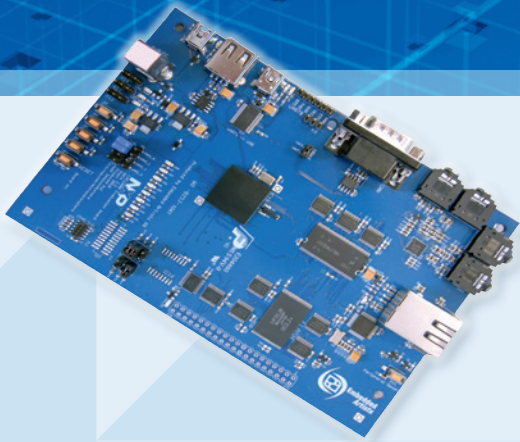


# ARM Development Platform



The Development Platform consists of the following basic elements:

- **C/C++ compiler with leading optimization technology**
- **Eclipse™ integrated development environment**
- **Universal Debug Engine**

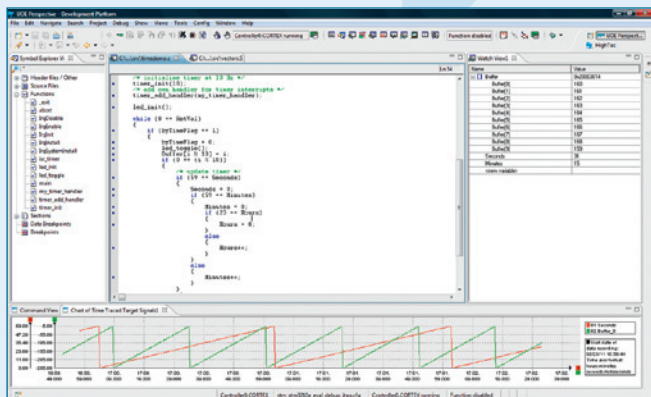
The ARM Development Platform is a well integrated joint product of HighTec and PLS Development Tools. The platform can be extended by the PXROS-HR realtime operating system with integrated MPU management.

## Features

The Development Platform includes powerful wizards, and supports different ARM Cortex-M3 and M4 derivatives. It manages the project settings and the entire build process for compiler, assembler and linker.

- **Project management**
- **Setup wizards**
- **Version control**

PLS have integrated their UDE debugger into the standard Eclipse™ environment. Launching the debugger within Eclipse™ will open the new UDE perspective with high-end debugging features.



The Eclipse™-based ARM Development Platform allows the simple definition of projects. After having selected a particular microcontroller derivative, the generation of a project with a correct startup code, the necessary hardware initialization, a valid memory layout, and the corresponding header file for the names and bit fields of the peripheral register can be prompted virtually at the push of a button. The project contains a simple main function, which allows the implementation to be started immediately.

## C/C++ compiler

- **Robust, compact and fast executing code**
- **Thumb2 instruction set**
- **VFP support**
- **Long-term support**

The ARM GNU C/C++ compiler is the fastest build system on the market. Furthermore, it can be started several times simultaneously for speeding up the build process by parallel compilation processes.

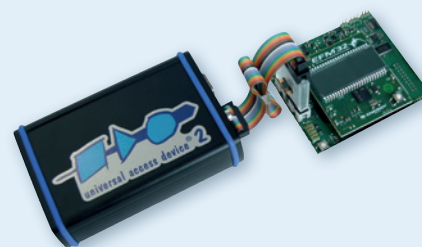
The GNU compiler is fully EABI-compliant and conforms to all relevant language and ISO standards.

## Universal Debug Engine

For debugging and testing the applications, the Development Platform includes a direct connection to the Universal Debug Engine (UDE) from PLS. The UDE can be started with all necessary settings directly from the IDE, thus hugely simplifying program function tests on the target hardware. The UDE, of course, also allows testing PXROS-HR applications with memory protection.

- **Target access via JTAG and SWD**
- **FLASH programming**
- **Real time data monitoring and graphical view**
- **Execution time measurement**
- **Instruction pointer profiling**
- **Simulated I/O**
- **PXROS-HR debug support**

## Debugger Hardware



- **High speed host interfaces: USB 2.0, FireWire, Ethernet-100Mbit**
- **Support of latest Coresight debug features:**
  - **Serial Wire Viewer (SWV)**
  - **Instrumentation Trace Macrocell (ITM)**
  - **Data Watchpoint and Trace (DWT)**
- **Coresight ETM trace**

# PXROS-HR

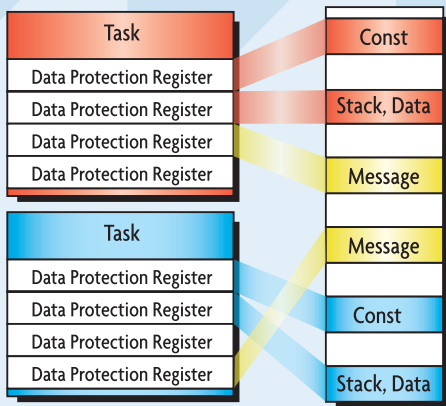
- Microkernel RTOS - No interrupt latency
- Integrated MPU management
- SIL-3 certification support
- Simplified debugging and error diagnosis

The PXROS-HR microkernel realtime operating system with integrated MPU management is an optional component of the Development Platform. Modern microcontrollers, such as ARM Cortex-M3 and Cortex-M4, TriCore and Power Architecture, include a Memory Protection Unit, which guarantees safe integration of functionalities consisting of different software components. With these features PXROS-HR is ideally suited for safety-critical applications and rugged industrial solutions.

The ingenious thing about PXROS-HR is that protection and communication between the individual encapsulated tasks are not realised on a software basis but by utilising the Memory Protection Unit (MPU) of a controller instead – as if each component (capsule) ran on a controller of its own. This technology ensures that no capsule can impair the integrity of the overall system.

Hardware-based memory protection is realised by means of the data protection registers and code protection registers of a memory protection unit, and is individually adjustable for different privilege levels. Each protection register uses an upper and lower boundary as well as specific access rights (reading, writing, executing) for configuring a “view window” within the memory. Switching between the protection registers for the respective software modules is handled by PXROS-HR. Any unauthorized access to a module outside the view window during the operating system’s runtime is detected and prevented, thereby avoiding any propagation of software errors.

The typical configuration of software modules (PXROS-HR tasks) is shown here.



Two of the existing data protection registers configure the sectors for the constants, stack and data of a task. The remaining registers can be used for exchanging messages. Sending a message from Task1 and receiving this message in Task2 corresponds to handing over an encapsulated memory sector (object) together with its access right. The protected message object can thus have any size.

## PXview

A graphic interface is used for visualising the tracing information of PXROS-HR tasks and services such as scheduling, message passing and event handling.



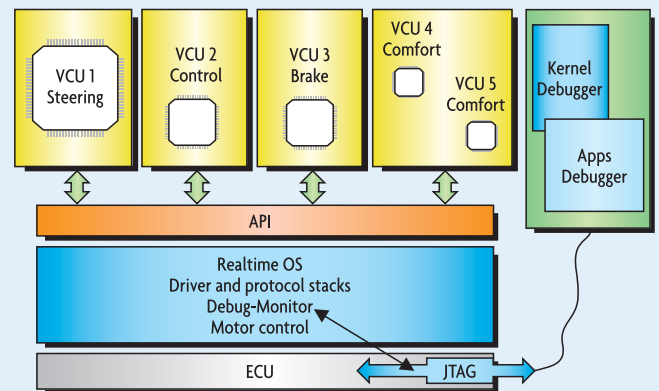
PXview allows a detailed view into a running PXROS-HR application; the system can be analysed by the debugger without having to stop the application.

Furthermore, a library, especially developed for the PXROS-HR debug monitor PXmon, allows the Universal Debug Engine to use the JTAG debug channel as a fast communication vehicle for the exchange of data with running PXROS-HR applications.

System conditions, such as the stack consumption of individual tasks, the process sequence of tasks, the processor workload due to individual application parts, or the workload of resources, are graphically displayed. The user is thus provided with an effective tool for testing PXROS-HR based programs and optimising the performance parameters of the application.

## Debugging and error diagnosis

The first operating mode uses a JTAG debugger for debugging the PXROS-HR kernel.



The second operating mode enables debugging of reloadable PXROS-HR tasks. Since it is not desirable to halt the hardware in this case, debugging of the application takes place via a debug monitor, which is integrated into the target operating system. The monitor encapsulates task-specific breakpoints, context handling and call stacks for the debugger, which is largely transparent and can be addressed by means of a gdb-compatible command interface. Ethernet, JTAG, CAN and serial interfaces serve as standard connections to the target monitor.

‘Eclipse’ and ‘Built on Eclipse’ are trademarks of Eclipse Foundation, Inc.