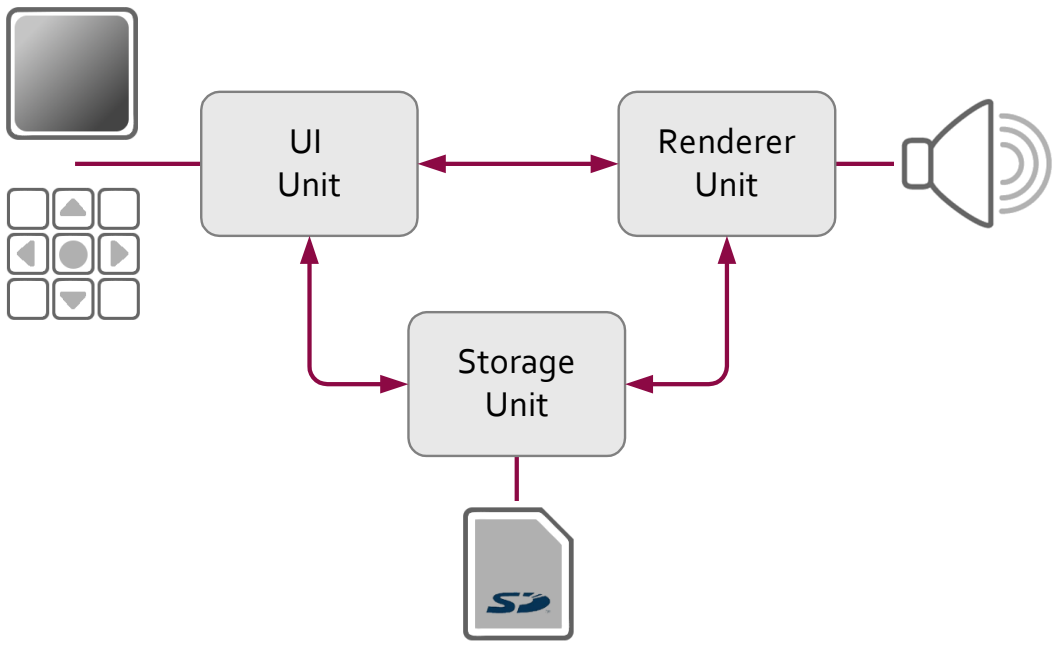
Designing Multi-Core Applications

This white paper offers an introduction into the design of multi-core applications using the embenatics tool suite and foundation layer based on a simple MP3 player example.

Introduction to the Sample Application

In this series of white papers a simple MP3 player is used as a sample application to show the various features and working steps of the embenatics tool suite. The figure below shows the building blocks for the MP3 player example. A short description of each block will facilitate the understanding of the basic functionality of the player. Each white paper will focus on different subjects of the sample application design to highlight the specific topic of the paper.



Block Diagram of the MP3 Sample Application

The UI Unit handles the user interaction, which comprises user commands as well as displaying status information, like which track is playing, the list of selected titles, etc. Access to the Storage Unit is used to provide all information about the available music tracks. The UI Unit interfaces with the Renderer Unit to pass on user commands, as well as to display the current status of the player.

The Storage Unit keeps track of the available music titles and provides access to the stored MP3 files. It interfaces with the UI Unit to provide track information. The interface with the Renderer Unit provides access to the coded music data that should be replayed.

The Renderer Unit is responsible for handling the music track to be played, managing a play list of titles, displaying information on the state of the player and converting the coded MP3 data into audible music. It receives track information and control commands via the UI Unit interface. Access to the MP3 data is obtained by interfacing with the Storage Unit.

In this white paper the MP3 player application is implemented by using five threads. The DISPLAY thread performs the graphic commands for drawing icons and text, and utilizes the LCD driver for visualization purposes. The KEYPAD thread receives information about key presses from the keypad driver. It forwards the key presses to the UI thread that controls the entire MP3 player. Additionally, there is the STORAGE thread managing a database containing the information about tracks, albums and artists stored on the memory card. A RENDERER thread is responsible for accessing the audio hardware and informing the UI about the audio renderer's state.

Multi-Core Support

Today multi-core hardware platforms find their way into more and more embedded applications. The advantage is not only the increased performance but also the ability to design applications split up into different subsystems. Unfortunately, the software support to fully exploit the increased hardware capabilities is still inadequate.

The embenatics foundation layer and tool chain assists you in designing your application in an asymmetric multiprocessing (AMP) environment on homogeneous as well as on heterogeneous systems. Due to the characteristics of the communication model, the software components are able to communicate with each other independently of their own and their peer's location. For further information on the communication model, see another document of this series of white papers [mbLay Inter-Process Communication Model]. According to the embenatics design methodology, there is a dedicated system description document for each subsystem and threads can be moved among subsystems by a simple a drag & drop operation in the system resource description editor mbEdit. For further information on system descriptions please see another document of this series of white papers [embenatics System Description]. The thread code itself does not need to be changed when moving a thread to a different core. [1] The communication model will handle the modified communication channels. Different alignment and endianness in the multi-core environment will be handled and converted automatically.

The possibility of moving threads among multiple processors can also be utilized if you consider establishing a distributed development environment where you develop, run and test SW parts on a PC instead of the target system. This will give you the full bandwidth of debugging capabilities on the PC before you move the validated software part back to the target system.

[1] Additional actions are required if the moved thread uses resources e.g. semaphores that are shared among threads

Distributed MP3 Player

Now we will distribute the MP3 player shown at the beginning of this document to run on two cores. The UI Unit will run on the embedded system whereas the renderer and the storage units are located on a Windows based PC.

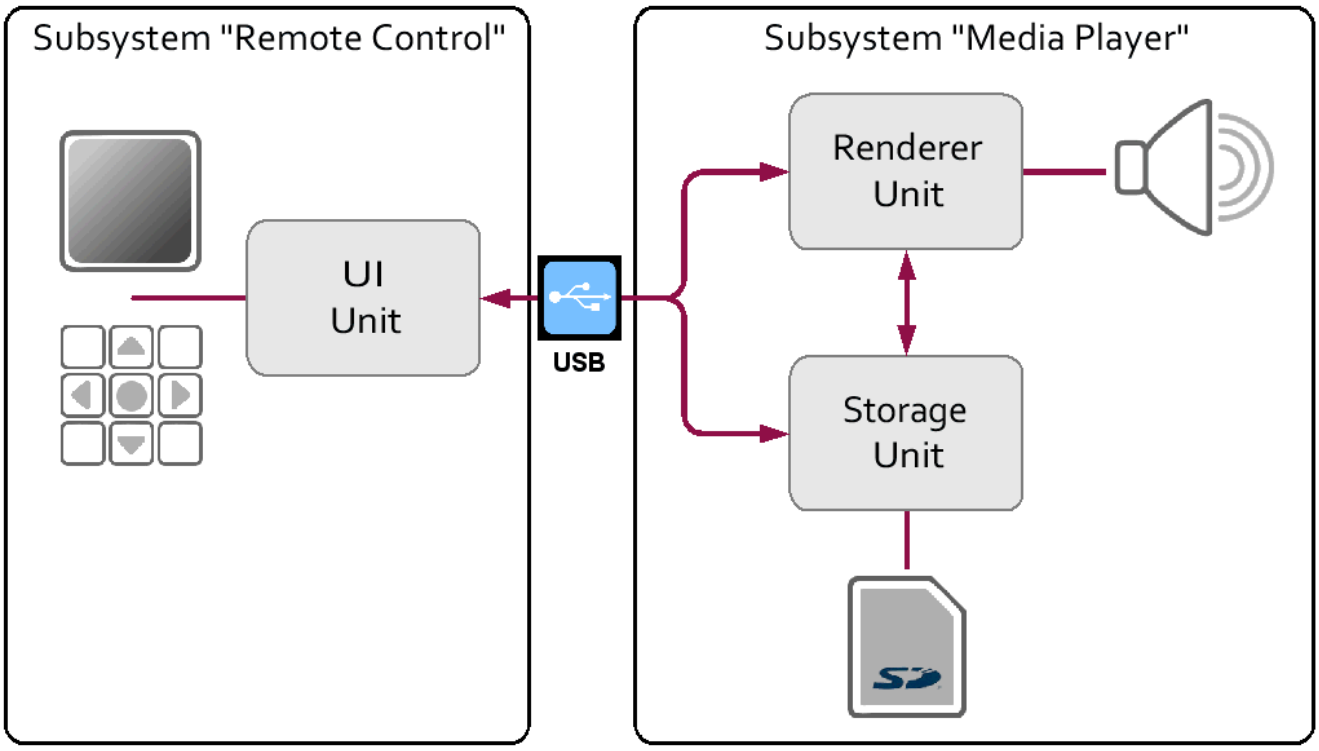


Figure 1 MP3 Player Example in Multi-Core Configuration

The following picture shows the resource description file of the MP3 player example that was already introduced in the [embenatics System Description] where the player is entirely running on the embedded system (in this example Raisonance Primer2 with FreeRTOS™).

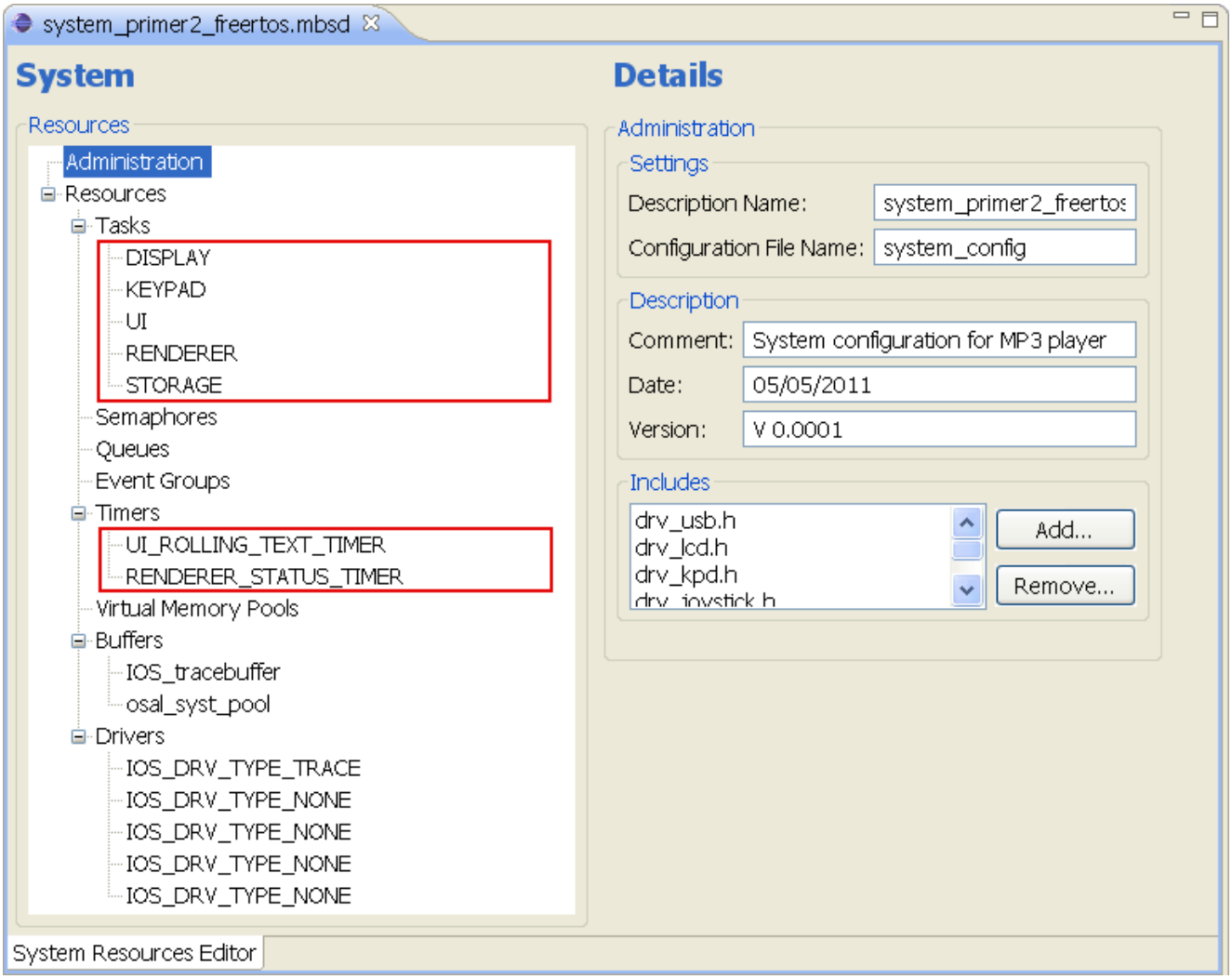


Figure 2 MP3 Player System Description in Single-Core Configuration

In order to demonstrate the multi-core support of the embenatics foundation layer and tool suite, we split the MP3 player and run DISPLAY, KEYPAD and UI threads on an embedded system (primer2) and the RENDERER and STORAGE threads on a Windows based PC.

To split an application, we simply take a template for a system description document, open it in the system description editor and fill in the administration data form. Next we drag the RENDERER and STORAGE threads and their timers to the newly created system description file for the PC based part of the player.

The following picture shows the two system configuration files for the distributed MP3 player example that is created by a simple drag & drop operation of resources from the primer2 stand-alone configuration into a configuration for the Windows based subsystem.

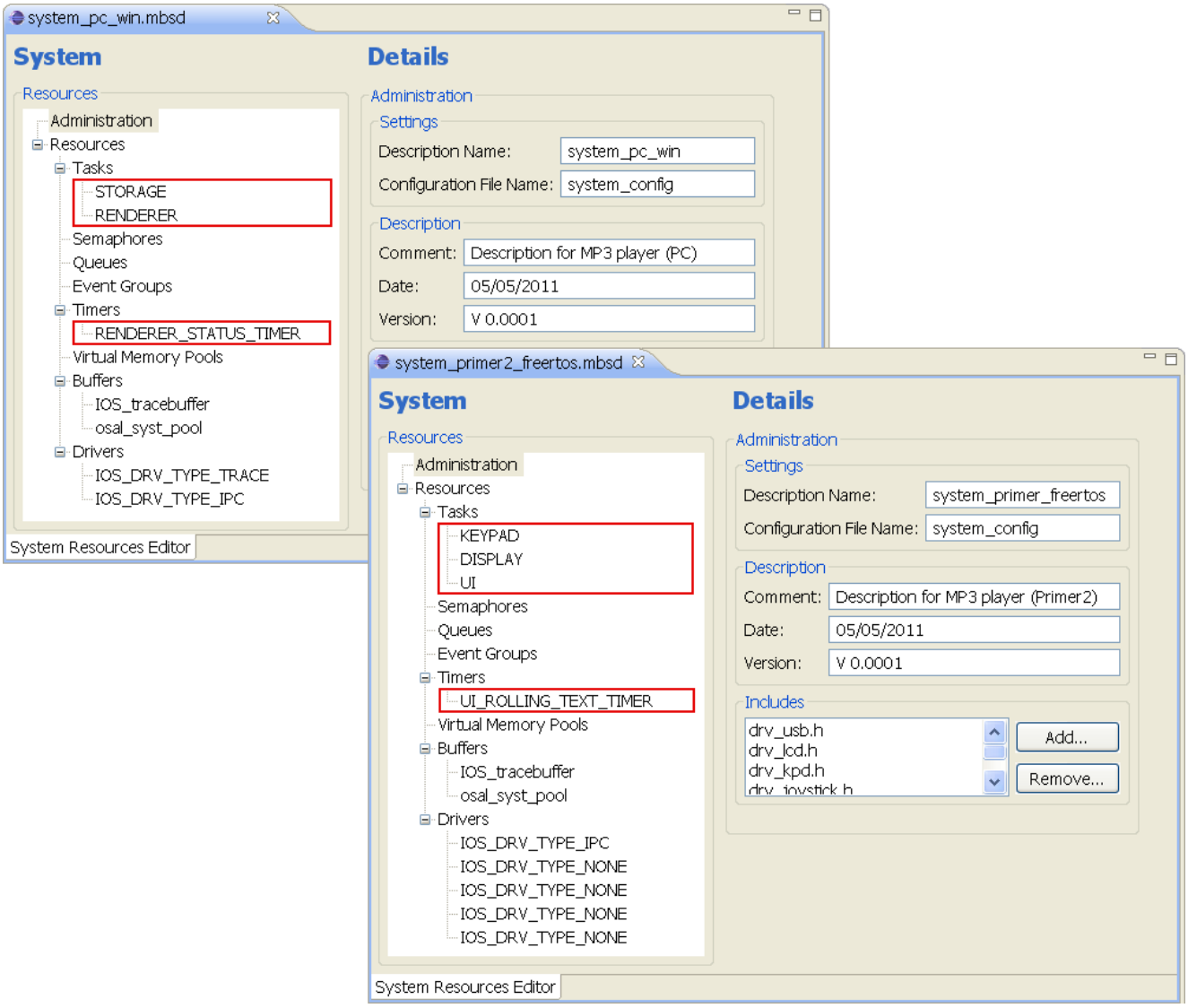


Figure 3 MP3 Player System Descriptions in Multi-Core Configuration

In order to enable the multi-core communication capabilities, a new driver for the inter-processor communication is required in both subsystems of the distributed MP3 player. These so-called IPC drivers are added to both system descriptions and need to be configured appropriately. The following picture shows the driver configuration forms of the two system descriptions of the MP3 player in a multi-core configuration.

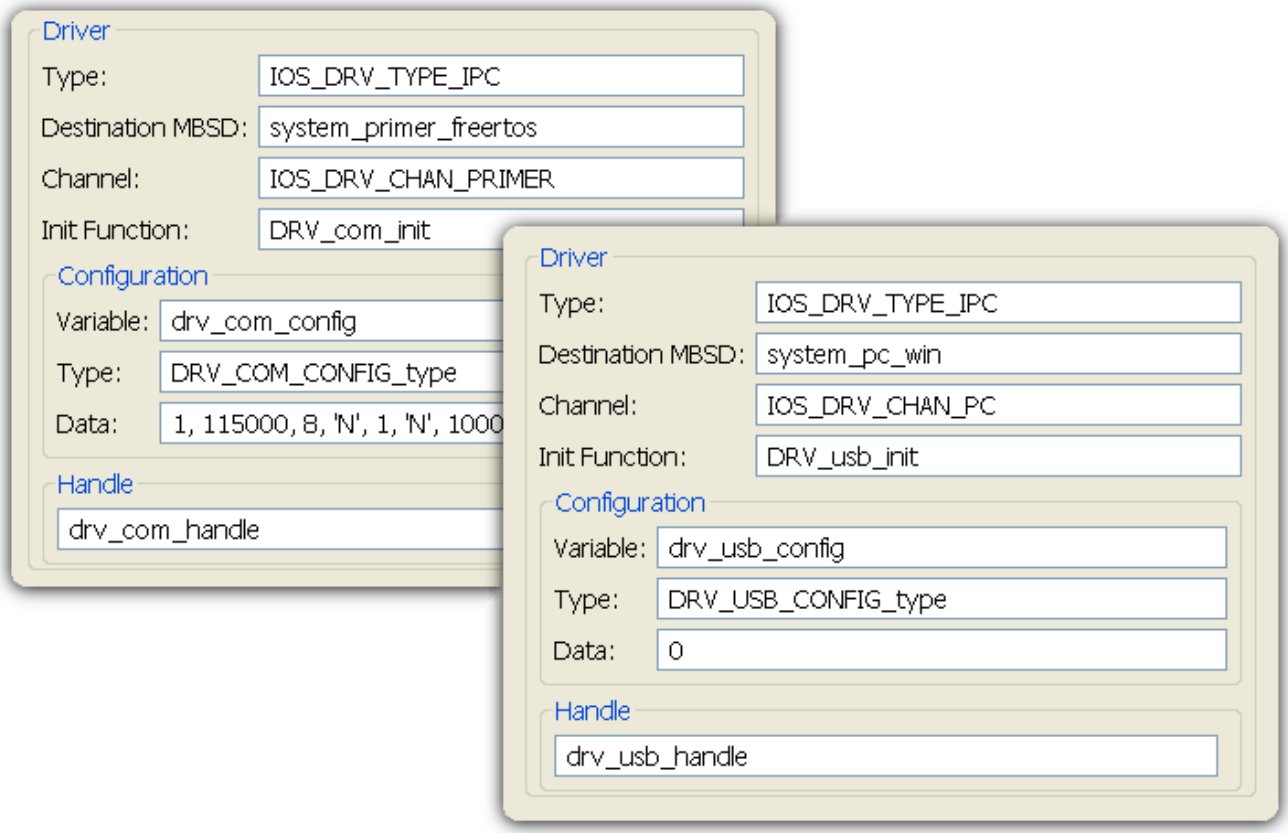


Figure 4 MP3 Player IPC Driver Descriptions in Multi-Core Configuration

In this example, both subsystems of the MP3 player are connected via a USB interface. The embedded part of the MP3 player is using a USB driver to access the PC. In the PC-based part of the player application, a simple COM driver is used to access the USB via a virtual COM port. Apart from the driver configuration itself, it is important to connect the communication infrastructures of both systems. The IPC drivers in both subsystems need to know which subsystem they are connected to; therefore, the name of the connected system description needs to be entered into the destination MBSD (embenatics system description) field.

Of course the embenatics foundation layer allows to connect the subsystems of a distributed application via various interfaces like Ethernet, CAN, shared memory and others.

Conclusion

The embenatics foundation layer and tool suite supports the distribution of an embedded application into subsystems running on different cores. Due to the methodology of keeping all system resources in one central system description document, it is very convenient to split a single-core application into a multi-core application. This design methodology, together with the embenatics inter-process communication model, ensures that only the system description files need to be modified, whereas the application's source code itself remains unchanged.

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