Engineering Support Application Note
Number: 276
*Issue: ***DRAFT****

Differentiating Between Hardware Platforms in Software

This document describes the software mechanisms required to distinguish between ARM processor cores, and where appropriate, between processors which share the same core.

Applicable Hardware :	All 32-bit Acorn hardware	Related Application Notes:	None

Engineering Support
Acorn Risc Technologies
Acorn House
645 Newmarket Road
Cambridge
CB5 8PB

Introduction

Application software is often required to run on a wide range of Acorn systems, from ARM2 to StrongARM; the only restrictions which are generally imposed concern the version of the OS in use.

In order to makesoftware run at acceptable speeds on this wide range of platforms, developers may wish to use different algorithms optimised for different variants of the ARM architecture within their code, selectable according to the processor fitted; on ARM2 machines, a common approach to produce acceptable speed when compared to later systems is to reduce the graphic resolution or colour depth relative to the StrongARM version. This document describes how processors may be distinguished from one another in application software.

Determining the CPU Core Type

ARM2, the earliest ARM processor produced in quantity, does not have the SWP instruction (which exchanges the contents of two registers without using a user-visible intermediate third register) implemented; thus an ARM2 can be uniquely identified by issuing a SWP on two registers and setting a trap on the undefined instruction vector to catch it if it fails.

By contrast, ARM250 uses the ARM2as core cell; SWP was implemented on this cell, so a test for SWP will not distinguish an ARM250 from an ARM3.

ARM3 utilises a precursor of the Architecture 3 system for determining CPU type; coprocessor 15 register 0 contains &03 in the range bit 15-bit 8. An MRC operation similar to the one detailed below (remember to have a trap on the undefined instruction vector in case the chip you are examining is actually an ARM250) will uniquely identify an ARM3. As ARM250 has no actual coprocessor interface, presence of SWP but absence of meaningful data in coprocessor 15 register 0 uniquely identifies an ARM250.

From ARM Architecture 3 onwards (Architecture 3 encompasses processors with ARM6 and ARM7 cores, Architecture 4 encompasses SA-110 and ARM8xx), register 0 of coprocessor 15 is designated as the ID register; bits [15:4] contain a 3-digit part number in binary-coded decimal format (eg &810 for ARM810). Owing to the variations in ARM cores across versions, it is recommended that only the most significant digit of the three-digit set is checked.

From SVC mode, this can be done with

```
MRC    CP15,0,r2,C0,C0,0
AND    r2,r2,#&F000
CMP    r2,#<whatever> (where <whatever> is, for example, &A000 for SA-110)
```

having first pushed R2 to the stack or otherwise stored it if it contained anything important to your program.

It is recommended that these test are done working from the least instruction-rich processor to the most, ie test for ARM2, then ARM3, then Architecture 3/4. Attempting a coprocessor data transfer on an ARM2 will, unsurprisingly, result in an unknown instruction error.

Thus, for all CPUs installed in 32-bit Acorn systems:

CPU	Core identifies as
ARM2	No formal ID method; test for absence of SWP
ARM250	No formal ID method; test for presence of SWP and absence of MRC
ARM3	ARM3
ARM610	ARM6
ARM710	ARM7
ARM7500	ARM7
ARM7500FE	ARM7
SA-110	SA-110

Resolving CPU Ambiguities

The table clearly illustrates that testing the core does not always give unique information; several processors share the same core, but have different surrounding functionality. These processors may be uniquely distinguished as follows:

ARM710, ARM7500 and ARM7500FE

These may be distinguished by examining the ID number within the area assigned to IOMD. In common with IOC, IOMD base address is &03200000; the chip ID number is stored at offset &94 (lo byte of ID is lo byte in word) and &98 (hi byte of ID is lo byte in word).

CPU	ID lo byte	ID hi byte
ARM710	&E7	&D4
ARM7500	&98	&5B
ARM7500FE	&7C	&AA

These addresses should only ever be read from; if written to, they may cause the system to enter a low-level test mode.