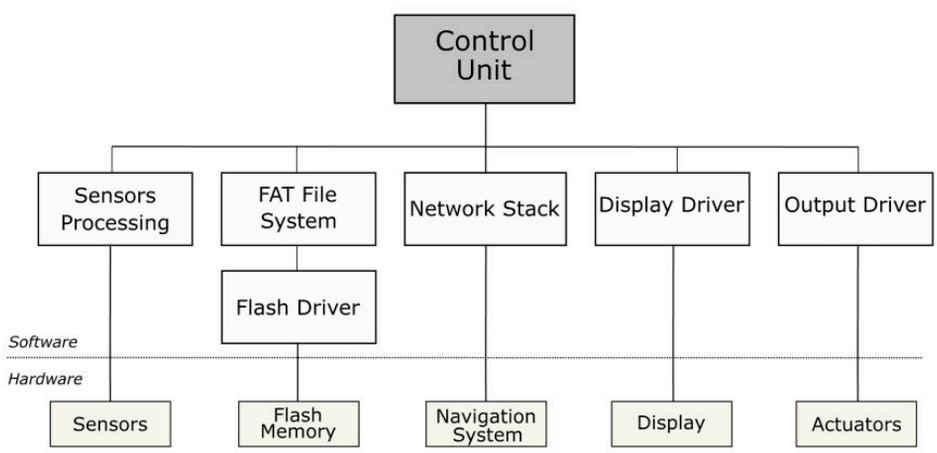
Separation of Safety Critical and Non-Safety Critical Tasks in Medical Embedded Systems

Efficient Separation

Embedded systems can be found in many medical applications such as MRI scanners, defibrillators and robotic assisted surgery. Apart from the main functionality, medical devices are more and more required to offer a likeable and easy-going experience by supporting feature-rich graphical interfaces, data storage and networking communications. For this reason, today's device designers face the challenge of having to develop a system that provides both safety and feature-rich functionalities. From a safety perspective, we can identify safety critical functionalities, which carry out actions that can injure or even kill a person in case of failure, and non-safety critical functionalities, which do not represent a life-threat in case of failure. The software system, therefore, consists of numerous software components having different safety levels.

Modern medical embedded systems usually have a similar architecture and share some or all the elements depicted in Figure 1. From a safety point of view, each system element can suffer from failures (random or systematic). Different elements, however, may have a different criticality level. In order to avoid harm to people or the environment, safety critical software is required to ensure the correct operation of safety critical elements and their functional isolation from non-safety critical components.



Pictures: embeX GmbH

Figure 1: Example of a Typical Medical Device and its Interaction with the Environment.

Medical software has to be partitioned

In the past, allocating functionalities to physically separated hardware components has been presented as a valid solution to deal with the requirements for safety critical systems. However, the complexity of current devices, the hardware area restrictions as well as the features of modern processing platforms (e.g. symmetric multiprocessing and multicore processors) turn physical separation into an unrealistic solution. It is, therefore, desirable that safety critical and non-safety critical software components can safely coexist on the same platform. This goal can be achieved by partitioning the software on the embedded system in a way that ensures that software from lower safety levels cannot interfere with software on higher safety levels. Resulting from software partitioning, the safety related components are kept small and concise as well as isolated from non-safety critical components (e.g. third-party components that are used for realizing non-safety critical functions).

Virtualization as a separation technique

Recently, embedded systems have adopted hypervisors as virtualization technology in order to create software virtual machines (or partitions), where each partition is isolated and has a set of virtual resources mapped to the available physical resources. This article presents a methodology that guarantees the separation of functionalities of different criticality levels by means of a hypervisor. Hypervisors enable functional separation, fault containment and the execution of several independent environments, thus allowing the coexistence of various execution environments on the same platform (e.g. real-time OS for safety critical tasks, general-purpose OS for non-safety critical tasks).

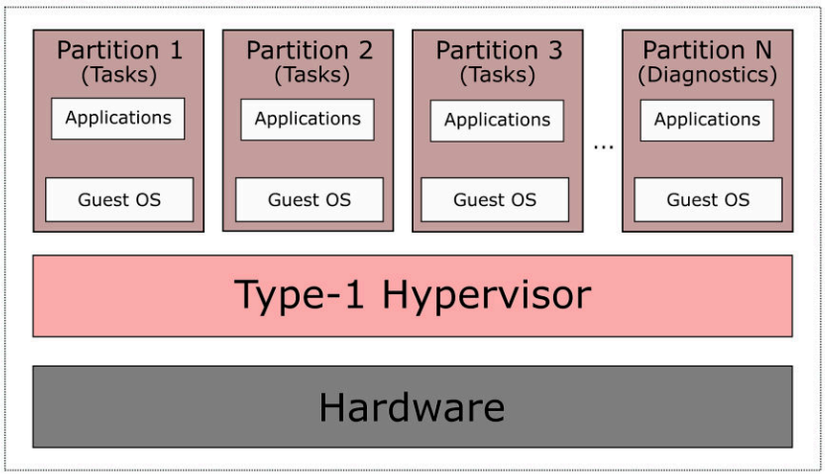


Figure 2: Type-1 Hypervisor with Paravirtualization

The strategy presented in this article uses a Type-1 hypervisor with paravirtualization in order to separate functionalities of different criticality levels and achieve both spatial and temporal isolation. Type-1 hypervisors, also called bare-metal hypervisors, run directly on the native hardware (see Figure 2) and, in combination with paravirtualization, are highly suitable for embedded systems due to their simplicity, high speed and reduced size. The separation provided by a hypervisor can be complemented with hardware mechanisms such as Memory Management Units (MMU) to achieve stronger spatial separation and a fixed cyclic scheduler for strong temporal isolation.

Develop a safety chain to deal with failure

Software partitioning provides separation among software components but failures affecting single partitions or software components such as OSes and the hypervisor are still to be taken care of. A common feature among medical devices is that, in case of failure, they are brought to a safe state, which is a state free from hazardous situations. A safe state can be reached by means of a set of carefully designed safety functions, which intend to achieve or maintain a state free from hazardous situations for an equipment under control.

In the described architecture, the system is decomposed into several Task partitions, which carry out various functions, including safety functions, and one Diagnostics partition, which has system privileges and checks the status of all partitions and of the hypervisor. In order to deal with failure situations, each Task partition can execute several safety functions. These safety functions control and mitigate failure situations such as control flow errors, OS errors and hardware faults. If the Task partition is executed successfully, it sends a confirmation message to the Diagnostics partition, which will then carry out further checking. On the other hand, whenever a failure is detected (e.g. missing confirmation of any of the Task partitions), the system is brought to a safe state by the Diagnostics partition.

In this design, all safety functions rely on the proper behaviour of the hypervisor, as it is managing the system execution. This creates a strong dependency on the hypervisor and therefore, there must exist a safety chain to bring the system to a safe state in case of a hypervisor failure. In our proposal (see Figure 3), the diagnostics partition reads out diagnostic information about the hypervisor and refreshes an external hardware watchdog when the hypervisor is functioning properly. On the other hand, in case of hypervisor failure or a failure of the Diagnostics partition, the safety chain is broken, as the watchdog will stop being refreshed. When the watchdog's timer expires, the system will immediately be brought to a safe state (e.g. by de-energizing a set of safety relays), thus avoiding an undefined system behaviour and a resulting hazardous situation.

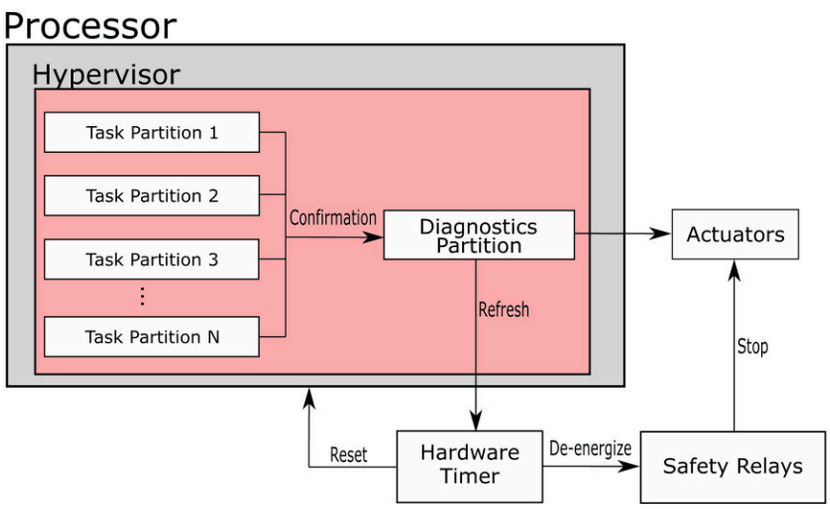


Figure 3: Safety Chain to bring the System to a Safe State in case of Hypervisor Failure

This methodology has been successfully applied and offered positive results in terms of its applicability in the medical field. Furthermore, this is a systematic approach for the separation of tasks and the integration of safety functions in a medical embedded system by use of a hypervisor as virtualization technology. This approach can also be complemented with medical device-related safety norms and with aspects of the functional safety norm IEC 61508.

CONTACT

embeX GmbH
Heinrich-von-Stephan-Str. 23
D-79100 Freiburg im Breisgau
Tel. +49 761 479 79 90
www.embeX.de