

# ARM<sup>®</sup> DS-5<sup>™</sup> Glossary

**Version 5.2**

# ARM® DS-5™ Glossary

Copyright © 2010 ARM. All rights reserved.

## Release Information

The following changes have been made to this book.

**Table 1 Change History**

Date	Issue	Confidentiality	Change
June 2010	A	Non-Confidential	First release for DS-5
September 2010	B	Non-Confidential	Update for DS-5

## Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM® in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”.

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

## Product Status

The information in this document is final, that is for a developed product.

## Web Address

<http://www.arm.com>

## 1 About this Glossary

This Glossary describes the terminology used in the documentation provided with ARM® DS-5™.

See *Glossary of terms used in DS-5 documentation* on page 1-4.

## Glossary of terms used in DS-5 documentation

### A

**AAPCS** See Procedure Call Standard for the ARM Architecture (AAPCS).

**ABI for the ARM Architecture (base standard) (BSABI)**

The ABI for the ARM® Architecture is a collection of specifications, some open and some specific to ARM architecture, that regulate the inter-operation of binary code in a range of ARM architecture-based execution environments. The base standard specifies those aspects of code generation that must be standardized to support inter-operation and is aimed at authors and vendors of C and C++ compilers, linkers, and runtime libraries.

**Adaptive clocking** A technique used by DSTREAM™ and RealView® ICE where it sends out a clock signal and then waits for the returned clock before generating the next clock pulse. The technique enables the DSTREAM and RealView ICE run control unit to adapt to differing signal drive capabilities and differing cable lengths.

*See also* DSTREAM and RealView ICE.

**Advanced eXtensible Interface (AXI)**

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

**Advanced High-performance Bus (AHB)**

A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM Limited recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

*See also* Advanced Microcontroller Bus Architecture (AMBA) and AHB-Lite.

**Advanced Microcontroller Bus Architecture (AMBA)**

The *Advanced Microcontroller Bus Architecture* (AMBA®) is an on-chip communications standard for high-performance 32-bit and 16-bit embedded microcontrollers.

**Advanced Trace Bus (ATB)**

The Advanced Trace Bus transfers trace data through CoreSight™ infrastructure in a SoC. Trace sources are ATB masters, and sinks are ATB slaves. Link components provide both master and slave interfaces.

*See also* CoreSight.

**AHB** See Advanced High-performance Bus (AHB).

**AHB Access Port (AHB-AP)**

CoreSight supports access to a system bus infrastructure using the *AHB Access Port* (AHB-AP) in the *Debug Access Port* (DAP). The AHB-AP provides an AHB master port for direct access to system memory. If an alternate bus protocol is implemented, you can use an AHB bridge to map transactions. For example, you can use an AHB to AXI bridge to enable access to an AXI bus matrix.

CoreSight also supports AHB bus tracing using an *AHB Trace Macrocell* (HTM).

*See also* Advanced eXtensible Interface (AXI), AHB Trace Macrocell (HTM), CoreSight, and Debug Access Port (DAP).

**AHB-AP** *See* AHB Access Port (AHB-AP).

**AHB-Lite** A subset of the full AMBA AHB protocol specification. It provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.

#### **AHB Trace Macrocell (HTM)**

The *AHB Trace Macrocell* (HTM) is a trace source that makes bus information visible that cannot be inferred from the processor trace using an ETM™:

- An understanding of multi-layer bus utilization.
- Software debug. For example, visibility of access to memory areas and data accesses.
- Bus event detection for trace trigger or filters, and for bus profiling.

*See also* Advanced High-performance Bus (AHB).

**AMBA** *See* Advanced Microcontroller Bus Architecture (AMBA).

**APB-AP** *See* Debug Access Port (DAP).

**armar** The ARM librarian, that enables you to create libraries of files, such as object files.

*See also* ARM Compiler toolchain.

**armasm** The ARM assembler.

*See also* ARM Compiler toolchain.

**armcc** The ARM compiler for C and C++ code. It also includes the NEON™ vectorizing compiler.

*See also* ARM Compiler toolchain.

**armlink** The ARM linker.

*See also* ARM Compiler toolchain.

#### **ARM Advanced SIMD Extension**

ARM Advanced SIMD Extension is an optional component of ARMv7 architecture. It is a 64/128 bit hybrid SIMD technology targeted at advanced media and signal processing applications and embedded processors. It is implemented as part of the ARM core, but has its own execution pipelines and a register bank that is distinct from the ARM core register bank.

ARM Advanced SIMD Extension supports integer, fixed-point, and single-precision floating-point SIMD operations. These instructions are available in both ARM and Thumb®-2.

ARM Advanced SIMD Extension is also known as *ARM NEON Technology* (NEON).

#### **ARM Compiler toolchain**

ARM Compiler toolchain is the only compiler co-developed with the ARM processors and specifically designed to optimally support the ARM architecture. ARM Compiler toolchain supersedes RealView Compilation Tools.

*See also* armar, armasm, armcc, armlink, fromelf.

<b>ARM DS-5</b>	<p>The suite of software development tools, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of processors. DS-5 supersedes RealView Development Suite.</p> <p><i>See also</i> RealView Development Suite (RVDS), DS-5 Debugger, RealView ICE, DSTREAM, Eclipse for DS-5, ARM Streamline Performance Analyzer.</p>
<b>ARM instruction</b>	<p>A word that encodes an operation for an ARM processor operating in ARM state. ARM instructions must be word-aligned.</p> <p><i>See also</i> Thumb instruction, Thumb-2 instruction, <i>and</i> Thumb-2EE instruction.</p>
<b>ARM state</b>	<p>A processor that is executing ARM instructions is operating in ARM state. The processor switches to Thumb state (and to recognizing Thumb instructions) when directed to do so by a state-changing instruction such as BX or BLX.</p> <p><i>See also</i> Jazelle state, Thumb state, <i>and</i> ThumbEE state.</p>
<b>ARM Streamline Performance Analyzer</b>	<p>ARM Streamline™ is a graphical performance analysis tool. Combining a kernel driver, target daemon, and an Eclipse-based user interface, it transforms sampling data and system trace into reports that present the data in both visual and statistical forms. ARM Streamline uses hardware performance counters with kernel metrics to provide an accurate representation of system resources. ARM Streamline is provided with DS-5.</p>
<b>ARM TrustZone® technology</b>	<p>The hardware and software that enables security features to be integrated throughout a SoC device.</p>
<b>ATB</b>	<p><i>See</i> Advanced Trace Bus (ATB).</p>
<b>AXI</b>	<p><i>See</i> Advanced eXtensible Interface (AXI).</p>
<b>B</b>	
<b>Bare metal</b>	<p>A hardware platform considered independently of any software. Writing code for bare metal involves directly accessing all the required hardware features, without recourse to a hardware abstraction layer such as an operating system.</p>
<b>Base Platform Application Binary Interface (BPABI)</b>	<p>The <i>Base Platform Application Binary Interface</i> (BPABI) is the base standard for the interface between executable files, such as dynamic shared objects and DLLs, and the systems that execute them.</p> <p><i>See also</i> Base Platform ABI for the ARM® Architecture.</p>
<b>BCD file</b>	<p><i>See</i> Board/Chip Definition (BCD) file.</p>
<b>Big-endian</b>	<p>In the context of the ARM architecture, big-endian is defined as the memory organization in which the least significant byte of a word is at a higher address than the most significant byte.</p> <p><i>See also</i> Little-endian.</p>
<b>Board file</b>	<p>ARM debuggers uses this term to refer to the top-level configuration file that references one or more other configuration files. A board file contains:</p> <ul style="list-style-type: none"> <li>• the Debug Configuration (connection-level) settings</li> <li>• references to the Debug Interface configuration file that identifies the targets on the development platform</li> </ul>

- references to any *Board/Chip Definition* (BCD) files assigned to a Debug Configuration.

*See also* Board/Chip Definition (BCD) file, Debug Configuration, Debug Interface, Development platform, *and* Target.

#### **Board/Chip Definition (BCD) file**

In the context of RealView Debugger, a BCD file enables you to define the memory map and memory mapped registers for a target development board or processor.

Eclipse for DS-5 enables you to import BCD files into the Target Configuration Editor.

*See also* Board file, Debug Configuration, *and* Eclipse for DS-5.

#### **BPABI**

*See* Base Platform Application Binary Interface (BPABI).

#### **Breakpoint unit**

In the context of an ARM debugger, a unit within a Chained breakpoint that combines with other breakpoint units to create a complex hardware breakpoint.

*See also* Chained breakpoint *and* Hardware breakpoint.

#### **BSABI**

*See* ABI for the ARM Architecture (base standard) (BSABI).

### **C**

#### **Canonical Frame Address (CFA)**

In DWARF, this is an address on the stack specifying where the call frame of an interrupted function is located.

#### **Captive thread**

Captive threads are all threads that can be brought under the control of RVDS. Special threads, called non-captive threads, are essential to the operation of *Running System Debug* (RSD) and so are not under debugger control. Non-captive threads are grayed out in the GUI.

*See also* Running System Debug (RSD).

#### **CFA**

*See* Canonical Frame Address (CFA).

#### **Chained breakpoint**

In the context of an ARM debugger, a complex breakpoint that comprises multiple hardware breakpoint units.

*See also* Breakpoint unit, Chained breakpoint, Data breakpoint, *and* Hardware breakpoint.

#### **CMM**

In the context of an ARM debugger, an existing scripting language provided for compatibility with other debuggers.

If you are writing new scripts it is recommended that you use the GDB scripting commands because they offer more functionality in an ARM debugger.

*See also* DS-5 Debugger.

#### **Conditional breakpoint**

A breakpoint that has one or more condition qualifiers assigned. The breakpoint is activated when all assigned conditions are met, and either stops or continues execution depending on the action qualifiers that are assigned. The condition normally references the values of program variables that are in scope at the breakpoint location.

*See also* Chained breakpoint, Data breakpoint, Hardware breakpoint, Instruction breakpoint, Software breakpoint, *and* Unconditional breakpoint.

#### **Core module**

In the context of an ARM Integrator development board, an add-on development board that contains an ARM architecture-based processor and local memory. Core modules can run standalone, or can be stacked onto Integrator development boards.

*See also* Integrator.

<b>CoreSight</b>	<p>CoreSight is an infrastructure that enables the debugging, monitoring, and optimization of performance of a complete <i>System on Chip</i> (SoC) design.</p> <p><i>See also</i> CoreSight ECT, CoreSight ETB, CoreSight ETM, Trace Funnel, <i>and</i> Trace Port Interface Unit (TPIU).</p>
<b>CoreSight ECT</b>	<p>CoreSight ECT is a control and access component that supports the interaction and synchronization of multiple triggering events within a SoC:</p> <p><i>See also</i> CoreSight, Cross Trigger Interface (CTI), Cross Trigger Matrix (CTM), <i>and</i> Embedded Cross Trigger (ECT).</p>
<b>CoreSight ETB</b>	<p>CoreSight ETB is a trace sink that provides on-chip storage of trace data using a configurable sized RAM.</p> <p><i>See also</i> CoreSight, CoreSight ETB, Embedded Trace Buffer (ETB), <i>and</i> Embedded Trace Macrocell (ETM).</p>
<b>CoreSight ETM</b>	<p>CoreSight ETM is a trace source that provides processor driven trace through an ATB compliant trace port.</p> <p><i>See also</i> Advanced Trace Bus (ATB), CoreSight, CoreSight ETB, <i>and</i> Embedded Trace Macrocell (ETM).</p>
<b>CPSR</b>	<i>See</i> Current Program Status Register (CPSR).
<b>Cross Trigger Interface (CTI)</b>	<p>The Cross Trigger Interface provides the interface between a component or subsystem and the Cross Trigger Matrix. The system requires a CTI for each subsystem that supports cross triggering.</p> <p><i>See also</i> CoreSight, CoreSight ECT, Cross Trigger Matrix (CTM), <i>and</i> Embedded Cross Trigger (ECT).</p>
<b>Cross Trigger Matrix (CTM)</b>	<p>The Cross Trigger Matrix combines the trigger requests generated from CTIs and broadcasts them to all CTIs as channel triggers. This enables subsystems to interact, cross trigger, with one another. CTMs can be connected together to increase the number of CTIs</p> <p><i>See also</i> CoreSight, CoreSight ECT, Cross Trigger Interface (CTI), <i>and</i> Embedded Cross Trigger (ECT).</p>
<b>CTI</b>	<i>See</i> Cross Trigger Interface (CTI).
<b>CTM</b>	<i>See</i> Cross Trigger Matrix (CTM).
<b>Current Program Status Register (CPSR)</b>	<p>A register containing the current state of control bits and flags.</p> <p><i>See also</i> Program Status Register (PSR) <i>and</i> Saved Program Status Register (SPSR).</p>
<b>D</b>	
<b>DAP</b>	<i>See</i> Debug Access Port (DAP).
<b>Data breakpoint</b>	<p>A hardware breakpoint that activates when a given location is accessed in a specific way. The breakpoint can also check for a specific data value being access at the given location, if required.</p> <p><i>See also</i> Chained breakpoint, Conditional breakpoint, Hardware breakpoint, Instruction breakpoint, Software breakpoint, <i>and</i> Unconditional breakpoint.</p>
<b>DCC</b>	<i>See</i> Debug Communications Channel (DCC).



**Debug Agent (DA)** The Debug Agent resides on the target to provide target-side support for *Running System Debug* (RSD) in RealView Debugger. The Debug Agent can be a thread or built into the RTOS. The Debug Agent and RealView Debugger communicate with each other using the *Debug Communications Channel* (DCC). This enables data to be passed between the debugger and the target using the ICE interface, without stopping the program or entering debug state.

*See also* Debug Communications Channel (DCC) and Running System Debug (RSD).

**Debug Access Port (DAP)**

The Debug Access Port is a control and access component that enables debug access to the complete SoC through system master ports.

External read/write access to the internal interface is provided by the *JTAG Debug Port* (JTAG-DP). The JTAG-DP is a standard JTAG interface for debug access and provides standard JTAG access to an SoC through the DAP. It interfaces to the DAP internal bus.

Internal access to on-chip busses and other interfaces is provided by the *Access Ports* (APs). The three APs are:

- the *AHB Access Port* (AHB-AP) that provides an AHB-Lite master for access to a system AHB bus
- the *APB Access Port* (APB-AP) that provides an AMBA 3™ APB master for access to the Debug APB that configures all CoreSight components
- the *JTAG Access Port* (JTAG-AP) that provides JTAG access to on-chip components and operates as a JTAG master port to drive JTAG chains throughout the SoC.

*See also* CoreSight.

**Debug Communications Channel (DCC)**

A debug communications channel enables data to be passed between an ARM debugger and the EmbeddedICE logic on the target using the JTAG interface, without stopping the program flow or entering debug state.

**Debug Configuration**

In the context of an ARM debugger, a Debug Configuration defines a debugging environment for the development platform that is accessed through a particular Debug Interface. Multiple Debug Configurations can be created for a Debug Interface, each providing a separate debugging environment to different development platforms, or different debugging environments to the same development platform.

*See also* Board file, Board/Chip Definition (BCD) file, Debug Interface and Target.

**Debug illusion**

The experience that a debugger creates in the mind of the software developer. The key features of debug illusion include:

- mixed source code and disassembly
- a function call stack showing symbolic function prototypes with names and argument types
- display of variables using their source code name
- source level stepping and breakpoints.

This illusion is created by the debugger using data from the system being debugged and symbolic debug information from the code generation tool chain.

<b>Debug Interface</b>	<p>In the context of an ARM debugger, the Debug Interface identifies the targets on your development platform, and provides the mechanism that enables the debugger to communicate with those targets. The Debug Interface corresponds directly to a piece of hardware or a software simulator.</p> <p><i>See also</i> Debug Configuration, Simulator, <i>and</i> Target.</p>
<b>Development platform</b>	<p>Contains the components, either hardware or simulated, that you are using to develop your application. It can include:</p> <ul style="list-style-type: none"> <li>• a development board, such as an Integrator/CP</li> <li>• peripherals</li> <li>• one or more ARM architecture-based processors</li> <li>• CoreSight components</li> <li>• one or more DSPs.</li> </ul> <p><i>See also</i> CoreSight, Simulator, <i>and</i> Target.</p>
<b>DS-5</b>	<p><i>See</i> ARM DS-5.</p>
<b>Device</b>	<p>In the context of an ARM debugger, a component on a target containing the application that you want to debug.</p> <p><i>See also</i> Target.</p>
<b>Doubleword</b>	<p>In the context of the ARM architecture, a 64-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.</p>
<b>DS-5 Debugger</b>	<p>An ARM software development tool that enables you to make use of a debug agent in order to examine and control the execution of software running on a debug target. It is fully integrated into Eclipse for DS-5.</p> <p><i>See also</i> Eclipse for DS-5.</p>
<b>DSTREAM</b>	<p>A combined debug and trace unit that enables you to:</p> <ul style="list-style-type: none"> <li>• connect a software debugger to an ARM processor-based target using a hardware interface such as JTAG or <i>Serial Wire Debug</i> (SWD)</li> <li>• collect trace from a target device for non-intrusive debug and code optimization</li> <li>• collect profiling data using ARM Streamline to provide real-time analysis of embedded software.</li> </ul> <p><i>See also</i> JTAG interface unit, RealView ICE, RealView Trace, RealView Trace 2, <i>and</i> Serial Wire Debug (SWD).</p>
<b>E</b>	
<b>Eclipse for DS-5</b>	<p>Eclipse for DS-5 is based around the Eclipse IDE, and provides additional features to support the ARM development tools provided in DS-5.</p> <p><i>See also</i> DS-5.</p>
<b>ECT</b>	<p><i>See</i> Embedded Cross Trigger (ECT).</p>
<b>Embedded assembler</b>	<p>Embedded assembler is assembler code that is included in a C or C++ file, and is separate from other C or C++ functions.</p>

**Embedded Cross Trigger (ECT)**

The Embedded Cross Trigger provides a standard interconnect mechanism to pass debug or profiling events around the SoC. It comprises:

- *Cross Trigger Interface (CTI)*
- *Cross Trigger Matrix (CTM).*

*See also* CoreSight and CoreSight ECT.

**Embedded Trace Buffer (ETB)**

The Embedded Trace Buffer™ (ETB™) provides logic inside the core that extends the information capture functionality of the Embedded Trace Macrocell™.

*See also* CoreSight ETB and Embedded Trace Buffer (ETB).

**Embedded Trace Macrocell (ETM)**

The Embedded Trace Macrocell™ (ETM) is a block of logic, embedded in the hardware, that is connected to the address, data, and status signals of the processor. It broadcasts branch addresses, and data and status information in a compressed protocol through the trace port. It contains the resources used to trigger and filter the trace output.

*See also* CoreSight ETM and Embedded Trace Macrocell (ETM).

**EmbeddedICE logic**

The EmbeddedICE® logic is an on-chip logic block that provides TAP-based debug support for ARM architecture-based processors. It is accessed through the TAP controller on the ARM architecture-based processor using the JTAG interface.

*See also* IEEE 1149.1.

**Emulator**

In the context of target connection hardware, an emulator provides an interface to the pins of a real core (emulating the pins to the external world) and enables you to control or manipulate signals on those pins.

**Enhanced Program Status Register (EPSR)**

An Enhanced Program Status Register (EPSR) is a Program Status Register that contains an additional bit (the Q bit, signifying saturation) used by some ARM architecture-based processors, including the ARM9E™.

*See also* Current Program Status Register (CPSR), Program Status Register (PSR), and Saved Program Status Register (SPSR).

**ETB**

*See* Embedded Trace Buffer (ETB).

**ETM**

*See* Embedded Trace Macrocell (ETM).

**ETV**

*See* Extended Target Visibility (ETV).

**Execution vehicle**

Part of the debug target interface, execution vehicles process requests from the client tools to the target.

*See also* Debug Interface.

**Execution view**

The address of regions and sections after the image has been loaded into memory and started execution.

**Extended Target Visibility (ETV)**

Extended Target Visibility enables a debugger to access features of the underlying target, such as chip-level information provided by the hardware manufacturer or SoC designer.

## F

### Fast Models from ARM

Fast Models from ARM are instruction-accurate models that enable you to perform early software development on ARM architecture-based systems.

### FIQ

Fast Interrupt.

### fromelf

The ARM image conversion utility. This accepts ELF format input files and converts them to a variety of output formats. `fromelf` can also generate text information about the input image, such as code and data size.

*See also* ARM Compiler toolchain.

## H

### Halfword

In the context of the ARM architecture, defined as a 16-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.

### Halted System Debug (HSD)

Usually used for OS aware debugging, *Halted System Debug* (HSD) means that a target can only be debugged when it is not running. Any target must be stopped before carrying out any analysis of the system. With the target stopped, RealView Debugger presents OS awareness information by reading and interpreting target memory.

*See also* Running System Debug (RSD).

### Hardware breakpoint

A breakpoint that is implemented using non-intrusive additional hardware. Hardware breakpoints are the only method of halting execution when the location is in *Read Only Memory* (ROM) or Flash. Using a hardware breakpoint often results in the processor halting completely. This is usually undesirable for a real-time system.

*See also* Chained breakpoint, Conditional breakpoint, Data breakpoint, Instruction breakpoint, Software breakpoint, *and* Unconditional breakpoint.

### Hint instruction

A hint instruction provides information to the hardware that the hardware can take advantage of. An implementation can choose whether to implement hint instructions or not. If they are not implemented, they execute as NOP.

### HSD

*See* Halted System Debug (HSD).

### HTM

*See* AHB Trace Macrocell (HTM).

## I

### ICE Extension Unit

A hardware extension to the EmbeddedICE logic that provides more breakpoint units.

### IEEE 1149.1

The IEEE Standard that defines TAP. Commonly (but incorrectly) referred to as JTAG.

*See IEEE Std 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture* specification available from the IEEE Standards Association, <http://standards.ieee.org>.

*See also* Joint Test Action Group (JTAG), and Test Access Port (TAP).

### Immediate values

Values that are encoded directly in the instruction and used as numeric data when the instruction is executed. Many ARM and Thumb instructions enable small numeric values to be encoded as immediate values within the instruction that operates on them.

**Implementation defined**

In the context of the ARM architecture, this means that the behavior is not architecturally defined, but must be defined and documented by individual implementations.

**In-Circuit Emulator**

A device enabling access to and modification of the signals of a circuit while that circuit is operating.

**Input section**

Contains code or initialized data or describes a fragment of memory that must be set to zero before the application starts.

*See also* Output section and Region.

**Instruction breakpoint**

A location in the image containing an instruction that, if executed, activates a breakpoint. The breakpoint activation can be delayed by assigning condition qualifiers, and subsequent execution of the image is determined by any actions assigned to the breakpoint.

*See also* Conditional breakpoint, Data breakpoint, Hardware breakpoint, Software breakpoint, and Unconditional breakpoint.

**Instruction Register (IR)**

When referring to a TAP controller, a register that controls the operation of the TAP.

**Integrator**

A range of ARM hardware development platforms. Core modules are available that contain the processor and local memory.

*See also* Core module.

**Interworking**

A method of working that enables branches between ARM and Thumb code.

**IRQ**

Interrupt Request.

**IT block**

A block of up to four instructions following the 16-bit Thumb-2 *If-Then* (IT) instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others.

**J****Jazelle®**

The Jazelle architecture extends the existing ARM architecture to enable direct execution of selected *Java Virtual Machine* (JVM) opcodes.

**Jazelle state**

A processor that is executing Jazelle bytecode instructions is operating in Jazelle state.

*See also* ARM state, Thumb state, and ThumbEE state.

**JTAG**

*See* Joint Test Action Group (JTAG).

**JTAG-AP**

*See* Debug Access Port (DAP).

**JTAG-DP**

*See* Debug Access Port (DAP).

**JTAG interface unit**

In the context of the ARM tools, a protocol converter that converts low-level commands from ARM debuggers into JTAG signals to the processor, for example to the EmbeddedICE logic and the ETM.

*See also* DS-5 Debugger, Embedded Trace Macrocell (ETM), and EmbeddedICE logic.

**Joint Test Action Group (JTAG)**

An IEEE group focussed on silicon chip testing methods. Many debug and programming tools use a *Joint Test Action Group* (JTAG) interface port to communicate with processors.

*See* IEEE Std 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture specification available from the IEEE Standards Association, <http://standards.ieee.org>.

## L

- Little-endian** In the context of the ARM architecture, little-endian is defined as the memory organization in which the most significant byte of a word is at a higher address than the least significant byte.
- See also* Big-endian.
- Load view** The address of regions and sections when the image has been loaded into memory but has not yet started execution.

## M

- Memory hint** In the context of the ARM architecture, a memory hint instruction enables a programmer to provide advance information to memory systems about future memory accesses, without actually loading or storing any data.
- MPCore™** An integrated *Symmetric Multiprocessor System* (SMP) delivered as a traditional uniprocessor core. The chip contains up to four ARM1136J-S™ based CPUs with cache coherency.
- MPU** Memory Protection Unit.

## N

- NEON™** *See* ARM Advanced SIMD Extension.
- Normal and Secure Worlds** Effectively two virtual processors on a single physical processor. The Normal World processes operations that are not security-critical, and it delegates security-critical operations to the Secure World. Client applications reside and execute in the Normal World. Native services reside and execute in the Secure World. The secure parts of TrustZone Software run in the Secure World.
- See also* Secure monitor.
- Normal World** *See* Normal and Secure Worlds.
- nSRST** Abbreviation of *System Reset*. The electronic signal that causes the target system other than the TAP controller to be reset. This signal is known as **nSYSRST** in some documentation.
- See also* nTRST.
- nTRST** Abbreviation of *TAP Reset*. The electronic signal that causes the target system TAP controller to be reset. This signal is known as **nICERST** in some documentation.
- See also* nSRST.

## O

- OS-awareness** OS-awareness is a feature provided by the ARM tools that enables you to:
- debug applications running on an embedded OS development platform, such as a *Real-Time Operating System* (RTOS)
  - present thread information and scope some debugging operations to specific threads.
- Output section** A contiguous sequence of input sections that have the same RO, RW, or ZI attributes. The sections are grouped together in larger fragments called regions. The regions are grouped together into the final executable image.
- See also* Input section and Region.

## P

**PCH** *See* PreCompiled Header (PCH).

### **PreCompiled Header (PCH)**

A header file that is precompiled. This avoids the compiler having to compile the file each time it is included by source files.

### **Procedure Call Standard for the ARM Architecture (AAPCS)**

*Procedure Call Standard for the ARM Architecture* defines how registers and the stack will be used for subroutine calls.

### **Program Counter (PC)**

In the context of the ARM architecture, integer register R15.

### **Program Status Register (PSR)**

Contains some information about the current program and some information about the current processor. Also referred to as Current PSR (CPSR), to emphasize the distinction between it and the Saved PSR (SPSR). The SPSR holds the value the PSR had when the current function was called, and which will be restored when control is returned.

An Enhanced PSR (EPSR) contains an additional bit (the Q bit, signifying saturation) used by some ARM architecture-based processors, including the ARM9E.

*See also* Current Program Status Register (CPSR), Enhanced Program Status Register (EPSR), and Saved Program Status Register (SPSR).

**PSR** *See* Program Status Register (PSR).

**PU** Protection Unit.

## R

### **Read-Only Position Independent (ROPI)**

In the context of the ARM architecture, code or read-only data that can be placed at any address.

### **Read Write Position Independent (RWPI)**

In the context of the ARM architecture, read/write data addresses that can be changed at runtime.

### **Real-Time System Model (RTSM)**

An RTSM contains a hard-coded system containing one or more specific simulated processors and an emulation baseboard. Some RTSMs are provided with DS-5.

*See also* ARM Streamline Performance Analyzer, DS-5, Fast Models from ARM, and Simulator.

### **RealView Debugger**

An ARM software development tool that enables you to make use of a debug agent in order to examine and control the execution of software running on a debug target.

### **RealView Development Suite (RVDS)**

The suite of software development tools, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of processors.

*See also* ARM Compiler toolchain, DSTREAM, Eclipse for DS-5, DS-5, RealView ICE, RealView Trace, and RealView Trace 2.

### **RealView ICE**

A JTAG-based debug solution to debug software running on ARM architecture-based processors. RealView ICE host software is provided with DS-5. The RealView ICE run control unit must be purchased as a separate product.

*See also* DSTREAM, RealView Trace, RealView Trace 2.

<b>RealView Trace</b>	<p>Works in conjunction with RealView ICE to provide real-time trace functionality for software running in leading edge System-on-Chip devices with deeply embedded processor cores. The RealView Trace hardware unit must be purchased as separately.</p> <p><i>See also</i> DSTREAM, RealView ICE, RealView Trace 2.</p>
<b>RealView Trace 2</b>	<p>Works in conjunction with RealView ICE to:</p> <ul style="list-style-type: none"> <li>• provide real-time trace functionality for software running in leading edge System-on-Chip devices with deeply embedded processor cores.</li> </ul> <p>The RealView Trace 2 hardware unit must be purchased as separately.</p> <p><i>See also</i> DSTREAM, RealView ICE, RealView Trace.</p>
<b>Region</b>	<p>In an image, a region is a contiguous sequence of one, two, or three output sections (RO, RW, and ZI). A region typically maps onto a physical memory device, such as ROM, RAM, or peripheral.</p> <p><i>See also</i> Output section and Root region.</p>
<b>Root region</b>	<p>In an image, regions having the same load and execution address. A non-root region is a region that must be copied from its load address to its execution address.</p> <p><i>See also</i> Region.</p>
<b>ROPI</b>	<i>See</i> Read-Only Position Independent (ROPI).
<b>RSD</b>	<i>See</i> Running System Debug (RSD).
<b>RTSM</b>	<i>See</i> Real-Time System Model (RTSM).
<b>Running System Debug (RSD)</b>	<p>Used for OS-aware debugging, <i>Running System Debug</i> (RSD) means that a target can be debugged when it is running. This means that the debug target does not have to be stopped before carrying out any analysis of the system. RSD gives access to the application using a <i>Debug Agent</i> (DA) that resides on the target. The Debug Agent is scheduled along with other tasks in the system.</p> <p><i>See also</i> Debug Agent (DA) and Halted System Debug (HSD).</p>
<b>RVDS</b>	<i>See</i> RealView Development Suite (RVDS).
<b>RWPI</b>	<i>See</i> Read Write Position Independent (RWPI).
<b>S</b>	
<b>Saved Program Status Register (SPSR)</b>	<p>A register that holds a copy of what was in the Current Program Status Register before the most recent exception. Each exception mode has its own SPSR.</p>
<b>Scatter-loading</b>	Assigning the address and grouping of code and data sections individually rather than using single large blocks.
<b>Section</b>	<p>In the context of applications targeted at ARM architecture-based processors, a block of software code or data for an Image.</p> <p><i>See also</i> Input section and Output section.</p>
<b>Secure monitor</b>	Reliably switches the ARM processor between Normal World and Secure World execution environments. The Secure monitor is transparent to TrustZone Software developers.
<b>Secure World</b>	<i>See</i> Normal and Secure Worlds.



<b>Semihosting</b>	A mechanism to communicate <i>Input/Output</i> (I/O) requests from application code to a host workstation running a debugger. For example, you can use semihosting to enable functions in the C library, such as <code>printf()</code> and <code>scanf()</code> , to use the screen and keyboard on the host workstation instead of having a screen and keyboard on the target system.
<b>Serial Wire Debug (SWD)</b>	Serial Wire Debug is a two-pin, bi-directional, data signal plus clock that replaces the 5-pin or 6-pin JTAG interface. The Serial Wire/JTAG debug port provides access to system memory peripherals and debug configuration registers.
<b>Simulator</b>	In the context of the ARM tools, a simulator executes non-native instructions in software (simulating a core).  <i>See also</i> Fast Models from ARM <i>and</i> Real-Time System Model (RTSM).
<b>Software breakpoint</b>	A breakpoint that is implemented by replacing an instruction in memory with one that causes the processor to take exceptional action. Because instruction memory must be altered software breakpoints cannot be used where instructions are stored in read-only memory. Using software breakpoints can enable interrupt processing to continue during the breakpoint, making them more suitable for use in real-time systems.  <i>See also</i> Chained breakpoint, Conditional breakpoint, Data breakpoint, Hardware breakpoint, Instruction breakpoint, Software breakpoint, <i>and</i> Unconditional breakpoint.
<b>SPSR</b>	<i>See</i> Saved Program Status Register (SPSR).
<b>Stack Pointer (SP)</b>	Integer register R13.
<b>Supervisor Call (SVC)</b>	An instruction that causes the processor to call a programmer-specified subroutine. Used by the ARM standard C library to handle semihosting. This replaces <i>software interrupt</i> (SWI).
<b>SVC</b>	<i>See</i> Supervisor Call (SVC).
<b>SWD</b>	<i>See</i> Serial Wire Debug (SWD).
<b>SWI</b>	<i>See</i> Supervisor Call (SVC).
<b>T</b>	
<b>TAP Controller</b>	Logic on a device which enables access to some or all of that device for test purposes. The circuit functionality is defined in IEEE1149.1.  <i>See also</i> Test Access Port (TAP) <i>and</i> IEEE 1149.1.
<b>Target</b>	In the context of an ARM debugger, a Target is the part of your development platform to which the debugger can connect, and on which debugging operations can be performed. A target can be: <ul style="list-style-type: none"> <li>• A runnable target, such as an ARM architecture-based processor or a DSP. When connected to a runnable target, you can perform execution-related debugging operations on that target, such as stepping and tracing.</li> <li>• A non-runnable CoreSight component. CoreSight components provide a system wide solution to real-time debug and trace.</li> </ul> <i>See also</i> CoreSight, Debug Configuration, Debug Interface, <i>and</i> Device.

<b>Target Vehicle</b>	<p>Target vehicles provide DS-5 with a standard interface to disparate targets so that the debugger can connect easily to new target types without having to make changes to the debugger core software. The interface can be a hardware or software interface.</p> <p><i>See also</i> Real-Time System Model (RTSM) and RealView ICE.</p>
<b>TCK</b>	<p>The electronic clock signal that times data on the TAP data lines <b>TMS</b>, <b>TDI</b>, and <b>TDO</b>.</p> <p><i>See also</i> Test Data Input (TDI) and Test Data Output (TDO).</p>
<b>TCM</b>	Tightly Coupled Memory.
<b>TDI</b>	<i>See</i> Test Data Input (TDI).
<b>TDO</b>	<i>See</i> Test Data Output (TDO).
<b>Test Access Port (TAP)</b>	<p>The collection of four mandatory and one optional terminals that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are <b>TDI</b>, <b>TDO</b>, <b>TMS</b>, and <b>TCK</b>. The optional terminal is <b>nTRST</b>. This signal is mandatory in ARM processors because it is used to reset the debug logic.</p> <p><i>See also</i> TAP Controller, TCK, Test Data Input (TDI), Test Data Output (TDO), and TMS.</p>
<b>Test Data Input (TDI)</b>	<p><i>Test Data Input</i> (TDI) is the electronic signal input to a TAP controller from the data source (upstream). Usually this is seen connecting the RealView ICE run control unit to the first TAP controller.</p> <p><i>See also</i> RealView ICE, and TAP Controller.</p>
<b>Test Data Output (TDO)</b>	<p><i>Test Data Output</i> (TDO) is the electronic signal output from a TAP controller to the data sink (downstream). Usually this is seen connecting the last TAP controller to the RealView ICE run control unit.</p> <p><i>See also</i> RealView ICE, and TAP Controller.</p>
<b>Thumb instruction</b>	<p>One halfword or two halfwords that encode an operation for an ARM architecture-based processor operating in Thumb state. Thumb instructions must be halfword-aligned.</p> <p><i>See also</i> ARM instruction, Thumb-2 instruction, and Thumb-2EE instruction.</p>
<b>Thumb state</b>	<p>A processor that is executing Thumb instructions is operating in Thumb state. The processor switches to ARM state (and to recognizing ARM instructions) when directed to do so by a state-changing instruction such as <b>BX</b>, <b>BLX</b>.</p> <p><i>See also</i> ARM state, Jazelle state, and ThumbEE state.</p>
<b>Thumb-2 instruction</b>	<p>Thumb-2 is a major enhancement of the Thumb instruction set, and is defined by ARMv6T2 and ARMv7M architectures. Thumb-2 provides almost exactly the same functionality as the ARM instruction set. It has both 16-bit and 32-bit instructions, and achieves performance similar to ARM code, but with code density similar to Thumb code.</p> <p><i>See also</i> ARM instruction, Thumb instruction, and Thumb-2EE instruction.</p>
<b>Thumb-2EE instruction</b>	<p><i>Thumb-2 Execution Environment</i> (Thumb-2EE) is defined by ARMv7 architecture. The Thumb-2EE instruction set is based on Thumb-2, with some changes and additions to make it a better target for dynamically generated code, that is, code compiled on the device either shortly before or during execution.</p> <p><i>See also</i> ARM instruction, Thumb instruction, and Thumb-2 instruction.</p>

<b>ThumbEE state</b>	<p>A processor that is executing Thumb-2EE instructions is operating in ThumbEE state. In this state, the instruction set is almost identical to the Thumb instruction set. However, some instructions have modified behavior, some instructions are not available, and some new instructions become available.</p> <p><i>See also</i> ARM state, Jazelle state, <i>and</i> Thumb state.</p>
<b>TMS</b>	Test Mode Select.
<b>TPA</b>	<i>See</i> Trace Port Analyzer (TPA).
<b>TPIU</b>	<i>See</i> Trace Port Interface Unit (TPIU).
<b>Trace Funnel</b>	<p>The Trace Funnel combines up to eight trace sources (ETM or HTM) on a single funnel. However, in this release, trace data can be captured only from a single ETM at a time.</p> <p><i>See also</i> AHB Trace Macrocell (HTM), CoreSight, CoreSight ETM, <i>and</i> Embedded Trace Macrocell (ETM).</p>
<b>Trace Port Analyzer (TPA)</b>	A logic analyzer that can capture the details of program execution in real time. RealView Trace is the ARM trace port analyzer.
<b>Trace Port Interface Unit (TPIU)</b>	<p>The Trace Port Interface Unit is a trace sink that drains trace data off-chip to a TPA, such as RealView Trace.</p> <p><i>See also</i> CoreSight, CoreSight ETB, CoreSight ETM, RealView Trace, <i>and</i> RealView Trace 2.</p>
<b>Trigger</b>	In the context of breakpoints, a trigger is the action of noticing that the breakpoint has been reached by the target and that any associated conditions are met.
<b>TrustZone Software</b>	A secure software framework that enables best use of security extensions built into the ARM architecture. Used in single-processor ARM cores that can operate as two virtual CPUs.
<b>U, V, W</b>	
<b>Unconditional breakpoint</b>	<p>A breakpoint that does not have a conditional qualifier assigned. The breakpoint activates immediately it is hit, but subsequent image execution is determined by any actions assigned to the breakpoint.</p> <p><i>See also</i> Conditional breakpoint, Data breakpoint, Hardware breakpoint, Instruction breakpoint, <i>and</i> Software breakpoint.</p>
<b>Undefined</b>	In the context of the ARM architecture, an attempt to execute an undefined instruction causes an Undefined Instruction exception.
<b>Unpredictable</b>	<p>In the context of the ARM architecture, the result of an unpredictable instruction cannot be relied upon. Unpredictable instructions or results must not represent security holes. Unpredictable instructions must not halt or hang the processor, or any parts of the system.</p>
<b>Veneer</b>	In the context of the ARM architecture, a small block of code used with subroutine calls when there is a requirement to change processor state or branch to an address that cannot be reached in the current processor state.
<b>VFP</b>	A standard for floating-point coprocessors where several data values can be processed by a single instruction.

**Watch**

In an ARM debugger, a watch is a variable or expression that you want the debugger to display at every step or breakpoint so that you can see how its value changes. You can set watchpoints in a debugger to display the content of a variable or expression.

**Watchpoint**

In DS-5, this is a hardware breakpoint.

**Word**

In the context of the ARM architecture, a word holds a value held in four contiguous bytes. A 32-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.