

Micrium

RTOS
and Tools

FOR
THE WAY
ENGINEERS
WORK



Micrium

FOR THE WAY ENGINEERS WORK

Micrium provides a full portfolio of embedded software components that not only shorten time-to-market throughout all development cycles, they are uniquely designed to address the way you work.

Our engineer-friendly source code, unsurpassed documentation, unprecedented online training and assistance, and a broad base of customer support, were developed for engineers, by engineers. Our products feature the cleanest code in the industry. What it means to you is that our code is written from the ground up. We don't take, borrow, or patch code into our programs anything that isn't written by us. The result of this clean code for you is an unprecedented ease-of-use and rapid implementation. Micrium's strict coding standards, code reviews, and clear and concise documentation is well respected industry-wide by embedded systems engineers developing any product imaginable.

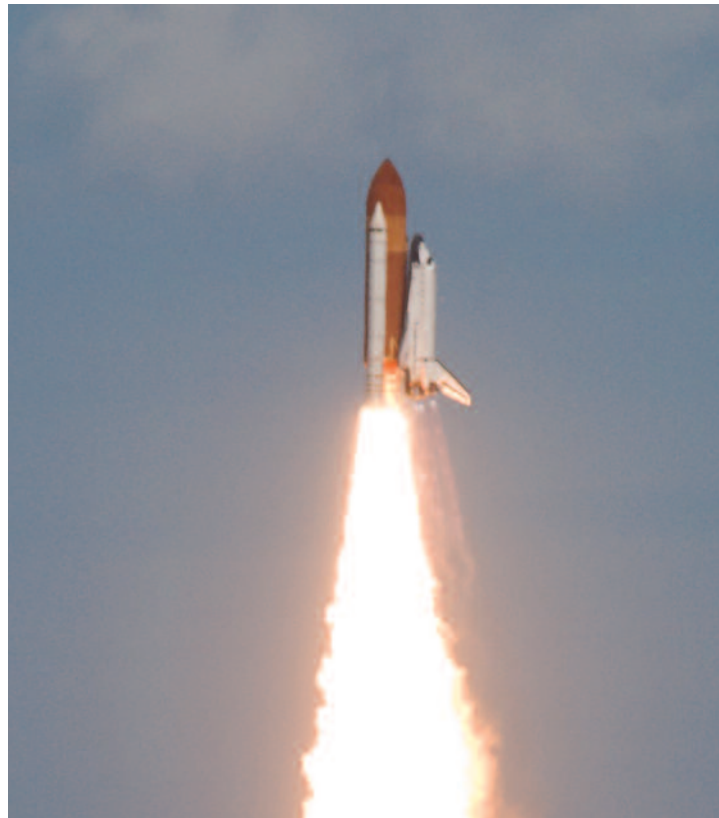
Micrium software components address both deeply embedded and high-reliability applications across a wide variety of markets, including:

- Military/Aerospace
- Industrial Applications
- Medical
- Other Embedded
- Communications
- Consumer Electronics
- Automotive

Our products enable companies to implement and manage embedded device software faster, at a lower cost and with more reliability. In today's complex design environment, when even a short time advantage can mean the difference between success and failure, see how Micrium provides a time-to-market advantage for your next embedded design.

INDEX

Try before you buy	3
Certifications	3
A Note About Drivers	3
μC/OS-II RTOS	4
μC/OS-III RTOS	5
RTOS Add-ons	6
μC/TCP-IP	7
μC/Probe	8
μC/USB Host	9
μC/USB Device	10
μC/USB-OTG (On-The-Go)	11
μC/Bluetooth	12
μC/CAN	13
μC/Modbus	14
μC/GUI (Graphical User Interface)	15
μC/FS (File System)	16
μC/Building Blocks	17
μC/FL (Flash Loader)	18
IAR Systems	19
Distributors	20
Micrium Wrote the Book	22



TRY BEFORE YOU BUY

At Micrium, we're so proud of our products, and we know they're easy to use. As a result, we've created a 'Try before you buy' program that allows you to register, download our source code, and use it for a 30-day trial period. This offer includes:

- μC/OS-II
- μC/TCP-IP

Download our source code, use it with your designs, and "kick the proverbial tires." We're sure you'll be asking yourself why you waited so long. Just go to Downloads at www.micrium.com, and try it yourself.



HOW RELIABLE IS MICRIUM?

Some say that Micrium products are easy enough to be used by hobbyists. And, well, they're right. However, given the stringent development process for all of Micrium's products, its strict coding standards, code reviews, and clear and concise documentation hobbyists aren't our focus. This stringent process was the foundation for a Validation Suite™ for μC/OS-II that provides the documentation required in pre-certifiable software components for such safety critical systems as avionics RTCA DO-178B and EUROCAE ED-12B; medical FDA 510(k); and IEC 61508 standard for transportation and nuclear systems. The μC/OS-II is used in devices adhering to DO-178B Level A, Class III medical devices, and SIL3/SIL4 IEC-certified systems.

μC/OS-II is also 99% compliant with the Motor Industry Software Reliability Association (MISRA) C Coding Standards. These standards improve the reliability and predictability of C programs in critical automotive systems. So, if you're a hobbyist and want reliable software, by all means select us. However, if you're in charge of the highest level of safety critical designs, we're definitely ready and able to take on your complex designs.

A NOTE ABOUT DRIVER DEVELOPMENT

Don't see your driver? If your hardware is not listed in this catalog or on the Micrium web page, consider contracting Micrium to develop it for you. Please call 1-954-217-2036 to receive a quote for non-recurring engineering.



Micrium's flagship product, **µC/OS-II** is a portable, scalable, and preemptive real-time, deterministic, multitasking RTOS for microprocessors, microcontrollers and DSPs. **µC/OS-II** runs on a large number of processors, manages 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 PROCESSOR ARCHITECTURE SUPPORT

Micrium's **µC/OS-II** runs on a large number of processor architectures and ports are available for download from the Micrium website. The vast number of ports is evidence of Micrium's commitment to embedded system design portability.

µC/OS-II MONITORING

Want to see **µC/OS-II** work? Micrium's **µC/Probe**, an award-winning monitoring tool, allows the engineer to "see" inside the **µC/OS-II-based** application, while it is running. A **µC/OS-II** plug-in, provided with **µC/Probe**, features intuitive data screens help keep track of stack sizes, CPU usage, and other key indicators of an application's status. **µC/Probe**'s Windows application receives information from the embedded system via RS-232C, USB, TCP/IP, or even JTAG (See **µC/Probe**, Page 8).

µC/OS-II KERNEL AWARENESS PLUG-IN

µC/OS-II KA allows for the display of **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 information about each of the active tasks in the target application—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 is particularly useful when testing and debugging applications. Other debuggers also provide Kernel Awareness for **µC/OS-II**, including Lauterbach and iSystem.

SCALABILITY AND EXECUTION

The **µC/OS-II** footprint can be scaled to contain only the features required by a specific application. The execution time for most services provided by **µC/OS-II** is both constant and deterministic. Execution times, therefore, do not depend on the number of tasks running in the application.

INTUITIVE AND EASY-TO-USE

Micrium uses conventions throughout its products that enable engineers to become proficient rapidly, thereby shortening the design cycle. For example:

- All services provided by **µC/OS-II** begin with the prefix 'OS,' making it apparent that the functions refer to RTOS services in the application.
- Services are neatly grouped by categories: **OSTask???**() relates to task management functions, **OSQ???**() relates to message queue management, **OSSem???**() relates to semaphore management etc.

Processor Architectures

Architectures supported by **µC/OS-II** include:

Company	Architecture
Actel	Cortex-M1
Altera	Nios II, Cortex-M1
Analog Devices	AduC7xxx (ARM7), ADSP-21xx, Blackfin 5xx, SHARC
ARM	ARM7, ARM9, ARM11, Cortex-M1, Cortex-M3
Atmel	ARM7, ARM9, AVR, AVR 32
Freescale	9S08, 9S12, Coldfire, PowerPC, i.MX
Fujitsu	FR50
Infineon	TriCore, 80C16x
Intel	80x86
Lattice	Micro32
Luminary Micro	Cortex-M3
Microchip	PIC24, dsPIC33, PIC32 (MIPS)
MIPS	R3000, R4000
NEC	78Kx, V850
NXP	ARM7, ARM9, Cortex-M3
Remesas	H8, M16C, M32C, R32C, SH
Samsung	ARM7, ARM9
ST	80C16x, STR7 (ARM7), STR9 (ARM9), STM32 (Cortex-M3)
TI	MSP430, TMS320, TMS470 (ARM7)
Toshiba	Cortex-M3
Xilinx	MicroBlaze, PowerPC
ZILOG	Z80, eZ80

Micrium's newest RTOS, **µC/OS-III** is designed for use by developers who need to save time on their current and next embedded system project. **µC/OS-III** includes many of the same features you've experienced using **µC/OS-II**, but it also manages an unlimited number of application tasks and features an interrupt disable time of near-zero.

µC/OS-III PROCESSOR ARCHITECTURE SUPPORT

Micrium's **µC/OS-III** supports ARM7/9, Cortex-Mx, Nios-II, PowerPC, Coldfire, i.MX, Microblaze, H8, SH, M16C, M32C, Blackfin, and more. Ports are available for download from the Micrium website.

PREEMPTIVE MULTITASKING

µC/OS-III, a preemptive multitasking kernel, always runs the most important task that is ready.

UNLIMITED TASKS, PRIORITIES, KERNEL OBJECTS

µC/OS-III operates in a world of unlimited, including support of an unlimited number of tasks (outside the limitations of a processor's access to memory). **µC/OS-III** supports an unlimited number of priority levels. Typically, configuring **µC/OS-III** for between 32 and 256 different priority levels is adequate for most applications. **µC/OS-III** allows for an unlimited number of tasks, semaphores, mutexes, event flags, message queues, timers, and memory partitions. All kernel objects are allocated by the user at run time. **µC/OS-III** provides features to allow stack growth of tasks to be monitored. While **µC/OS-III** does not impose limitations on task size, the exception is that they need to have a minimum size based on the CPU used.

ROUND ROBIN SCHEDULING

µC/OS-III allows multiple tasks to run at the same priority level. When equal, priority tasks are ready to run, **µC/OS-III** runs each tasks for a user-specified time. Each task can define its own time quanta and give up its time slice if it does not require the full time quanta.

ERROR CHECKING

µC/OS-III ensures that NULL pointers are not passed, task level services from ISRs aren't called, arguments are within allowable range, specified options are valid, and more. Each **µC/OS-III** API function provides an error code regarding the outcome of the function call.

NEAR ZERO INTERRUPT DISABLE TIME

µC/OS-III has a number of internal data structures and variables that it needs to be accessed atomically. It protects these critical regions by disabling interrupts for almost zero clock cycles ensuring that it is able to respond to some of the fastest interrupt sources. With its near zero interrupt disable time, interrupt response with **µC/OS-III** is deterministic.

SCALABILITY AND EXECUTION

The **µC/OS-III** footprint can be scaled to contain only the features required by a specific application. The execution time for most services provided by **µC/OS-III** is both constant and deterministic. Execution times, therefore, do not depend on the number of tasks running in the application.

µC/OS-III MONITORING

Want to see **µC/OS-III** work? Micrium's **µC/Probe**, an award-winning monitoring tool, allows the engineer to "see" inside the **µC/OS-III**-based application, while it is running. A **µC/OS-III** plug-in, provided with **µC/Probe**, features intuitive data screens help keep track of stack sizes, CPU usage, and other key indicators of an application's status. **µC/Probe**'s Windows application receives information from the embedded system via RS-232C, USB, TCP/IP, or even JTAG (See **µC/Probe**, Page 8).

µC/OS-III

Who should use this RTOS?	Developers who want to save time on their current and next embedded system project, and who want the cleanest, most popular, and robust RTOS on the market.
Supported Processors	See complete list in Processor Chart for µC/OS-II (See page 4)
Maximum ROM Footprint (Unscaled)	24 Kbytes
Minimum ROM Footprint (Scaled)	6 Kbytes
Number of Kernel Services	10 different using 80 API calls
Multitasking Model	Preemptive
Code Execution Entities	Tasks, ISRs
Dynamic Objects	Static and Dynamic
Data Movement	Message Queues (unlimited)
Semaphores - Full Counting	Yes (unlimited)
Mutexes - With Priority Inheritance	Yes (priority ceiling)
Event Flags	Yes (unlimited), configurable for 8, 16, or 32 bits
Memory Partitions - RAM Management	Yes (unlimited)
Timers	Yes (unlimited)
Number of task	Unlimited
Interrupt Disable time	Near-Zero

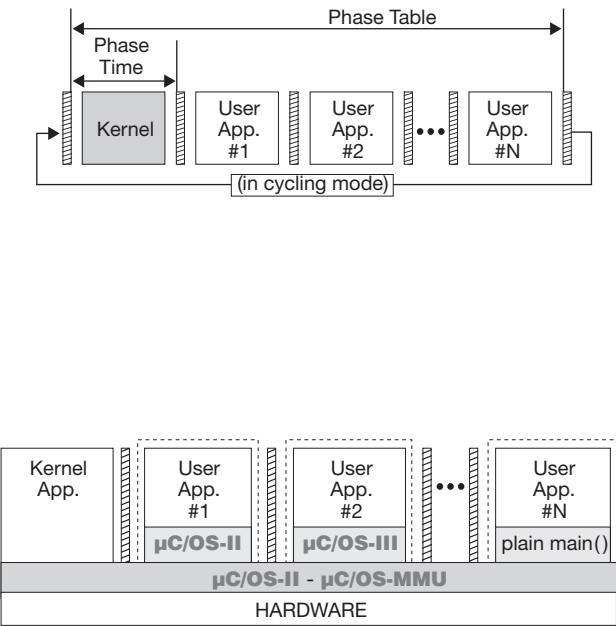
$\mu\text{C}/\text{OS-MMU}^{\text{TM}}$

Memory Management Unit

$\mu\text{C}/\text{OS-MMU}$ 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 guest real-time operating systems (RTOS), including $\mu\text{C}/\text{OS-II}$, $\mu\text{C}/\text{OS-III}$, or without an RTOS. Each application within a protected memory space, or partition, can be developed as though no other partition exists.

$\mu\text{C}/\text{OS-MMU}$ includes a failure handling capability that identifies applications performing incorrect actions and allows the action to be stopped, deleted, or recreated. This simplifies the development of complex control units that often include applications from several vendors, since each vendor has its own partition, functioning as its own virtual CPU.

$\mu\text{C}/\text{OS-MMU}$ also guarantees runtime of an application, as system architects must define time slots for applications during system design. These are managed in phase tables and can be activated in the kernel application. Within a phase table, it is possible to define multiple phases for one application and, if an application is idle, the application can forfeit time back to the kernel. Each phase table ensures a static timing behavior, even if applications are activated, deactivated, or removed. DO178B and IEC61508 certifications of code are available. Certified 510(k) devices are able to use the $\mu\text{C}/\text{OS-MMU}$ as well.



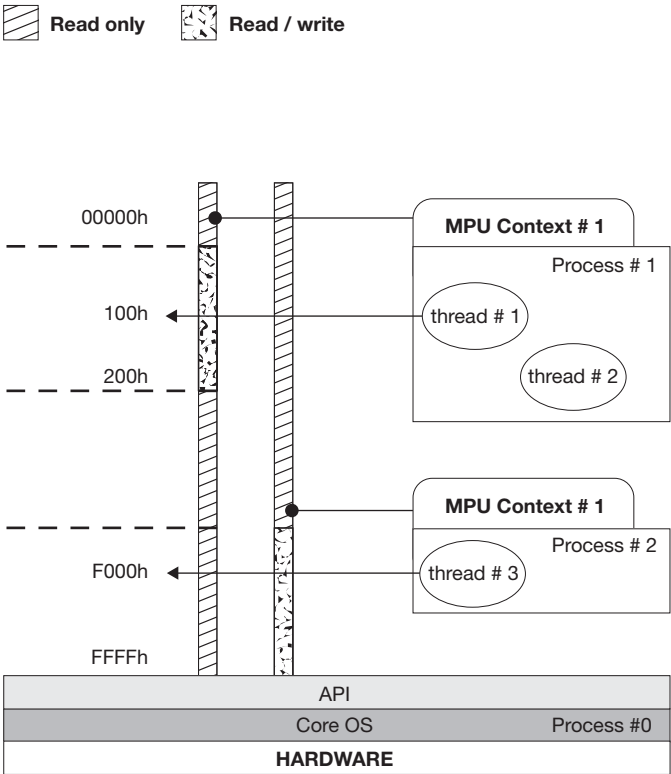
$\mu\text{C}/\text{OS-MPU}^{\text{TM}}$

Memory Protection Unit

$\mu\text{C}/\text{OS-MPU}$ offers memory protection for CPUs incorporating a memory protection unit (MPU). This extension prevents applications from accessing forbidden locations, thus protecting against damage to safety-critical applications including medical and avionics.

$\mu\text{C}/\text{OS-MPU}$ builds a system with MPU processes, each containing one or more tasks or threads. Each process has an individual read, write, and execution rights. Exchanging data between threads can be accomplished in the same manner as $\mu\text{C}/\text{OS-II}$ or $\mu\text{C}/\text{OS-III}$ tasks, however handling across different processes is achieved by the core operating system.

This system facilitates integration of such third-party software as protocol stacks, graphics modules, file system libraries, or other components. Debugging and error diagnosis is simplified as an error management system provides information on the different processes. The hardware protection mechanism cannot be bypassed by software. Existing $\mu\text{C}/\text{OS-II}$ applications can be adapted with minimal effort. $\mu\text{C}/\text{OS-MPU}$ is available for any microcontroller (MCU) with MPU. Certification support is also available.



µC/TCP-IP is a compact, reliable, high-performance TCP/IP protocol stack. Built from the ground up with Micrium quality, scalability and reliability, **µC/TCP-IP** enables the rapid configuration of required network options to minimize time-to-market.

CLEANEST SOURCE CODE

µC/TCP-IP provides the cleanest, highest-quality source code in the industry. **µC/TCP-IP** is a clean-room design, not derived from publicly available Unix stacks, yet compatible with the Berkeley 4.4 (BSD) socket layer interface. As with all Micrium products, **µC/TCP-IP** is written in ANSI C, enabling its usage with a wide array of best-in-class cross-development tools.

PORTABLE

µC/TCP-IP can be used on 16-, 32-, and even 64-bit CPUs or DSPs.

HIGH PERFORMANCE

µC/TCP-IP was designed specifically for the demanding requirements of embedded systems. Critical sections are kept to a minimum, while selected run-time validations can be disabled to enhance performance. **µC/TCP-IP** implements zero copy buffer management for greatest efficiency. With **µC/TCP-IP**, the full advantage of high-performance DMA-enabled Ethernet controllers can be harnessed so that TCP/IP stack is a true ZERO COPY stack.

MEMORY FOOTPRINT

µC/TCP-IP allows for the memory footprint to be adjusted based upon design requirements. **µC/TCP-IP** can be configured to include only those network modules necessary to the system. When a module is not used, it's not included in the build, saving valuable memory space.

MULTI-HOMING

µC/TCP-IP currently supports multiple simultaneous Network Interface Controller (NIC) interfaces

Add-on Modules

µC/DHCPc	Dynamic Host Configuration Protocol (client)
µC/DNSc	Domain Name System (client)
µC/FTPc	File Transfer Protocol (client)
µC/FTPs	File Transfer Protocol (server)
µC/HTTPs	HyperText Transport Protocol (server) a.k.a. Webserver
µC/POP3c	Post Office Protocol (client)
µC/SMTPc	Simple Mail Transfer Protocol (client)
µC/SNTPc	Simple Network Time Protocol (client)
µC/TFTPc	Trivial File Transfer Protocol (client)
µC/TFTPs	Trivial File Transfer Protocol (server)
µC/TELNETs	Telnet (server)

NETWORK INTERFACE CONTROLLER DRIVERS

µC/TCP-IP supports Ethernet and Serial NICs. Additional drivers are added on a regular basis. Check www.micrium.com for a complete list of drivers.

THIRD-PARTY CERTIFICATION

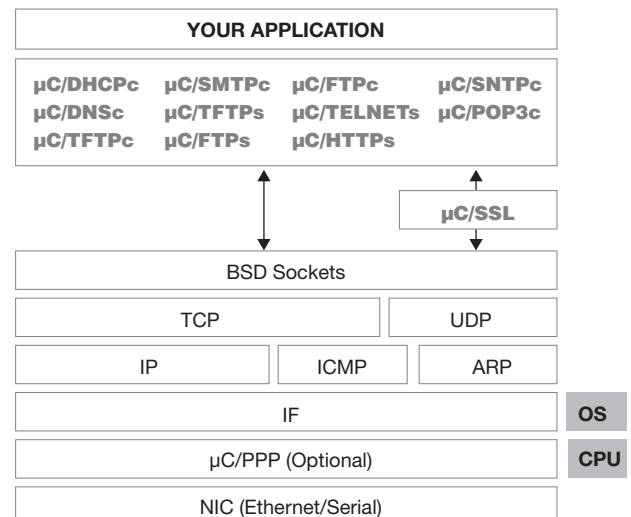
µC/TCP-IP is designed to be certifiable for use in avionics, compliant for use in FDA-certified devices, and in other safety-critical products. The source code for **µC/TCP-IP** is an extremely robust and highly reliable TCP/IP solution.

RTOS REQUIRED

µC/TCP-IP requires the presence of a RTOS for task scheduling and mutual exclusion. To meet this requirement, Micrium provides source code to allow network applications to readily accommodate **µC/OS-II** and **µC/OS-III**. However, based on the module's ostensible Kernel interface, other Kernels can be adapted to **µC/TCP-IP**. Using an RTOS with **µC/TCP-IP** is a design criteria. Because of the processing time required to process IP packets, Micrium ensures that the end-application will always be able to process higher priority tasks.

TIMESAVING TOOLS

Helpful status information that corresponds to the actual data structures of **µC/TCP-IP** is never far away when Micrium's revolutionary visualization tool, **µC/Probe**, is used. With the plug-in, a Windows PC becomes a convenient way to monitor the inner workings of the network application. Non-intrusive, **µC/Probe** gathers useful statistics without stopping or slowing program execution or demanding an undue share of resources (See Probe, Page 8).



See into your embedded design while it's running — no matter what that design may be, or the design's source code. Micrium's Windows-based **μC/Probe** displays a value, at run time, of virtually any variable, memory location, and I/O port in a target product. **μC/Probe** displays such variables as analog gauges, numeric indicators, thermometers, charts, spreadsheets, LEDs, and more, whatever you select for your design.

Display the values of application variables, state variables, timers and counters, computed values, execution times, stack usage, I/O registers and ports, tables and arrays, and more. Display Booleans, integers, floats, and ASCII strings. Make write code, printf() statements, or including a user interface a thing of the past.

HOW μC/PROBE WORKS

μC/Probe reads the .ELF or IEEE695 file downloaded into a target for debugging. From this file, names and addresses of global variables, I/O ports and memory locations are available. Open data screens containing variables to view or change. **μC/Probe** collects a variable's current value and display it. Create data screens where variables are displayed to logically group diverse 'views' into the product. Need to make changes? Change specific values using sliders, switches, buttons, knobs, etc., changing configuration variables setpoints, gains, offsets, limits, and operating modes.

ENVIRONMENT

μC/Probe enables custom displays used to monitor embedded systems with an impressive collection of high-quality graphical components. Two views support this functionality:

- Use Design View to create custom displays. **μC/Probe** Workspace Explorer gives control over workspaces to manage complex projects using multiple data screens. Simply choose from a wide selection of components. **μC/Probe** grid feature allows for the accurate positioning of components. Custom bitmaps let you place unique graphics alongside components. Once satisfied, it's easy to associate components with the embedded system's data.
- To monitor or change the variable(s) associated with a given component, use **μC/Probe** Run-Time View. Just press "start" and **μC/Probe** queries the embedded target via supported communication protocols. The data that **μC/Probe** gathers will be displayed on any components associated with variables.

DATA LOGGING

The data monitored by **μC/Probe** can be logged to a file in ASCII format. Users can review these logs for any specific event they would like to identify.

INTEGRATED SUPPORT

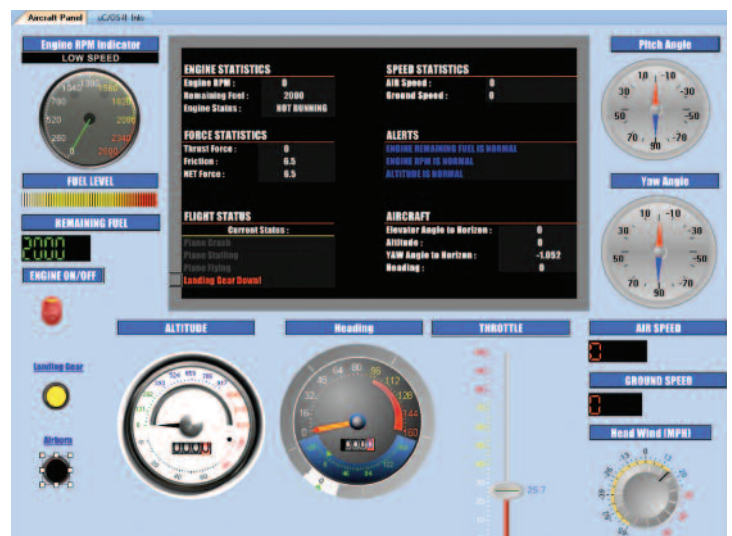
Although it's not necessary to use an RTOS, **μC/Probe** offers unique benefits to applications using Micrium software. Micrium support is integrated to

μC/Probe and provides a convenient way to monitor variables. For example, for **μC/OS-II** and **μC/OS-III**, kernel awareness is provided.

COMMUNICATIONS PROTOCOLS

Communications protocols supported between **μC/Probe** and the embedded system include:

- RS-232C
- J-Link or SAM-ICE Interface (ARM JTAG)
- TCP/IP
- ARM Cortex SWD
- USB
- HEW Target Server



μC/Probe Features and Benefits

Display/change target	Use it with any variable, any memory location, any data at runtime I/O port
Works with any processor	8-, 16-, 32-, 64-bit or DSP, uses minimal resources, no target resident code is needed with ARM Cortex SWD
Works with any compiler	Compatible with any toolchain capable of producing an ELF/DWARF or IEEE 695 file
Does not require an RTOS	Works with or without an RTOS, support for single and multi-task embedded applications
Fast update rate	>300 variables/sec HEW Target Server >200 variables/sec J-Link Interface >300 variables/sec ARM Cortex SWD >500 variables/sec RS-232C Interface, >4,000 variables/sec TCP/IP Interface
Variables can be displaced as:	Analog gauges, numeric indicators, thermometers, bar charts, pie charts, plots, spreadsheets, LEDs, bitmap graphics, etc.
Variable can be changed using:	Sliders, knobs, buttons, pick lists, radio buttons, checkboxes, edit boxes
Display bitmaps	User 'icons', graphics change based on value of variables, pictures
Interfaces via:	RS-232C, TCP/IP, USB, J-Link or SAM-ICE Interface (ARM JTAG), Cortex SWD, HEW Target Server
Project Management	Supports project hierarchy, project auto-save/restore, create data screens in minutes, unlimited number of data screens, Import/Export data screens

µC/USB-Host is a full-featured, high performance, small footprint USB host software stack designed for embedded systems equipped with a USB host or OTG controller. Kernel independent, **µC/USB-Host** includes API, class drivers (Mass Storage, HID, CDC, Printer and Audio) and framework for developing custom class drivers.

ARCHITECTURE

µC/USB-Host uses a modular architecture with three software layers between the application and the hardware. Class drivers that constitute the uppermost layer provide class-specific services to the application. For example, the Mass Storage Class Driver includes interface functions for reading and writing sectors from a storage device. The USB Driver Layer configures the device, loads a matching class driver, and provides the mechanism for data transfers. Within the bottom layer, the Host Controller Driver interfaces with the host controller hardware to enable data transfers and detect devices.

CLASS SUPPORT

By combining the Mass Storage Class (MSC) driver and a file system, an application can access files on connected USB flash drives, hard drives, and DVD drives, etc. A file system is necessary since the MSC driver implements only a USB protocol, offering an application interface for reading and writing sectors and obtaining basic device information such as the number of sectors and sector size. The file system interprets the data as necessary for reading and writing files. This driver can be used with **µC/FS**, Micrium's file system, or with any file system.

When using the Human Interface Device (HID) Class driver, an application can communicate with standard and vendor-specific HID devices. This driver provides routines for receiving and setting reports, and mechanisms for parsing report descriptors.

The Communication Device Class allows the embedded system to use serial devices such as modems. A Printer Class is also available, enabling the embedded system to print to industry standard printers.

The Audio Class driver is compatible with the USB Audio Device Class 2.0 Specifications. Speaker, MIC and Mixer (optional) Units are supported. Audio Streaming through I2S to external Audio Codec and Audio Control through I2C to external Audio Codec are supported. Volume Control and Mute are implemented. One Control, one Isochronous-Out and one Isochronous-In endpoint are used.

The Printer Class is compatible with USB 2.0 Specifications. It is for printers which supports mandatory Bulk OUT endpoint and both Bulk OUT and optional Bulk IN endpoints. It is compatible with almost all HP LaserJet printers which supports PCL5 language. Simple Printer API is exposed to the end user such that NO knowledge of underlying Printer protocols is necessary to develop applications.

A driver for other standard classes or for a vendor-specific class can be developed from a template driver, using host stack documentation describing the class driver architecture.

SPECIFICATIONS

- USB 1.1 and USB 2.0 compatible
- Control, Bulk, interrupt, and isochronous data transfers
- OHCI-, EHCI-, and UHCI-compatible
- ANSI C source code
- Host class drivers (Hub, Mass Storage, HID, CDC, Printer)

BENEFITS

- High performance
- ROMable and scalable (to reduce footprint)
- Used with a Real-Time Kernel (commercial or proprietary)
- Easy to use API
- Minimal USB knowledge required.
- Outstanding documentation and source code (included)
- Extensive test cases and test harness to verify stack integration
- Device drivers are available to support different USB Host controllers

RTOS SUPPORT

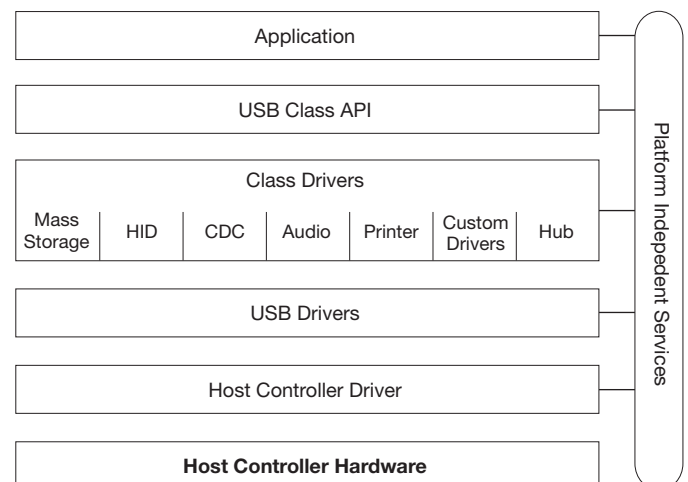
µC/USB-Host can be used with or without an RTOS.

MEMORY FOOTPRINT

µC/USB-Host's footprint is scalable to contain only the features required by the application.

USB HOSTCONTROLLER DRIVERS

µC/USB-Host features a hardware abstraction layer allowing rapid addition of support for the new devices. Additional drivers are added on a regular basis. Visit Micrium's Website for a current list of available drivers.





µC/USB-Device is a device stack designed for embedded systems equipped with a USB device controller. A hardware abstraction layer makes it easy to port **µC/USB-Device** to new USB controllers by simply modifying existing hardware access routines. Drivers for several common device classes (Mass Storage, CDC, HID, Audio) are offered, supplemented by a framework for easily developing new class drivers.

ARCHITECTURE

Each target requires a device controller driver to interface with device controller hardware to process interrupts, notify the stack core of bus events, and receive and transmit packets. The core controls packet reception and transmission and responds to standard host requests during enumeration (when a host discovers the features of a device). The target provides functionality to the host with one or more class drivers (e.g., the mass storage class driver). Each class driver responds to class-specific requests and may provide an API for controlling the feature or receiving and transmitting information.

CLASS SUPPORT

The Mass Storage Class (MSC) driver enables the use of the embedded target device as a USB mass storage device. Typical applications include digital cameras, USB stick, MP3 player, DVD player, and any target with a USB interface. Micrium provides the interface layer to use **µC/FS** media drivers and to mitigate the use of the media between the File System and the USB Device stack.

The Communication Device Class (CDC) driver converts the target device into a serial communication device. The target is recognized by the host as a serial interface (USB2COM, virtual COM port). Typical applications include modems, telephone systems and fax machines.

The Human Interface Device (HID) Class driver allows the target to use a standard USB class without a special host driver, for a vendor-specific communication protocol.

The Audio Class driver is compatible with the USB Audio Device Class 2.0 Specifications. Speaker, MIC and Mixer (optional) Units are supported. Audio Streaming through I2S to external Audio Codec and Audio Control through I2C to external Audio Codec are supported. Volume Control and Mute are implemented. One Control, one Isochronous-Out and one Isochronous-In endpoint are used.

A Bulk Device Class driver enables the target to use bulk transfer to exchange data with the host. Unlike the HID class, a special host driver is required, as Windows does not natively support this type of device. A Windows driver is provided as an executable (*.sys), although it is optionally sold as source code.



FEATURES AND BENEFITS

SPEED

µC/USB-Device supports full-speed (12 Mbit/sec) and high speed (480 Mbit/sec) controllers. The highest achievable transfer rate on full-speed devices is approximately 1.1 MByte/sec. This data rate is achievable on a fast system such as an ARM7 platform. While better transfer rates will be achieved with high-speed controllers, most platforms cannot reach the theoretical maximum (approximately 50 Mbytes/sec). However, a high-speed controller should provide an order-of-magnitude performance increase over a full-speed controller on a comparable MCU/MPU.

MEMORY FOOTPRINT

µC/USB-Device's footprint can be scaled to contain only the features required for the application at hand.

RTOS SUPPORT

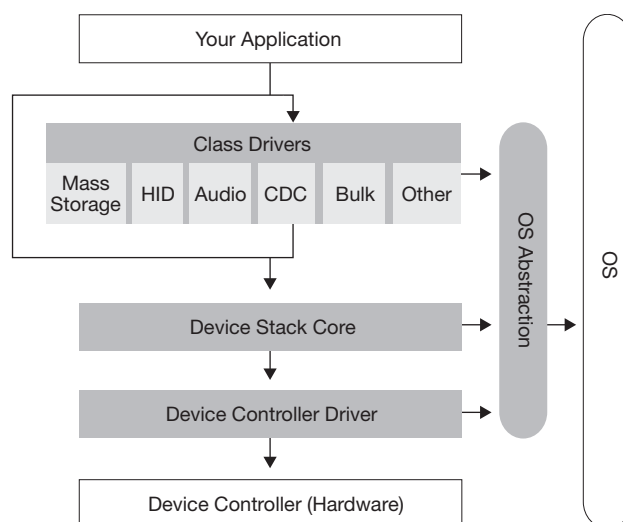
µC/USB-Device can be used with or without an RTOS.

USB DEVICE CONTROLLER DRIVERS

µC/USB-Device features a hardware abstraction layer allowing rapid addition of support for the new devices. Additional drivers are added on a regular basis. Visit Micrium's Website for a current list of available drivers.

START/TEST APPLICATIONS

Micrium provides simple target test applications for all USB device classes in all significant configurations. For bulk, mass storage, HID, Audio and CDC classes, simple Windows applications are included, constituting an end-to-end demonstration of USB functionality.





μC/USB-OTG is an OTG software stack targeting embedded devices featuring a USB OTG controller. It implements HNP and SRP protocols, and is fully compatible with OTG supplement. **μC/USB-OTG** provides an RTOS adaptation layer and code for **μC/OS-II** and **μC/OS-III**. **μC/USB-OTG** is compatible with **μC/USB Host** and **μC/USB Device** stacks, and can be easily ported to any RTOS and native Host and Device stacks. The stack is highly configurable for devices with Dual-Role and Device-Only capabilities.



μC/USB-OTG enables point-to-point data exchange between two USB devices. The devices compliant with OTG specifications may support both traditional host-based (PC) and point-to-point connectivity. USB connectivity is implemented in such mobile consumer electronics products as PDAs, mobile phones, digital cameras, portable storage devices, etc., to exchange data with PCs. **μC/USB-OTG** allows the devices to communicate a host.

μC/OTG introduces the dual-role device, capable of functioning as either host or peripheral. Although most OTG devices are dual-role capable, an OTG device may or may not be capable of functioning as a host. Hence, the OTG devices are classified into two groups:

- OTG Dual-Role (devices that can function as host or peripheral)
- OTG Peripheral (devices that function as peripheral only)

HOST NEGOTIATION PROTOCOL (HNP)

When two OTG devices connect, the cable orientation decides the initial roles of the devices. Once connected, OTG dual-role devices exchange host and peripheral roles by using the Host Negotiation Protocol (HNP). This eliminates the need to manually switch cable connections to accomplish the role reversal. HNP is typically initiated in response to user input or by an application on the dual-role device.

SESSION REQUEST PROTOCOL (SRP)

The OTG supplement defines Session Request Protocol (SRP), which is used by the Peripheral only device to request a Dual-Role Device to turn on VBUS power and start a session. The OTG session starts when the Dual-Role Device supplies VBUS power, and ends when the Dual-Role Device turns off VBUS power to conserve energy.

FEATURES

DEVELOPMENT ENVIRONMENT

μC/USB-OTG is written in portable ANSI C and can therefore be used with most 8-, 16-, 32- and 64-bit CPUs or DSPs. A C++ compiler is not required but can be used with C++ based applications. **μC/USB OTG** is designed for resource constrained embedded applications and is portable across 8- to 64-bit processors and DSPs.

MEMORY FOOTPRINT

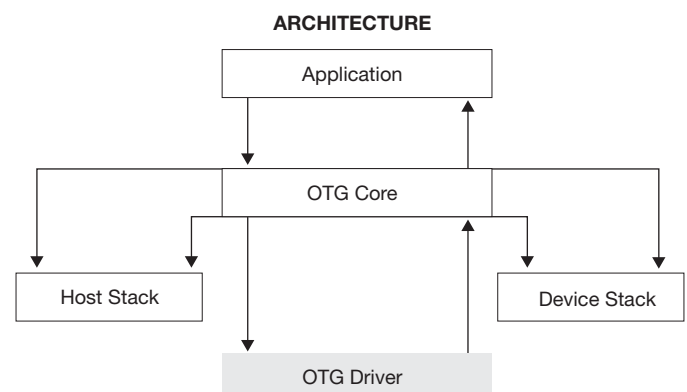
The **μC/USB-OTG** footprint can be scaled to contain only the features required by the application.

RTOS SUPPORT

μC/USB-OTG can be used with or without an RTOS.

USB OTG CONTROLLER DRIVERS

μC/USB-OTG features a hardware abstraction layer allowing rapid addition of support for the new devices. Additional drivers are added on a regular basis. Visit Micrium's Website for a current list of available drivers.





μC/Bluetooth stack is a highly configurable, resource-efficient protocol stack targeting embedded Bluetooth applications. It is written entirely in ANSI C with simple and well-defined interfaces.

ARCHITECTURE

The **μC/Bluetooth** architecture and design strictly conforms to Bluetooth Core Specifications (See block diagram for supported layers within the Bluetooth stack). All layers are user configurable. There are a number of Bluetooth profiles defined in the Bluetooth specification. **μC/Bluetooth** currently supports the Serial Port Profile (SPP) and Dial-Up Networking (DUN) profiles. **μC/Bluetooth** provides a framework to develop new profiles very easily.

BLUETOOTH CONTROLLER

μC/Bluetooth supports any Bluetooth controller implementing the Host to Controller Interface (HCI) standard. Additional drives are added on a regular basis. Visit the Micrium Web site for a current list of available drivers at: www.micrium.com/products/Bluetooth/drivers.html.

BLUETOOTH CORE

The **μC/Bluetooth** core contains the following higher layers of the stack:

RFCOMM is a transport protocol based on L2CAP that emulates RS-232 serial ports. The protocol supports up to 60 simultaneous connections between two Bluetooth devices. RFCOMM provides a data stream interface for high-level applications and profiles.

Service Discovery Protocol (SDP) provides a means for applications to discover the services available and determine the characteristics of those services. SDP uses an existing L2CAP connection and further connection to Bluetooth devices are established using information obtained via SDP.

Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to upper-layer protocols with data packets up to 64 KB in length. L2CAP performs the segmentation and reassembly of I/O packets from the baseband controller.

HCI Generic Driver implements the HCI Interface standardized by Bluetooth SIG. It establishes the communication between the stack and the HCI firmware in the Bluetooth hardware, and communications with the Bluetooth controller via the HCI Bus driver.

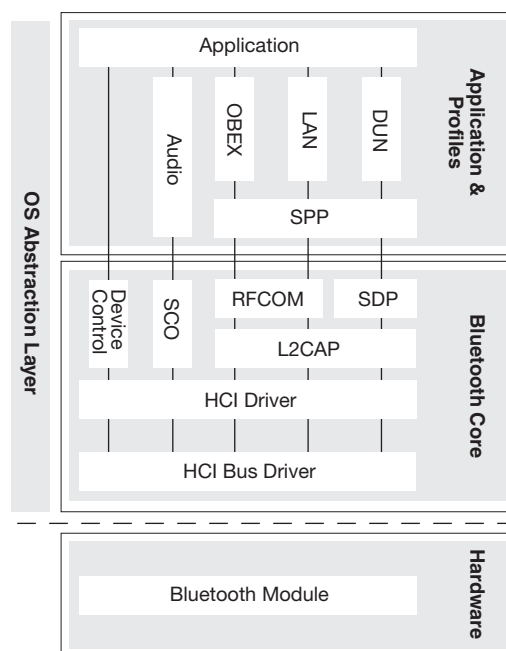
HCI Bus Driver - Bluetooth controllers are connected to the host using interface-like UART, USB, SDIO, etc. The HCI Bus Driver provides bus abstraction to the rest of the Bluetooth stack software. Currently the HCI Bus Driver supports UART and USB connectivity. SDIO support will be available in future releases.

RTOS SUPPORT

μC/Bluetooth assumes the presence of an RTOS. **μC/Bluetooth** contains an OS abstraction layer that allows it to be used with most RTOSes. The interface to **μC/OS-II** is included. The use of an RTOS is a Micrium design decision to make sure the customer application will have enough CPU cycles to process the application data.

BLUETOOTH DEVICE DRIVERS

Device drivers are available to support different Bluetooth controllers. Please check the Micrium Web site for a current list of device drivers.



µC/CAN is a CAN protocol framework that enables easy and clean implementation of CAN communication paths. **µC/CAN** is a source code library optimized for speed, flexibility, and size. High portability and clean documentation was the focus throughout development. The goal of **µC/CAN** is to reduce the development effort of a developer wanting to embed CAN. The developer only needs to understand signals, messages and bus configurations. Different abstraction layers can be used independently. **µC/CAN** can be used with or without an RTOS.

PROTOCOLS

µC/CAN enables the easy and quick implementation of custom CAN protocols such as CANopen. The framework allows customers to develop CAN protocols with a completely abstracted CAN interface. Once a protocol is developed, any microcontroller or CAN controller can be used by switching a **µC/CAN** layer. The customer can work with multiple CAN busses, messages, and signals.

CANopen STACK

The CANopen stack is a scalable solution for embedded systems with limited resources. The stack is delivered in ANSI C source code and can be compiled with any ANSI C compliant compiler on the market. There are three variants of the stack depending on application, and a Windows application for the automatic source code generation of the object directly and EDS files.

CANopen SENSOR SLAVE

The CANopen Sensor Slave is designed for CANopen protocol handling on minimal slave systems such as intelligent sensors, digital I/O modules, etc. This variant supports the following features:

- SDO Server Support
- SDO Expedited Transfers
- PDO Producer and Consumer with static mapping and communication parameters
- NMT Slave
- NMT Heartbeat Producer
- EMCY Producer
- LSS Slave
- Object Strings with up to 4 Characters

CANopen SLAVE

The CANopen Slave is full-featured for more complex slave applications. This variant includes the features of CANopen Sensor Slave with the following additions:

- SDJO Client Support
- SDO Normal and Block Transfers
- Dynamic PDO mapping and communication parameters
- EMCY History
- Dynamic Object Directory

CANopen MASTER

The CANopen Master is a fully featured for network management applications. This variant includes the features of CANopen Slave with the following additions:

- NMT Master
- NMT Heartbeat Consumer
- LSS Master
- Object Strings with Unlimited Length
- Object Domains Supported

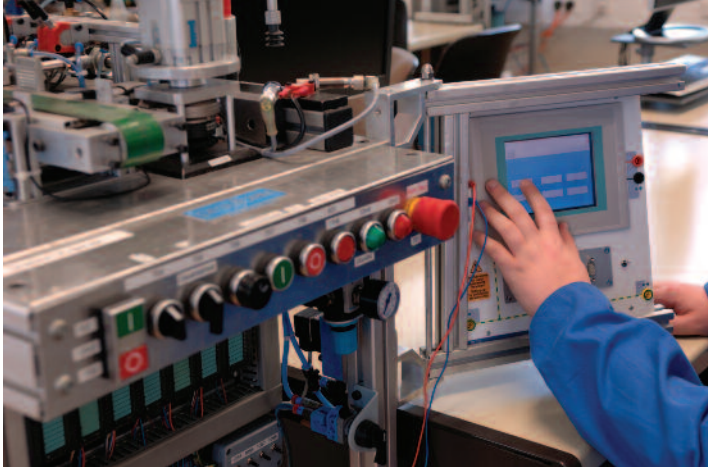
CANopen CONFIG

This is a Windows application for configuration of the object director and the EDS-file of all CANopen stack variants. The tool allows the template-based generation of ANSI C code and EDS files.

CAN CONTROLLER DRIVERS

Device drivers are available to support a variety of CAN controllers. Please consult the Web sit for a current list.





μC/Modbus software module allows an embedded system to communicate to other devices using the MODBUS protocol. **μC/Modbus** provides both Master (**μC/Modbus-M**) and Slave (**μC/Modbus-S**) capabilities to products. Embedded systems can act as a Modbus Slave, Modbus Master, or both, concurrently. Each is offered separately.

PORTABILITY

μC/Modbus is written in C and is highly portable; it supports both ANSCII and RTU protocols, RS-232C and RS-485. **μC/Modbus** enables the designer to read or write integer, floating-point (assuming the Daniels extensions), and discrete values from or to the target system.

MULTIPLE CHANNELS

μC/Modbus allows for multiple serial interfaces or channels on a single target system. Each channel is independently configurable at run-time to be either a Master or a Slave and communicate using Modbus ASCII or RTU. **μC/Modbus** supports Modbus ASCII and RTU on an individual channel basis.

RS-232C OR RS-485

μC/Modbus allows for multiple RS-232C or RS-485 ports on the same target.

APPLICATION DATA

With **μC/Modbus**, assign just about any application variable to any Modbus holding register, input register, coil (discrete output), or input status (discrete input).

MODBUS REGISTERS

- Read and Write from up to 65536 16-bit Integer or Floating-Point Holding Registers.
- Read from up to 65536 16-bit Integer or Floating-Point Input Registers.
- Read and Write from up to 65536 Coils.
- Read from up to 65536 Input Status.

MEMORY REQUIREMENTS

The amount of code space required by **μC/Modbus** depends on which function codes are included and varies between 8 and 20 Kbytes. The amount of RAM depends on the number of Modbus channels, each requiring approximately one Kbytes.

DEVELOPMENT ENVIRONMENT

Since **μC/Modbus** is written in portable ANSI C, it can be used with most 8-, 16-, 32- and even 64-bit CPUs or DSPs. A C++ compiler is not required, but can be used in C++ based applications.

RTOS SUPPORT

μC/Modbus works with or without an RTOS. **μC/Modbus** includes interface code for **μC/OS-II** and **μC/OS-III**.

Supported Function Codes

FUNCTION	CODE DESCRIPTION
1	Coil Read
2	Discrete Input Read
3	Holding Register Read
4	Input Register Read
5	Write Single Coil
6	Write Single Holding Register
7	Diagnostic Loopback
15	Write Multiple Coils
16	Write Multiple Holding Registers
20	File Read
21	File Write

μC/GUI is universal graphical software for embedded applications. It provides an efficient processor and LCD-controller independent GUI to applications using a graphical LCD. It is designed for single and multi-task environments, and is adaptable to nearly any size physical or virtual display with an LCD controller and CPU. **μC/GUI** is compatible with nearly all CPUs and, unlike other GUIs that require a C++ compiler, **μC/GUI** is written entirely in ANSI C. Processors ranging from 8- to 32-bits run **μC/GUI**. 16-bit CPUs (or better) are advisable for optimal performance.

ADD-ON OPTIONS

Optional add-on modules allow for memory footprint and feature customization based on the application. Modules include:

- **Memory Devices** — Used to prevent the display from flickering when using drawing operations of overlapping items.
- **Anti-aliasing** — Smooths curves and diagonal lines by “blending” the background color with that of the foreground.
- **Window Manager** — allows for creation and handling of different Windows of any size.
- **Widgets** — Widgets (or controls) are windows with object-type properties: buttons, radio buttons, scroll bars, check boxes, list boxes, etc.
- **Dialogs** — Typically windows that appear in order to request user input. They may contain multiple widgets, requesting data through a variety of selections, or take the form of a message box to simply provide information such as a note or warning.
- **Touch Screen** — Optional touch-screen support contains a low-level driver to handle analog input (from two 8-bit or better A/D converter channels), debouncing, and calibration of the touch screen.
- **Multiple Displays/Layers** — Windows can be placed, and drawing operations used, in any layer or display. Multiple layers/displays are handled in the same fashion, and are simply referred to as multiple layers (even if the embedded system uses multiple displays). The **μC/GUI** viewer allows the user to look at every individual layer (display), and in the case of multi-layer systems, to look at the actual output (composite view).
- **Virtual Screen** — A display area greater than the physical size of the display, requiring additional video memory. It allows for instantaneous switching between different screens, even on slow CPUs.
- **Virtual Network Computing** — VNC is a client-server system based on a simple display protocol that allows users to view a computing ‘desktop’ environment not only on the machine where it is running, but from anywhere on the Internet and from a wide-variety of machine architectures.

LCD DRIVERS

Micrium provides drivers for the most popular display controllers. Visit Micrium’s web site for a current list of drivers.

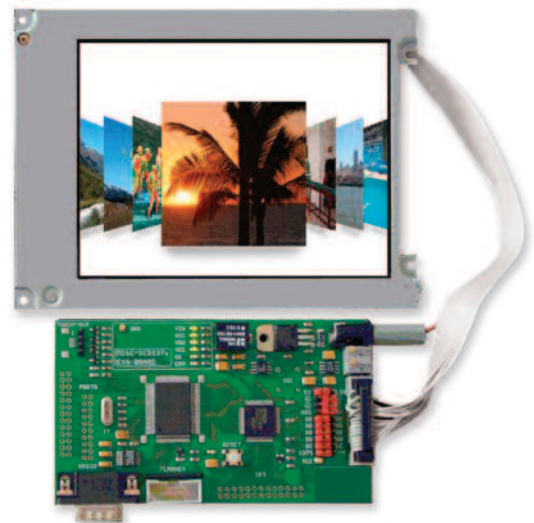
DEVELOPMENT ENVIRONMENT

Given that **μC/GUI** is written in portable ANSI C, it can be used with most 8-, 16-, 32- and 64-bit CPUs or DSPs. A C++ compiler is not required, but can be used in C++ applications.

PC TOOLS

μC/GUI includes a suite of development tools to facilitate project development. **μC/GUI** includes a simulation environment under Microsoft Visual C++ to write and test the entire user interface on the PC. All routines are identical to the embedded application, regardless of the CPU or LCD used, making debugging and development easy and convenient. Generating screen shots of the LCD that can be inserted into documentation is a snap.

Micrium ImageFlow

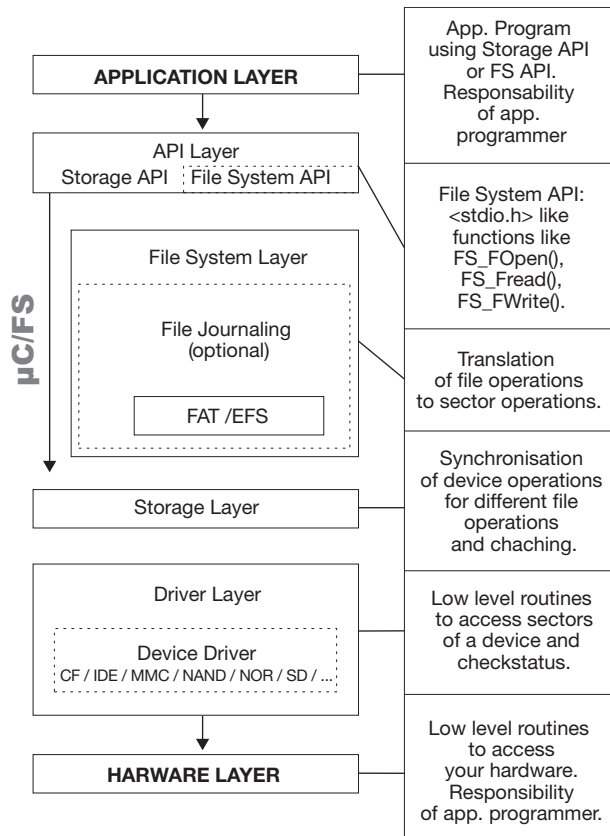


μC/FS is a FAT file system, written in ANSI C, and used on any media, for which the user can provide basic hardware access functions. It is a high-performance library optimized for speed, versatility, and memory footprint. **μC/FS** is designed to cooperate with virtually any type of hardware that has a device driver. The device driver consists of basic I/O functions to access hardware and a global table, which holds pointers to these functions. The **μC/FS** footprint is scalable to contain exactly the features required for an application.

μC/FS is written in ANSI C and is usable on virtually any CPU. A partial list of features includes:

- MS-DOS/MS-Windows compatible — FAT12, FAT16 and FAT32 support or Micrium FAT-free proprietary Embedded File System (EFS).

- Multiple device driver support — Use a variety of device drivers with **μC/FS** to access different types of hardware with the file system simultaneously.
- RTOS Support — **μC/FS** is suitable for use within nearly any multi-threaded environment. To ensure that tasks access the file system concurrently, implement a few RTOS-dependent functions. **μC/OS-II** functions are included. **μC/FS** can also be used without an RTOS.
- ANSI C stdio/h-like API — An application using a standard C I/O library can easily be ported to use **μC/FS**.
- Simple device driver structure — **μC/FS** device drivers need only very basic functions for reading and writing blocks. Custom hardware, therefore, is very easy to support.
- Generic device drivers — Include: NOR Flash; NAND Flash; Serial DataFlash; SmartCards; Secure Digital; MultiMedia Card; CompactFlash; Integrated Device Electronics; RAMdisk; and Windows to allow for simulation of software in a Windows environment.



LAYERED ARCHITECTURE

JOURNALING OPTIONAL

μC/FS Journaling sits on top of the file system, making the file system layer fail-safe. FAT file systems without journaling support are not failsafe. Journaling requires that a file system logs all changes to a journal before committing them to the main file system.

DRIVER FAIL SAFETY

Data can be lost as a result of an unexpected Reset in either a file system layer (FAT or EFS), or in the driver layer. Systems are failsafe only if both layers are failsafe. The journaling add-on makes the file system layer fail-safe. Micrium device drivers are failsafe.



Micrium embedded software components include **μC/Shell**, **μC/CRC**, **μC/Clk**, and **μC/LCD**.

μC/Shell

μC/Shell is a stand-alone module allowing a string containing a command and its argument to be parsed and executed. **μC/Shell** can be used with **μC/OS-II**, **μC/OS-III**, **μC/TELNETs**, **μC/FTPs**, **μC/TFTPs**, **μC/FS**, and more. **μC/Shell** provides a shell interface for Micrium components, but can be used within many applications as it allows for the additional of any number of commands. Modules in need of a shell facility (such as **μC/TELNETs**) interact with it by means of an application callback function.

μC/CRC

μC/CRC allows an application to compute CRCs on arrays of bytes or on a stream of data. Any 16-or 32-bit CRC can be calculated. Parameters and tables are provided for commonly used checksums. **μC/CRC** uses fast table-driven calculations and certain core functions may be optionally replaced by assembly language analogues to further accelerate the computation. Such assembly code is provided for several processor families including ARM7/9 and Cortex-M3.

μC/Clk

μC/Clk implements a Year 2000-compliant clock/calendar offering such features as:

- Maintains time using a 32-bit running counter starting at 2000/01/01 00:00:00 GMT.
- Allows applications to obtain timestamps to mark the occurrence of events.
- A **μC/Clk** timestamp is a copy of the internal 32-bit running counter.

THREE TIMESTAMP TYPES ARE SUPPORTED:

- **μC/Clk** timestamps, which cover years 2000 to 2135 with one-second resolution
- UNIX timestamps covering 1970 to 2105 with one-second resolution
- Network Time Protocol (NTP) timestamps, which cover years 1900 to 2035 with one-second resolution

μC/Clk allows applications to receive the current date and time in a structured data type named `CLK_DATE_TIME` which contains the year, month, day, day of week, hour minute, second, and time zone. It allows conversion from and to all types of timestamps and `CLK_DATE_TIME`, receive and set the internal 32-bit running counter using any of the four data types. **μC/Clk** is compatible with **μC/SNTPc** to obtain the current data and time from a NTP server.

μC/LCD

μC/LCD allows for the interface with LCD modules based on the highly popular HD44780 Dot Matrix LCD Controller and Driver. **μC/LCD** allows for the following actions:

- Control LCD modules containing up to 80 characters
- Display ASCII characters
- Display ASCII strings
- Define up to eight symbols based on a 5x7 dot matrix
- Display horizontal and vertical bar graph





An increasing number of embedded applications use in-circuit reprogrammable memory chips enabling flexibility at any time, i.e. after assembly of the hardware or even after shipment to a customer's site in case of firmware revision, or the uploading of new features.

µC/Flash is software designed for program updates of embedded applications via serial interface from a PC. **µC/Flash** consists of:

- **µC/FlashLoader**, a Windows application that runs on a PC
- Target resident code (called bootloader or BTL0), provided in C source code form.
- An embedded application with a FLASH-type memory for program storage

FEATURES

µC/Flash features include:

- Portable to any CPU and FLASH
- Interfaces via an RS-232C serial port
- Configurable
- Optional password
- 100% safe and fast: CRC-check implemented. Easy-to-use.

FILE TYPES AND CONVERSIONS

- Intel HEX (*.HEX)
- Motorola S (*.MOT, S7, S8, and S9)
- **µC/FlashLoader** analyses, loads and displays file contents. Files can be relocated and converted.

µC/FlashLoader is 32-bit Windows application and can be started from Windows Explorer or from a command line. **µC/FlashLoader** is very easy to use. Any HEX file (Intel, Hex, Motorola S-records, etc.) can be loaded and transferred to the bootloader to update the embedded target.

UPDATER

Updater is an optional add-on for **µC/Flash**. Designed to provide easy firmware update without the use of **µC/FlashLoader**, it is shipped as source code. Updater enables the user the ability to set up the PC COM port. Once the updater tool is in contact with the target, a notification message is provided in the application window. Users need only to press "start" or "enter" to update the target. The tool clears the flash, programs the new file into the flash, makes a CRC check, and makes the target application valid.



IAR SYSTEMS

Micrium, in addition to having its RTOS family of products, is a worldwide-authorized distributor of IAR Systems development tools for embedded systems, including:

- Integrated development environments (IDE)
- C/C++ compilers and debuggers
- Development kits
- Hardware debug probes
- State machine tools

PRODUCTS INCLUDE:

IAR KICKSTART KIT™

Whether the goal is to get started rapidly or evaluate a combination of tools, IAR KickStart Kit includes the hardware and software needed to be up and running quickly. The kit contains a development board, the IAR J-Link JTAG debugger with USB connector, a suite of software development tools including IAR visualSTATE, and an IAR Embedded Workbench project complete with full **μC/OS-II** source code, a limited edition of **μC/Probe**, and optionally, **μC/TCP-IP** source code.

IAR EMBEDDED WORKBENCH®

IAR Embedded Workbench is the most complete, integrated, development environment for embedded applications on the market. It features the same intuitive user interface regardless of the microprocessor used. Ready-made project templates and configuration files provide a flying start. IAR Embedded Workbench incorporates the IAR C/C++ compiler, assembler, linker, librarian, text editor, project manager, and C-SPY debugger into one integrated development environment, providing an uninterrupted workflow and single toolbox so that all components integrate seamlessly.

Manage project, create source files, build and run applications. A Kernel Awareness plug-in enables the user to view and interact with information from **μC/OS-II** in dedicated windows within the Embedded Workbench IDE.

OPTIMIZING C AND C++ COMPILERS

IAR Systems claims more compilers developed for embedded microprocessors than anyone. That experience is evident. For example, the IAR C/C++ is available for several architectures ranging from 8-bit MCUs from Atmel, NEC, TI, Renesas, and Freescale. Every C/C++ compiler contains both generic global optimizations as well as low-level chip-specific optimizations to ensure small code size, yet takes advantages of specific features of the device used.

IAR J-TRACE AND IAR J-LINK

IAR J-Trace and IAR J-Link are small ARM hardware debug devices that connect via USB to a PC host running Windows. They integrated seamlessly into IAR Embedded Workbench for ARM and are fully plug-and-play compatible.



Distributors

CORPORATE OFFICE AND DIRECT SALES

Micrium

1290 Weston Road, Suite 306
Weston, FL 33326
Tel: +1 954 217 2036
Fax: +1 954 217 2037
www.micrium.com
sales@micrium.com

NORTH AMERICA

UNITED STATES

IAR Systems Software, Inc.

Century Plaza
1065 E. Hillsdale Blvd.
Foster City, CA 94404
Tel: +1 650 287 4250
Fax: +1 650 287 4253
www.IAR.com
info@IAR.com

IAR Systems Software, Inc.

2 Mount Royal
Marlborough, MA 01752
Tel: +1 508 485 2692
Fax: +1 508 485 9126
Info.us-east@IAR.com
www.IAR.com

IAR Systems Software, Inc.

555 Republic Drive, Suite 109
Plano, TX 75074
Tel: +1 972 526 0190
Fax: +1 972 526 0195
Info.us-central@IAR.com
www.IAR.com

CANADA

Micrium

1 Place du commerce
Verdun, Quebec H3E 1A2
Canada
Tel: +1 514 768 8108
Fax: +1 954 217 2037
www.Micrium.com
sales@Micrium.com

JORAL Technologies

Head Office
Ottawa Canada
4322 Donnelly Drive Kemptville
Ontario, Canada K0G 1J0

JORAL Technologies

Calgary Canada
10659 Shillington Cres. SW Calgary
Alberta, Canada T2W 0N8
Tel: 403 668 8587
Tel: 613 258 9683
Direct: 613 851 2155
Toll Free: 1 800 GO JORAL
Fax: 613 822 5137
www.JORALTECHNOLOGIES.com
Robert.Campbell@Joral.ca

SOUTH AMERICA

BRAZIL

IAR Systems Software, Inc.

Sao Paulo Representative Office
Rua Sapopemba, 25
Cep 13104-170 - Campinas - SP
Brazil
Tel: +55 19 3258-1118
Fax: +55 19 3258-1118
info@IAR.com

EUROPE

AUSTRIA

Embedded Office

Amann & Hillmann GbR
August-Braun-Str.1
88239 Wangen, Germany
Tel: +49 (0) 7522 970008-0
Fax: +49 (0) 7522 970008-99
Mr. Thomas Amann
amann@embedded-office.de
Mr. Michael Hillmann
hillmann@embedded-office.de
www.embedded-office.de

DENMARK

Inside Technology Nordic ApS

P.O. Box 4075
DK-8260 Viby J
Denmark
Tel: +45 87 34 11 00
Fax: +45 87 34 11 20
inside@insidetechnology.dk
www.insidetechnology.dk

FINLAND

Inside Technology Nordic ApS

P.O. Box 273
SE 55114 Jonkoping
Sweden
Tel: +46 (0) 36 340715
Fax: +46 (0) 36 340716
inside@insidetechnology.se
www.insidetechnology.se

FRANCE

NeoMore

ZA Ste Apolline
23 rue des Poiriers
78370 Plaisir
France
Tel: +33 (0) 1 30 64 15 81
Fax: +33 (0) 1 30 64 08 83
Mr. Jean-Luc Trassard
JL.Trassard@NeoMore.com
www.neomore.com

GERMANY

Embedded Office

Amann & Hillmann GbR
August-Braun-Str.1
88239 Wangen, Germany
Tel: +49 (0) 7522 970008-0
Fax: +49 (0) 7522 970008-99
Mr. Thomas Amann
amann@embedded-office.de
Mr. Michael Hillmann
hillmann@embedded-office.de
www.embedded-office.de

ITALY

Themis Embedded Innovation S.r.l.

Via Marsala, 34/A
21013 Gallarate (VA)
Italy
Tel: +39 0331 775 119
Fax: +39 0331 775 526
Mr. Luca Foglia
Luca.foglia@isystem.com

ISRAEL

Sightsys Ltd.

7 Imber St.
Kiryat Arie, P.O.B. 10267
Petach-Tikva 49002 Israel
Tel: 972 3 9222771
Fax: 972 3 9222059
Mr. Zvika Almog
Zivka@sightsys.co.il
www.sightsys.com

NORWAY

Inside Technology Nordic ApS

P.O. Box 4075
DK-8260 Viby J
Denmark
Tel: +45 87 34 11 00
Fax: +45 87 34 11 20
inside@insidetechnology.dk
www.insidetechnology.dk

SWEDEN

IAR Systems AB

P.O. Box 23051
SE-750 23 Uppsala, Sweden
Tel: +46 18 16 78 00
Fax: +46 18 16 78 38
info@iar.se

Inside Technology Nordic ApS

P.O. Box 273
SE-55114 Jonkoping
Sweden
Tel: +46 (0) 36 340715
Fax: +46 (0) 38 340716
inside@insidetechnology.se
www.insidetechnology.se

Combitech Systems AB

Slottsgatan 15
553 22 Jonkoping
Sweden
Tel: +46 36 194611
Fax: +46 36 194361
Mr. Carl Andersson
carl.andersson@combitechsystems.com



SWITZERLAND

Embedded Office

Amann & Hillmann GbR
August-Braun-Str.1
88239 Wangen, Germany
Tel: +49 (0) 7522 970008-0
Fax: +49 (0) 7522 970008-99
Mr. Thomas Amann
amann@embedded-office.de
Mr. Michael Hillmann
hillmann@embedded-office.de
www.embedded-office.de

THE NETHERLANDS

Impactions

Handelsweg 8S
9804 TK Noordhorn
The Netherlands
Tel: +31 (0) 594 79 51 37
Fax: +31 (0) 594 59 11 65
Mr. Jeroen Verheul
j.verheul@impactions.eu

UNITED KINGDOM

IAR Systems Ltd.

Beckett House
14 Billing Road
Northampton
NN1 5AW
United Kingdom
Tel: +44 (0) 1604 250 440
Fax: +44 (0) 1604 250 330
info@iarsys.co.uk

IAR Systems China

Room 2510, Central Plaza
No. 227, North Huangpi Road
Shanghai, 200003
P.R. China
Tel: +86 21 6375 8658
Fax: +86 21 6375 8650
info@iar.se

INDONESIA

Testech Electronics Pte Ltd

Block 118
Aljunied Ave 2, 05 110
Singapore 380118
Tel: +65 67492162
Fax: +65 67494246
Mr. Kevin Quek
sales@testech-elect.com
www.testech-elect.com

MALAYSIA / PHILIPPINES / SINGAPORE

Testech

Electronics Pte Ltd

Block 118
Aljunied Ave 2, #05-110
Singapore 380118
Tel: +65 67492162
Fax: +65 67494246
Mr. Kevin Quek
sales@testech-elect.com
www.testech-elect.com

SOUTH KOREA

DIOIZ

7th Floor Acorn Place
757-3 Neson dong, Uiwang
Kyonggido 437-081
South Korea
Tel: +82 2 785 5709
Fax: +82 2 2653 0433
Mr. Won Ho Sung
dioiz@dioiz.com
www.DIOIZ.com

TAIWAN

Micetek

International Inc.

8F, 184, Sec. 1, Chung-Cheng Road
Taipei 111, Taiwan, R.O.C.
Tel: +886 2 28386826
Fax: +886 2 28386827
Mr. Mos Lin
mos.lin@micetek.com
www.Micetek.com

THAILAND

Testech

Electronics Pte Ltd

Block 118
Aljunied Ave 2, #05-110
Singapore 380118
Tel: +65 67492162
Fax: +65 67494246
Mr. Kevin Quek
sales@testech-elect.com
www.testech-elect.com

AUSTRALIA

/ NEW ZEALAND

MacroDynamics Pty. Ltd.

Suite 7, 322 Mountain Hwy.
Wantima, Victoria, 3152
Tel: +61 3 9720 2399
Fax: +61 3 9720 2486
rod@macrodynamics.com.au
www.macrodynamics.com.au



ASIA

CHINA

BMRTech

Beijing Office Headquarters
D403, JinYuJiaHua Building
No. 9 3rd Street
Shangdi Haidian District
Beijing, China
Tel: +86 10 6297 5900
Fax: +86 10 6297 5227
General Manager: Allan He
allan.he@bmrtech.com

BMRTech

Shanghai Office Room A, 23/F
North Tower, Yin Tong Building
Ding Xi Road
Shanghai, China 200050
Tel: +86 21 6212 7690
Fax: +86 21 6212 8032
Manager: Yin Lijie
lijie.yin@bmrtech.com

BMRTech

Shenzhen Office
Room 1001 - 1002
HangGang FuChun
Commercial Building
No. 6031, Shennan Road
Futian District
ShenZhen 518040, China
Tel: +86 75 5883
54481/82/83/85/86
Manager: Liang Changjing
changjing.liang@bmrtech.com

INDIA

Vi Microsystems Pvt. Ltd.

Chennai (Head Office)
Plot No. 75, Electronics Estate,
Perungudi, Chennai - 600 096
Tamil Nadu
Tel: 044 24961842, 24961852
sales@vimicrosystems.com

JAPAN

IAR Systems K.K.

Toyo Sudacho Bldg. 6F
1-5 Kanda-Sudacho, Chiyoda-ku
101-0041 Tokyo
Japan
Tel: +81 3 5298-4800
Fax: +81 3 5298-4801
info@iarsys.co.jp

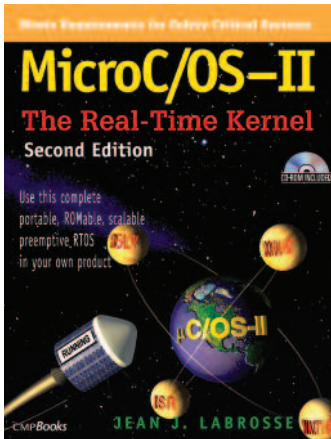
TechnoLogic

5F DAIICHI Bldg. 1-3-18
Namba-naka Naniwa-ku
Osaka City
556-0011, Japan
Tel: +81 6 4397 6620
Fax: +81 6 4397 6630
Mr. Masayuki Wada
info@t-logic.jp
www.t-logic.jp

Micrium wrote the book

MICRIUM WRITES THE BOOKS FOR EMBEDDED DESIGN

You hear it all the time: someone is knowledgeable so it's said that they “wrote the book” on a certain subject. In Micrium’s case, however, it’s literally true.



The Real-Time Kernel

Micrium “wrote the book” on **μC/OS-II**. Micro **μC/OS-II**, *The Real-Time Kernel* (ISBN 1-57820-103-9) by Jean J. Labrosse, CEO of Micrium, is used globally as a definitive RTOS text, ensuring a high number of design engineers that are both familiar and proficient with its use. Industry surveys, such as those conducted annually by TechInsights, consistently list **μC/OS-II** as one of the top three RTOSes selected in several categories by design engineers, and the book is ranked number one in the embedded design industry.

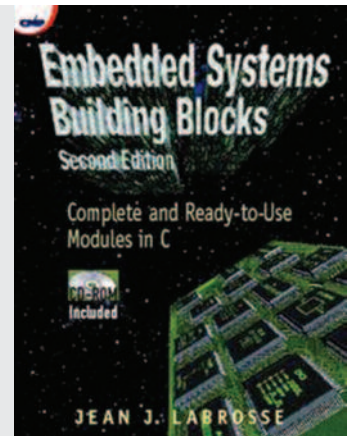
“If you currently do not use an RTOS, read this book to gain a deep understanding of what these powerful tools can do for your code. If you’re looking at commercial products, read the book to understand some of the trade-offs and gain an appreciation for what happens when in an OS.”

— Jack Ganssle, consultant, teacher, and writer

Also Written by Jean Labrosse:

Embedded Systems Building Blocks: Complete and Ready-to-Use Modules in C

Get a clear explanation of microcontroller theory and functional code modules that can be used to create basic embedded system functions. And, take the mystery out of embedded systems design with concrete programming examples.



Micrium

For the way Engineers work