



ARM7TDMI and ARM7TDMI-S

ARM 32-Bit RISC Core with 16-Bit System Costs

The ARM7TDMI core is the industry's most widely used 32-bit embedded RISC microprocessor. The ARM7TDMI-S is a synthesizable version of the ARM7TDMI. Optimized for cost- and power-sensitive applications, the ARM7TDMI solution provides the low power consumption, small size, and high performance needed in portable, embedded applications.

The ARM7TDMI core is a 32-bit embedded RISC processor delivered as a hard macrocell optimized to provide the best

combination of performance, power, and area characteristics. The ARM7TDMI core enables system designers to build embedded devices requiring small size, low power, and high performance.

The ARM7 family also includes the ARM7TDMI processor, the ARM7TDMI-S processor, the ARM720T processor, and the ARM7EJ-S processor, each of which has been developed to address different market requirements.

RISC Advantages ▶

The ARM architecture is based on the Reduced Instruction Set Computer (RISC) principles. The RISC instruction set and related decode mechanism are much simpler than those of the Complex Instruction Set Computer (CISC) designs. This simplicity has the following advantages:

- A high instruction throughput
- An excellent real-time interrupt response
- A small, cost-effective, processor macrocell

The Instruction Pipeline ▶

The ARM7TDMI core uses a pipeline to increase the speed of the flow of instructions to the processor. This allows several operations to take place simultaneously.

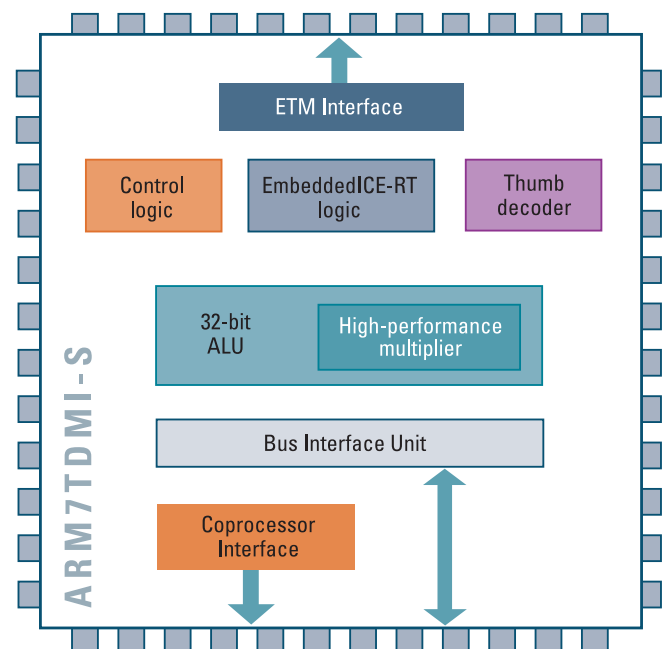
A three-stage pipeline is used, so instructions are executed in three stages:

- Fetch (the instruction is fetched from memory)
- Decode (decoding of registers used in the instruction)
- Execute (register/s read from register bank; shift and ALU operations; write register/s back to register bank)

During normal operation, while one instruction is being executed, its successor is being decoded and a third instruction is being fetched from memory.

Memory Access ▶

The ARM7TDMI core has a Von Neumann architecture with a single 32-bit data bus carrying both instructions and data. Only load, store, and swap instructions can access data from memory. This simplifies the internal logic of the processor memory interface using less die area.



Memory Interface ▶

The ARM7TDMI processor memory interface has been designed to allow performance potential to be realized while minimizing the use of memory. Speed-critical control signals are pipelined to allow system control functions to be implemented in standard low-power logic. These control signals facilitate the exploitation of fast-burst access modes supported by many on-chip and off-chip memory technologies.

EmbeddedICE-RT Logic ▶

The EmbeddedICE-RT logic provides integrated on-chip debug support for the ARM7TDMI core. You use the EmbeddedICE-RT logic to program the conditions under which a breakpoint or watchpoint can occur.

The EmbeddedICE-RT logic contains a Debug Communications Channel (DCC), which is used to pass information between the target and the host debugger. The EmbeddedICE-RT logic is controlled through the Joint Test Action Group (JTAG) test access port.

Architecture ▶

The ARM7TDMI processor has two instruction sets:

- The 32-bit ARM instruction set
- The 16-bit Thumb® instruction set

Having both 32-bit ARM instructions and 16-bit Thumb instructions gives the ARM7TDMI processor two advantages: instruction compression and higher performance over typical 16-bit architectures.

Microprocessor architectures traditionally have the same width for instructions and data. In comparison with 16-bit architectures, 32-bit architectures exhibit higher performance when manipulating 32-bit data and can access a large address space much more efficiently.

Typically, 16-bit architectures have higher code density than 32-bit architectures, but they have approximately half the performance.

The Thumb instructions implement a 16-bit instruction set on a 32-bit architecture to provide:

- Higher performance than a 16-bit architecture
- Higher code density than a 32-bit architecture

The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Thumb instructions are each 16 bits long and have a corresponding 32-bit ARM instruction. This has the same effect on the processor model. Thumb instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and Thumb states.

On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bit ARM instructions in real time without performance loss.

Applications ▶

- Industrial
- Automotive
- Personal audio (MP3, WMA, and AAC players)

Features ▶

- 32-/16-bit RISC architecture (ARM v4T)
- 32-bit ARM instruction set for maximum performance and flexibility
- 16-bit Thumb instruction set for increased code density
- Unified bus interface; 32-bit data bus carries both instructions and data
- Three-stage pipeline
- 32-bit ALU
- Very small die size and low power consumption
- Fully static operation
- Coprocessor interface
- Extensive debug facilities:
 - EmbeddedICE-RT real-time debug unit
 - JTAG interface unit
 - Interface for direct connection to Embedded Trace Macrocell (ETM)

Benefits ▶

- Generic layout can be ported to specific process technologies
- ARM and Thumb instruction sets can be mixed with minimal overhead to support application requirements for speed and code density
- Small die size reduces overall SoC area, cost, and power consumption
- EmbeddedICE-RT and optional ETM units enable extensive, real-time debug facilities

Performance Characteristics ▶			
	0.18 μm	0.13 μm	0.090 μm
	Speed Optimized	Speed Optimized	Speed Optimized
Frequency* (MHz)	115	133	236
Area (mm ²)	0.59	0.26	0.18
Power** (mW/MHz)	0.21	0.06	-

*Worst-case conditions—0.18 μm process—1.62V, 125°C, slow silicon; 0.13 μm process—1.08V, 125°C, slow silicon; 90 nm process—0.9V, 125°C, slow silicon

**Typical-case conditions—0.18 μm process—1.8V, 25°C, typical silicon; 0.13 μm process—1.2V, 25°C, typical silicon; 90 nm process—1V, 25°C, typical silicon