Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

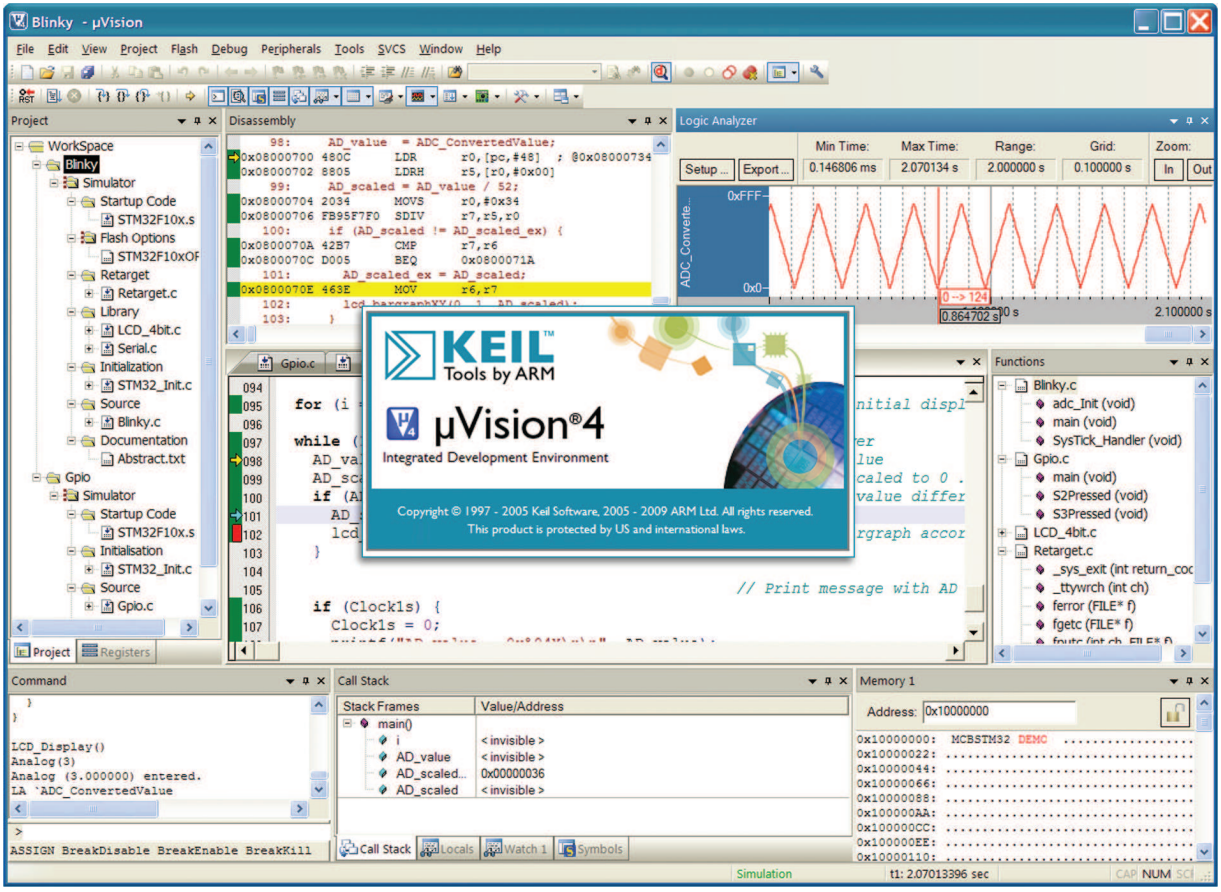
Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

KEIL™
Tools by ARM

Getting Started

Creating Applications with µVision®4

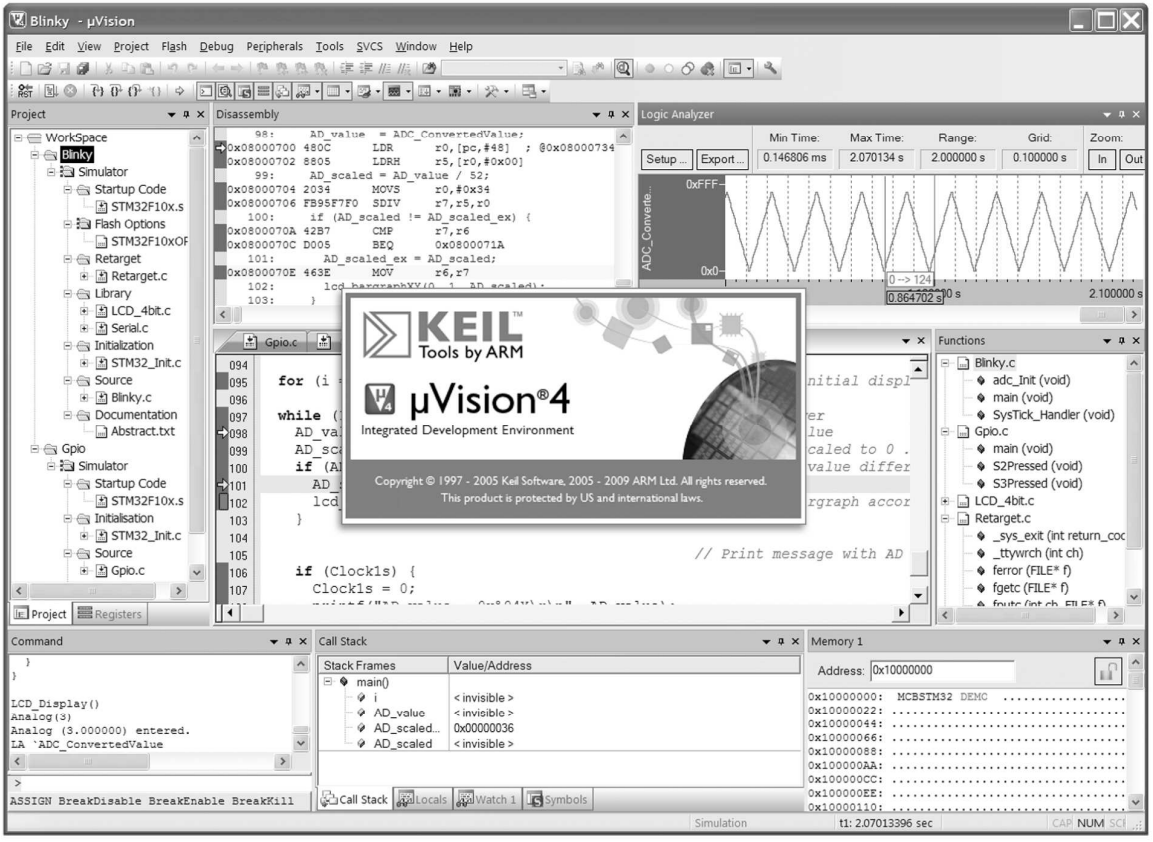


For 8-bit, 16-bit, and 32-bit Microcontrollers

KEIL™
Tools by ARM

Getting Started

Creating Applications with µVision®4



For 8-bit, 16-bit, and 32-bit Microcontrollers

NOTE
This manual assumes that you are familiar with Microsoft Windows and the hardware and instruction set of the ARM7, ARM9, Cortex-Mx, C166, XE166, XC2000, or 8051 microcontroller.

Every effort was made to ensure accuracy in this manual and to give appropriate credit to persons, companies, and trademarks referenced herein.

Preface

This manual is an introduction to the Keil development tools designed for Cortex-Mx, ARM7, ARM9, C166, XE166, XC2000, and 8051 microcontrollers. It introduces the µVision Integrated Development Environment, Simulator, and Debugger and presents a step-by-step guided tour of the numerous features and capabilities the Keil embedded development tools offer.

Who should Read this Book

This book is useful for students, beginners, advanced and experienced developers alike.

Developers are considered experienced or advanced if they have used µVision extensively in the past and knowledge exists of how the µVision IDE works and interacts with the debugger, simulator, and target hardware. Preferably, these developers already have a deep understanding of microcontrollers. We encourage this group of engineers to get familiar with the enhancements introduced and to explore the latest features in µVision.

Developers are considered students or beginners if they have no working experience with µVision. We encourage this group of developers to start by reading the chapters related to the µVision IDE and to work through the examples to get familiar with the interface and configuration options described. They should make use of the ample possibilities the simulator offers. Later on, they should continue with the chapters describing the RTOS and microcontroller architectures.

However, it is assumed that you have a basic knowledge of how to use microcontrollers and that you are familiar with a few instructions or with the instruction set of your preferred microcontroller.

The chapters of this book can be studied individually, since they do not strictly depend on each other.

Chapter Overview

"Chapter 1. **Introduction**", provides an overview of product installation and licensing and shows how to get support for the Keil development tools.

"Chapter 2. **Microcontroller Architectures**", discusses various microcontroller architectures supported by the Keil development tools and assists you in choosing the microcontroller best suited for your application.

"Chapter 3. **Development Tools**", discusses the major features of the µVision IDE and Debugger, Assembler, Compiler, Linker, and other development tools.

"Chapter 4. **RTX RTOS Kernel**", discusses the benefits of using a Real-Time Operating System (RTOS) and introduces the features available in Keil RTX Kernels.

"Chapter 5. **Using µVision**", describes specific features of the µVision user interface and how to interact with them.

"Chapter 6. **Creating Embedded Programs**", describes how to create projects, edit source files, compile, fix syntax errors, and generate executable code.

"Chapter 7. **Debugging**", describes how to use the µVision Simulator and Target Debugger to test and validate your embedded programs.

"Chapter 8. **Using Target Hardware**", describes how to configure and use third-party Flash programming utilities and target drivers.

"Chapter 9. **Example Programs**", describes four example programs and shows the relevant features of µVision by means of these examples.

Document Conventions

Examples	Description
README.TXT[1]	Bold capital text is used to highlight the names of executable programs, data files, source files, environment variables, and commands that you can enter at the command prompt. This text usually represents commands that you must type in literally. For example: **ARMCC.EXE DIR LX51.EXE**
`Courier`	Text in this typeface is used to represent information that is displayed on the screen or is printed out on the printer This typeface is also used within the text when discussing or describing command line items.
Variables	Text in italics represents required information that you must provide. For example, *projectfile* in a syntax string means that you must supply the actual project file name Occasionally, italics are also used to emphasize words in the text.
Elements that repeat…	Ellipses (…) are used to indicate an item that may be repeated

Omitted code
.
.
.
	Vertical ellipses are used in source code listings to indicate that a fragment of the program has been omitted. For example:
	void main (void) {
	.
	.
	.
	while (1);

«Optional Items»	Double brackets indicate optional items in command lines and input fields. For example: **C51 TEST.C PRINT** «filename»
{ opt1 | opt2 }	Text contained within braces, separated by a vertical bar represents a selection of items. The braces enclose all of the choices and the vertical bars separate the choices. Exactly one item in the list must be selected.
Keys	Text in this sans serif typeface represents actual keys on the keyboard. For example, “Press Enter to continue

[1] *It is not required to enter commands using all capital letters.*

Contents

Chapter 1. Introduction

Thank you for allowing Keil to provide you with software development tools for your embedded microcontroller applications.

This book, **Getting Started**, describes the µVision IDE, µVision Debugger and Analysis Tools, the simulation, and debugging and tracing capabilities. In addition to describing the basic behavior and basic screens of µVision, this book provides a comprehensive overview of the supported microcontroller architecture types, their advantages and highlights, and supports you in selecting the appropriate target device. This book incorporates hints to help you to write better code. As with any **Getting Started** book, it does not cover every aspect and the many available configuration options in detail. We encourage you to work through the examples to get familiar with µVision and the components delivered.

The Keil Development Tools are designed for the professional software developer, however programmers of all levels can use them to get the most out of the embedded microcontroller architectures that are supported.

Tools developed by Keil endorse the most popular microcontrollers and are distributed in several packages and configurations, dependent on the architecture.

- **MDK-ARM**: Microcontroller Development Kit, for several ARM7, ARM9, and Cortex-Mx based devices
- **PK166**: Keil Professional Developer's Kit, for C166, XE166, and XC2000 devices
- **DK251**: Keil 251 Development Tools, for 251 devices
- **PK51**: Keil 8051 Development Tools, for Classic & Extended 8051 devices

In addition to the software packages, Keil offers a variety of evaluation boards, USB-JTAG adapters, emulators, and third-party tools, which completes the range of products.

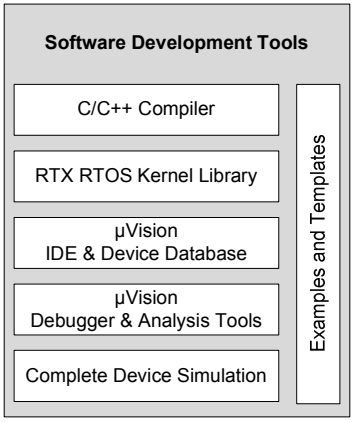
The following illustrations show the generic component blocks of µVision in conjunction with tools provided by Keil, or tools from other vendors, and the way the components relate.

Software Development Tools

Like all software based on Keil's µVision IDE, the toolsets provide a powerful, easy to use and easy to learn environment for developing embedded applications.

They include the components you need to create, debug, and assemble your C/C++ source files, and incorporate simulation for microcontrollers and related peripherals.

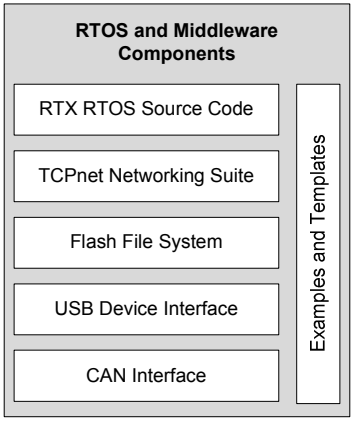
The RTX RTOS Kernel helps you to implement complex and time-critical software.



RTOS and Middleware Components

These components are designed to solve communication and real-time challenges of embedded systems. While it is possible to implement embedded applications without using a real-time kernel, a proven kernel saves time and shortens the development cycle.

This component also includes the source code files for the operating system.



Hardware Debug Adapters

The µVision Debugger fully supports several emulators provided by Keil, and other vendors. The Keil ULINK USB-JTAG family of adapters con nect the USB port of a PC to the target hardware. They enable you to download, test, and debug your embedded application on real hardware.

Last-Minute Changes

As with any high-tech product, last minute changes might not be included into the printed manuals. These last-minute changes and enhancements to the software and manuals are listed in the **Release Notes** shipped with the product.

Licensing

Each Keil product requires activation through a license code. This code is obtained via e-mail during the registration process. There are two types of product licenses:

- **Single-User License** is available for all Keil products. A Single-User License grants the right to use a product on a maximum of two computers to one user. Each installation requires a license code that is personalized for the computer on which the product is installed. A Single-User license may be uninstalled and moved to another computer.
- **Floating-User License** is available for many Keil products. The Floating-User license grants the right to use that product on several computers by several different developers at the same time. Each installation of the product requires an individual license code for each computer on which the product is installed.

Installation

Please check the minimum hardware and software requirements that must be satisfied to ensure that your Keil development tools are installed and will function properly. Before attempting installation, verify that you have:

- A standard PC running Microsoft Windows XP, or Windows Vista
- 1GB RAM and 500 MB of available hard-disk space is recommended
- 1024x768 or higher screen resolution; a mouse or other pointing device
- A CD-ROM drive

Keil products are available on CD-ROM and via download from **www.keil.com**. Updates to the related products are regularly available at **www.keil.com/update**.

Installation using the web download

1. Download the product from **www.keil.com/demo**
2. Run the downloaded executable
3. Follow the instructions displayed by the **SETUP** program

Installation from CD-ROM

1. Insert the CD-ROM into your CD-ROM drive. The CD-ROM browser should start automatically. If it does not, you can run **SETUP.EXE** from the CD-ROM.
2. Select **Install Products & Updates** from the CD Browser menu
3. Follow the instructions displayed by the **SETUP** program

Product Folder Structure

The **SETUP** program copies the development tools into subfolders. The base folder defaults to **C:\KEIL**. The following table lists the default folders for each microcontroller architecture installation. Adjust the examples used in this manual to your preferred installation directory accordingly.

Microcontroller Architecture	Folder
MDK-ARM Toolset	**C:\KEIL\ARM**
C166/XE166/XC2000 Toolset	**C:\KEIL\C166**
8051 Toolset	**C:\KEIL\C51**
C251 Toolset	**C:\KEIL\C251**
µVision Common Files	**C:\KEIL\UV4**

Each toolset contains several subfolders:

Contents	Subfolder
Executable Program Files	**\BIN**
C Include/Header Files	**\INC**
On-line Help Files and Release Notes	**\HLP**
Common/Generic Example Programs	**\EXAMPLES**
Example Programs for Evaluation Boards	**\BOARDS**

Requesting Assistance

At Keil, we are committed to providing you with the best embedded development tools, documentation, and support. If you have suggestions and comments regarding any of our products, or you have discovered a problem with the software, please report them to us, and where applicable make sure to:

1. Read the section in this manual that pertains to the task you are attempting
2. Check the update section of the Keil web site to make sure you have the latest software and utility version
3. Isolate software problems by reducing your code to as few lines as possible

If you are still having difficulties, please report them to our technical support group. Make sure to include your license code and product version number. See the **Help – About** Menu. In addition, we offer the following support and information channels, all accessible at **www.keil.com/support**[1]**.**

1. The **Support Knowledgebase** is updated daily and includes the latest questions and answers from the support department
2. The **Application Notes** can help you in mastering complex issues, like interrupts and memory utilization
3. Check the on-line **Discussion Forum**
4. Request assistance through **Contact Technical Support** (web-based E-Mail)
5. Finally, you can reach the support department directly via **support.intl@keil.com** or **support.us@keil.com**

[1] *You can always get technical support, product updates, application notes, and sample programs at* ***www.keil.com/support.***

Chapter 2. Microcontroller Architectures

The Keil µVision Integrated Development Environment (µVision IDE) supports three major microcontroller architectures and sustains the development of a wide range of applications.

- **8-bit (classic and extended 8051)** devices include an efficient interrupt system designed for real-time performance and are found in more than 65% of all 8-bit applications. Over 1000 variants are available, with peripherals that include analog I/O, timer/counters, PWM, serial interfaces like UART, I^2C, LIN, SPI, USB, CAN, and on-chip RF transmitter supporting low-power wireless applications. Some architecture extensions provide up to 16MB memory with an enriched 16/32-bit instruction set.

  The µVision IDE supports the latest trends, like custom chip designs based on IP cores, which integrate application-specific peripherals on a single chip.

- **16-bit (Infineon C166**, **XE166**, **XC2000)** devices are tuned for optimum real-time and interrupt performance and provide a rich set of on-chip peripherals closely coupled with the microcontroller core. They include a Peripheral Event Controller (similar to memory-to-memory DMA) for high-speed data collection with little or no microcontroller overhead.

  These devices are the best choice for applications requiring extremely fast responses to external events.

- **32-bit (ARM7 and ARM9 based)** devices support complex applications, which require greater processing power. These cores provide high-speed 32-bit arithmetic within a 4GB address space. The RISC instruction set has been extended with a Thumb mode for high code density.

  ARM7 and ARM9 devices provide separate stack spaces for high-speed context switching enabling efficient multi-tasking operating systems. Bit-addressing and dedicated peripheral address spaces are not supported. Only two interrupt priority levels, - Interrupt Request (IRQ) and Fast Interrupt Request (FIQ), are available.

- **32-bit (Cortex-Mx based)** devices combine the cost benefits of 8-bit and 16-bit devices with the flexibility and performance of 32-bit devices at extremely low power consumption. The architecture delivers state of the art implementations for FPGAs and SoCs. With the improved Thumb2 instruction set, Cortex-Mx[1] based microcontrollers support a 4GB address space, provide bit-addressing (bit-banding), and several interrupts with at least 8 interrupt priority levels.

Selecting an Architecture

Choosing the optimal device for an embedded application is a complex task. The Keil Device Database (**www.keil.com/dd**) supports you in selecting the appropriate architecture and provides three different methods for searching. You can find your device by architecture, by specifying certain characteristics of the microcontroller, or by vendor.

The following sections explain the advantages of the different architectures and provide guidelines for finding the microcontroller that best fits your embedded application.

8051 Architecture Advantages

- Fast I/O operations and fast access to on-chip RAM in data space
- Efficient and flexible interrupt system
- Low-power operation

8051-based devices are typically used in small and medium sized applications that require high I/O throughput. Many devices with flexible peripherals are available, even in the smallest chip packages.

[1] *Cortex-M0 devices implement the Thumb instruction set.*

C166, XE166 and XC2000 Architecture Advantages

- Extremely fast I/O operations via the Peripheral Event Controller
- High-speed interrupt system with very well-tuned peripherals
- Efficient arithmetic and fast memory access

These devices are used in medium to large sized applications that require high I/O throughput. This architecture is well suited to the needs of embedded systems that involve a mixture of traditional controller code and DSP algorithms.

ARM7 and ARM9 Architecture Advantages

- Huge linear address space
- The 16-bit Thumb instruction set provides high code density
- Efficient support for all C integer data types including pointer addressing

ARM7 and ARM9-based microcontrollers are used for applications with large memory demands and for applications that use PC-based algorithms.

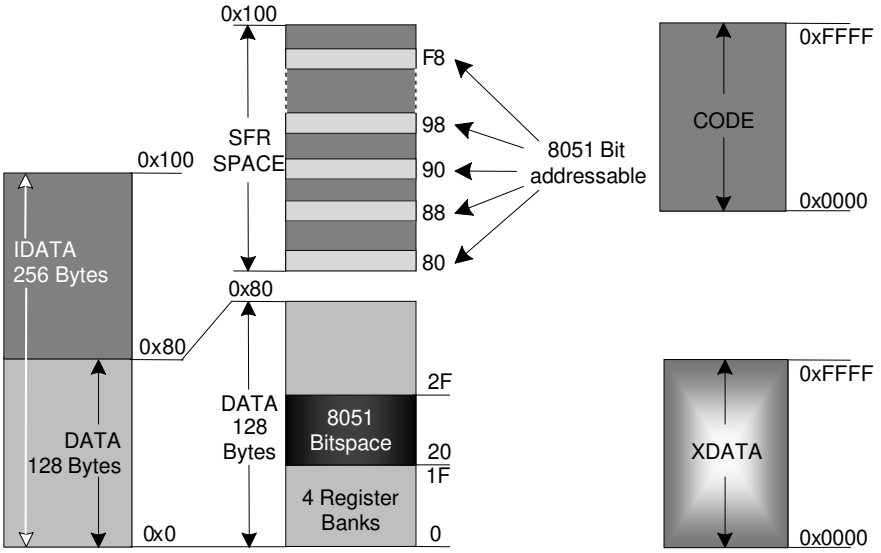
Cortex-Mx Architecture Advantages

- One instruction set, Thumb2, reduces the complexity of the program code and eliminates the overhead needed for switching between ARM and Thumb instruction mode
- The Nested Vector Interrupt Controller (NVIC) removes interrupt prolog and epilog code, and provides several, configurable priority levels
- Extremely low power consumption with a variety of sleep modes

The Cortex-Mx microcontroller architecture is designed for hard real-time systems, but can be used for complex System-on-Chip applications as well.

Classic and Extended 8051 Devices

8051 devices combine cost-efficient hardware with a simple but efficient programming model that uses various memory regions to maximize code efficiency and speed-up memory access. The following figure shows the memory layout of a classic 8051 device.



The 8051 architecture provides three different physical memory regions:

- **DATA/IDATA** memory includes a 256 Bytes on-chip RAM with register banks and bit-addressable space that is used for fast variable accessing. Some devices provide an extended data (**EDATA**) space with up to 64KB.
- **CODE** memory consists of 64KB ROM space used for program code and constants. The Keil linker supports code banking that allows you to expand the physical memory space. In extended variants, up to 16MB ROM space is available.
- **XDATA** memory has a 64KB RAM space for off-chip peripheral and memory addressing. Today, most devices provide some on-chip RAM that is mapped into **XDATA**.

- **SFR** and **IDATA** memory are located in the same address space but are accessed through different assembler instructions
- For extended devices, the memory layout provides a universal memory map that includes all 8051-memory types in a single 16MByte address region

8051 Highlights

- Fast interrupt service routines with two or four priority levels and up to 32-vectored interrupts
- Four register banks for minimum interrupt prolog/epilog
- Bit-addressable space for efficient logical operations
- 128 Bytes of Special Function Register (SFR) space for tight integration of on-chip peripherals. Some devices extend the SFR space using paging.
- Low-power, high-speed devices up to 100 MIPS are available

8051 Development Tool Support

The Keil C51 Compiler and the Keil Linker/Locator provide optimum 8051 architecture support with the following features and C language extensions.

- Interrupt functions with register bank support are written directly in C
- Bit and bit-addressable variables for optimal Boolean data type support
- Compile-time stack with data overlaying uses direct memory access and gives high-speed code with little overhead compared to assembly programming
- Reentrant functions for usage by multiple interrupt or task threats
- Generic and memory-specific pointers provide flexible memory access
- Linker Code Packing gives utmost code density by reusing identical program sequences
- Code and Variable Banking expand the physical memory address space
- Absolute Variable Locating enables peripheral access and memory sharing

8051 Memory Types

A memory type prefix is used to assign a memory type to an expression with a constant. This is necessary, for example, when an expression is used as an address for the output command. Normally, symbolic names have an assigned memory type, so that the specification of the memory type can be omitted. The following memory types are defined:

Prefix	Memory Space
C:	Code Memory (CODE)
D:	Internal, direct-addressable RAM memory (DATA)
I:	Internal, indirect-addressable RAM memory (IDATA)
X:	External RAM memory (XDATA)
B:	Bit-addressable RAM memory
P:	Peripheral memory (VTREGD – 80x51 pins)

The prefix **P:** is a special case, since it always must be followed by a name. The name in turn is searched for in a special symbol table that contains the register's pin names.

Example:

C:0x100	Address 0x100 in CODE memory
ACC	Address 0xE0 in DATA memory, D:
I:100	Address 0x64 in internal RAM
X:0FFFFH	Address 0xFFFF in external data memory
B:0x7F	Bit address 127 or 2FH.7
C	Address 0xD7 (PSW.7), memory type B:

Infineon C166, XE166, XC2000

The 16-bit architecture of these devices is designed for high-speed real-time applications. It provides up to 16MB memory space with fast memory areas mapped into parts of the address space. High-performance applications benefit from locating frequently used variables into the fast memory areas. The below listed memory types address the following memory regions:

Memory Type	Description
bdata	Bit-addressable part of the **idata** memory.
huge	Complete 16MB memory with fast 16-bit address calculation. Object size limited to 64KB.
idata	High speed RAM providing maximum access speed (part of **sdata**).
near	Efficient variable and constant addressing (max. 64KB) with 16-bit pointer and 16-bit address calculation.
sdata	System area includes Peripheral Registers and additional on-chip RAM space.
xhuge	Complete 16MB memory with full address calculation for unlimited object size.

C166, XE166, XC2000 Highlights

- Highest-speed interrupt handling with 16 priority levels and up to 128 vectored interrupts
- Unlimited register banks for minimum interrupt prolog/epilog
- Bit instructions and bit-addressable space for efficient logical operations
- ATOMIC instruction sequences are protected from interrupts without interrupt enable/disable sequences
- Peripheral Event Controller (PEC) for automatic memory transfers triggered by peripheral interrupts. Requires no processor interaction and further improves interrupt response time.
- Multiply-Accumulate Unit (MAC) provided for high-speed DSP algorithms

C166, XE166, XC2000 Development Tool Support

The Keil C166 Compiler supports all C166, XE166, XC2000 specific features and provides additional extensions such as:

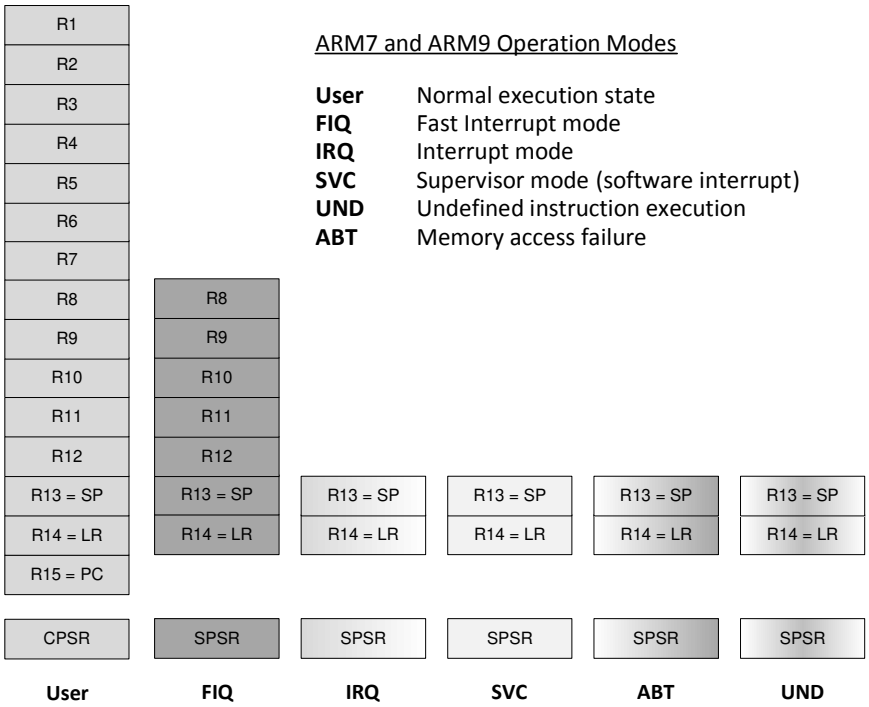
- Memory type support and flexible digital pattern processing for extremely fast variable access
- Function inlining eliminating call/return overhead
- Inline assembly for accessing all microcontroller and MAC instructions

ARM7 and ARM9 based Microcontrollers

The ARM7 and ARM9 based microcontrollers run on a load-store RISC architecture with 32-bit registers and fixed op-code length. The architecture provides a linear 4GB memory address space. In contrast to the previously mentioned 8/16-bit devices, no specific memory types are provided, since memory addressing is performed via 32-bit pointers in microcontroller registers. Peripheral registers are mapped directly into the linear address space. The Thumb instruction set improves code density by providing a compressed 16-bit instruction subset.

The ARM7 and ARM9 cores are easy to use, cost-effective, and support modern object-oriented programming techniques. They include a 2-level interrupt system with a normal interrupt (IRQ) and a fast interrupt (FIQ) vector. To minimize interrupt overhead, typical ARM7/ARM9 microcontrollers provide a vectored interrupt controller. The microcontroller operating modes, separate stack spaces, and Software Interrupt (SVC) features produce efficient use of Real-Time Operating Systems.

The ARM7 and ARM9 core provides thirteen general-purpose registers (R0–R12), the stack pointer (SP) R13, the link register (LR) R14, which holds return addresses on function calls, the program counter (PC) R15, and a program status register (PSR). Shadow registers, available in various operating modes, are similar to register banks and reduce interrupt latency.



ARM7 and ARM9 Highlights

- **Linear 4 GB memory space** that includes peripherals and eliminates the need for specific memory types
- **Load-store architecture with efficient pointer addressing**. Fast task context switch times are achieved with multiple register load/store.
- **Standard (IRQ) and Fast (FIQ) interrupt**. Banked microcontroller registers on FIQ reduce register save/restore overhead.
- **Vectored Interrupt Controller** (available in most microcontrollers) optimizes multiple interrupt handling
- **Processor modes** with separate interrupt stacks for predictable stack requirements
- **Compact 16-bit Instruction Set (Thumb)**. Compared to ARM mode, Thumb mode code is about 65% of the code size and 160% faster when executing from a 16-bit memory system.

ARM7 and ARM9 Development Tool Support

The ARM compilation tools support all ARM-specific features and provide:

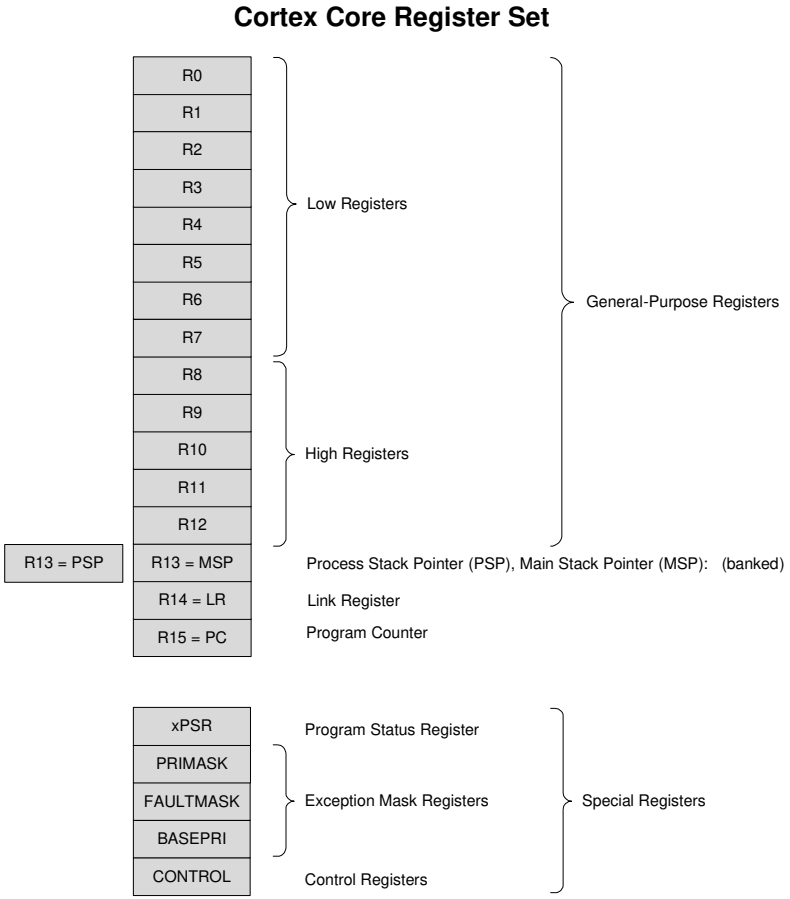
- **Function Inlining** eliminates call/return overhead and optimizes parameter passing
- **Inline assembly** supports special ARM/Thumb instructions in C/C++ programs
- **RAM functions** enable high-speed interrupt code and In-System Flash programming
- **ARM/Thumb interworking** provides outstanding code density and microcontroller performance
- **Task function and RTOS support** are built into the C/C++ compiler

Cortex-Mx based Microcontrollers

Designed for the 32-bit microcontroller market, the Cortex-Mx microcontrollers combine excellent performance at low gate count with features only previously found in high-end processors.

With 4GB of linear, unified memory space, the Cortex-Mx processors provide bit-banding features and supports big and little endian configuration. Predefined memory types are available, while some memory regions have additional attributes. Code can be located in the SRAM, external RAM, but preferably in the Code region. Peripheral registers are mapped into the memory space. Code density is improved by the Thumb or Thumb2 instruction set, depending on the processor version.

General-purpose registers rank from R0 to R12. R13 (SP) is banked, with only one copy of the R13 (MSP, PSP) being visible at a time. Special registers are available, but are not used for normal data processing. Some of the 16-bit Thumb instructions can access R0-R7 (low) registers only. There is no FIQ; however, nested interrupts and interrupt priority handling is implemented via the Nested Vector Interrupt Controller (NVIC), greatly reducing interrupt latency.



Cortex-Mx Highlights

- **Nested Vectored Interrupt Controller** optimizes multiple external interrupts (up to 240 + 1 NMI, with at least eight priority levels)
- **R0-R3**, **R12**, **LR**, **PSR**, **and PC are pushed automatically** to the stack at interrupt entry and popped back at interrupt exit points
- **Only one instruction set (Thumb2)**, assuring software upward compatibility with the entire ARM roadmap
- **Several Extreme Low-Power Modes** with an attached Wake-Up Interrupt Controller (WIC)