KEIL™
An ARM® Company

Newsletter 2007

8-, 16-, and 32-Bit Device Support

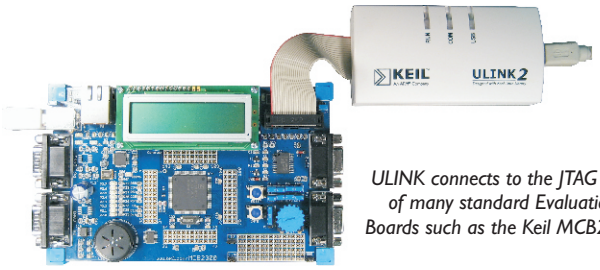
During the last year Keil has continued to improve the 8-, 16-, and 32-bit software development tools for microcontrollers.

The RealView® **Microcontroller Development Kit** (MDK) combines the ARM RealView Compiler with the Keil™ µVision® IDE/Debugger, providing developers with a feature-rich environment optimized for ARM processor-based microcontrollers. The industry-standard RealView Compiler delivers the tightest, highest performing code and is optimized for 32-bit ARM and 16-bit Thumb® instruction sets while supporting full ISO standard C and C++.

Keil **C51 Version 8** and Keil **C166 Version 6** support the latest 8051 and XC16x devices and the new high-performance **ULINK2** USB/JTAG Adapter.

ULINK2 USB/JTAG Adapter

ULINK2 connects the USB port of your PC to the JTAG or OCDS port of your target hardware. The ULINK2 adapter supports several microcontroller architectures including ARM7™, ARM9™, ARM Cortex™-M3, Infineon XC800/XC16x, and ST µPSD. The ULINK2 adaptor enables you to download programs to on-chip and external Flash, set breakpoints, view memory contents, and single-step through your program.



ULINK connects to the JTAG Port of many standard Evaluation Boards such as the Keil MCB2370

For standard ARM7 and ARM9 development ULINK2 introduces the **Real-Time Agent** for 'on-the-fly' debugging. Via a standard JTAG interface, you may now examine variables and memory in a running system, use serial I/O via the JTAG communciation channel, or set breakpoints even on variable access with value ranges.

µVision Development Process

Software Debugging with Complete Device Simulation

Code Optmization with Keil C51 and Keil C251

Real-Time Solutions for CAN, USB, and TCP/IP Networking

ULINK2 JTAG Debugger and Flash Programmer

ULINK2 Highlights

- On-chip Debugging (using on-chip JTAG, SWD, or OCDS)
- Up to 10MHz JTAG clock including auto-adjust (RTCK)
- Flash Memory Programming (using pre-configured and user-configurable algorithms)
- Real-Time Agent for debugging of running systems
- Plug-and-Play installation using standard Windows USB drivers
- Serial Wire Debug (SWD) for ARM Cortex-M3 based devices
- New 10-pin 0.05" JTAG connector

2007

Efficient Development Cycle

The µVision IDE/Debugger is common to all Keil development tools, offering the same environment for 8051, C166 or ARM processor-based architectures. It provides full control of the Compiler, Assembler, Real-Time OS, Project Manager, and Debugger in a seamless, intelligent environment.

The µVision Device Database® avoids typical user errors and reduces the complexity of the software development process. The µVision IDE is versatile, powerful, and flexible and is designed to manage even the most demanding embedded applications. But a major feature is the ease of use that is reflected in the 4 Steps of the software development process outline below.

Improving Software Quality

The µVision design process lets you focus on the application code and enables you to deliver better products faster. The µVision Debugger supports complete device simulation using only your PC and provides trace capture, execution profiler, code coverage, and logic analyzer for detailed analysis of the application code. This in-turn improve overall software quality.

Once the application code is running with the µVision simulator, the same debugger interface can be used for program testing in target hardware (by using the ULINK2 adapter or a third-party emulator). As simulation already uncovered most of the problems in your application code, the time spent for target testing is reduced.

Step 1

Select Device and Specify Target Hardware

The Keil Device Database provides detailed information on more than 1,200 microcontrollers. The parametric search tool available at www.keil.com/dd helps you find the best matching device for your application. To configure the tools, just select the same device in the ì Vision IDE and specify target parameters such as clock speed and external memory.

Step 2

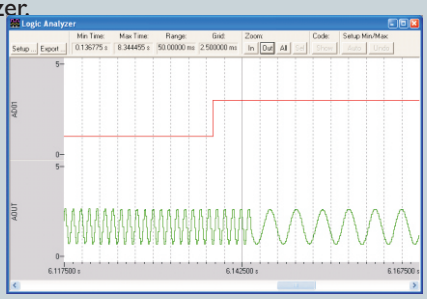
Configure Device and Create Application

The µVision IDE provides preconfigured device start-up code. A built-in configuration wizard simplifies device set-up and lets you focus on your application code. Also included in the µVision IDE are extensive project templates, project examples and application notes you may use to jump start your application.

Step 3

Verify and Optimize Application Program in Device Simulation

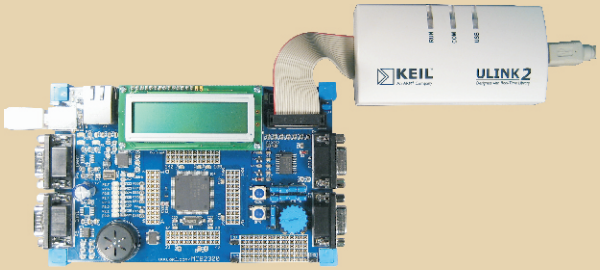
Use the µVision Simulator to accelerate your project development cycle. Develop software before hardware is available. Verify and optimize your application using advanced features such as instruction trace, code coverage, execution profiling and µVision Logic Analyzer.



Step 4

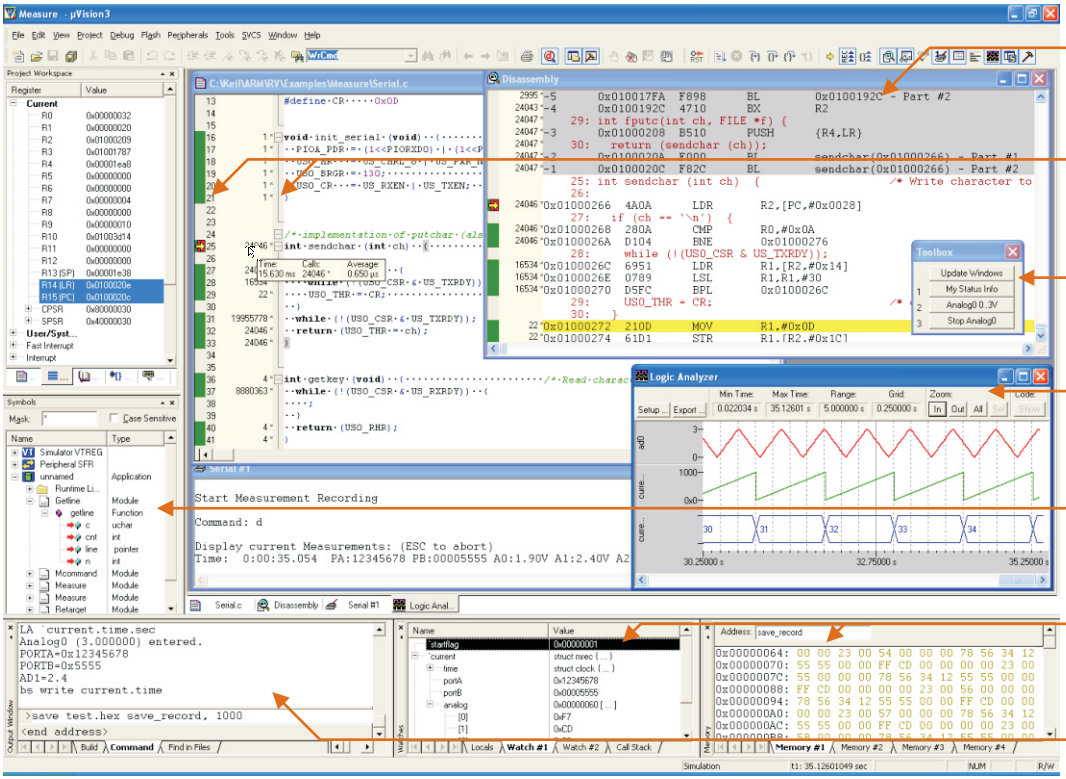
Download to Flash and Test Application

Download your code to the Flash ROM of your target system using a serial cable or the Keil ULINK2™ USB-JTAG Adapter, which connects directly to the JTAG pins of your microcontroller. The ULINK2 adapter supports Flash programming, single stepping, real-time program execution with breakpoints, and access to memory and CPU registers.



The 4 Steps of the µVision Development Process helps you to reduce development time and improves software quality.

µVision Debugging



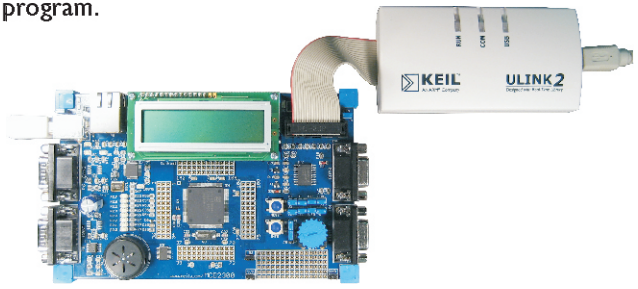
Accurate Device Simulation

The µVision Debugger simulates a complete ARM Powered® microcontroller including the instruction set and on-chip peripherals. These powerful simulation capabilities provide serious benefits and promote rapid, reliable embedded software development.

- Simulation allows software testing on your desktop with no hardware environment
- Early software debugging on a functional basis improves overall software reliability
- Simulation allows breakpoints that are not possible with hardware debuggers
- Simulation allows for optimal input signals (hardware debuggers add extra noise)
- Signal functions are easily programmed to reproduce complex, real-world input signals
- Single-stepping through signal processing algorithms is possible. External signals stop when the CPU halts
- It is easy to test failure scenarios that would destroy real hardware peripherals

Target Debugging

The ULINK2 adapter connects the USB port of your PC to the JTAG port of your target board. When used with the µVision IDE, the ULINK2 adapter enables you to download programs to on-chip and external Flash, set breakpoints, view memory contents, and single-step through your program.



Evaluation Hardware

Keil provides a wide range of evaluation boards you may use to jump-start your product development and rapidly evaluate the performance of ARM Powered MCU's. For more information, refer to www.keil.com/boards.

Use Cases for Device Simulation

User Cases

The µVision IDE provides a cycle-accurate model of the complete MCU device including the CPU instruction set and all on-chip peripherals such as I/O prts, UART's, I²C, A/D, D/A and CAN.

The following real-life use cases demonstrate how complete device simulation allows you to fully develop, optimize and verify your application without the need for target hardware and enables you to test and debug your application in ways that are impossible using hardware debug.

Illegal Memory Accesses

Simulation automatically checks for a variety of illegal memory accesses including:

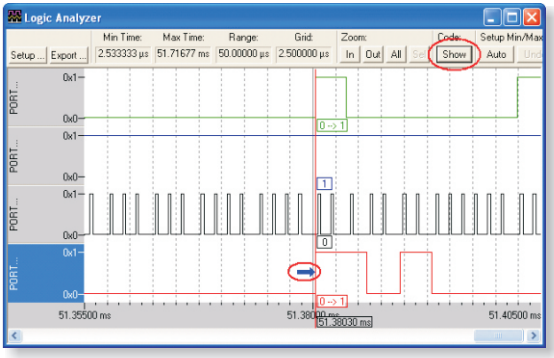
- Un-aligned memory accesses
- Writes to ROM that do not modify memory content
- Access to non existent memory locations

In these cases, complete device simulation enables better software verification without any extra verification effort.

More details: www.keil.com/download/docs/323.asp

Detect I/O Glitches

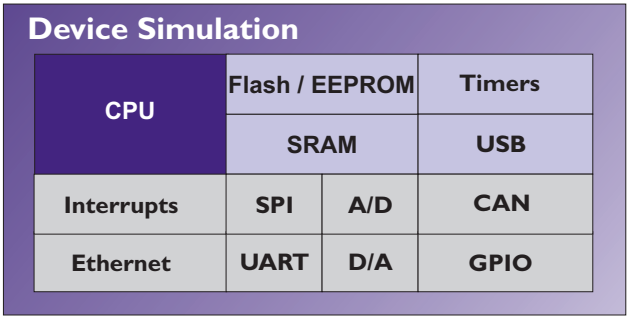
Glitches of the I/O pins can be detected and verified using the Logic Analyzer in the µVision Simulator. This type of glitch significantly increases the EMC behavior of electronic equipment and are often hard to detect.



Logic Analyzer window showing signals after glitch correction

When using the µVision Logic Analyzer these glitches are easy to find and by clicking the signal name you can immediately access the source code for correction.

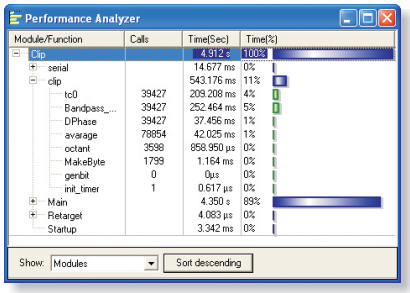
More details: www.keil.com/download/docs/322.asp



Application Optimization

Sophisticated simulation tools make optimizing applications much easier. The performance analyzer allows you to view the CPU time required by all application tasks; full source code and execution time can be easily viewed by double-clicking a function name.

More details: www.keil.com/download/docs/326.asp

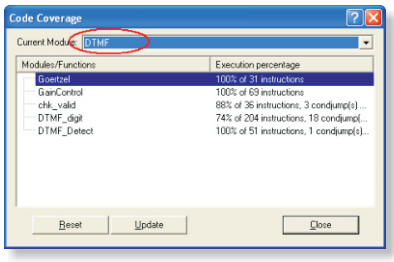


Performance Analyzer shows execution time for tasks and programs.

Code Verification

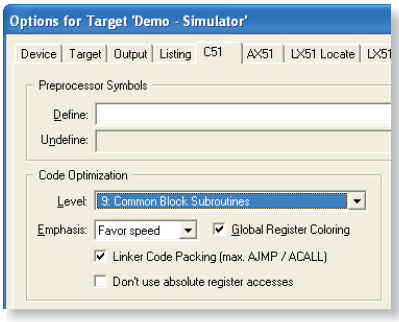
Today's applications require a high level of program verification and algorithm testing. By using simulation and the µVision Code Coverage Window you may check that all paths of your application or algorithm are executed. This is useful during program verification and helps you to even identify redundant code in a program.

More details: www.keil.com/download/docs/326.asp



Code Coverage Window verifies program execution.

Many 8051 programs are reaching the code size limits of the architecture or the single chip device. The Cx51 and C251tools offer several unique code optimization features. When you enable these code optimizations, the Keil tools allow devlopers to put more features into the limited address space of single-chip 8-bit devices.



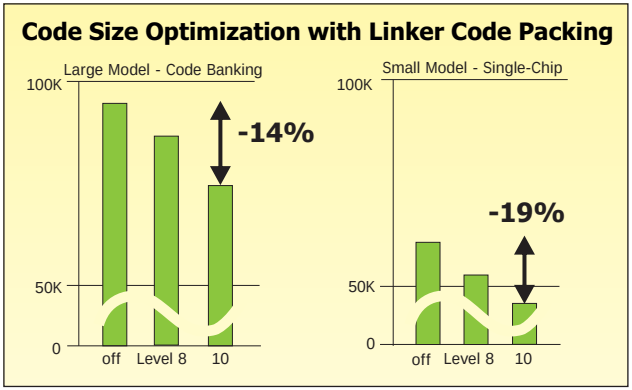
*In the µVision IDE you Compiler Optimizations are configured in the dialog **Project - Options for Target - Cx51**.*

***Linker Code Packing** and **Global Register Coloring** are unique code optimization features of the Keil development tools.*

Linker Code Packing

Linker Code Packing (available with the LX51 and L251 Linker/Locater) analyzes and optimizes your entire program. With Linker Code Packing, the linker performs the following optimizations depending on the level:

- **Level 0 - 7 Maximize AJMP / ACALL** which are shorter than LJMP and LCALL instructions. Just this optimization reduces code size in average by 3%
- **Level 8 Reuse of Common Entry Code** when multiple calls are made to a single function
- **Level 9 Common Block Subroutines** are used to replace Recurring instruction sequences
- **Level 10 Rearrange Code** when detecting common block subroutines to maximize sizes



The examples were on the physical memory limits before using linker code packing. Up to Level 8 no run-time overhead is introduced.

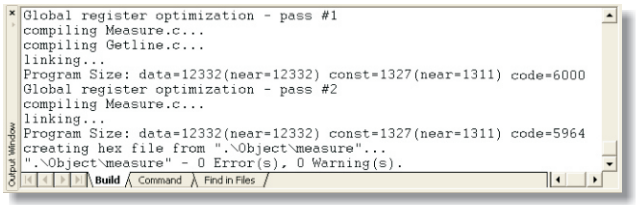
Tips & Tricks for Optimal 8051 Code

- Use the smallest possible data type for variables. Prefer **unsigned char** and **bit**.
- Prefer the **SMALL** memory model. Locate large arrays with **xdata** or **pdata** memory types.
- When using other memory models, apply the memory type **data** to frequently used variables.
- Learn how to use the **pdata** memory type on your specific device. **pdata** memory is a 256-byte memory page in **xdata** memory with fast access.
- Use **memory-typed pointers** to access specific memory types. By default, generic pointers are used that can access any memory type.
- Reduce the usage of **reentrant functions**, since the 8051 lacks support for stack variables.
- Use **Linker Code Packing** to generate common code block sub-routines and optimize the usage of 2-byte AJMP/ACALL instructions.
- Optimize CPU register utilization with **Global Register Coloring** and apply **REGUSE** information to assembly code.
- **Order functions used before caller functions** within a module for module register optimization.

Global Register Coloring

Global Register Coloring optimizes CPU register usage across a complete application. The linker generates a file (REGFILE) that lists the CPU registers used by each function. This REGFILE is used as feedback to the compiler to obtain register usage for external functions. Registers that are not altered can then be used for variables and the code generated is smaller and faster.

The performance gain depends on the application code whereby the most effect is on low-level interface routines. Therefore you should specify the register usage even in your assembly routines with the **REGUSE** directive.



*The µVision **Build** command re-compiles modules to maximize register variables when **Global Register Coloring** is enabled.*

RTX Kernel

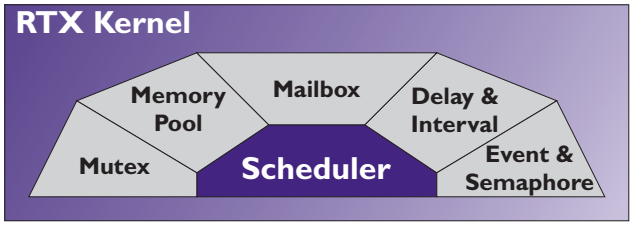
Today, microcontroller applications demand more critical control, often requiring simultaneous execution of multiple tasks in a real-time environment.

While it is possible to implement an embedded program without using a real-time kernel, the proven Keil RTX enables developers to concentrate on application development, save time, produce a reliable, expandable system and makes software development easier.

A real-time kernel should be viewed as an essential building block of a modern embedded system, allowing developers to create complex, flexible but easy to implement and maintain systems.

The Keil range of real-time kernels have each been developed specifically for 8051, 166 or ARM processor-based devices and are fully integrated into the Keil tools and are royalty-free.

	RTX51 Tiny2	ARTX 166	RL-ARM
Task Scheduler			
Round Robin	Yes	Yes	Yes
Pre-emptive	No	Yes	Yes
Cooperative	Yes	Yes	Yes
Task Specifications			
Priority Levels	1	256	256
Defined Tasks (max)	16	256	Unlimited
Active Tasks (max)	16	256	256
Context Switch	100-700	<500 cycles	<300 cycles
Interrupt Latency	<50	4 cycles	100 cycles
Memory Requirments			
CODE Space	900 Bytes	1.5K Bytes	<5K Bytes
RAM Space	7 Bytes	8 Bytes	~500 Bytes
Timer/Signals/Events			
Timeouts	Yes	Yes	Yes
Intervals	Yes	Yes	Yes
User Timers	Unlimited	Unlimited	Unlimited
Signals (max)		16 per task	16 per task
Inter-Task Communication			
Semaphores	8	Unlimited	Unlimited
Mailboxes	8	Unlimited	Unlimited



Real-Time Kernel Benefits

A real-time kernel offers significant advantages to embedded systems developers

- Manages system resources and task scheduling
- Takes care of system 'housekeeping' tasks
- Enables deterministic system behavior
- High-speed real-time operation (if required)
- Developers focus on application development
- Insulation layer between software and hardware
- Easy implementation of complex systems

High speed may not be your primary concern and real-time applications can certainly be high speed, however real-time operation simply means that tasks or events should happen within a defined time period or in a predetermined order, real-time systems can be as fast or slow as you need.

RTX51 Tiny2 for 8051

The RTX51 Tiny2 multitasking real-time kernel works on all classic 8051 device variants and makes implementing complex, time-critical software projects easy.

ARTX166 for C166, ST10, XC16x

ARTX166 includes a dedicated RTX real-time kernel for C166, ST10, and XC16x that is combined with the TCPnet Networking Suite and a Flash File System.

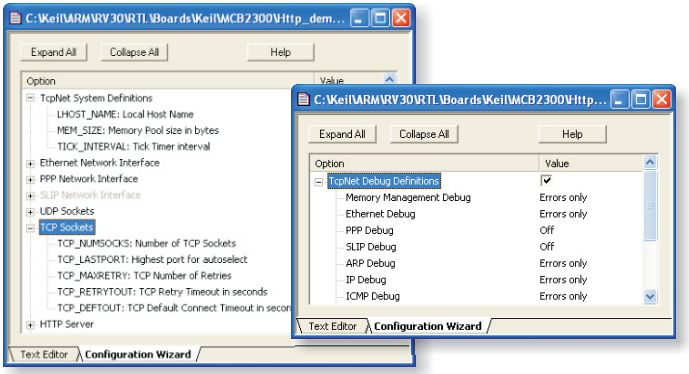
Real-Time Library for ARM

The RealView **Real-Time Library** (RL-ARM) expands RealView MDK with highly efficient middleware components specifically developed for ARM processor-based MCU devices and contains the RTX Kernel (including source code), TCPnet Networking Suite, Flash File System, together with CAN and USB device drivers.

TCPnet Networking Suite

TCP/IP or UDP software layers are easy to implement using the ARTX166 or Real-Time Library TCPnet Networking Suite. TCPnet provides standard Internet protocols (TCP, UDP, ARP, DHCP) and offers flexible connections (with Ethernet or UART/Modem interface).

Parameters and the functionality of the TCPnet Networking Suite can be tailored for your software requirements. TCPnet supports a wide range of features from reliable TCP/IP connections to multi-language HTTP configurations. To create highly interactive applications, TCPnet includes a file conversion utility that creates images for a ROM File System from HTML and graphic files.



Menu-driven configuration is available for all TCPnet components including the Debug Interface.

CAN Driver

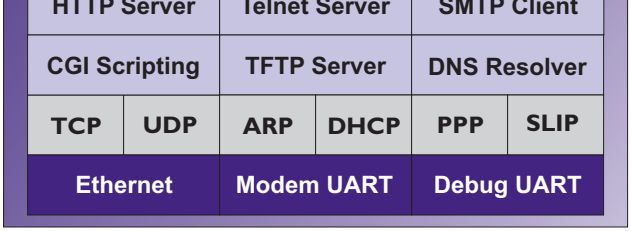
The CAN driver provides high-performance functions that transmit and receive CAN messages. The CAN driver interfaces to RTX via mailboxes and memory pools.

USB Device Interface

The USB interface uses standard device driver classes that are available with all Windows PCs. No Windows host driver development is required. The USB Device interface uses a generic software layer using RTX kernel features.

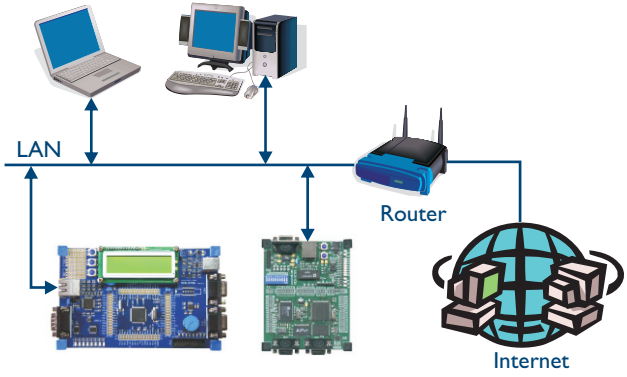
Flash File System

The Flash File System is designed to be fast, simple and efficient while allowing you to create, save, read, and modify files. Files may be stored in standard Flash ROM or RAM devices or on SD memory cards using a FAT file system.



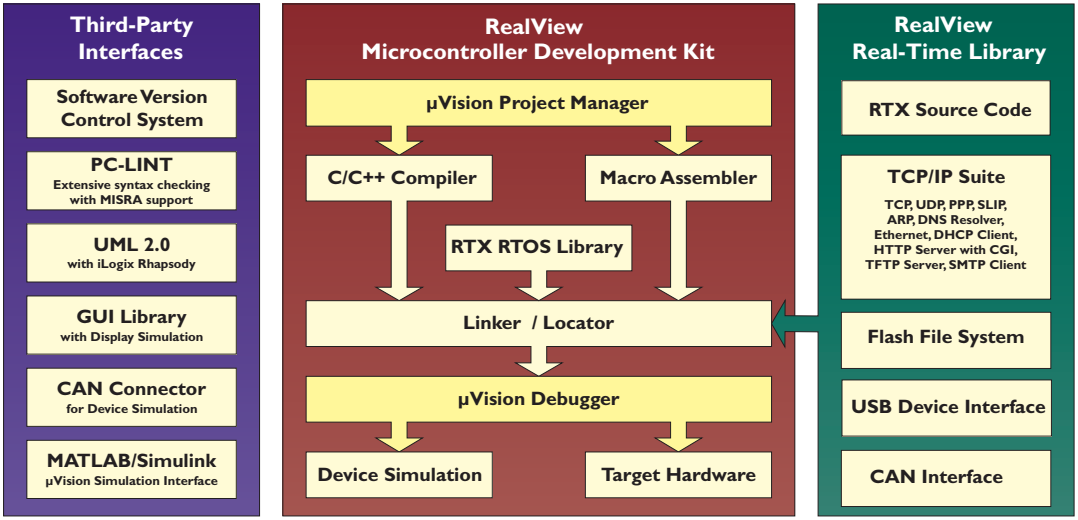
TCPnet Protocol Overview

- **TCP** (Transmission Control Protocol) is a full-duplex, reliable, connection-oriented protocol between network applications
- **UDP** (User Datagram Protocol) is a protocol for sending data packages (with optional checksum) without the overhead or reliability of TCP
- **ARP** (Address Resolution Protocol) translates IP addresses to Ethernet MAC addresses
- **DHCP** (Dynamic Host Configuration Protocol) configures network parameters automatically
- **PPP** (Point to Point Protocol) is used for reliable dial-up modem connections
- **SLIP** (Serial Line Internet Protocol) is a simple protocol for data transmission over serial lines



Template applications help you to get started quickly and are royalty-free when used for product development. The Real-Time Libary components let you therefore focus on the specific requirements of your application.

Keil Microcontroller Development Tools 8, 16, and 32-Bit



The RealView **Microcontroller Development Kit** is a complete software development environment for ARM7, ARM9, and Cortex-M3 processor-based microcontrollers. It includes the tools you need to create, and debug C, C++, and assembly source files. Like all tools based on the Keil µVision IDE, it is easy to learn and easy to use, yet powerful enough for the most demanding ARM processor-based embedded applications.For a complete list of currently supported devices, refer to **www.keil.com/dd**.

The **Real-Time Library** is based on a real-time kernel that simplifies the design and implementation of complex, time-critical applications. A Flash file system, TCP/IP networking suite, and other communication protocols are included. For more information, refer to **www.keil.com/rl-arm**.

Debug Solutions

Keil is the world-wide leader for microcontroller development tools.

Legendary is the µVision **Device Simulation** that allows software testing without actual hardware and enables detailed algorithm verification.

The **ULINK2** adapter supports Flash programming and software debugging via the JTAG/OCDS interface available on all ARM powered MCU's and many 8051 and 166 devices.

The **Real-Time Agent** solves the challenge of debugging an embedded system without stopping the program execution.

And with Signum JTAG-JET you have access to trace recording via the ARM **Embedded Trace Macrocell**™ connector.

Europe:
Keil
Bretonischer Ring 15
D-85630 Grasbrunn
Germany

Phone +49 89 / 45 60 40 - 0
Support +49 89 / 45 60 40 - 24
FAX +49 89 / 46 81 62

Email sales.intl@keil.com
support.intl@keil.com

United States:
Keil
1501 10th Street, Suite 110
Plano, Texas 75074
USA

Phone +1 800 348 8051
+1 972 312 1107
FAX +1 972 312 1159

Email sales.us@keil.com
support.us@keil.com

KEIL™ An ARM® Company

RealView® Tools by ARM®