The Embedded Experts

Embedded Studio

Debug Probes

RTOS

Data Management

Connectivity

Security

User Interface

Production

Developing with SEGGER Embedded Studio

SEGGER Embedded Studio is a streamlined and powerful C/C++ IDE (Integrated Development Environment) for ARM microcontrollers. It is specifically designed to provide everything needed for professional embedded development: an all-in-one solution providing continuous workflow.

Cross-Platform Support

SEGGER Embedded Studio is available for Windows, OS X and Linux. Its look and feel is similar on all platforms to provide the best experience regardless of the operating system.

Target Support

SEGGER Embedded Studio can be used with ARM®7, ARM®9, and the complete ARM® Cortex® microcontroller series.
SEGGER Embedded Studio offers a reduced cost license for those working only with Cortex®-M, and alternatively a license which covers the full range of supported ARM® microcontrollers.

Powerful Project Manager

An advanced Project Manager is included with SEGGER Embedded Studio which enables simple management of extremely large projects and multi-project solutions.
SEGGER Embedded Studio's Package Manager provides access to Support Packages for various ARM MCUs which can be installed on demand and updated when a new version is available. The Support Packages make starting a new project for new target hardware as simple as clicking a button.

Compiler Included

SEGGER Embedded Studio comes with advanced GCC and LLVM compilers for ARM® microcontrollers. With its highly-optimized, royalty-free, ANSI/ISO-C compliant standard C library, which has been developed specifically for embedded applications, you can expect the highest performance for your applications.

Feature-packed Debugger

SEGGER Embedded Studio integrates a feature-packed graphical Debugger with enhanced J-Link integration for direct debugging on your target hardware. All of the industry leading J-Link features have been tightly integrated into SEGGER Embedded Studio.
The debugger includes various debugging windows, which make it possible to inspect and manipulate advanced information concerning the running application and its execution, including mixed-mode disassembly, source code, an I/O Terminal for semihosting, SWO, SEGGER's Real-Time Transfer, and a scriptable Threads Window to be used with any (real-time) operating system.

First-Class Editor

The first-class Source Code Editor does not only support user-defined syntax highlighting, automatic code indention and matching bracket highlighting, it also provides a code completion feature for symbols, functions and keywords of your application, as well as configurable code and comment templates to easily match your coding and documentation standards.
The Editor is highly integrated into the Project Manager for efficient and advanced search and replace functionality in your files, projects and solutions. The behavior of all features is fully user-configurable.

Features

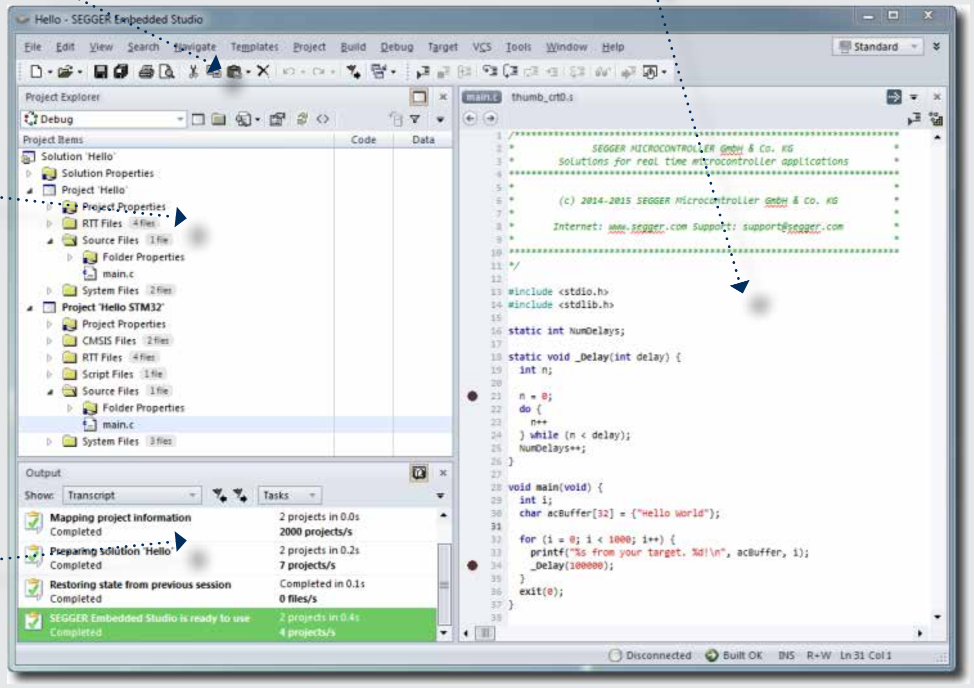
- Windows, Mac OS X and Linux support
- Powerful Project Manager, even for huge projects
- Advanced first-class Editor
- Package-based Project Generator for all common microcontrollers
- Pre-built C/C++ Compiler, GCC and LLVM included for an immediate start
- Royality-free ANSI / ISO C compliant C library for embedded systems
- Feature-packed Debugger with seamless J-Link integration

Feature packed Debugger
A graphical Debugger with J-Link integration and mixed-mode disassembly. I/O Terminal for semihosting. Threads Windows to be used with any (real-time) OS.

First-Class Editor
Syntax highlighting with automatic code indention and matching bracket highlighting. Code completion for symbols, functions and keywords.

Powerful Project Manger
Capable of managing huge and multi-project solutions easily. Provides access to Support Packages for various ARM MCUs.

Compiler included
LLVM based and industry standard GCC compiler. Pre-built for ARM microcontrollers. Highly-optimized, royalty-free ANSI / ISO C compliant standard C/C++ library.



```
Mapping project information                   2 projects in 0.0s
Completed                                     2000 projects/s
Preparing solution 'Hello'                    2 projects in 0.2s
Completed                                     7 projects/s
Restoring state from previous session         Completed in 0.1s
Completed                                     0 files/s
SEGGER Embedded Studio is ready to use        2 projects in 0.4s
Completed                                     4 projects/s
```

```c
/*********************************************************************
*                SEGGER MICROCONTROLLER GmbH & Co. KG                *
*        Solutions for real time microcontroller applications        *
**********************************************************************
*                                                                    *
*        (c) 2014-2015 SEGGER Microcontroller GmbH & Co. KG          *
*                                                                    *
*       Internet: www.segger.com Support: support@segger.com         *
*                                                                    *
**********************************************************************
*/

#include <stdio.h>
#include <stdlib.h>

static int NumDelays;

static void _Delay(int delay) {
  int n;

  n = 0;
  do {
    n++
  } while (n < delay);
  NumDelays++;
}

void main(void) {
  int i;
  char acBuffer[32] = {"Hello World"};

  for (i = 0; i < 1000; i++) {
    printf("%s from your target. %d!\n", acBuffer, i);
    _Delay(100000);
  }
  exit(0);
}
```

J-Link & J-Trace Debug Probes

SEGGER J-Links are the most widely used line of debug probes available today. They have proven their worth for more than 10 years with over 400,000 units sold. This popularity stems from the unparalleled performance, extensive feature set, large number of supported CPUs, and compatibility with all popular development environments.

Debug Smarter and Faster with J-Link!

With up to 3 MByte per second download speed to RAM and record-breaking flashloaders, and with the ability to set an unlimited number of breakpoints in flash memory of MCUs, J-Link debug probes are undoubtedly the best choice to optimize your debugging and flash programming experience.

Extensive Device & IDE Support

J-Link debug probes support all ARM 7/9/11, Cortex™, Microchip PIC32™, Renesas RX™ CPUs and are supported by all major IDEs such as Keil MDK-ARM, GDB-based IDEs and SEGGER Embedded Studio.

Direct Download into Flash Memory

J-Link can program the internal flash of almost all popular microcontrollers as well as external CFI compliant flashes. From a debugger perspective, the flash area can be treated just like RAM, so this great feature works with any debugger, including GDB.

Intelligence in the Firmware

In contrast to other Debug Probes J-Link has intelligence for different CPU cores in its firmware. For most emulators, the CPU communication handling is done completely from the PC side while the emulator simply outputs some PC-generated sequences. This makes special handling for scenarios like "low-power" and "very slow CPU speed at very high debug interface speed" almost impossible to handle. Having intelligent emulator firmware which is able to handle such cases itself, makes J-Link more robust in these situations.

Software Development Kit (SDK)

For customers who want to build their own applications using J-Link, and for IDE vendors who implement J-Link support for their IDE, SEGGER offers a J-Link SDK which comes with the J-Link DLL + API documentation + implementation samples. The SDK is available for Windows and Linux.

Cross-Platform Support

The Following platforms are supported: Windows, Linux, Mac OS X. The versions are fully usable and contain the following components: J-Link Commander, Command line GDB-Server, Shared library (DLL-equivalent).

Supports SWV/SWO

J-Link fully supports ARM's SWV/SWO feature which is available for most devices which support the SWD interface. SWO is a single pin output from the core which can be used to transfer terminal data (printf) and also, real-time trace data. The later enables monitoring of variable read and write accesses in compatible processors.

Features

- Support for ARM® Cortex®-M / R / A cores and ARM® 7 / 9 / 11, Microchip PIC32, Renesas RX and Silicon Labs 8051
- Maximum JTAG speed 15 MHz, J-Link ULTRA+ / PRO: 50 MHz
- Download speed up to 1.5 MB/s (J-Link® / J-Link® PLUS), 3 MB/s (J-Link® ULTRA+ / PRO)
- Very fast flash loader
- Supported by all popular debuggers
- Support for different debug interfaces: JTAG / SWD / FINE / SPD / ICSP
- Serial Wire Viewer (SWV) with up to 7.5 / 25 MHz supported
- Host interface: USB, Ethernet
- Power over USB
- Support for adaptive clocking
- Mulit-core debugging supported
- Wide target voltage range: 1.2V - 5.0V tolerant
- J-Link Remote Server included, which allows using J-Link via TCP/IP networks
- SDK available

Real Time Transfer

Real Time Transfer (RTT) is SEGGER's new technology for interactive user I/O in embedded applications. It combines the advantages of SWO and semihosting at very high performance, with data transfer speed reaching up to 2 MByte per second while retaining the real-time behaviour of the target system.

Unlimited Flash Breakpoints

The Unlimited Flash Breakpoints feature allows the user to set an unlimited number of breakpoints when debugging in flash memory. Without this feature, the number of breakpoints which can be set in flash is limited to the number of hardware breakpoints supported by the debug unit of the CPU. Unlimited Flash Breakpoints works in both internal and external flash, even with memory-mapped flashes.

Monitor Mode Debugging

Monitor Mode Debugging enables an embedded system based on a Cortex-M3, M4 or M7 core to maintain essential functionality while being debugged. This offers the possibility to maintain real-time, user-defined functions in selected interrupt services, such as motor control, data acquisition, or any application that needs some kind of continuous operation.

Remote Server Debugging in Tunnel Mode

The J-Link Remote Server effortlessly debugs target hardware and application in remote locations over TCP/IP as if the target was on the developer's desk. Taking this concept to the next level, SEGGER offers a tunnel mode for remote debugging anywhere in the world.
Tunnel mode initiates the connection sending the serial number of the J-Link to the tunnel server. The J-Link DLL then is capable of creating a tunnel connection via the server just by using the serial number of the target J-Link.
Support engineers can debug unwieldy hardware at the customer's site without having to travel there, just by sending a J-Link. Distributed development teams can share early prototypes even in remote locations.

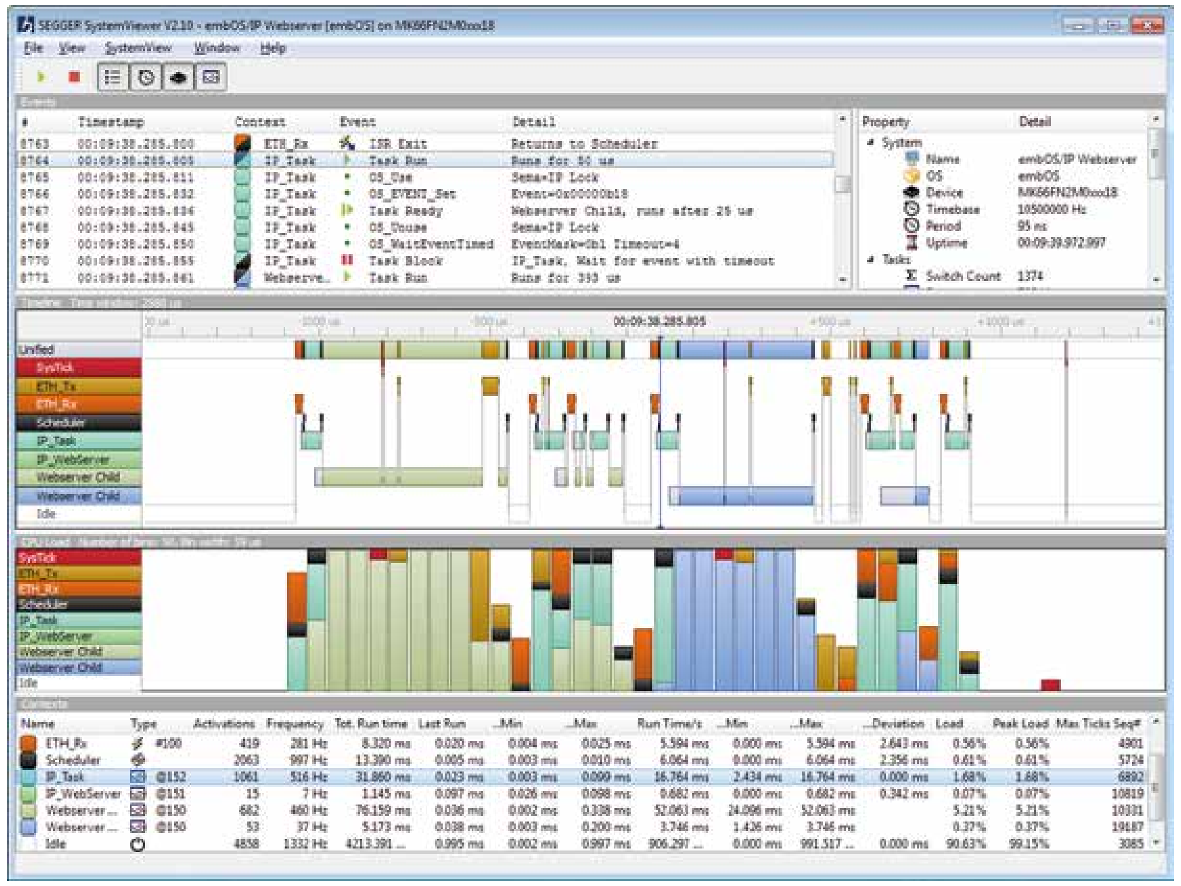
Sophisticated Tools for use with J-Link Technology

SEGGER SystemView

SystemView is a new tool for the visual analysis of any embedded system. It provides complete insight into the runtime behavior of an application, with minimal side effects on the observed embedded system. Using SystemView with J-Link gives the additional benefit of unlimited recording capacity and real-time analysis. In order to ensure real-time delivery of data, SystemView makes use of RTT.

J-Scope

J-Scope is a focused tool that visualizes data on a microcontroller in real-time, while the target is running. It does not require features like SWO or any extra pin on the target, but it only uses the existing debug port. J-Scope can show the value of multiple variables in an oscilloscope-like style. It reads an ELF file and allows selection of a number of variables to visualize. Simply connect the target microcontroller to a J-Link, flash an application and start J-Scope.
A visual graph and data is now available for analysis and manipulation, such as scrolling, zooming in and out, or saving the data to a file for further analysis.
J-Scope can be used in parallel with your debugging environment and extends your IDE's debugging experience.


SEGGER SystemViewer V2.10 - embOS/IP Webserver [embOS] on MK66FN2M0xxx18

#	Timestamp	Context	Event	Detail
8763	00:09:38.285.800	ETH_Rx	ISR Exit	Returns to Scheduler
8764	00:09:38.285.805	IP_Task	Task Run	Runs for 50 us
8765	00:09:38.285.811	IP_Task	OS_Use	Sema=IP Lock
8766	00:09:38.285.832	IP_Task	OS_EVENT_Set	Event=0x00000b18
8767	00:09:38.285.836	IP_Task	Task Ready	Webserver Child, runs after 25 us
8768	00:09:38.285.845	IP_Task	OS_Unuse	Sema=IP Lock
8769	00:09:38.285.850	IP_Task	OS_WaitEventTimed	EventMask=0x1 Timeout=4
8770	00:09:38.285.855	IP_Task	Task Block	IP_Task, Wait for event with timeout
8771	00:09:38.285.861	Webserve...	Task Run	Runs for 393 us

Property	Detail
System	
Name	embOS/IP Webserver
OS	embOS
Device	MK66FN2M0xxx18
Timebase	10500000 Hz
Period	95 ns
Uptime	00:09:39.972.997
Tasks	
Switch Count	1374

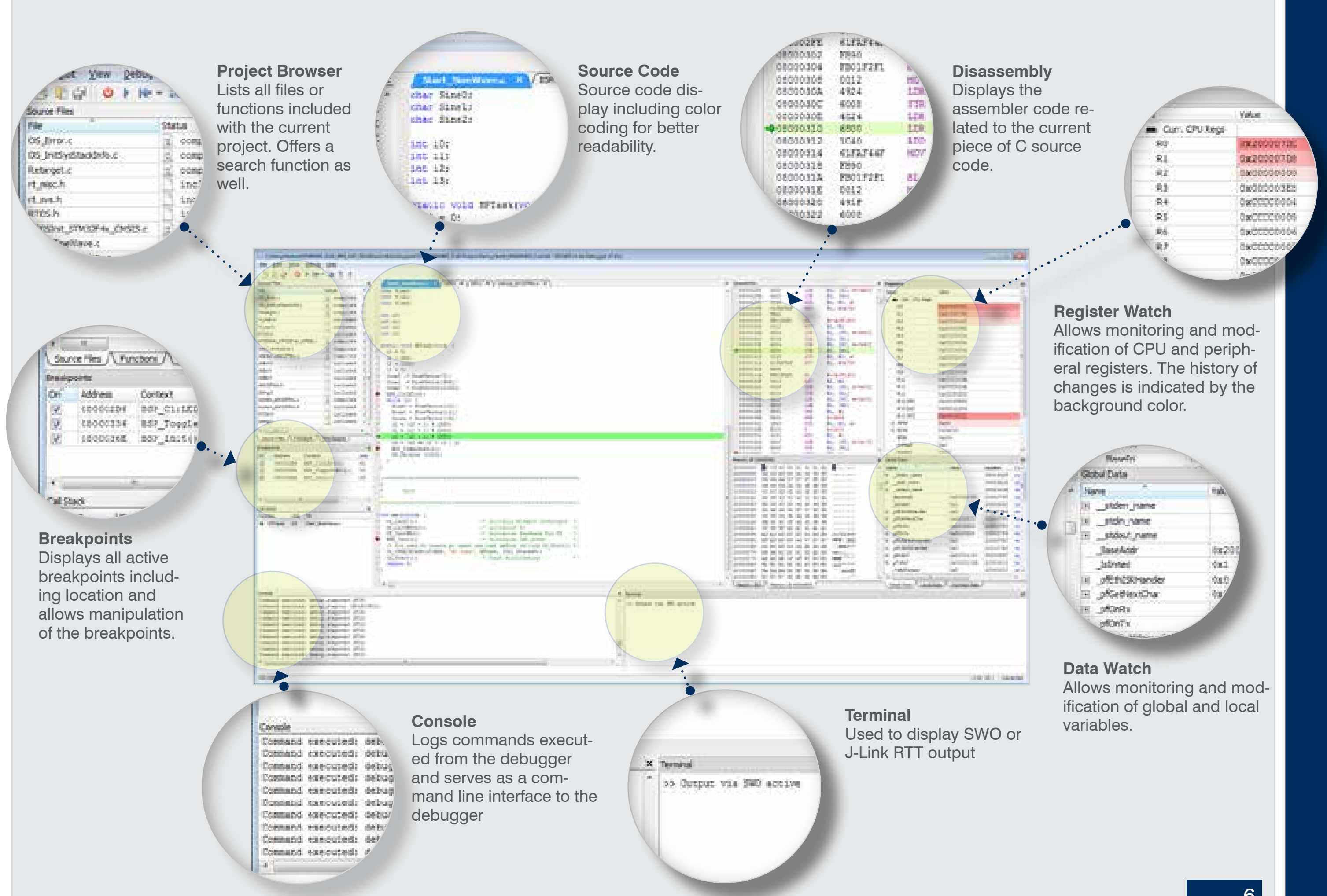
Name	Type	Activations	Frequency	Tot. Run time	Last Run	...Min	...Max	Run Time/s	...Min	...Max	...Deviation	Load	Peak Load	Max Ticks Seq#
ETH_Rx	#100	419	281 Hz	8.320 ms	0.020 ms	0.004 ms	0.025 ms	5.594 ms	0.000 ms	5.594 ms	2.643 ms	0.56%	0.56%	4901
Scheduler		2063	997 Hz	13.390 ms	0.005 ms	0.003 ms	0.010 ms	6.064 ms	0.000 ms	6.064 ms	2.356 ms	0.61%	0.61%	5724
IP_Task	@152	1061	516 Hz	31.860 ms	0.023 ms	0.003 ms	0.099 ms	16.764 ms	2.434 ms	16.764 ms	0.000 ms	1.68%	1.68%	6892
IP_WebServer	@151	15	7 Hz	1.145 ms	0.097 ms	0.026 ms	0.098 ms	0.682 ms	0.000 ms	0.682 ms	0.342 ms	0.07%	0.07%	10819
Webserver ...	@150	682	460 Hz	76.159 ms	0.036 ms	0.002 ms	0.338 ms	52.063 ms	24.096 ms	52.063 ms		5.21%	5.21%	10331
Webserver ...	@150	53	37 Hz	5.173 ms	0.038 ms	0.003 ms	0.200 ms	3.746 ms	1.426 ms	3.746 ms		0.37%	0.37%	19187
Idle		4858	1332 Hz	4213.391 ...	0.995 ms	0.002 ms	0.997 ms	906.297 ...	0.000 ms	991.517 ...	0.000 ms	90.63%	99.15%	3085

Debugging with J-Link Debugger

J-Link Debugger is a full-featured graphical debugger for embedded applications. With J-Link Debugger it is possible to debug any embedded application at C source and assembly level. J-Link Debugger can load applications built with any toolchain / IDE or debug the target's resident application without any source. J-Link Debugger includes extensive debug information windows and makes use of the best performance of J-Link debug probes. The user interface is designed to be intuitive and is fully configurable. All windows can be moved, re-sized and closed to fit the need of any developer.

Features

- Blindingly fast debug and step performance
- Modern, intuitive user interface
- Precision stepping at source or assembly level
- Thread-aware debugging (customizable)
- C code source level debugging and assembly instruction debugging
- Direct use of J-Link built-in features
- Extensive debug & processor state windows
- Scriptable project files



Project Browser
Lists all files or functions included with the current project. Offers a search function as well.

Source Code
Source code display including color coding for better readability.

Disassembly
Displays the assembler code related to the current piece of C source code.

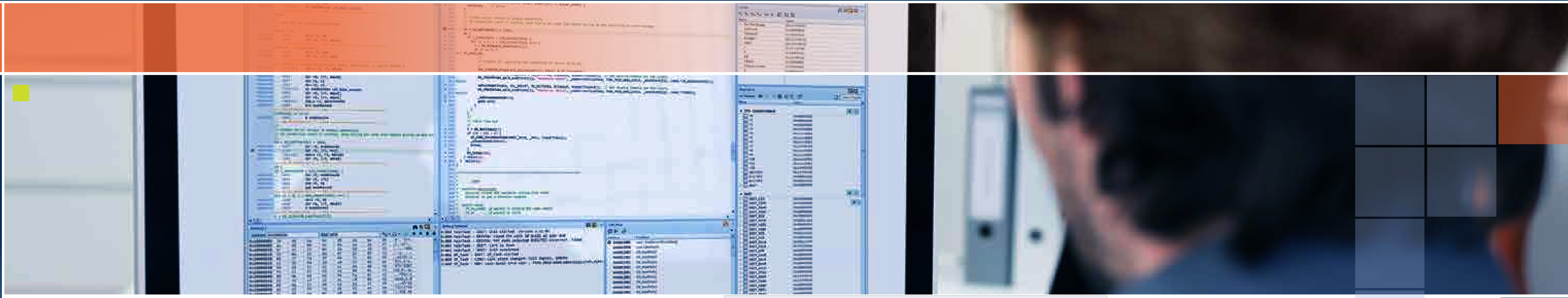
Register Watch
Allows monitoring and modification of CPU and peripheral registers. The history of changes is indicated by the background color.

Breakpoints
Displays all active breakpoints including location and allows manipulation of the breakpoints.

Data Watch
Allows monitoring and modification of global and local variables.

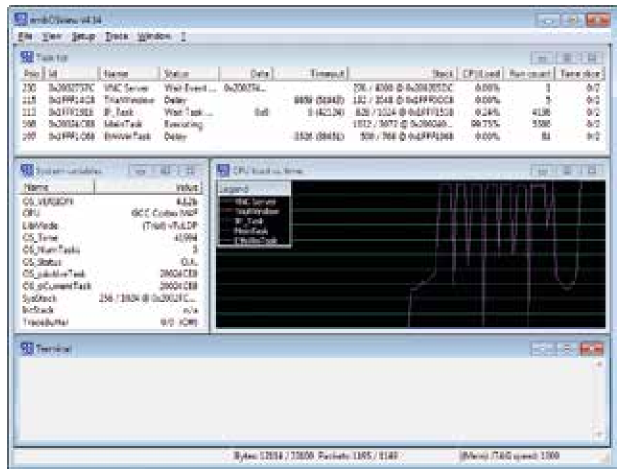
Console
Logs commands executed from the debugger and serves as a command line interface to the debugger

Terminal
Used to display SWO or J-Link RTT output



The Real Time Operating System embOS

embOS is a real time operating system for embedded applications designed to offer the benefits of a fully featured multitasking system, even for hard real time applications using minimal resources.


(embOSView)

Developing applications with embOS

embOS is available in source or object code form. Both come with a ready-to-go start project: The first multi-task program will be running within five minutes. The start application and usable samples are supplied in source code form. Libraries for all memory models and initialization of the controller in C-source are included to tailor the system to any application. Check out the free fully functional trial versions, which can be downloaded from our website www.embos.com.

Profiling using embOSView

embOSView communicates with embOS running on the target over a UART and displays all available information of the tasks and major system variables. All communication is done from within the communication interrupt routines. This means that it is non-intrusive if embOSView is not connected and minimally intrusive while embOSView is connected. On most CPUs a profiling build of the libraries is available. In profiling build, embOS collects precise timing information for every task, which enables embOSView to show the CPU load.

Zero Latency Interrupts

embOS is perfectly suited for hard real time conditions as it does not block high priority or zero latency interrupts. High priority interrupts do not add additional latency or jitter from the operating system. When an RTOS modifies its data structures, it has to block access to its data structures. In order to achieve that, low priority interrupts are blocked. High priority interrupts cannot call OS functions.

Simulation environment

A simulation environment running under MS Windows is available. It can be used to write and test the entire application on your PC (all routines are 100% identical to your embedded application). This makes debugging and development easy and convenient and saves development time. The simulation is an open environment, which also allows adding C-code to simulate the target specific hardware. embOS Simulation comes with a ready-to-go start project for MSVC, but may also be used with other tool chains. A trial version is available upon request.

Power-Saving Tickless Mode

embOS tickless low power support reduces the power consumption for e.g. battery powered devices. Instead of having a timer interrupt for each system tick the timer is reprogrammed to be able to spend as much time as possible in low power mode.

Features

- Preemptive scheduling
- Unlimited priorities
- Round-robin scheduling for tasks with identical priorities.
- No configuration needed
- Intertask communication
- Software timer
- No royalties
- Zero interrupt latency
- Fast & efficient
- Small footprint
- Easy to use start project
- Versatile
- Supported directly by developers
- System analysis tool included
- Profiling support

Technical info

Kernel size (ROM)	1100 - 1600 byte *
Kernel RAM usage	18 - 25 byte *
Kernel CPU usage at 1 ms Interrupts with 10MHz M16C	Less than 3%
RAM usage mailbox	9 - 15 byte *
RAM usage binary and counting semaphore	3 byte
RAM usage resource semaphore	4 - 5 byte *
RAM usage timer	4 - 5 byte *
RAM usage event	0
Basic time unit (One Tick)	Default 1 ms, can be configured, Min. 100 μs (M16C@10MHz) *
Task activation time	Independent of no. of tasks (e.g. typically 12 us M16C@10MHz)
Interrupt latency	Zero
No. of tasks	Unlimited (by available RAM only)
No. of mailboxes	Unlimited (by available RAM only)
No. of semaphores	Unlimited (by available RAM only)
No. of s/w timers	Unlimited (by available RAM only)
No. of priorities	Unlimited
No. of tasks with identical priorities (Round robin scheduling)	Unlimited

** Depends on CPU, compiler and library model used*

Supported processor families

All major 8-, 16-, and 32-bit processor families are supported.

ALTERA, NORDIC, ANALOG DEVICES, NXP, ARM, RENESAS, Atmel, SILICON LABS, CYPRESS, ST, freescale, SPANSION, infineon, TEXAS INSTRUMENTS, MICROCHIP, TOSHIBA, MIPS, XILINX

Data Storage with emFile

emFile is a file system for embedded applications which can be used on any type of storage device. emFile is a high performance library that has been optimized for minimum memory consumption in RAM and ROM, high speed and versatility. It is written in ANSI C and can be used on any CPU.

Device Drivers

emFile is designed to cooperate with any kind of embedded system and storage device. To use a specific medium with emFile, a device driver for that medium is required. The device driver consists of basic I/O functions for accessing the hardware and a global table, which holds pointers to these functions.
If you need to use a proprietary storage device, you can write your own device driver. Currently the following device drivers are available:
MultiMediaCard (MMC), Secure Digital (SD), RAM disk, Compact Flash, IDE, NOR flash, and NAND flash.

NOR/NAND Flashes

The Universal NAND flash driver works with all modern SLC and MLC NAND flashes. It can use the ECC engine built into NAND flashes to correct multi bit errors. The driver also works with SLC flashes which require 1-bit error correction and supports ATMEL's DataFlashes.
To enable the use of large NAND flash memories, the NAND flash driver allows block grouping to save memory required for administration of the memory blocks. The NOR flash driver can be used with any CFI compliant 16-bit chip. The Common Flash Memory Interface (CFI) is an open specification which may be implemented freely by flash memory vendors in their devices.

SEGGER NAND flash evaluation board

Wear Levelling

Wear levelling is supported by the NOR/NAND driver. Wear levelling makes sure that the number of erase cycles remains approximately equal for each sector, thus prolonging the life span of the whole flash memory. Maximum erase count difference is set to 5. This value specifies a maximum difference of erase counts for different physical sectors before the wear levelling uses the sector with the lowest erase count. In contrast to other products on the market, SEGGER's emFile offers both static and dynamic wear levelling. In order to keep erase cycles on the same level for all sectors, static data is regularly moved around to different sectors.

MMC and SD Cards

MMC and SD cards can be accessed through two different modes: either SPI MODE or MMC/SD card mode. For both modes drivers are available. To use one of these drivers, you need to configure the MMC driver and provide basic I/O functions for accessing your card reader hardware.

Encryption

The emFile Encryption add-on provides a simple way to encrypt individual files or the storage media as a whole. Encryption can be used with both available file systems - EFS and FAT. All storage types such as NAND, NOR, SD/MMC/CompactFlash cards are supported. To use encryption, only minor changes to the application program are required in order to select the encryption method and a password for volume or individual files.

Features

- MS DOS/MS Windows-compatible FAT12, FAT16 and FAT32 support, proprietary EFS file system
- Support for long file names
- Multiple device driver support.
- Multiple media support. A device driver allows you to access different media at the same time
- Cache support. Improves the performance of the file system by keeping the last recently used sectors in RAM
- Works with any operating system to accomplish a thread-safe environment
- ANSI C stdio.h-like API for user applications.
- Very simple device driver structure. emFile device drivers need only basic functions for reading and writing a block
- Optional NOR flash (EEPROM) driver. Any CFI-compliant NOR flash is supported. Wear levelling included
- Optional device driver for NAND flash devices. Very high read/write speeds. ECC and wear levelling included
- An optional device driver for MultiMedia & SD cards using SPI mode or card mode that can be easily integrated
- An optional IDE driver, which is also suitable for CompactFlash using either True IDE or Memory Mapped mode
- An optional proprietary file system (EFS) with native long file name support
- An optional journaling add-on. It protects the integrity of file system against unexpected resets
- NAND flash evaluation board available

Journaling

Journaling is an additional component for emFile which sits above the file system and makes the file system layer fail-safe. File systems without journaling support (for example, FAT) are not fail-safe. Journaling means that a file system logs all changes to a journal before committing them to the main file system. To prevent corruptions from unexpected interruptions, caused for example by a power failure, the Journaling Layer caches every write access to achieve an always consistent state of the file system.

Memory requirements*

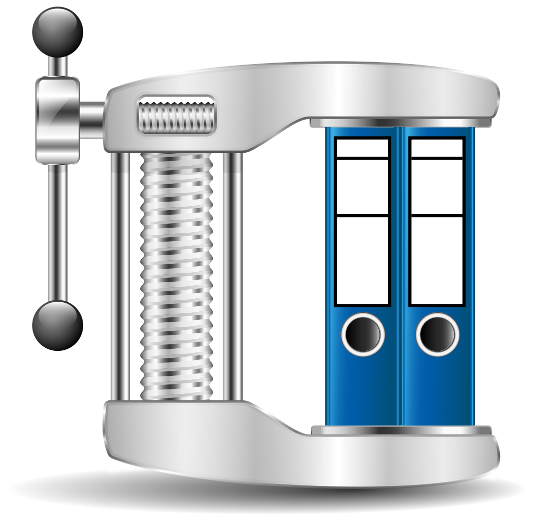
Memory requirements depend on the used CPU, compiler, memory model, as well as on various other factors such as configuration switches and selected drivers.

ROM:	app. 9-40 kb
RAM:	app. 2 kb

* Precise values depend on the functionality used. Values are measured on a specific target system and will be different on other systems.

Shrink data with emCompress

emCompress is a compression system that is able to reduce the storage requirements of data to be embedded into an application. A compressed version of the data is stored in the flash memory of the target system. In the target, a small, fast decompressor can decompress the data on-the-fly, whenever it is required. The decompressor can decompress into RAM or send the output to an application defined function.



Software only grows in one direction

With increasing complexity of today's devices, customers expect firmware updates over the lifetime of a device. It's important to be able to replace both firmware images and FPGA configuration bitstreams in the field. Typically, firmware upgrades and other static content grow in size over the lifetime of a device: features are added to software, not taken away, it's just what happens.

This is where emCompress can help. emCompress will compress your data so that it takes much less space on the target device. The decompressors are tiny in ROM, but the benefits of compression means you reclaim more space in your device for the features you're trying to add.

Because emCompress decompressors can be tailored for RAM use, you can decompress static content early in your application and not devote RAM to decompression buffers.

Typical example – Configuring an FPGA

For instance, configuring an FPGA is one of the first things that an application will do, decompressing a bitstream and sending it to the FPGA. In this case, a small decompression buffer can be held on the stack even though the compressed bitstream is hundreds of kilobytes in size: the temporary buffer is a known size that is configured at compression time, and that RAM is free for reuse the moment any decompression completes.

emCompress enables microcontrollers with small internal flash memory to store FPGA image bitstreams which otherwise would require the system designer to use a bigger version of the microcontroller.

Decompressing and Processing Data

emCompress features two decompression functions. The first one is decompression into memory. The complete data is decompressed and stored in a user-provided memory buffer. Although the buffer can be temporary, this requires to have enough free memory for the complete uncompressed data and the workspace. Decompression into memory can for example be useful for dynamic firmware images.

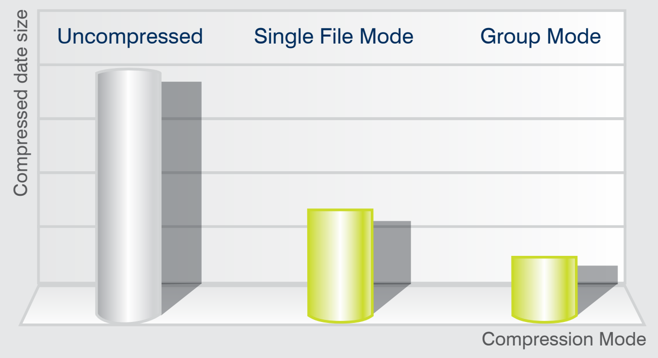
The second function is decompression in streamed mode. emCompress will use a small temporary buffer whose size is set by the user when compressing. Once a fragment of data is decompressed and the buffer is filled, the user-provided output function is called and the next buffer filled again with the next fragment. Streamed decompression is particularly effective for programming FPGAs or serving web content.

Features

- Highly efficient compression
- Small decompressor ROM footprint
- Fixed decompressor RAM use, chosen when compressing
- Wide range of codecs to choose from
- Automatic selection of best codec for each file
- Works with any operating system to accomplish a thread-safe environment
- Easy-to-understand and simple-to-use
- Simple to configure and integrate
- Royalty free

Group mode offers better compression!

emCompress can boost compression ratios using group mode when compressing many small files. In group mode, the uncompressed source files are concatenated and compressed as a whole, increasing the opportunities for the compressor to find ways to compress the combined data. This proves especially effective when compressing small, static HTML files for embedded web servers.



Performance and Memory Footprint

ROM use

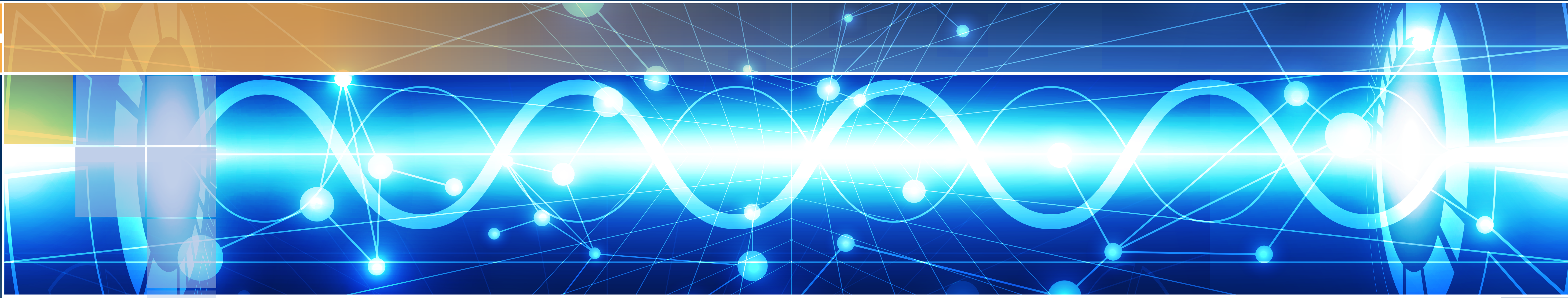
The amount of ROM that emCompress uses for decompression varies with the codec selected between 0.5 kByte and 2.1 kByte. The total ROM requirement includes a single decoder, and all supporting functions, excluding integrity checks.

RAM use

The amount of RAM that emCompress uses is under complete control as it is specified at compression time. Typically, 2KB of temporary RAM for decompression yield good compression ratios, but even without temporary RAM, good compression ratios can be achieved in many cases. No static RAM is required, stack usage is well below 512 bytes.

Typical uses of emCompress

Compression has many fields of application. Static data that is not frequently used and or has exceptionally high compression rates are the prime target applications. Typical examples are the configuration bitstreams to program FPGA and CPLD devices, permanent files for embedded web server static content, firmware updates and the user interface messages for multiple languages.



Get connected with embOS/IP

embOS/IP is a TCP/IP stack that provides a small memory footprint, high performance solution for embedded networking solutions. The stack has been optimized for use in real-time, memory-constrained embedded systems. It offers RFC-compliant TCP/IP and a standard socket API. embOS/IP works seamlessly with the embOS operating system. Several higher-level protocols are also available.

Easy to use

The stack comes with a variety of confidence tests and example applications. It runs out of the box. For most microcontrollers, sample projects are available. All modules can output debug messages and warnings in debug builds. The modules to output this information can be selected at run-time, allowing the developer to focus on the aspect he is analyzing.

Configuration free

The entire stack can be compiled into a library. Setup is reduced to a minimum, performed at run-time. This, along with a wealth of sample programs, gets you up and running quickly. Since inter-module dependencies are limited to the parts required for the functionality of the stack, unused parts of the code are automatically excluded by the linker.

Portability

embOS/IP is written in ANSI C and, apart from the Link-Layer-Driver, hardware-independent.

Performance

The stack has been optimized for both, performance and code size. The standard socket interface is complemented by the zero-copy API, which allows reading and writing of data without additional protocol buffers, if the target hardware and driver support DMA.

Multi-task support

embOS/IP allows any number of tasks to call API functions concurrently. The stack itself uses typically two tasks (one for housekeeping and one for packet reception), but can also be used with just a single or even no task (polled mode). The two-task model allows minimum interrupt latency on systems without nested interrupts.

embOS/IP structure

embOS/IP is organized in different layers:

Application Layer	DHCP, DNS, FTP, HTTP, Telnet
Transport Layer	TCP, UDP
Network Layer	IPv6, IPv4, ICMP, ARP, RARP, …
Link Layer	Ethernet (IEEE 802.3), PPP, …

embOS/IP Software Products

embOS/IP is offered in a BASE package which includes the most important protocols related to Ethernet communication and the stack itself. Depending on the engineer's needs there are several protocols available as add-ons, as well as the embOS/IP PRO software which includes a device driver, and all add-ons for extended communication via Internet.

The following protocols are part of the basic package:

- ARP (Address resolution protocol)
- IP (Internet protocol)
- ICMP (Internet control message protocol)
- UDP (User datagram protocol)
- TCP (Transmission control protocol)
- Standard Socket API
- Zero-Copy API
- DNS client
- DHCP client
- Telnet server (sample)

You can find the full list of additional modules here: **www.segger.com**

Features

- Standard sockets interface
- High performance
- Small footprint
- Runs "out-of-the-box"
- No configuration required
- Works with any RTOS in a multitasking environment (embOS recommended)
- Zero-copy for ultra-fast performance
- Drivers for most popular microcontrollers and external MACs available
- Easy to use!
- Raw socket support
- Non-blocking versions of all functions
- Unlimited Connections
- Re-assembly of fragmented packets
- Fully runtime configurable
- Developed from ground up for embedded systems
- PPP/PPPoE available
- Various upper layer protocols available
- Royalty-free

Optional Products

embOS/IP WEB SERVER
The embOS/IP web server allows embedded systems to present web pages with dynamically generated content. It has all features typically required by embedded systems: multiple connections, authentication, forms. The RAM usage of the web server has been kept to a minimum by smart buffer handling. The sockets interface can be used with any TCP/IP stack.

embOS/IP FTP SERVER and FTP CLIENT
The embOS/IP FTP server can use the same file system as the web server. It can be used in r/o or in r/w mode and allows reading and modifying of configuration data or web content. With the FTP client add-on data can be exchanged with any FTP server

emSSL Transport Layer
emSSL is a Transport Layer Security solution which allows secure and private connections with single-chip systems using as little as 7 kBytes of RAM. emSSL works perfectly with embOS/IP. For more details about emSSL, please refer to the product description on page 18.

Available Add-ons

embOS/IP PPP LINK LAYER
As an alternative to Ethernet PPP allows the use of IP via modem or GSM. Further add-ons are available.

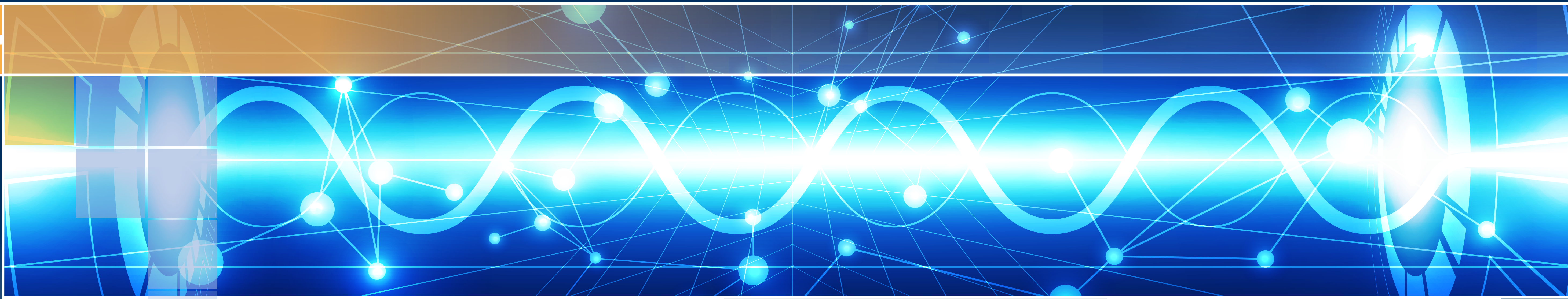
embOS/IP UPnP
Universal Plug&Play module.

embOS/IP SNTP Client
Simple Network Time Protocol.

embOS/IP SMTP Client
The Client allows to send emails from your embedded system via an email server.

Memory requirements

Memory requirements depend on the used CPU, compiler, memory model, as well as on various other factors.
A typical ROM size for a system using ARP, IP, ICMP, UDP, TCP and sockets is about 18kB (on typical 32-bit microcontroller with size optimization). Mimimum RAM usage is about 6KB for simple applications.



Get connected with emUSB

emUSB Device is a high-speed USB device stack specifically designed for embedded systems. The software is written in ANSI C and can run on any platform. emUSB can be used with embOS or any other supported RTOS. A variety of target drivers are already available. Support for new platforms can usually be added at no extra charge.

emUSB Components

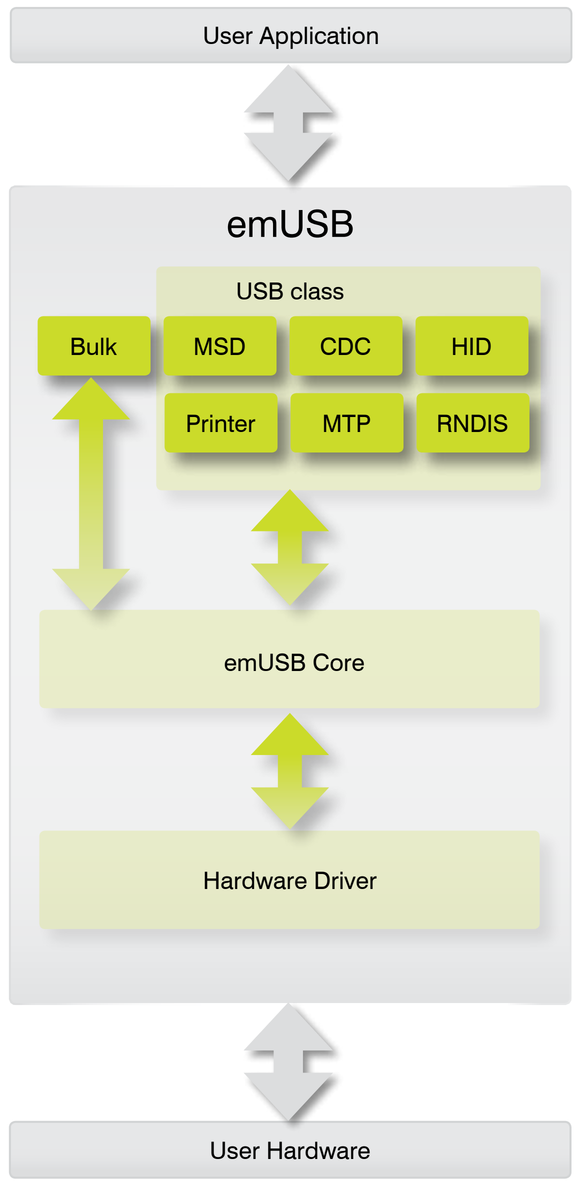
SEGGER offers a flexible USB device stack structure. The typical emUSB stack package consists of a target driver designed for your target hardware, the emUSB core and the Bulk, MSD, CDC or HID component,l or any combination thereof. The different available hardware drivers, the USB class drivers and the Bulk communication component are additional packages, which can be combined and ordered depending on project requirements.

Bulk Communication Component

emUSB-Bulk allows developers to quickly and painlessly develop software for an embedded device that communicates with a PC via USB. The communication is like a single reliable high-speed channel (very similar to a TCP connection). It allows the PC to send and receive data with the embedded target. This permits usage of the full bandwidth of the USB bus.

MSD Component

emUSB-MSD converts your embedded target device into a USB mass storage device. Your target device can then be plugged into a USB host like a PC and accessed as an ordinary disk drive, without the need to develop a kernel mode driver for the host operating system. This is possible because the mass storage class is one of the standard device classes, defined by the USB Implementers Forum. Every major operating system on the market supports these device classes out of the box. Since every major OS already provides host drivers for USB mass storage devices, there is no need to implement your own. The target device will be recognized as a mass storage device and can be accessed directly.



Components overview

Features

ISO/ANSI-C source code	CDC component available
Supports USB 1.1 / 2.0	RNDIS component available
Low/Full/High-Speed support	HID component available
Bulk communication component with Windows kernel mode driver available	Printer component available
MSD component available	Start/test applications supplied
Smart MSD works without file system	No royalties
	MTP component available

MTP Component

The MTP class supports file-based communication with the host system for all types of files. MTP is an alternative to MSD, without some of the latter's weaknesses. File based communication gives access to the file system from the host system (PC) and the device at the same time. Using the MTP class also allows selectively exposing content of the file system to the host system, typically a PC. Sudden removal of the USB cable does not endanger the data integrity of the device's file system.

CDC Component

emUSB-CDC converts the target device into a serial communication device. The host will recognize it as a virtual COM port (USB2COM). It allows the use of software which is not designed to be used with USB, such as a device datalogger or terminal program.

RNDIS Component (Internet via USB)

RNDIS enables developers to transform low end standalone products into connected devices with the same functionality as other devices on a local network. The most noteworthy application is a USB based web server. The USB web server allows users to configure their device and to view data using a web browser. This saves development costs as there is no need to develop an application for every major operating system.

HID Component

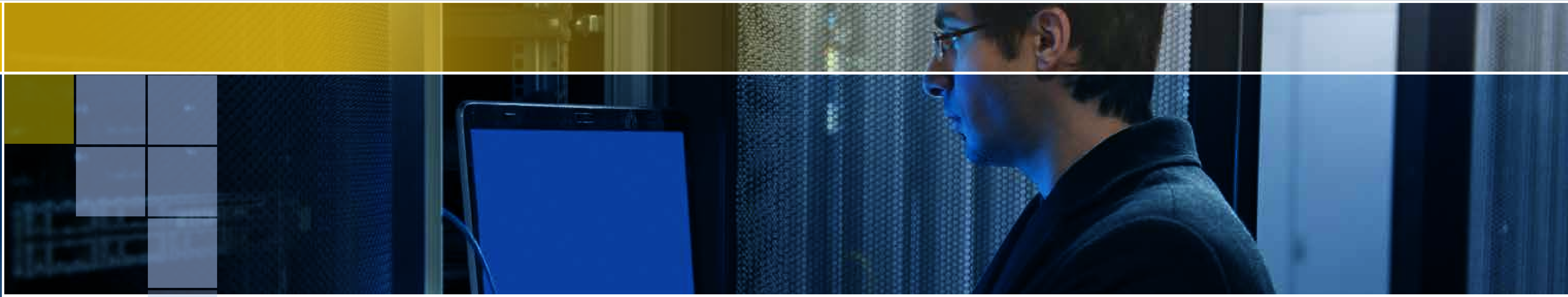
The Human Interface Device class (HID) is an abstract USB class protocol defined by the USB Implementers Forum. This protocol was defined for handling devices which are used by humans to control the operation of computer systems. An installation of a custom-host USB driver is not necessary, because the USB human interface device class is standardized and every major OS already provides host drivers for it. emUSB device also allows “Vendor specific HIDs.” These are HID devices communicating with an application program. The host OS loads the same driver as for any “true HID” and automatically enumerates the device. The application communicates with the particular device using API functions offered by the host. This allows an application program to communicate with the device without the need for loading a custom driver. The HID class is a good choice if ease of use is important and high communication speed is not a requirement.

Printer Component

The USB class protocol for printers was defined for handling output devices like printers and plotters. emUSB printer receives data from the host and forwards the data to a parser. The printer component provides automatic error handling routines, in case of events like the device running out of paper. The USB protocol is completely hidden from the developer and he can concentrate on developing the parser.

emUSB Host

SEGGER's USB host software stack implements full USB host functionality, including external hub support, and optionally provides device class drivers. It enables developers to easily add USB host functionality to embedded systems. The software stack supports all transfer modes (control, bulk, interrupt, isochronous) at low, full and high speed.



Digital Asset Protection with emSecure

In today's world, protecting one's reputation as a device manufacturer is more critical than ever before. No company wants to make headlines for the wrong reason. emSecure can help you in the fight against firmware hacking and device cloning.

Keep Customers Safe

emSecure is a software solution to securely authenticate digital assets, based on the concept of digital signatures. Deploying emSecure can help authenticate any plug-in card or attached device that contains a microprocessor capable of running emSecure.
It is much simpler than digital certificates to implement and deploy, but offers the same level of protection and more flexibility. Critically, emSecure is not licensed on a per-device basis, lowering your costs for production runs.

Protect against Cloning

One important feature is that it protects an embedded device against the creation of a clone by simply copying hardware and firmware.

Indispensable for Critical Devices

And it can do much more, such as securing firmware updates for any kind of embedded device, licenses, serial numbers or other sensitive data. emSecure is therefore indispensable for critical devices such as election machines and payment terminals or any other tamper proof application in the industrial, automotive and health care market. Based on asymmetric encryption with two keys, it cannot be broken by reverse engineering.

Created for Embedded Systems

The source code has been created from scratch for embedded systems, to achieve highest portability with a small memory footprint and high performance. However, usage is not restricted to embedded systems, but includes for example PC, and smartphone apps.

Features

- Alternative key generation schemes RSA and ECDSA - Dual keys, private and public make it 100% safe
- Hardware-independent, any CPU, no additional hardware needed
- High performance, small memory footprint
- Simple API, easy to integrate
- Applicable for new and existing products
- Complete package, key generator and tools included
- Drag-and-drop Sign And Verify application included
- Full source code
- Free 'Sign & Verify' Windows Version to Protect Personal Files

Developer Friendly

It is licensed in the same way as other SEGGER middleware products and does not rely on any foreign code or code licensed under an open-source or even "viral" GPL-style license.
With its easy usage, it takes less than half a day to add and integrate emSecure into an existing product.

Using emSecure is Easy

emSecure has a simple yet powerful API. It can be easily integrated into an existing application. The code is completely written in ANSI C and can be used platform- and controller-independent.

Key pairs can be generated on a computer, as well as on any embedded system itself. The generated keys can be exported into different formats to be stored in the application code or loaded from a key file. This allows portability and exchangability between different platforms.

Safe Data Transport with emSSL

emSSL is a SEGGER software library that enables secure connections across the Internet. emSSL offers both client and server capability. SSL/TLS is a must-have in nearly every application which is connected to the Internet. Products for IoT, smart grid or home automation markets benefit from securing their communication.

Suitable for Single-Chip Systems

The mimized RAM usage enables operation of emSSL on single-chip systems. a secure connection between browser and the web server requires only 7 KB of RAM. This way, even small embedded devices can establish secure connections.

Secured Connections

emSSL offers the possibility to establish a secured connection to any server application from your product. It can be used both target independent in native computer applications, and in embedded targets.

The emSSL Package

emSSL is a complete package and comes with everything needed to secure communication. It includes all modules to implement the required functionality to use SSL. They are provided in source code to allow complete control of the code used in the product and create transparency to avoid worries about possible back doors or weakness in code, which cannot be checked in precompiled libraries.

emSSL Makes it Easy

emSSL comes with a simple, yet powerful API to make using emSSL in your product as easy as possible.

It also includes sample applications in binary and source code, which demonstrate how and when emSSL can be used in real life scenarios. For a list of included applications, see the chapters below.

Features

- ISO/ANSI C source code
- Supports USB 1.1 / 2.0 devices
- Full/High Speed support
- Bulk communication component with Windows kernel mode driver available
- MSD component available
- Virtual MSD works without file system
- Sign & Verify drag-and-drop
- FIPS specifications issued by NIST (FIPS 186-4)

Performance

emSSL is built for high performance with target independent code. It is completely written in ANSI-C and can be used in any embedded application, as well as in PC applications.

Configurable

emSSL is created for high performance and a low memory footprint. The library can be configured to fit any speed or size requirements. Unused features can be excluded, additional features can easily be added.

Supported Cipher Suites

emSSL includes the most commonly used cipher suites, which allows connection to nearly every TLS-supporting server.

With emSSL the cipher suites can be added dynamically. When the required cipher suites are known it is possible to create a minimal size configuration by not linking in unused algorithms. This is can be done by the compiler/linker automatically. With the included scan suites application it is possible to find out the required cipher suite(s) to connect to a server.

Graphical User Interface with emWin

emWin is designed to provide an efficient, processor- and LCD controller-independent graphical user interface (GUI) for any application that operates with a graphical LCD. It is compatible with single-task and multitask environments, with a proprietary operating system or with any commercial RTOS. emWin is shipped as C source code. It may be adapted to any physical and virtual display with any LCD controller and CPU.

Attractive and useful display with reduced time and cost

One of the most challenging aspects of many development projects is designing an attractive and useful display. In addition to creating images that look exactly how you want them to appear, the implementation of windows techniques, complex drawing routines, different fonts and flicker-free updates are also expected. As part of a complex process which can take months or even years, the developer only has a short time available for the implementation of the display functionality.

Example speedometer

emWin, probably the most efficient and comprehensive embedded GUI available, helps developers beat their timelines and development costs. It is written in ANSI C and supports any b/w, gray-scale or color display. Drivers for all popular LCD controllers are available. All types of graphical displays (STN-LCD, TFT, CRT, OLED, Plasma, ...) are supported.

Drivers for Display Controllers

Run-time configurable drivers can be written for all types of displays and display controllers, including monochrome, gray-scale, passive and active color (TFT) displays. Drivers for all popular display controllers already exist.

GUIBuilder

The GUIBuilder gives developers and designers a comprehensive tool to create the initial framework of the user interface. The code generated by the GUIBuilder can be modified and read back into the GUIBuilder again which allows easier modifications even at a later stage of the user interface development.

Bitmap Converter

The Bitmap Converter can convert any bitmap into standard C code or into a binary format which can be located on any media and loaded at run-time. It supports palette conversion for palette based color modes as well as high color, true color or semi transparent images as PNGs. For efficiency, bitmaps may also be saved without palette data and in compressed form.

Fonts

A variety of fonts are shipped with the software. The default set of fonts includes quite small fonts up to fairly large fonts, mono spaced and proportional fonts, special digit fonts and framed fonts. Additional fonts can easily be generated from PC fonts using the Font Converter. Monotype and TrueType fonts support is available.

Features

- ISO/ANSI C source code
- Low resource usage
- Alpha-Blending
- Anti-Aliased drawing
- Anti-Aliased fonts
- Auto double / triple buffering
- Multi-Language-Support with ASCII resource files
- Multi-Layer-Support
- Memory devices for flicker-free animation
- Free rotation and scaling
- Run-time-configurable drivers
- Start/test applications supplied
- No royalties
- Any 8/16/32-bit CPU; only an ANSI “C” compiler is required!
- Any (monochrome, grayscale or color) LCD with any controller supported(if the right driver is available)
- May work without LCD controller on smaller displays
- Any interface supported using configuration macros
- Display-size configurable
- Characters and bitmaps may be written at any point on the LCD, not just on even-numbered byte addresses

Font Converter

Font Converter is a Windows program that makes it easy to add new fonts to emWin. It can convert any installed PC font into a C file that can be compiled and linked with the application or the binary formats ".sif" and ".xbf", which can be loaded during runtime. Simply load a font which is installed on your system into the program, edit its appearance if necessary, and save it. The generated file can then be used by emWin and be shown on the display like any other standard emWin font.

Color Management

emWin features an integrated, very efficient color management system. This system allows conversion of logical colors (RGB format) into physical colors, which can be displayed at run time. As a result, your application does not need to be concerned the available colors, and displays can easily be interchanged.

Virtual Screen Support

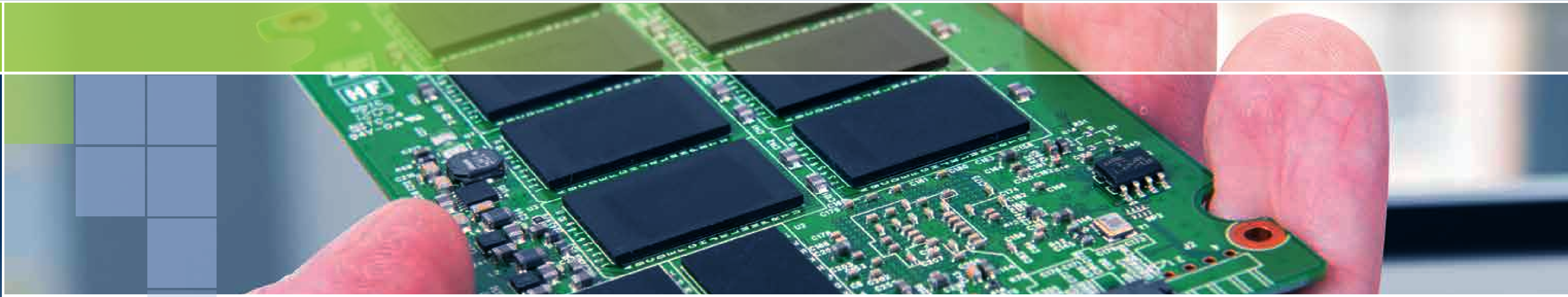
The virtual screen feature supports display areas larger than the physical size of the display. It allows switching between different screens even on slow CPUs.

Window Manager/Widgets

The window manager allows the creation of windows of arbitrary size at any point of the display. It is an optional component, which is fully integrated into the software. Child windows and the exchange of messages between windows and their children/parents are supported.
The window manager allows windows to be transparent and overlapping. Windows can freely move and resize. Additionally the window manager allows fading in and out.
The window manager performs any necessary clipping. If callback routines are used, it also manages the redrawal of invalidated areas.

Touch Screen Support

emWin supports touch, gesture and multitouch events. The window manager deals with touch messages and widgets such as button objects. It takes only one line of code to create a button or another widget, which then handles touch messages and reacts accordingly. For resistive touch screens support is available as a driver for analog touch panels, which handles the analog input (from an AD-converter), debouncing and calibration of the touch screen is included.



Programming with SEGGER Flasher

The Flasher production tools are a family of in-circuit-programmers with stand-alone programming support. Each Flasher is optimized for high programming speed and engineered for robustness. The hardware and software interfaces allow an easy integration into production environments.

Simply press a button

Using the stand-alone option, service technicians can update devices in the field by simply pressing a button after the device has been set up and loaded with the necessary programming information. With the battery powered Flasher Portable, service technicians do not even need to carry an external power supply to program their targets.

The programming information is stored together with CRC data which has been generated on the host PC. This CRC data is used to verify the integrity of the data stored inside the Flasher and to verify the programming success.

Interfaces for every need

In addition to the features like stand-alone operation and RS232 interface, current SEGGER Flasher models support the host interfaces Ethernet and USB to control the Flasher and transfer the programming data into the Flasher. The programmers can also be used in MSD mode to transfer the programming data. Additionally they have an internal web-server.

Proven programming solution

Serial number programming and patch programming to add additional variable information are supported. The internal memory offers more than enough memory for programming data and configuration files. The programmers operate at 5V via USB interface. The Flasher family of stand-alone In-Circuit- Programmers has been offered for many years now and is a proven programming solution for numerous customers.

Multiple Images

Multiple firmware images and configurations can be stored in the internal memory of the Flasher. Which image or configuration is selected, can be selected via the remote control interface using UART terminal or Telnet via Ethernet.

Authorized Flashing

Authorized Flashing allows the customer to limit the number of flash programming cycles and to protect the Flasher against non-authorized access to the customers software, thus preventing the production of unwanted quantities by third parties. Authorized Flashing is available for ARM, RX and PowerPC targets.

Flasher 5 Pro

Flasher 5 Pro is the successor of the Flasher 5. It has 64MB of flash memory to store programming data and configuration and has a significantly higher performance. The programmer supports the following devices: Renesas R8C/M16C/R32C/M32C.

Features

- Stand-alone programming
- Verification of data integrity
- Remote Control via RS232 & Ethernet
- High programming speed
- Easy integration into production environments
- USB interface
- Ethernet interface
- USB-powered
- MSD mode
- Internal Web-Server
- Serial number programming
- Patch programming
- Authorized Flashing

Flasher ST7

The Flasher ST7 supports the RS232 interface only and offers 512kB of flash memory to store programming data and configuration. Option byte programming is supported. The programmer supports the following devices: STMicroelectronics ST7.

Flasher ARM

Flasher ARM uses JTAG or SWD as target interface. Flasher ARM supports programming of the internal flash for many MCUs with different ARM cores like: ARM7/9, Cortex-M0/M0+/M1/M3/M4/M7.
Additionally Flasher ARM allows programming of external flash memories connected to any ARM-core. CFI-compliant flash memories are recognized automatically and can be programmed directly. Other external memories (like NAND, SPI-NOR, …) require an additional RAM code, which is typically available for popular evaluation boards.

Flasher STM8

The Flasher STM8 connects via SWIM interface with the target. The target is optically isolated from the host side. The configuration and operation can be handled with the Flasher software for STM8. Option byte programming is supported. The Flasher STM8 supports the following devices:
STMicroelectronics STM8.

Flasher RX

The Flasher RX uses the fastest flash programming algorithm currently available for the Renesas RX. The supported device families are: Renesas RX600/RX200/RX100.

Flasher PPC

The Flasher PPC supports the following devices: Freescale Pictus/STMicroelectronics Bolero.

Flasher Pro

The top model Flasher PRO combines the features and programming targets of Flasher ARM, Flasher PPC and Flasher RX.

Flasher Portable

The Flasher Portable is a battery powered programmer for targets based on ARM, PPC or RX architectures. It has a simple interface to allow the selection of different firmware settings and images stored inside the Flasher.

About SEGGER

SEGGER Microcontroller is a full-range supplier of hardware and software development tools as well as middleware for embedded systems – in brief: **The Embedded Experts**.

The company offers support throughout the whole development process with affordable, high quality, flexible and easy-to-use tools and components: Starting with its own IDE **SEGGER Embedded Studio**, debugging with the industry-leading **J-Link**, tying in Microcontroller peripherals via its Middleware, and finally implementing the software on the target through its own Flasher programmers.

SEGGER relies on closed-loop development: Its products have to prove their worth in daily use by the developers. That's why it simply works.

Software products include, among others: **embOS** (RTOS), **emWin** (GUI), **emFile** (File System), **emUSB** (USB host and device stack) and **embOS/IP** (TCP/IP stack). With **emSSL** and **emSecure**, a unique software to generate and verify digital signatures, **SEGGER** is offering solutions for secure communication as well as data and product security, meeting the needs of the rapidly evolving IoT.

The highly integrated, cost-effective tools comprise the **Flasher** flash programmer and the **J-Link/J-Trace** debug probe.

SEGGER was founded in 1997, is privately held, and is growing steadily. Based in Hilden with distributors in all continents and a local office in Massachusetts, **SEGGER** offers its full product range worldwide.

SEGGER Microcontroller GmbH & Co. KG
In den Weiden 11
40721 Hilden
Germany

www.segger.com
Tel: +49-2103-2878-0
E-mail: info@segger.com

SEGGER Microcontroller Systems LLC
106 Front Street
Winchendon, MA 01475
United States of America

www.segger-us.com
Tel: +1617-874-1255
E-mail: info@segger.com