Application Note 99

Core Type & Revision Identification

Document number: ARM DAI 0099C
Issued: November 2003

ARM

Application Note 99
Core Type & Revision Identification

Release information

The following changes have been made to this Application Note.

Change history

Date	Issue	Change
July 2002	A	First release
September 2002	B	Second release
November 2003	C	Updated to include ARM1136J-S & ARM1136FJ-S cores. Also updated ARM Manufacture ID.

Proprietary Notice

Confidentiality status

This document is Open Access. This document has no restriction on distribution.

Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

ARM web address

http://www.arm.com

Table of Contents

1 Introduction

1.1 Two ways ARM cores are identified

ARM cores are identified through two mechanisms. The first is through Register 0 of the System Control Coprocessor, also referred to as coprocessor 15, or CP15. CP15 is available on processor cores containing an MMU or MPU only and contains a number of configuration registers (the actual number dependent on the core). CP15 Register 0 is hard-wired, readable by the core, and defined by ARM Limited. It defines information such as the core's part number, revision number, architecture version, implementation variants, and who implemented the core. The value of Register 0 must not be changed by the implementer; otherwise, operating systems may not correctly identify the host processor, and have difficulties with validation.

The second way that ARM cores are identified is through a hard-wired, IEEE 1149.1 (JTAG) standard compliant TAP ID. This TAP ID is used to configure debug software and indicates the part number, manufacturer, and revision of a particular ARM core. The TAP ID is configured by the implementer (the person integrating the core into a chip design).

Note *The TAP ID is not visible through CP15 and so the revision of a particular ARM core is not readable by the core. It is only read through the scan chain.*

1.2 What does the application note cover?

This application note, together with the appropriate *Technical Reference Manual* (TRM), describes CP15 Register 0, core TAP ID settings, and ways of implementing separate SoC identification numbers.

Cores in the SecurCore family do not normally have a TAP controller and so no TAP ID.

This application note indicates:

The bit allocation of CP15 Register 0

How to set the 32 bits of the ARM core TAP ID

The bit allocation of the ARM core TAP ID

Using a separate SoC TAP ID

2 Core Identification

2.1 Coprocessor 15 Register 0 core identification

Coprocessor 15 (CP15) Register 0 is the main identification (ID) register and defines the core implementation. This register is read-only and configured by ARM Limited within the core.

Note *CP15 Register 0 can have extra shadow registers, depending on the core. These shadow registers identify additional characteristics of the particular core. The following table shows some examples of this.*

Core	MRC Instruction opcode_2	Coprocessor 15 – Register 0 ID code and shadow registers
ARM720T	0	ID code register
ARM920T	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
ARM922T	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
ARM940T	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
ARM926EJ-S	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
	2	Shadow register – Tightly Coupled RAM type & size
ARM946E-S	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
	2	Shadow register – Tightly Coupled RAM type & size
ARM966E-S	0	ID code register
ARM1020T	0	ID code register
ARM1022E	0	ID code register
	1	Shadow register - Instruction & Data Cache type & size
ARM1026EJ-S	0	ID code register
	1	Shadow register – Cache Type
	2	Shadow register – TCM Status
ARM1136J-S / ARM1136JF-S	0	ID code register
	1	Shadow register – Instruction & Data Cache type & size
	2	Shadow register – Tightly Coupled RAM type & size
	3	Shadow register – Translation Look-Aside Buffer RAM type & size

All CP15 Register 0 registers are read using the MRC instruction. The syntax of the MRC instruction is:

```
MRC {cond} p15, opcode_1, Rd, CRn, CRm, opcode_2
```

For example:

```
MRC p15, 0, r0, c0, c0, 0
```

This instruction has two opcode fields. The second, opcode_2, identifies which register within Register 0 you access.

For specific CP15 Register 0 registers and their particular bit allocation, see the appropriate TRM for your core.

The bit allocation of the ID code register is different for ARM7 family cores and post ARM7 (ARM9, ARM10, etc.) family cores.

Note *ARM7TDMI, ARM7TDMI-S, ARM7EJ-S, and ARM9TDMI cores do not have a CP15.*

The bit allocation of CP15 Register 0 depends on the ARM core family as indicated by the following two tables.

ARM7 Core Family **Coprocessor 15 Register 0**				
[31:24]	**[23]**	**[22:16]**	**[15:4]**	**[3:0]**
Implementer	**A**	**Variant**	**Primary Part Number**	**Revision**

Where:

Revision (bits [3:0]) **Implementation-defined processor revision number**

Primary Part Number (bits [15:4]) **Implementation-defined processor primary part number**

Variant (bits [22:16]) **Implementation-defined variant number**

A (bit [23]) **ARM processor architecture**

- **0 Architecture 3**
- **1 Architecture 4T**

Implementer (bits [31:24]) **Implementer code indicates who designed the core**

e.g.

- **0x41 A (ARM Ltd)**
- **0x44 D (Digital Equipment Corporation)**
- **0x69 i (Intel Corporation)**

ARM9, ARM10, ARM11 …. Core Families Coprocessor 15 Register 0				
[31:24]	[23:20]	[19:16]	[15:4]	[3:0]
Implementer	Variant	Architecture	Primary Part Number	Revision

Where:

Revision (bits [3:0]) — **Implementation defined processor revision number**

Primary Part Number (bits [15:4]) — **Implementation defined processor primary part number**

Architecture (bits [19:16]) — **ARM processor architecture**

- **0x1 Architecture 4**
- **0x2 Architecture 4T**
- **0x3 Architecture 5**
- **0x4 Architecture 5T**
- **0x5 Architecture 5TE**
- **0x6 Architecture 5TEJ**
- **0x7 Architecture 6**

Variant (bits [23:20]) — **Implementation defined variant number**

Implementer (bits [31:24]) — **Implementer code indicates who designed the core**

e.g.

- **0x41 A (ARM Ltd)**
- **0x44 D (Digital Equipment Corporation)**
- **0x69 i (Intel Corporation)**

Revision and Variant fields

ARM uses the Revision field (bits[3:0]) and Variant field (bits[23:20]) to define the CPU revision and silicon revision according to the following table. A CPU revision can be viewed as a major change and is accompanied with updated documentation. A silicon revision can be viewed as a minor update.

Core	Variant field	Revision field	Most recent major revision as of Nov 2003
ARM720T	Minor CPU revision	Major CPU revision	Rev 4
ARM920T	Major CPU revision	Minor CPU revision	Rev 1
ARM922T	Major CPU revision	Minor CPU revision	Rev 0
ARM926EJ-S	Minor CPU revision	Major CPU revision	Rev r0
ARM940T	Minor CPU revision	Major CPU revision	Rev 2
ARM946E-S	Major CPU revision	Minor CPU revision	Rev r1
ARM966E-S	Major CPU revision	Minor CPU revision	Rev r2
ARM1020T	Major CPU revision	Minor CPU revision	Rev 0
ARM1022E	Major CPU revision	Minor CPU revision	Rev r0
ARM1026EJ-S	Major CPU revision	Minor CPU revision	Rev r0
ARM1136J-S / ARM1136JF-S	Major CPU revision	Minor CPU revision	Rev r0

2.2 JTAG TAP ID core identification

The JTAG TAP ID is a 32-bit “Device ID” register that can be read through the JTAG port by debug tools. The debug tools read this TAP ID and can automatically configure themselves for the appropriate core. Alternatively, if you are using Multi-ICE or RealView-ICE you can manually configure it for any particular core configuration. The partner implementing the core must set the TAP ID according to the following table as defined by ARM Limited and in accordance with the IEEE 1149.1 standard.

Note.

For the **ARM7TDMI**, **ARM720T**, and **ARM740T** cores, the TAP ID can only be set by hand-patching the layout. Usually, this is done in the top mask layer.

For the **ARM9** AND **ARM10** family cores, the TAP ID is configured by setting all values on a 32-bit TAPID[31:0] bus external to the macrocell. To change the TAP ID you must change the value on the 32-bit TAPID[31:0] bus.

For the **ARM11** family cores, only the version and manufacturer ID fields of the 32-bit TAP ID, are routed to the edge of the chip so that partners can create their own TAP ID device numbers by tying the pins to HIGH or LOW values. The remaining parts of the TAP ID, which are fixed for a particular core, are pre-configured within the core.

JTAG TAP ID						
Version	Part number				Manufacturer ID	Marker
	Processor Core	Capability	Family	Device number		
31:28	27	26:24	23:20	19:12	11:1	0

Where:

Marker [0]

Must always set to logic 1, as required by IEEE 1149.1.

Manufacturer ID [11:1]

The value of the Manufacturer ID field identifies the ARM partner that manufactured the chip.

Foundry customers can either use their own JEDEC issued manufacturers ID or use ARM's i.e. 0x477 including the Marker bit, bit0. (This 0x477 value is better than using the old number 0xF0F, again including the Maker bit.)

For the ARM7TDMI, ARM720T, and ARM740T the Manufacturer ID plus Marker is fixed in the layout. In the past, this value has been at 0xF0F. This is not a problem, but future all new implementations should use ARM’s new number 0x477.

For ARM test chips, ARM’s Manufacturer ID is used (0x477 including the Marker bit).

In production devices, the manufacturer ID may be set to the manufacturer’s JEDEC bank and company code, as described in the IEEE 1149.1 JTAG standard.

The implementer can change the number from the default value, but if you do, you must ensure the part number does not conflict with the part number of any other device with the same manufacturer number.

Part Number [27:12]

The Part Number has four fields as shown and must be set appropriately by the implementer. Multi-ICE uses this value to automatically detect the device type.

ARM has developed some general rules to predict in advance which ID codes will be used for various ARM cores. See section 2.2.1

If you change the part number from the default value, you must notify ARM Limited and other debug tool vendors, so that their debug tools can be updated, but no guarantees can be given as to when this will happen.

Version (Revision) [31:28]

The revision number is used by debug tools to decide which features and workarounds to enable. You should not normally need to change the revision from the default value.

Only major revisions are reflected in the JTAG ID, not minor revisions.

2.2.1 Device number (TAPID[19:12])

The 8 bits of the device number represent the last two digits in the part number (see following table).

TAPID[19:12] Device number	Core features
0x00 (b 0000 0000)	Core only
0x20 (b 0010 0000)	Core with MMU
0x22 (b 0010 0010)	Core with MMU and half-size caches
0x26 (b 0010 0110)	Core with MMU and TCM
0x40 (b 0100 0000)	Core with MPU
0x46 (b 0100 0110)	Core with MPU and TCM
0x66 (b 0110 0110)	Core with TCM
0x36 (b 0011 0110)	V6 architecture core with MMU

Where:
MMU – Memory Management Unit
MPU – Memory Protection Unit
TCM – Tightly Coupled Memory

2.2.2 Family (TAPID[23:20])

This field is used to represent the core family as shown.

TAPID[23:20]	Family
0x7 (b0111)	ARM7
0x9 (b1001)	ARM9
0xA (b1010)	ARM10
0xB (b1011)	ARM11

2.2.3 Capability (TAPID[26:24]) and Processor Core (TAPID[27])

The capability bits ([26:24]) define the core's capability in terms of:

extended math operations (E extension)

Jazelle extension (J extension) for running JAVA

whether the processor core is a hard or soft macrocell

TAPID bit 27 identifies the core as being an ARM processor core (logic 0) or a non-ARM processor core (logic 1).

The combined ARM core ID bit 27 and the capability bits [26:24] are interpreted according to one of the following two tables, depending on the core family.

Note *All ARM11 cores contain extended math operations, so there is no need for its option in the capability bits. These bits are left reserved for future use.*

Capability - ARM7, ARM9, ARM10 Core Families	
Processor core - TAPID[27] / Capability - TAPID[26:24]	**Description**
b0 000	ARM Processor pre E extension - hard macrocell
b0 001	ARM Processor pre E extension - soft macrocell
b0 010	Reserved
b0 011	Reserved
b0 100	ARM processor with E extension - hard macrocell
b0 101	ARM processor with E extension - soft macrocell
b0 110	ARM Processor with J extension - hard macrocell
b0 111	ARM Processor with J extension - soft macrocell
b1 000	Reserved
b1 001	Not a recognized executable ARM device (1)
b1 010	Reserved
b1 011	ARM Embedded Trace Buffer (2)
b1 100	Reserved
b1 101	Reserved
b1 110	Reserved
b1 111	Typically used for test chip boundary scan IDs

Capability - ARM11 Core Family	
Processor core - TAPID[27] / Capability - TAPID[26:24]	**Description**
b0 000	Reserved
b0 001	Reserved
b0 010	Reserved
b0 011	Reserved
b0 100	Reserved
b0 101	Reserved
b0 110	ARM Processor with J extension - hard macrocell
b0 111	ARM Processor with J extension - soft macrocell
b1 000	Reserved
b1 001	Not a recognized executable ARM device (1)
b1 010	Reserved
b1 011	ARM Trace Buffer (2)
b1 100	Reserved
b1 101	Reserved
b1 110	Reserved
b1 111	Typically used for test chip boundary scan IDs

Example part numbers

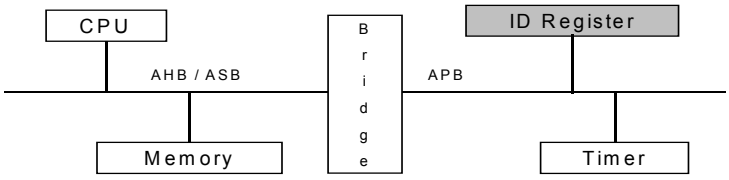
Example TAP ID part numbers are shown in the following table. The ‘E’ means extended math operations and the ‘S’ means it is a synthesizable core.

Core name	Part Number				Part number in Hex format
	TAPID [27]	TAPID [26:24]	TAPID [23:20]	TAPID [19:12]	
	ARM core ID	Capability	Family	Core	
ARM9TDMI	0	000	1001	0000 0000	X0900xxx
ARM966E-S	0	101	1001	0110 0110	X5966xxx

2.3 Customer specific SoC identification

You might want your SoC implementation to have its own ID to identify different revisions or configurations of the chip. These must not conflict with the ID registers already described in this document.

To implement this, ARM Limited recommends you add a separate memory-mapped read-only register on the system bus, possibly on the AMBA bus if you are using AMBA. See the diagram below.



Alternatively, you might want to implement the ID in a separate scan chain, using the existing ARM TAP controller.

The ARM7TDMI core uses scan chains 0-4 and 8 for internal purposes. Additionally, scan chain 15 is used by the system control coprocessor in the ARM710T and ARM720T cores. This means scan chains 5-7 and 10-14 can be used by the ASIC designer for their own purpose.

For the ARM9 family, scan chains 0 to 15 are used by ARM leaving 16 to 31 which can be used by the ASIC designer.

Details of how to add scan chains can be found in the FAQ (frequently asked questions) titled “How do I add scan chains to the ARM TAP controller?” This FAQ uses a latch-based scan cell, but applies equally well when using a flip-flop based scan cell.

Note *The ARM TAP controller is IEEE compliant. It is recommended that you follow the IEEE1149.1 specifications when adding scan chains to the TAP Controller.*