mbLay Architecture and Interface

This white paper gives an introduction into the embenatics foundation layer mbLay. It describes the basic design principles of the architecture and presents the interface that is provided to applications running on top of mbLay.

mbLay Architecture

The embenatics foundation layer mbLay consists of two sub-layers, the OS abstraction layer (OSAL) and the embedded foundation layer (EFL). With this approach, we separate OS dependent functionality from the mbLay core services. Because of this layered structure, we are able to port mbLay to a new operating system without modification of the core functionality in EFL. This reduces the development effort as well as the risk of a regression.

OS Abstraction Layer

The OS abstraction layer (OSAL) is designed to abstract from the underlying operating system. This layer implements basic services that are needed to run the upper embedded foundation layer on top of the supported operating system. In addition, it provides the embedded foundation layer with OS dependent information about the status of the system and its resources. The upper OSAL interface itself is operating system independent. Currently, embenatics provides OS abstraction layers for Windows™, Linux™, QNX™, FreeRTOS™, ThreadX™, VxWorks™ and Nucleus™. Adaptations to other operating system are available on request.

Embedded Foundation Layer

The embedded foundation layer (EFL) offers additional features that extend the pure OS abstraction. These core features are the basis for many of the embenatics key functions. The entire RPC based communication infrastructure uses the EFL services to establish the connections between the communication threads. EFL is managing the creation of the system resources according to the system resource description and boots the application without any additional action required by the developer. The central resource description ensures that the implementation of the application on top of mbLay does not need to be adapted if the resource configuration has changed. mbLay also provides additional system services that support the user in designing, testing and debugging the application. Therefore, mbLay includes a driver adaptation layer that supports the efficient use of communication drivers for the test and diagnosis interface.
Finally, and essential for the user, EFL provides an interface for system service access to applications running on top of mbLay. The EFL interface can be seen as a virtual operating system interface that

is identical for all operating systems below mbLay. This generic approach ensures that the implementation of an application based on mbLay does not need to be modified if the operating system is exchanged.

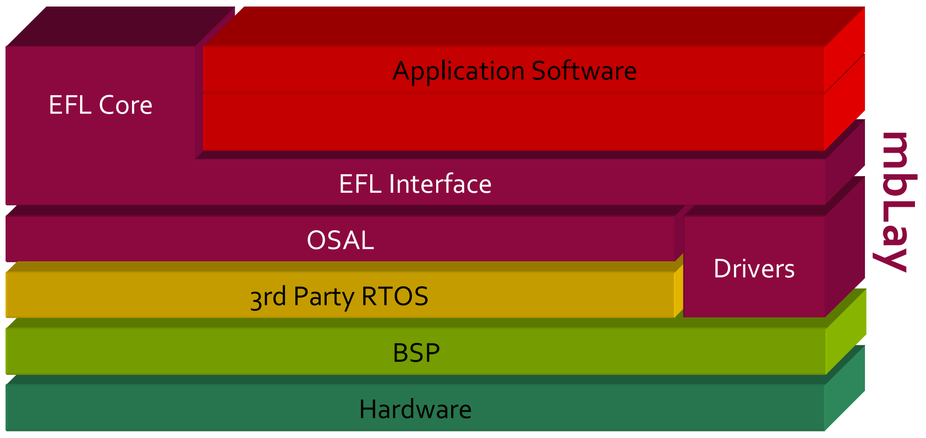

Figure 1 mbLay within the Basic Software Architecture of an Embedded System

mbLay and its built-in communication infrastructure, together with the approach to centrally describe a software system, supports the distribution of applications across multiple CPUs of multi-core systems. Threads created in the mbLay environment that use mbLay system interfaces can be moved between applications in multi-core systems. For further information on embenatics multi-core support please see another document of this series of white papers [Designing Multi-Core Applications].

mbLay Interface

The mbLay interface is used by any mbLay based application to request access to system resources. mbLay-based applications will always use exactly the same interface regardless of the used hardware platform or underlying operating system. Applications that use the mbLay interface to access operating system services can therefore easily be ported to different environments.

The EFL interface provides the following functional blocks:

- Thread management
- Memory management, virtual memory pools
- Synchronization (semaphores, timers, event flags, critical sections)
- Communication exchange (RPC, messages queues, event flags)
- Handlers
- Logging
- Statistics (history, profiling)

Most of these functional blocks are self-explanatory, some require a more in-depth look which follows below.

Virtual Memory Pools

Virtual memory pools provide the designer with a way to limit the memory consumption of a thread, group of threads or application. They are specified by their size and two thresholds. The user can set an upper and a lower allocation threshold and register a callback function that is called by EFL if the allocation level exceeds the upper allocation threshold or falls below the lower allocation threshold. Virtual memory pools help to segment memory management across the threads or thread pools of an application to avoid excessive memory allocation for a single process.

Handlers

The handlers in embenatics mbLay are functions that are periodically called by mbLay. A dedicated API function is provided that can be used by the designer to register a handler function to be called at regular specific intervals. This feature is a convenient way to cyclically call a function without the need for creating an OS thread or OS timer.

Logging

The logging feature of mbLay is responsible tracking the entire inter-process communication. Additionally, mbLay provides the user with a set of macros for code instrumentation to generate different classes of logging events that are sent to the embenatics logging tool mbLog via the test and logging interface. For further information on logging itself and the logging tool mbLog see the other two documents of this series of white papers [mbLay Logging] and [mbLog Logging and Diagnosis Tool]. Filters can be applied to the logging classes in order to adjust the logging interface load in case of a low bandwidth target connection.

Statistics

mbLay sends tracked system resource information via the test and logging interface. The status of all threads can be monitored and thread properties like the current and maximum stack consumption can be displayed. The CPU load can be displayed on a per thread basis. The current and peak load of the system memory pools, as well as the memory pool load over the time, is monitored and displayed in mbLog in order to support the developer during memory optimization tasks. Finally, every memory pool access can be logged, which is of great advantage when debugging out-of-memory conditions and memory leaks.

Conclusion

The embenatics foundation layer mbLay supports you in designing operating system independent and reusable software. EFL provides a virtual system interface that is the same for every underlying operating system. This ensures that an mbLay based application does not need to be adapted when exchanging the OS. The built-in test and diagnostic functionalities help to reduce the development time and increase the application software quality. The embenatics design methodology for keeping the system configuration in a single description document, together with automatic resource creation and start-up according to the system description document, keeps the effort for changes in the resource configuration at a minimum.

About Us

embenatics is a new company that entered the market in 2010. Our focus is on embedded software development; as such we offer a software foundation layer and tool suite that supports your development team in designing embedded software in an efficient, portable and maintainable way. Based on our wide and varied experience in embedded systems design and development, we know that future product requirements are hard to predict. Our goal is, therefore, to provide you with our technology to make the design of your products as flexible and adaptable as possible. Our approach allows your company to concentrate on the core competencies that differentiate your valuable product from those of your competitors.

Before embenatics was founded, we worked with well-known international companies over two decades and gained valuable experience in the embedded software business. While working as software developers and architects, we encountered the various challenges of the embedded software development life cycle. This wide range of experiences is the backbone of the software foundation products that are offered by embenatics.

Our business philosophy is to establish a close and trustful relationship with our customers in order to successfully promote and support projects over a long time period. For further information please contact

Joachim Pilz
Beerenstraße 29
14163 Berlin

info@embenatics.com
www.embenatics.com

Phone +49 30 26 34 75 28
Mobile +49 176 96 98 46 07