micrium.com/products/rtos/kernel/ports.html**

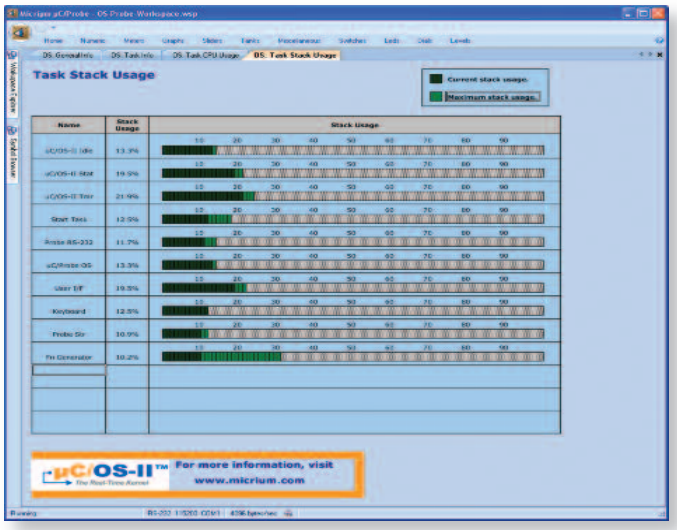
µC/OS-II's footprint can be scaled to only contain the features you need for your application. Consult the following webpage for more details: **www.micrium.com/products/rtos/kernel/benefits.html**

The execution time for most of the services provided by µC/OS-II is both constant and deterministic. This means that the execution times do not depend on the number of tasks running in your application.

µC/OS-II has been used in hundreds of products from companies all around the world. Many colleges and Universities worldwide are also using µC/OS-II in curriculum teaching the subject of real-time systems. This ensures that engineers in the workplace are trained and ready to use µC/OS-II in your products.

µC/OS-II monitoring

Via µC/Probe, Micrium's award-winning monitoring tool, you can use a Windows PC to visualize your µC/OS-II-based applications. The µC/OS-II plug-in that is provided with µC/Probe features intuitive data screens that allow you to keep track of stack sizes, CPU usage, and other key indicators of your application's status. µC/Probe's Windows application receives such information from your embedded system via either RS-232C, USB, TCP/IP, or even JTAG. For additional information relating to the exciting visualization capabilities that µC/Probe offers, please turn to page 18.



µC/OS-II KA

µC/OS-II KA (Kernel Awareness Plug-In) allows you to display µC/OS-II's internal data structures in a convenient series of Windows integrated with the C-SPY Debugger within the IAR Embedded Workbench. This provides you with information about each of the active tasks in the target application, about each semaphore, mutex, mailbox, queue and event flag group along with a list of all the tasks waiting on these kernel objects, and more. This can become very useful to the embedded developer when testing and debugging applications. Other debuggers also provide Kernel Awareness for µC/OS-II: Lauterbach, Nohau, iSystem and others.

µC/OS-OSEK™

OSEK/VDX Abstraction Layer

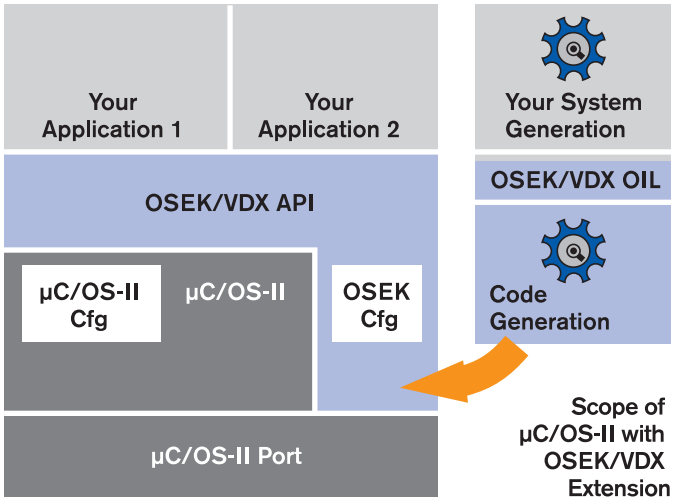
This µC/OS-II extension provides the user with a certified OSEK/VDX application programming interface. The OSEK/VDX Extension supports the OS conformance classes BCC1 and ECC1. The COM conformance classes CCCA and CCCB are provided for internal communication.

During the standard development process in OSEK/VDX system development, the system architect generates one or more system description files, containing the "OSEK/VDX Implementation Language" (OIL). The OSEK/VDX Extension includes a template based code generation tool, which allows the automatic generation of the necessary target configuration files. The code generation tool is a command line tool, which can be integrated in any build process.

Certified environments

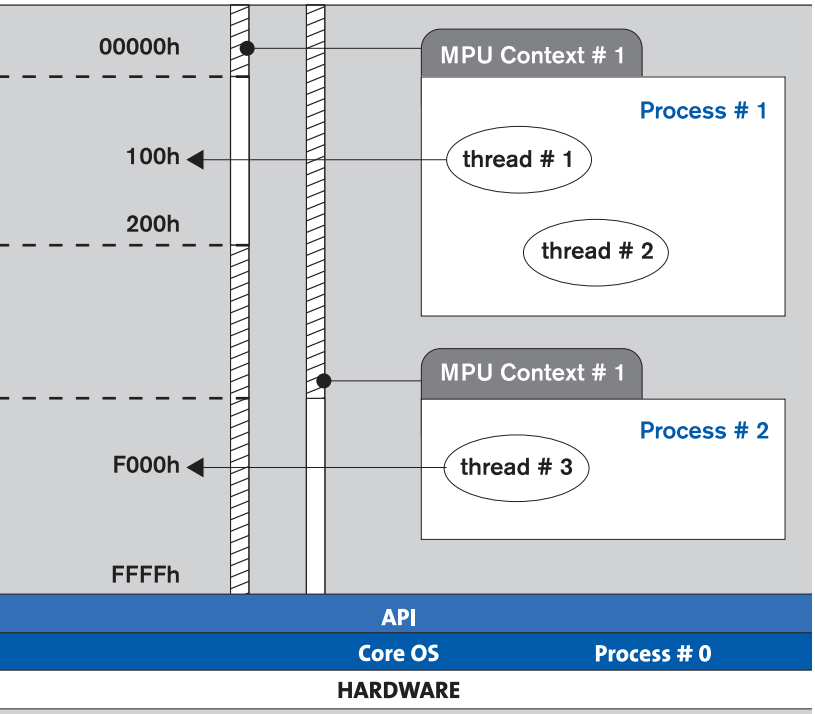
µC/OS-OSEK certified environments are added on a regular basis. Visit our website for the current list of available environments:
www.micrium.com/products/rtos/osek.html

Extension Overview



µC/OS-MPU™

Memory Protection Unit



µC/OS-MPU
The MPU Add-On for µC/OS-II™

µC/OS-II now offers memory protection for CPUs that incorporate a memory protection unit (MPU). This extension of µC/OS-II prevents applications from accessing forbidden locations to protect against damage to, especially important for safety-critical applications, including medical and avionics products.

µC/OS-MPU offers a runtime environment designed to protect a task's memory space in order to prevent damage from unauthorized access to system memory. µC/OS-MPU builds a system with MPU contexts (processes); a process can contain one or more tasks (threads). Each process has its individual read, write and execution rights. Exchanging data between threads can be done in the same manner as µC/OS-II tasks, but the handling across different processes is done by the core operating system.

This system facilitates integration of third party software such as protocol stacks, graphical modules, file system libraries, or other components. It also simplifies debugging and error diagnosis because an error management system provides information onthe different processes. Additionally, the hardware protection mechanism can not be bypassed by software. Existing µC/OS-II applications can be adapted with minimum effort.µC/OS-MPU is available for any microcontroller (MCU) with MPU, and certification support will be available shortly.

Read only **Read / write**

µC/OS-MMU™

Memory Management Unit

µC/OS-MMU
The MMU extension for µC/OS-II™

µC/OS-II now offers memory protection for CPUs that incorporate a memory management unit (MMU). This extension of µC/OS-II provides a configurable solution with minimum overhead that facilitates development of safety critical systems.

µC/OS-MMU offers a runtime environment with time and space protection for multiple independent applications. Each application is executed with the guarantee that no other application will influence, disturb or interact with its execution. Applications can be designed with different guest real time operating systems (RTOS), including µC/OS-II, µC/OSEK, or without an RTOS, and every application within a protected memory space (a partition) can be developed as if no other partition existed.

µC/OS-MMU includes a failure handling capability that identifies any application performing incorrect actions and allows it to be stopped, deleted or recreated. This simplifies the development of complex control units that often include applications from several vendors, since each vendor gets its own partition that functions like its own virtual CPU.

Additionally, µC/OS-MMU guarantees runtime of the applications, since system architects have to define time slots (phases) for the applications during system design. These phases are managed in phase tables and can be activated in the kernel application. Further, within a phase table, it is possible to define multiple phases for one application and if an application is idle, the application can forfeit this time back to the kernel. Each phase table ensures a static timing behavior, even if applications are activated, deactivated or removed. DO178B, 510(k) and IEC61508 certification of the source code is in progress.