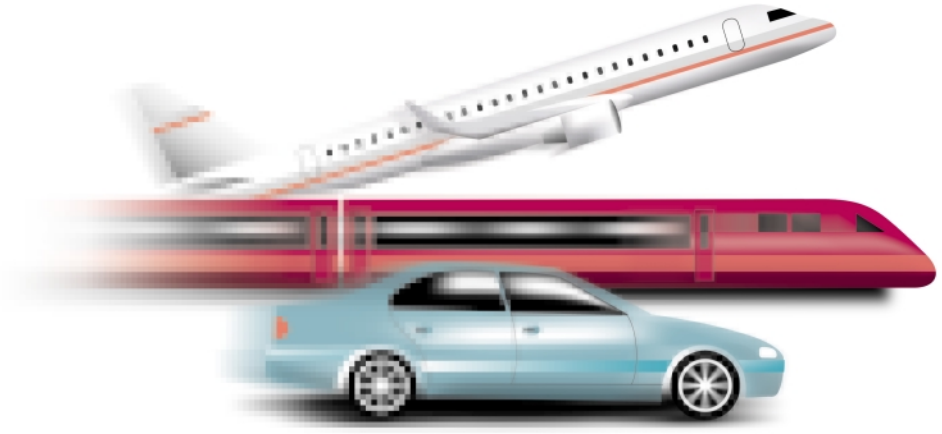
SCIOPTA

High Performance Real-Time Operating Systems



SCIOPTA

SCIOPTA Real-Time Kernel

High Real-Time Performance

The SCIOPTA architecture is specifically designed to provide excellent real-time performance and small size. Internal data structures, memory management, interprocess communication and time management are highly optimized.

SCIOPTA kernels are written in assembler and specifically tuned for the supported CPUs.

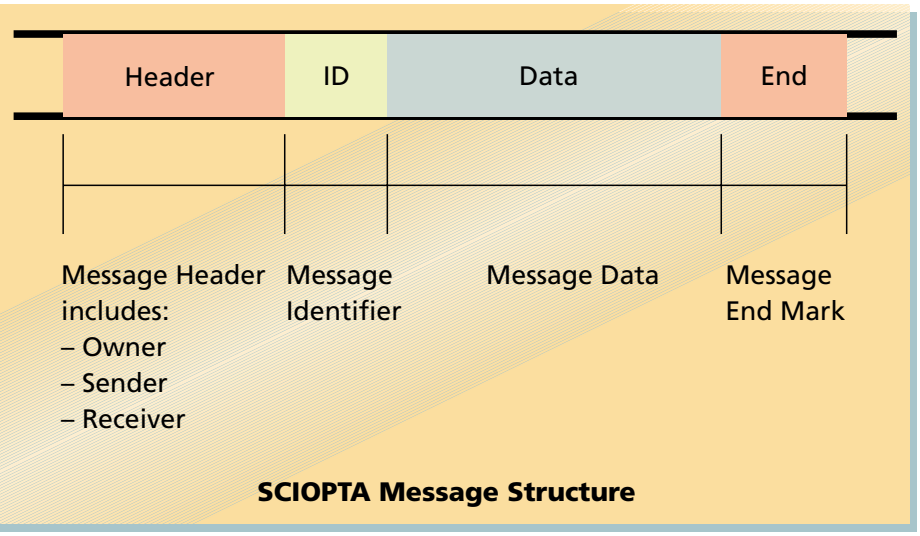
Message-Based Architecture

SCIOPTA is designed on a message-based architecture allowing direct message passing between processes. Messages are mainly used for interprocess communication and synchronization.

SCIOPTA messages are stored and maintained in memory pools. The kernel memory pool manager is designed for high performance and memory fragmentation is avoided.

Pre-emptive Real-Time Kernel

SCIOPTA is a pre-emptive real-time kernel. Interrupts can be serviced any time.



SCIOPTA Message Structure

Reduced Time-to-Market

A powerful set of system calls managing the message passing and the resources of SCIOPTA allows you to shorten the development time and thus to reduce the time-to-market for your products.

SCIOPTA is a message-based real-time operating system. Standardized processes and interprocess communication result in clear system designs and are easy to write, to read and to maintain.

As processes are communicating with well defined messages and processes can be grouped into modules, SCIOPTA systems are very well suited for teamwork in big projects.

Easy to Debug

A SCIOPTA system can be easily debugged by tracing messages until a breaking or blocking situation is reached. The message trace allows analysis the message sequence preceding a possible faulty system state.

A SCIOPTA message contains not only the message data but also administrative information such as the owner, sender and addressee process. The debugger can, therefore, quickly find lost messages or messages which have been sent to wrong processes, by analysing the message pools.

Error Handling

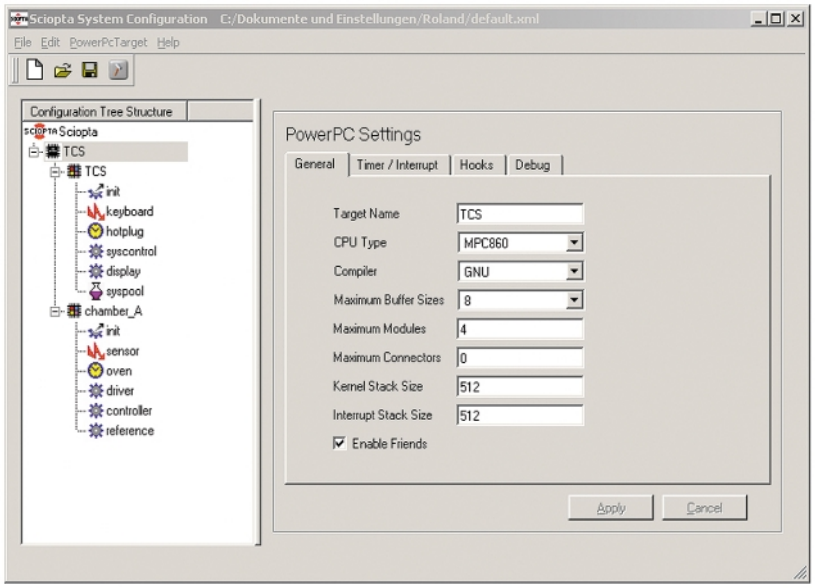
SCIOPTA features centralized error handling by using SCIOPTA error hooks. Each time SCIOPTA detects an error the error hook will be called. This guarantees consistent error handling covering the whole system.

Problems common in traditional operating systems when using individual error handling by different team members spread over the whole application code can be avoided in SCIOPTA.

Easy Configuration

All static system settings such as static modules, processes and pools can easily be configured with the universal SCIOPTA graphical configuration tool. You can configure a project, including different target CPUs, within the same configuration session.

The project settings are stored in an XML file. After the project configuration settings are satisfactory the configuration tool will generate all needed files be included in the software build process.



Fully Dynamic

All system components such as:

- Modules
- Interrupt processes
- Timer processes
- Prioritized processes
- Message pools

are fully dynamic in SCIOPTA.
They can be created and killed during run time.

Supervision

SCIOPTA has built-in support for process supervision.

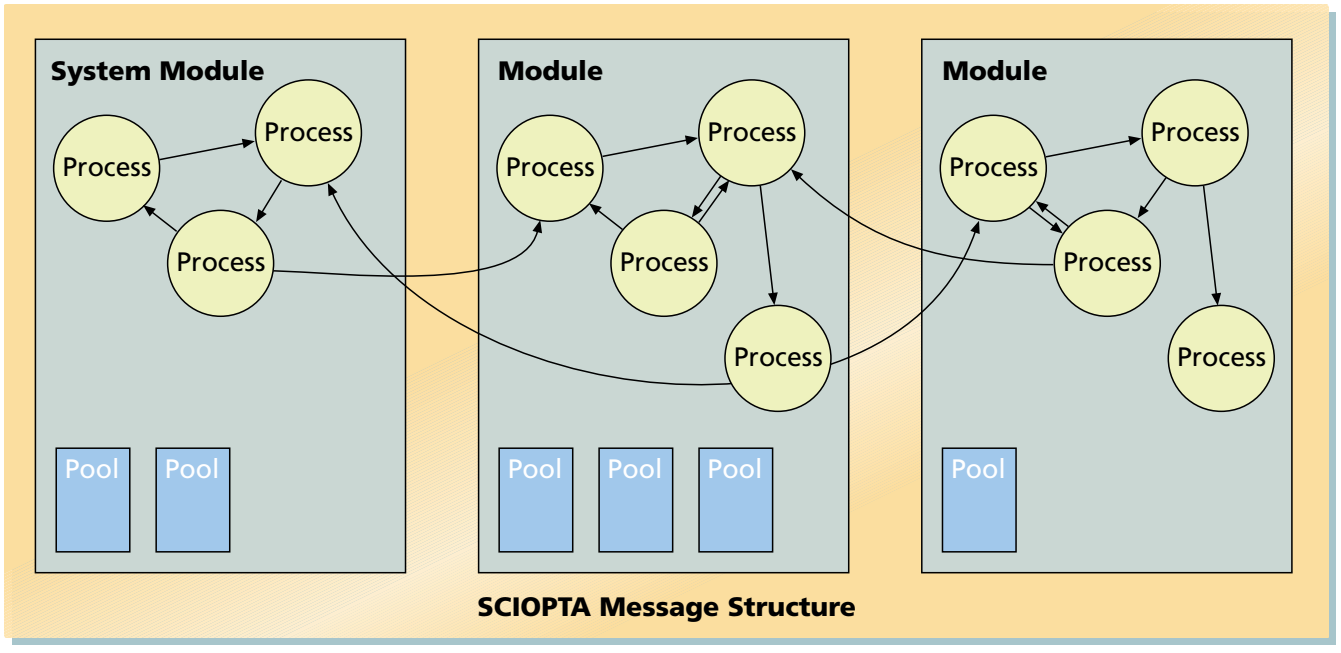
A process can register another process for supervision. If the supervised process dies, the kernel will send a message to the supervisor.

SCIOPTA Modules

Processes can be grouped in SCIOPTA modules, which allows you to design a very modular system. Modules can be static or created and killed during run time.

SCIOPTA also supports a friend concept. Modules can be friend to other modules. Friend modules have privileged access to either modules system resources.

SCIOPTA modules can be used to encapsulate whole system blocks (such as a communication stack) and protect them from other modules in the system.



SCIOPTA for Distributed Systems

SCIOPTA CONNECTOR

The CONNECTOR is a SCIOPTA communication process which connects SCIOPTA systems on different CPUs in a distributed environment.

The CONNECTOR is the essential part in a distributed multi-CPU system running SCIOPTA. Each node is controlled by a CONNECTOR process which has knowledge of distributed processes taking part in the communication system.

CONNECTORS are responsible for redirecting messages, of knowing the location of distributed processes and hiding the distributed system from the user programmer.

SCIOPTA CONNECTORS have an open interface and do not depend on any network topology. Therefore any network stacks (e.g. TCP/IP, CAN) can be used.

A SCIOPTA distributed system is not limited to one type of CPU. Any CPU where a SCIOPTA kernel and a CONNECTOR process is available can be included.

Transparent Communication

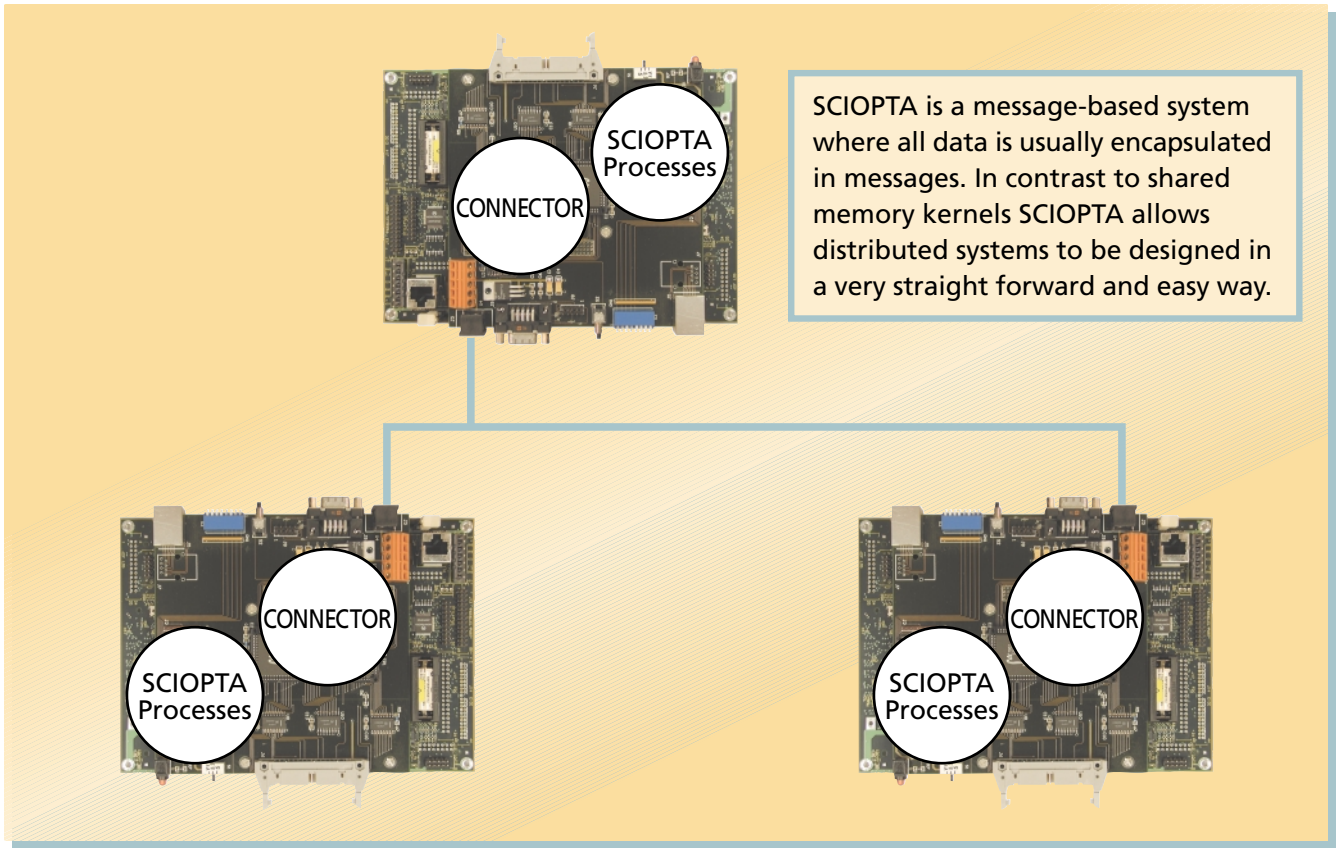
The clean message-based interface in SCIOPTA allows distributed systems to be designed as quickly and easily as a single CPU system. Message passing beyond processor boundaries uses the same system calls for transmitting and receiving although the processes reside on different CPUs.

The support provided for supervision in SCIOPTA can also be used in a distributed system. CONNECTOR processes, board support processes and the network system can be supervised by the kernel. Remotely connected processes will be informed by the kernel if resources in the communication path are no longer available.

In a SCIOPTA distributed system there is no need to have a master node. A node or CPU that disappears will not affect the whole distributed system.

Fault-Tolerant System

SCIOPTA supervision, transparent communication and the module concept allow you to built fault-tolerant distributed systems.



SCIOPTA is a message-based system where all data is usually encapsulated in messages. In contrast to shared memory kernels SCIOPTA allows distributed systems to be designed in a very straight forward and easy way.

SCIOPTA IPS Internet Protocols

Designed for Embedded Systems

SCIOPTA IPS has been specially designed to meet the requirements of modern internet protocol network applications in embedded systems.

This gives IPS the advantage over traditional internet stacks that were ported to embedded systems, of having a higher performance and a lower memory footprint.

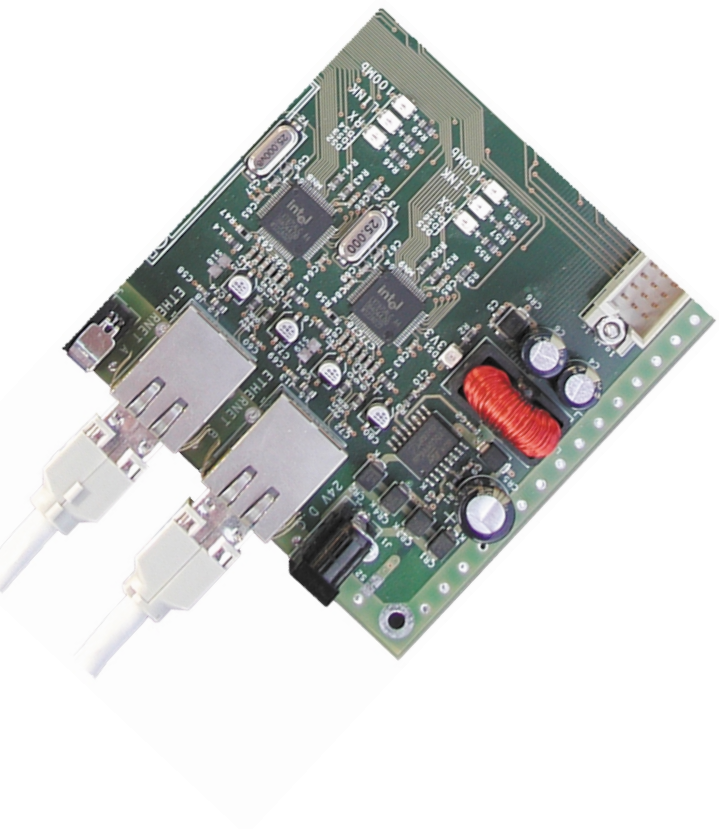
Flexible Interface

SCIOPTA IPS provides a standard BSD socket interface. This easily allows the porting of off-the-shelf internet protocols.

For designers who want to take full advantage of the speed, security and debug possibilities of SCIOPTA messages, an optimized asynchronous message interface is included. The IPS application can transmit and receive internet protocol data packages the same way as normal SCIOPTA messages.

Distributed Systems

With its modular design the SCIOPTA IPS stack cannot only be used in high performance single CPU embedded systems but is also suitable for distributed multi-CPU systems.



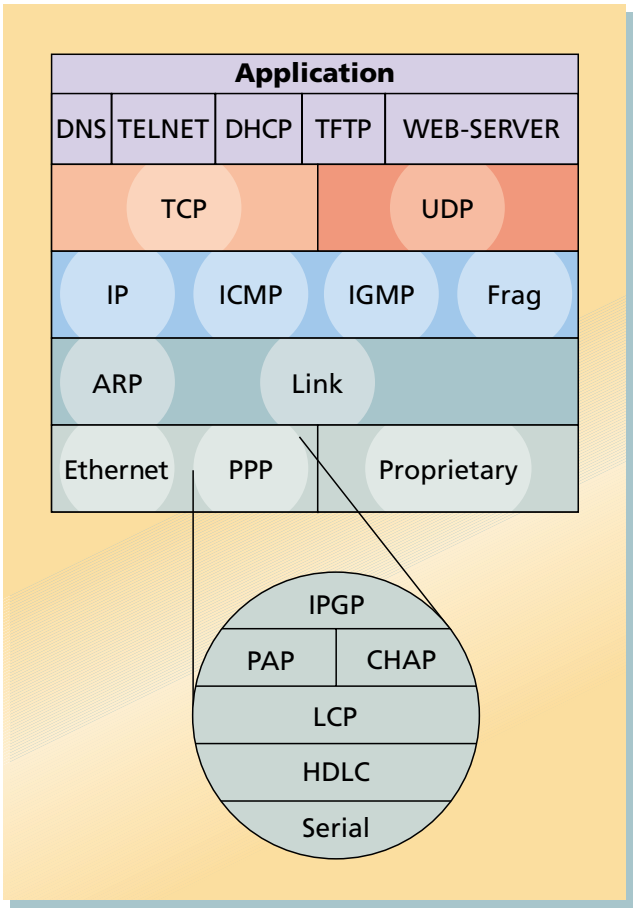
Scalable

Many parts of the SCIOPTA IPS stack are encapsulated in SCIOPTA static or dynamic processes which can be started and stopped individually.

This gives a highly modular design which can be scaled for specific applications.

SCIOPTA IPS Applications

A suite of well-known standard Internet Protocols applications such as TELNET, DNS, DHCP, TFTP are available for SCIOPTA IPS.

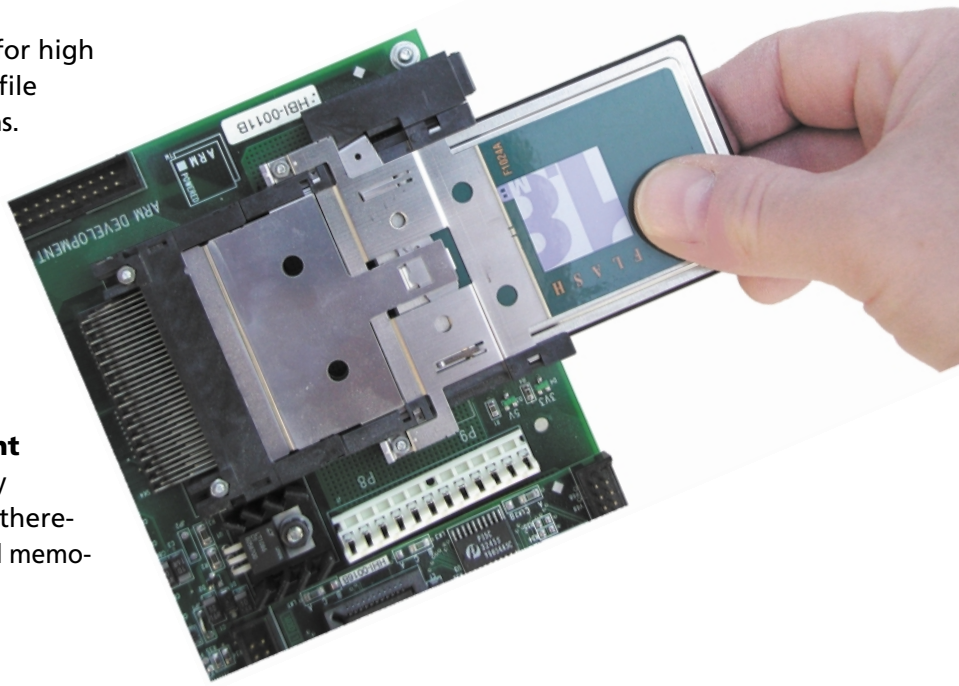


SCIOPTA File System

Flexible File System Architecture

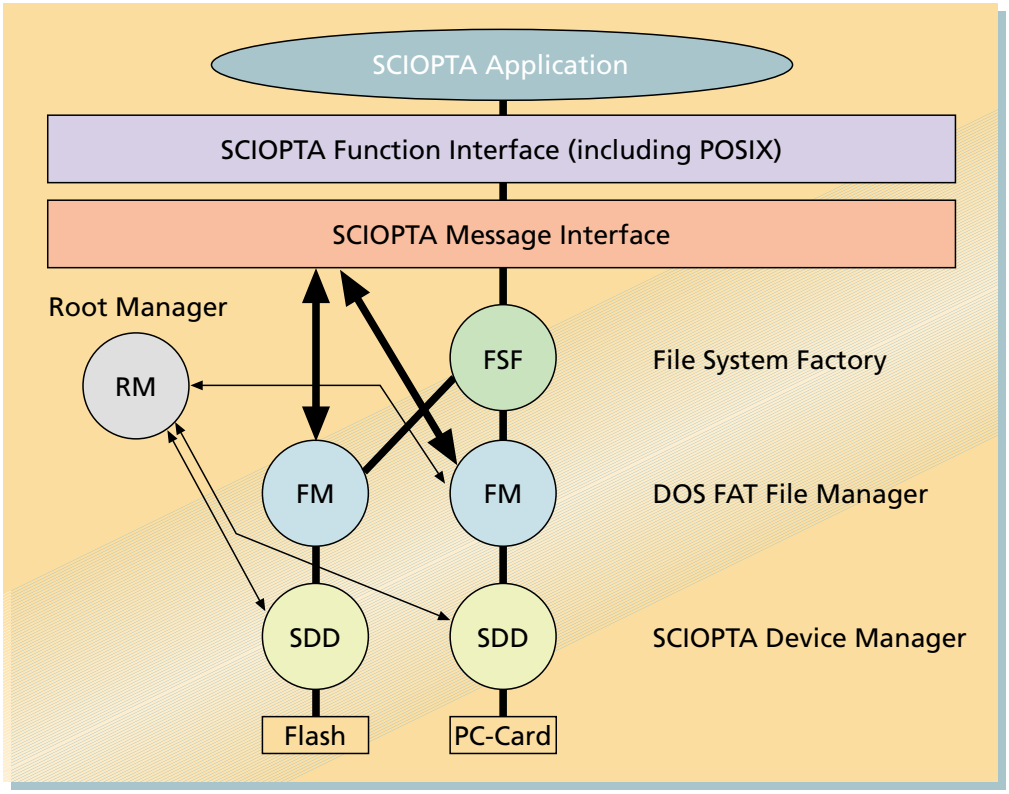
The SCIOPTA File System was designed for high flexibility allowing the user to adapt the file handling for various embedded applications.

An application can use standard POSIX system calls to access the SCIOPTA File System. If you want to take full advantage of the speed, security and debug possibilities of SCIOPTA messages you can directly use messages to communicate with a SCIOPTA File System.



High Performance and Small Footprint

The SCIOPTA File System was specifically designed for embedded systems and can therefore also be used in application with limited memory resources.



SCIOPTA DRUID Debugging System

System-Level Debugging

The SCIOPTA DRUID Debugging System consists of a range of plug-ins for different source-level debuggers to support SCIOPTA system level debugging.

In addition to all source-level windows and display functions the SCIOPTA DRUID Debugging System allows you to debug and view your system on a higher operating system level.

State and information about SCIOPTA objects such as modules, processes, pools and messages can be displayed and observed.

A system event trace can be used to store system events such as a message sequence or process swaps preceding a possible faulty system state.

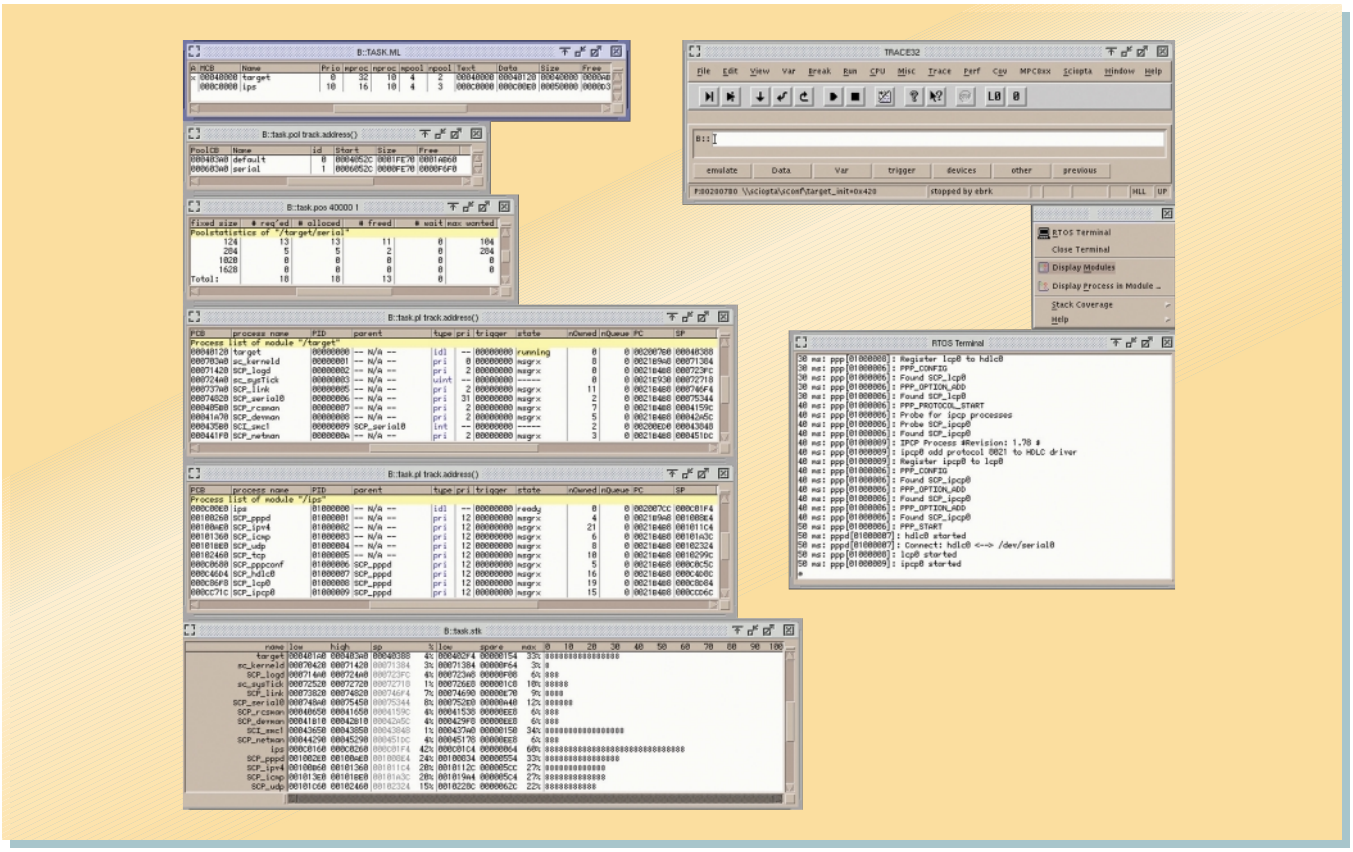
Efficient Debugging

A typical debugging case in a real-time system can quickly emphasize the extraordinary power of the SCIOPTA DRUID Debugging System.

It can happen that a user erroneously does not return memory after use or just has the return code at a wrong place. In SCIOPTA this would for instance be a missing sc_msgFree() system call after receiving a message.

After running for a while, the system will respond with an out of memory error message as one message will never be given back to the system, so eating up all memory.

With the SCIOPTA DRUID Debugging System you can find the source of the error in just a view minutes. You just need to analyse the message pool, which will show a large number of messages owned by the receiving process. As the sender, owner and type of message are stored you can immediately see the problem.



www.sciopta.com

High Performance Real-Time Operating Systems

Headquarters
Litronic AG
Gartenstrasse 76
CH–4052 Basel
Switzerland
Tel. +41 61 276 90 90
Fax +41 61 276 90 99
sales@sciopta.com

Germany
SCIOPTA Systems GmbH
Hauptstrasse 293
D–79576 Weil am Rhein
Germany
Tel. +49 7621 940 919 0
Fax +49 7621 940 919 19
sales@sciopta.com