ARM926EJ-S Errata List

Processor Division

CPU Group

Document number:	**ARM926EJ-PRDC-002691 5.0**
Date of Issue:	28th March 2007
Author:	
Authorized by:	

Abstract

This document describes the known errata in the ARM926EJ-S design indicating which errata are present in each product revision.

Keywords

Contents

1 ABOUT THIS DOCUMENT

1.1 References

This document refers to the following documents.

Ref.	Document No	Author(s)	Title
1	ARM DDI 0198	ARM	ARM926EJ-S Technical Reference Manual
2	ARM DDI 0222	ARM	ARM9EJ-S Technical Reference Manual
3	ARM IHI 0011	ARM	AMBA Specification

1.2 Scope

This document describes the errata discovered in the implementation of the ARM926EJ-S, categorised by level of severity. Each description includes:

- where the implementation deviates from the specification
- the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a ‘work-around’ where possible
- the status of corrective action.

1.3 Terms and Abbreviations

This document uses the following terms and abbreviations.

AHB	AMBA High Speed Bus (see ref. 3)
DHBL	Data AHB byte lane strobe signals
Endianness	Byte order significance within a word
INCR4	Four beat incrementing burst on AHB (see ref. 3)
MMU	Memory Management Unit
TCM	Tightly Coupled Memory
WFI	Wait for Interrupt
PTE	MMU Page Table Entry

2 CATEGORISATION OF ERRATA

Errata recorded in this document are split into three groups:

Category 1 Features which are impossible to work around and severely restrict the use of the device in all or the majority of applications rendering the device unusable.

Category 2 Features which contravene the specified behavior and may limit or severely impair the intended use of specified features but does not render the device unusable in all or the majority of applications.

Category 3 Features that were not the originally intended behavior but should not cause any problems in applications.

2.1 Current Revision Errata Summary

The ARM926EJ-S r0p5 is the current revision available. Known errata affecting this revision are indicated by the table in the section below.

2.2 Product Revision and Errata Summary Table

The errata associated with this product are categorised in the following way. Numbers in brackets after the errata description indicate the order in which the errata were found chronologically.

Errata Description (No.)	Rev r0p0	Rev r0p2	Rev r0p3	Rev r0p4	Rev r0p5
Category 1					
Duplicate cache lines for FCSE multiple address mappings (1)	**Yes**	**No**	**No**	**No**	**No**
Category 2					
Writeback eviction data incorrect (2)	**Yes**	**No**	**No**	**No**	**No**
Hardware Breakpoint Unit Addressing (5)	**Yes**	**Yes**	**No**	**No**	**No**
STANDBY mode not entered after execution of WFI instruction (6)	**Yes**	**Yes**	**Yes**	**No**	**No**
Incorrect behavior of HLOCK output (9)	**Yes**	**Yes**	**Yes**	**No**	**No**
DHPROT incorrect after data access to instruction TCM (11)	**Yes**	**Yes**	**Yes**	**Yes**	**No**
IRSEQ incorrect after data access to instruction TCM (12)	**Yes**	**Yes**	**Yes**	**Yes**	**No**
IRSEQ incorrect during instruction fetches to TCM (13)	**Yes**	**Yes**	**Yes**	**Yes**	**No**
Category 3					
Reset of cache round robin victim selection counters (3)	**Yes**	**No**	**No**	**No**	**No**
Destination register update in bounced coprocessor 14 MRC operations (4)	**Yes**	**Yes**	**No**	**No**	**No**
DHBL byte lane strobes endianness inconsistent for buffered writes (7)	**Yes**	**Yes**	**Yes**	**No**	**No**
Inefficient fill of instruction prefetch buffer (8)	**Yes**	**Yes**	**Yes**	**No**	**No**
Writes unbuffered in bufferable region with data cache disabled (10)	**Yes**	**Yes**	**Yes**	**No**	**No**
Setting CP15 Disable Write Back bit can cause data loss for dirty cachelines (14)	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Shift-IR produces incorrect output on DBGTDO when Scan chain 15, INTEST selected (15)	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

3 CATEGORY 1 ERRATA

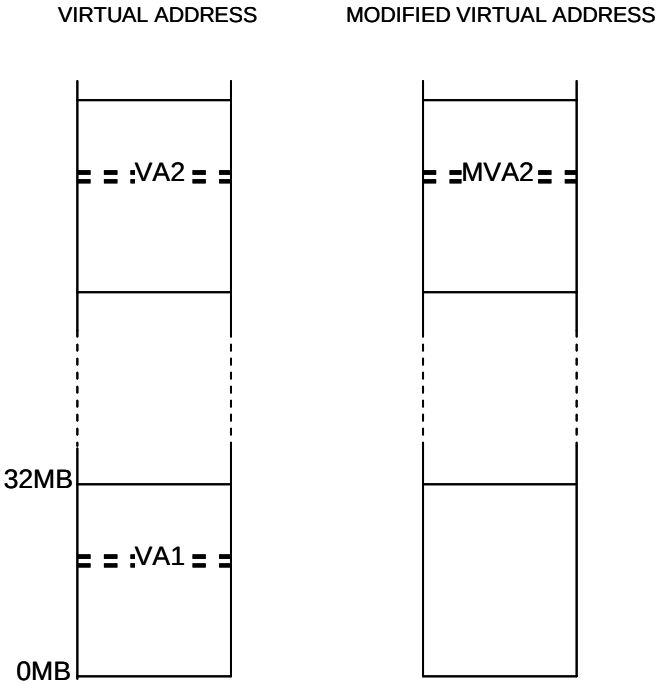
3.1 Duplicate cache lines for FCSE multiple address mappings (1)
(Revision r0p0)

3.1.1 Summary

Operating systems using FCSE may legitimately map two virtual addresses (VA) to a single modified virtual address (MVA). Under certain conditions it is possible for this type of multiple mapping to result in duplicate cache lines for the same MVA. The duplication can occur in either the instruction or data cache, however only duplication in the data cache can cause the device to function incorrectly. If duplicate cache lines exist in the data, then write operations can cause the data cache to become incoherent, if the write updates either of the duplicated cache lines.

3.1.2 Description

Consider the following address mapping:



Both VA1 and VA2 (virtual addresses) are mapped to MVA2. VA2 will be the same as MVA2, and VA1 will be mapped to MVA2 by virtue of the value contained in the FCSE PID register.

If the data cache does not contain an entry corresponding to MVA2, and two subsequent read accesses are made to both VA1 and VA2, then a linefill will occur for the access to VA1, and under certain conditions, a linefill will also (incorrectly) occur for the access to VA2.

3.1.3 Conditions

- The data cache must be enabled.
- The FCSE PID register must be non-zero.

- Multiple VA/MVA address mapping must exist.
- Two read accesses must be made from two different VA aliases for a given MVA address, with no FCSE PID register change between reads.

3.1.4 Implications

Currently, of the main platform operating systems targeted by ARM926EJ-S, only WindowsCE uses FCSE. Linux and Symbian OS are not affected by this fault.

3.1.5 Workaround

Either do not enable the data cache, or do not use the FCSE PID register.

4 CATEGORY 2 ERRATA

4.1 Writeback eviction data incorrect (2) (Revision r0p0)

4.1.1 Summary

The data cache contains an ancillary write buffer for data to be written into the cache data RAMs. Forwarding paths exist both for data read accesses performed by the ARM9EJ-S processor, and for those performed during a writeback operation. Under specific circumstances the forwarding logic used for writeback data does not function correctly. This can result in incorrect data being written out to main AHB memory, on the occurrence of a writeback eviction.

4.1.2 Description

The ancillary data cache RAM write-buffer contains two entries. If two store operations to the same address are made in succession then it is possible that both entries of the write-buffer correspond to the same address. If a writeback eviction is made to the cache line corresponding to this address, then forwarding logic must select the appropriate entry to be written into the eviction write-buffer. Under certain circumstances the forwarding logic functions incorrectly and data is forwarded from the wrong entry, causing incorrect data to be written back to memory.

4.1.3 Conditions

- The data cache must be enabled.
- Two store operations to the same address must have occurred in quick succession, such that both entries in the ancillary write-buffer contain the same address entry, and the data for the second store operation is placed in entry 0.
- The store address must be in a writeback region, and must be present in the data cache.
- The cache line corresponding to the address in the ancillary write-buffer is evicted by a subsequent linefill, and the ancillary write-buffer has not drained either of its entries.

An example is given by the following code sequence:

```
STR   R1,[R0]     ; store data placed in WB entry 1
STR   R2,[R0]     ; store data placed in WB entry 0
LDR   R3,[R4]     ; causes eviction of cache line corresponding to R0
```

In this particular example the value of R1 would be written back, instead of R2.

Note that forwarding is done on a byte basis, and incorrect behavior only occurs if the forwarding logic has to prioritize between entries. This means that the following sequence would not fail:

```
STRH  R1,[R0]     ; store bottom half of word (WB entry 1)
STRH  R2,[R0,#2]  ; store top half of word    (WB entry 0)
LDR   R3,[R4]     ; causes eviction of cache line corresponding to R0
```

4.1.4 Implications

Writeback cannot be used reliably.

4.1.5 Workaround

Do not use writeback.

4.2 Hardware Breakpoint Unit Addressing (5) (Revisions r0p0, r0p2)

4.2.1 Summary

The instruction address given to the hardware breakpoint units in ARM and Thumb states has bit 31 permanently set to zero. This can cause breakpoints not to be taken and others to be incorrectly taken. Multi-ICE can provide a basic workaround solution, and software breakpoints still work correctly. Note that this erratum also affects non-stopping hardware breakpoints, and the real-time debug system would have to be modified to work around this erratum.

4.2.2 Description

The hardware breakpoint units can be configured to match against instruction addresses. In order to do this comparison a 31-bit bus containing the top 31 instruction address bits is passed to the breakpoint hardware. The top bit (bit 31) is erroneously set to zero. This gives incorrect comparisons if either the instruction address has bit 31 set or the value being compared against has bit 31 set. In the case of the instruction address having bit 31 set the breakpoint hardware will see a zero instead, potentially allowing a false match with a value having bit 31 clear. In the case of the value in the breakpoint unit having bit 31 set, this will never match.

4.2.3 Conditions

This erratum applies to hardware breakpoint comparisons on the instruction address in ARM and Thumb states (not Jazelle state). It affects both stopping and non-stopping debug. Vector catching is not affected.

4.2.4 Implications

This erratum will confuse debuggers, potentially giving false breakpoints and potentially missing some breakpoints.

4.2.5 Workaround

There are several possible work around solutions:

1. Do not use virtual addresses in the range 2GB to 4GB.
2. Use software breakpoints instead of hardware breakpoints.
3. If using halting debug: Set the address comparison masks so that bit 31 is ignored. This ensures that breakpoints will not be missed, but false breakpoints are still possible. The debugger can then use the Method of Entry Bits to check that debug entry was caused by a breakpoint. If the breakpoint address was not that set by the debugger, then control can be returned to the CPU, otherwise the breakpoint can be taken as normal.
4. If using non-stopping debug: Set the address comparison masks so that bit 31 is ignored. This ensures that breakpoints will not be missed, but false breakpoints are still possible. The breakpoint handler needs to check that the breakpoint address corresponds to a breakpoint it set, if it does not then control can be returned to the main process, otherwise the breakpoint can be taken as usual.

The suitability of any of these workaround solutions is system dependent.

4.3 STANDBY mode not entered after execution of WFI instruction (6)
(Revisions r0p0, r0p2, r0p3)

4.3.1 Summary

Under certain circumstances the ARM926EJ-S processor may not fully switch into low-power state upon execution of the wait for interrupt instruction. This is dependent upon the processor state relating to prefetching of non-cacheable instructions.

4.3.2 Description

After execution of the wait for interrupt instruction, the switch into low power state (visible externally by the assertion of the STANDBYWFI signal), is delayed until all write-buffers have been drained and the memory system is in a quiescent state. It is possible for the block responsible for non-cacheable instruction fetches to incorrectly indicate that it is still active, when it is in fact idle. As the central memory sub-system controller will delay entering low power state until all the memory sub-system blocks indicate they are idle, low power state is never entered.

4.3.3 Conditions

The idle indication logic for non-cacheable instruction fetches is only asserted (other than immediately after system reset), when the four word prefetch buffer is full. If this buffer is not full when a wait for interrupt instruction is executed, then low power state will not be entered.

The prefetch buffer will not be full if prefetching is disabled via the CP15 debug override register, or if the last non-cacheable instruction fetch made by the ARM9EJ-S processor was within 4 words of a 1KB boundary. This is illustrated by the following diagram, where FE corresponds to the ARM9EJ-S fetch stage, and P1-P4 are the entries in the prefetch buffer.

	Instruction address [7:0]									
	EC	F0	F4	F8	FC	00	04	08	0C	10
1	FE	P1	P2	P3	P4					
2		FE	P1	P2	P3	P4				
3			FE	P1	P2	P3	P4			
4				FE	P1	P2	P3	P4		
5					FE	P1	P2	P3	P4	
6						FE	P1	P2	P3	P4

In case 1, the last non-cacheable instruction fetch is made at 0xEC, and the prefetch buffer is filled, as the addresses for P1-P4 are all before the 1KB boundary.

In cases 2-5 the prefetch buffer is not filled, as one or more of P1-P4 resides the next 1KB region.

In case 6, the last non-cacheable instruction fetch is the first word in 1KB region, and so like case 1, the addresses for P1-P4 all reside in the same 1KB region and the prefetch buffer will be filled.

There are a number of scenarios, which can cause cases 2-5 above, and consequently cause low power mode not to be entered correctly. These can be split into two categories: one where failure directly depends upon the alignment of the WFI instruction itself, and the other where it depends upon the underlying memory region type (non-cacheable, cacheable or TCM), and possibly the alignment of an unrelated instruction.

Failure due to Alignment of WFI Instruction

If the WFI instruction is run from a non-cacheable region, and located at an offset of between 0x3E8 and 0x3F4 (inclusive) from the previous 1KB boundary, then the WFI will execute with the prefetch buffer in a non-full state, and low-power state will not be entered correctly.

Failure due to underlying Memory Region

The second failure category involves the prefetching to be suspended due to the proximity of a 1KB boundary, and then a resulting change in memory region, which results in the prefetch buffer remaining in a non-full state. In this case the WFI instruction is run from either a cacheable or TCM region, and can be located anywhere within 1KB region. There are several ways for this condition to occur:

i. An instruction that causes the underlying code region to be no longer treated as non-cacheable is located at an offset of between 0x3E8 and 0x3F4 (inclusive). Such an instruction may be for example turning the cache or MMU on.

ii. A branch from a non-cacheable region into a region which is either cacheable, or a TCM region, with the branch instruction located at an offset of between 0x3E4 and 0x3F4 (inclusive).

iii. As ii, but the change in instruction flow is caused by an exception, rather than a branch.

iv. The instruction stream sequentially crosses a 1KB boundary, which also corresponds to an MMU page, or TCM region boundary, and the new page is either cacheable or a TCM region.

4.3.4 Implications

The programmers model behavior for the CP15 wait-for-interrupt instruction is not affected by this erratum. A software workaround however may be required to ensure low power state is correctly entered, in order for logic both internal and external to the ARM926EJ-S processor to be shutdown.

4.3.5 Workarounds

The complexity of the workaround required depends upon the usage model of non-cacheable code in a particular application. For applications which only run non-cacheable code for a short time during system initialization, then the workaround simply involves aligning the code which enables the instruction cache, such that it is offset from the previous 1KB, by less than 0x3E4. If there is mix of non-cacheable and cache/TCM regions used in the application, then different workarounds are required depending what memory region type the WFI instruction resides in.

WFI in Cacheable Region

If both non-cacheable and cacheable code is used in an application, and the WFI instruction resides in a cacheable region, then the following sequence will work around the problem:

1. Disable interrupts
2. Disable instruction cache
3. Execute WFI instruction
4. Enable instruction cache
5. Enable interrupts.

With this sequence the WFI instruction will complete execution once an interrupt is pending, and the interrupt will be acknowledged by the ARM9EJ-S processor once interrupts are enabled.

For this workaround to operate correctly, the instruction which disables the cache must be offset from the previous 1KB boundary by less than 0x3E4. The act of disabling the cache will cause instruction fetches to be

treated as non-cacheable, and instruction prefetching will recommence, with the prefetch buffer eventually becoming full.

The following code sequence can be used to implement this workaround with minimal impact on interrupt latency.

```
  MCR     p15, 0, r0, c7, c10, 4     ; drain write buffer
  MRS     r2, CPSR
  ORR     r4, r2, #3 :SHL: 6         ; set FIQ and IRQ disable bits
  MRC     p15, 0, r0, c1, c0, 0      ; read control register
  BIC     r1, r0, #1 << 12           ; clear ICache enable bit

; critical code section
  MSR     CPSR_c, r4                 ; disable interrupts
  MCR     p15, 0, r1, c1, c0, 0      ; disable ICache
  MCR     p15, 0, r0, c7, c0, 4      ; WFI
  MCR     p15, 0, r0, c1, c0, 0      ; restore I bit of control register
  MSR     CPSR_c, r2                 ; restore FIQ & IRQ disable bits of CPSR
```

WFI in TCM Region

If code is run from both non-cacheable code regions, and the instruction TCM region, then a workaround is possible, but relies on executing code in both the instruction TCM, and a specified non-cacheable region.

The workaround sequence is as follows:

1. Disable interrupts
2. Branch to specified location in non-cacheable region
3. Immediately return from branch
4. Execute WFI Instruction
5. Enable Interrupts.

This workaround is similar to the one given for WFI in a cacheable region, except for the mechanism used for filling the prefetch buffer.

The branch target in the non-cacheable region must be must be offset from the previous 1KB boundary by less than 0x3E4.

```
   MCR     p15, 0, r0, c7, c10, 4     ; drain write buffer
   MRS     r2, CPSR
   ORR     r4, r2, #3 :SHL: 6         ; set FIQ and IRQ disable bits
   LDR     r12,=branch_null           ; r12 = branch target
   ADD     r14, pc, #4                ; r14 = rtn address set to WFI instruction
```



```
;  critical code section
   MSR     CPSR_c, r4                ; disable interrupts
   MOV     pc,r12                    ; branch to non-cacheable region
   MCR     p15, 0, r0, c7, c0, 4     ; WFI
   MSR     CPSR_c, r2                ; restore FIQ & IRQ disable bits of CPSR

;  code in non-cacheable region
branch_null
   MOV          pc, r14
```

4.4 Incorrect behavior of HLOCK output (9) (Revisions r0p0, r0p2, r0p3)

4.4.1 Summary

Under certain limited conditions, the HLOCK indication, which is used during a swap instruction that accesses memory on the AHB, is not asserted at the correct point in the transaction preventing the swap instruction from being locked on the AHB. This can lead to another bus master being granted ownership of the bus during the swap instruction, resulting in incorrect semaphore operation.

4.4.2 Description

Figure 1 shows the required HLOCK behavior. HLOCK is asserted with HBUSREQ at least a cycle before the read address of the swap instruction is driven on the AHB. The ARM926EJ-S follows this behavior in 1:1 CLK:HCLK mode.

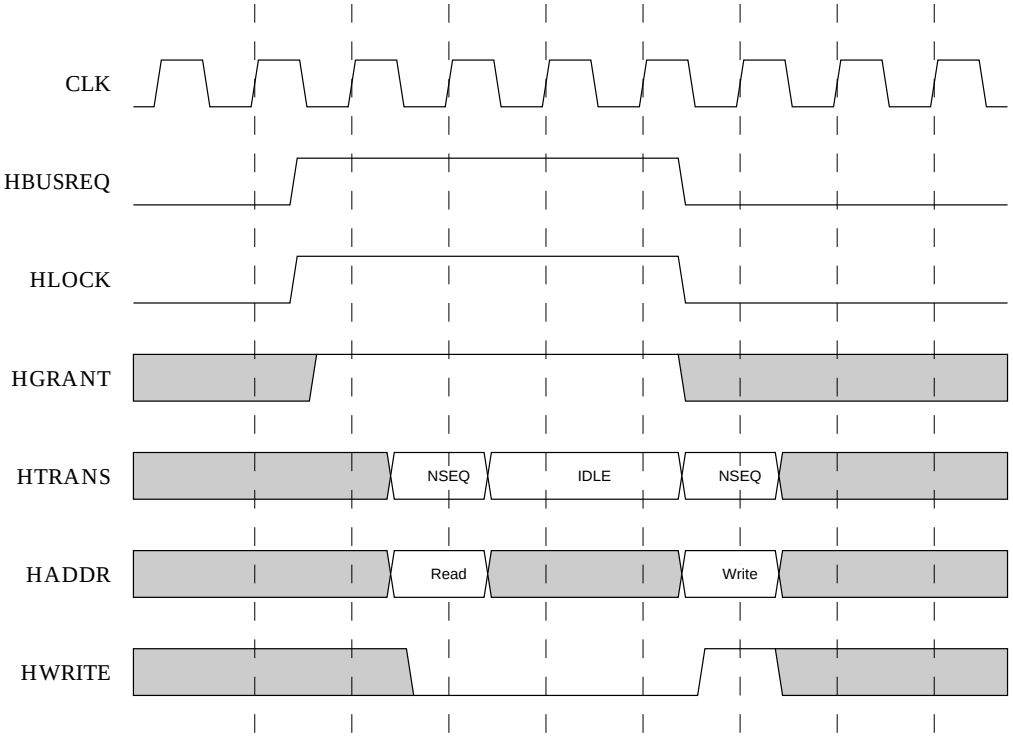


Figure 1: Correct HLOCK operation

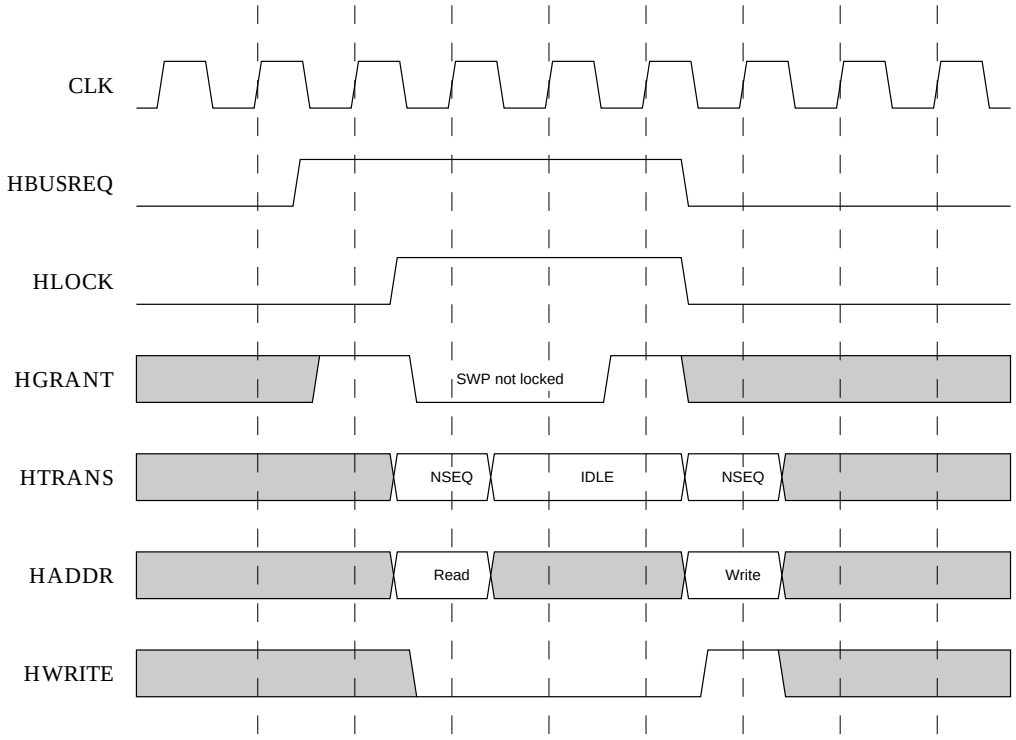


Figure 2: Incorrect HLOCK operation

Figure 2 shows the ARM926EJ-S behavior in non 1:1 CLK:HCLK modes. HLOCK is driven in the same cycle as the control signals (HTRANS, HADDR, etc) instead of in the previous cycle. The effect of the erratum is that the ARM926EJ-S may not gain locked ownership of the bus for the duration of the swap instruction and may be de-granted the bus after the read portion of the SWP. This allows another master to gain ownership of the bus after the read portion of the swap instruction, breaking the required semaphore behavior.

4.4.3 Implication

The effect of this erratum depends on the AHB system into which the ARM926EJ-S is integrated. This erratum will have no effect on the system if any of the following conditions apply:

- The system is running only in 1:1 CLK:HCLK mode.
- The ARM926EJ-S is the highest priority master in the system.
- The code being executed does not perform SWP's.
- Masters with a higher priority than the ARM926EJ-S do not have any dependency on SWP's performed by the ARM926EJ-S.
- Masters with a higher priority than the ARM926EJ-S do not perform SWP's to the same address as the ARM926EJ-S.

4.4.4 Workaround

If any of the conditions in section 4.4.3 are true for your design, the erratum will not occur and no workaround will be necessary.

If none of the conditions in section 4.4.3 are true for your design, the erratum may have an impact on your system and a workaround involving the modification of logic external to the ARM926EJ-S macrocell will be necessary:

The AHB arbiter can be modified to prevent the ARM926EJ-S being de-granted during the SWP instruction. The logic to do this is shown in Figure 3. HLOCK_926 is DHLOCK driven from the ARM926EJ-S. HGRANTlast_926 is a pipelined version of DHGRANT driven to the ARM926EJ-S, indicating that the ARM926EJ-S owns the current address cycle (note that DHGRANT may be low during the address cycle as shown in Figure 2).. HGRANTlast_926 is registered using the positive edge of HCLK and using HREADY as a qualifier.

HLOCK_926 in the errata case has the same timing as required for HMASTLOCK driven from the arbiter. Therefore, OR'ing HLOCK_926 with HMASTLOCK will achieve the correct HMASTLOCK timing. HLOCK_926 must be AND'ed with HGRANTlast_926 as it is only valid to OR when the ARM926EJ-S is the granted master.

HMASTLOCK_new must be driven to all the logic that HMASTLOCK previously drove, which will be the output of the arbiter plus any internal arbiter logic, such as the logic which controls the holding of HGRANT during a locked transaction. The original HMASTLOCK logic will only connect to the logic shown in Figure3.

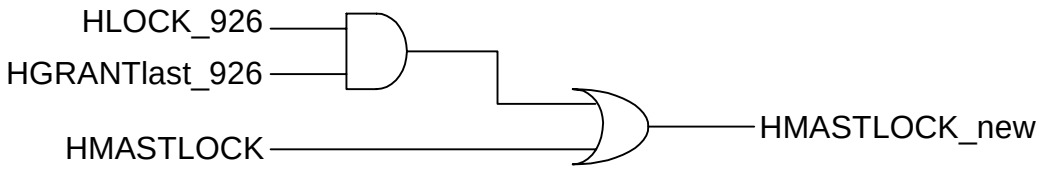


Figure 3: HLOCK workaround

4.5 DHPROT incorrect after data access to instruction TCM (11)

(Revisions r0p0, r0p2, r0p3, r0p4)

4.5.1 Summary

Under certain circumstances an error in the MMU can cause the AMBA AHB DHPROT bus to be incorrect, if an instruction TCM memory is used.

4.5.2 Description

After a data side access to instruction TCM the corresponding DHPROT value for the next data side instruction may be incorrect. In the event of the errata occurring DHPROT[3:2] will be 2'b00, which corresponds to the cacheable and bufferable attributes for the access.

4.5.3 Conditions

The error occurs when a load or store instruction to the instruction TCM is followed by either:

i) a store to a write-through region

ii) a store to a write-back region which misses the data cache

iii) a store to a non-cacheable, bufferable region

where the memory region type of this load or store is the same memory region type as the last memory operation before the instruction TCM load or store.

Any number of non load-store operations (excluding certain CP15 operations) may occur between the instruction accessing instruction TCM, and the instruction which causes the AHB access.

For example:

```
STR r0,[r1]   ; store to WT region
    .
    .       ; zero or more non load/store operations
    .
LDR r0,[r0]   ; load from I-TCM memory
    .
    .       ; zero or more non load/store operations
    .
STR r0,[r1]   ; store to WT region
```

4.5.4 Implications

The impact of this erratum on a system is dependant on how the AHB cacheable and bufferable attributes are used. AHB bridge or buffer structures may use the AHB attributes to determine whether an access should be

buffered or not. In the event of the error occurring an access would not be buffered when it should be. This will not cause functional failure but may degrade system performance.

A level 2 cache such as L210 may use the AHB attributes to determine whether cache memory should be accessed or not. In the event of the cacheable attribute being incorrect, functional failure can occur.

4.5.5 Workarounds

Depending on the way TCM is used, implementing one of the following workarounds will avoid the error:

Mutually Exclusive Use of I-TCM and L2 cache (L210 workaround)

Do not use instruction TCM and level 2 cache at the same time; prior to accessing instruction TCM memory, clean, invalidate, and disable the L2 cache.

Insert Dummy Operation between accesses

If copying from instruction TCM memory to cacheable memory, read or write a dummy location in memory before the write to cacheable memory.

For example:

```
LDR  r0,[r1]   ; load r0 from I-TCM memory addressed by r1
STR  r2,[r2]   ; r2 addresses dummy location
STR  r0,[r3]   ; store r0 to cacheable location
```

Note that performing the dummy access to D-TCM memory (if present), would avoid degrading bus performance, which would occur if the dummy store resulted in an AHB access.

Repeat Access (L210 workaround)

If the access following the instruction TCM access is repeated, then the DHPROT attributes will be correct for the repeated access.

For example:

```
LDR  r0,[r1]   ; load r0 from I-TCM memory addressed by r1
STR  r0,[r3]   ; store r0 to cacheable location, DHPROT[3:2] = 2'b00
STR  r0,[r3]   ; repeat access, DHPROT[3:2] now correct
```


4.6 IRSEQ incorrect after data access to instruction TCM (12)

(Revisions r0p0, r0p2, r0p3, r0p4)

4.6.1 Summary

Under certain circumstances the instruction TCM interface signal IRSEQ may be incorrect after data-side read (Load) from the instruction TCM.

4.6.2 Description

After a data load access to the instruction TCM, the IRSEQ signal can be incorrectly asserted, falsely indicating that the address on IRADDR is sequential to the previous access. The behavior of IRSEQ is correct following a write (store instruction) to the instruction TCM.

4.6.3 Conditions

The erroneous indication will occur, with few exceptions, whenever a load instruction for the instruction TCM is performed within sequential code being executed from instruction TCM.

4.6.4 Implications

The impact of this erratum is dependant on how the IRSEQ signal is used in the TCM system. If the IRSEQ signal is either used to determine wait-states (via IRWAIT), or to gate the memory chip-select (IRCS), then functional failure can occur.

4.6.5 Workarounds

There is no simple workaround that can be implemented in the software running from ITCM. The erratum can only be avoided by eliminating reliance on IRSEQ by one of the following means:

- Do not use IRSEQ in the instruction TCM sub-system.
- Do not access the instruction TCM using load or store instructions, whilst running code from the instruction TCM.

4.7 IRSEQ incorrect during instruction fetches to TCM (13)

(Revisions r0p0, r0p2, r0p3, r0p4)

4.7.1 Summary

Under certain circumstances, during sequential instruction fetch reads from the instruction TCM, the interface signal IRSEQ may incorrectly indicate that the accesses are non-sequential.

4.7.2 Description

While fetching sequences of some instruction types from the instruction TCM, the IRSEQ signal can be incorrectly deasserted, falsely indicating that the address on IRADDR is not sequential to the address used for the previous access.

4.7.3 Conditions

The erroneous indication will occur when executing code from instruction TCM in the following cases:

- While executing Thumb instructions
- While executing Java bytecodes
- When executing multi-cycle ARM instructions (e.g. MLA, LDM)

4.7.4 Implications

The impact of this erratum is dependant on how the IRSEQ signal is used in the TCM system. If the IRSEQ signal is either used to determine wait-states (via IRWAIT), or to gate the memory chip-select (IRCS), then the efficiency and performance of the design could be severely reduced.

4.7.5 Workarounds

There is no workaround that can be implemented in the software running from ITCM.

5 CATEGORY 3 ERRATA

5.1 Reset of cache round-robin victim selection counters (3) (Revision r0p0)

5.1.1 Summary

The precise behavior for round-robin victim selection after a CP15 invalidate All instruction, does not form part of the functional specification for ARM926EJ-S. However it is desirable to make the behavior deterministic, in such a way that it can easily be modeled by high level instruction set simulators. In order to achieve this round-robin victim selection, counters for each index should be reset after a CP15 invalidate all instruction.

This occurs in all cases except for one, which is an inconsistency.

5.1.2 Description

If the first linefill after a CP15 Invalidate All is to the same Index subset (X), but not actually the same Index, as the last linefill prior to the Invalidate All, then the Round-Robin counter for the linefill prior to the Invalidate All will not be reset, but will continue from its existing value.

i.e.

Linefill to Index y = <X,X+1,X+2,X+3>

CP15 Invalidate All

Linefill to Index z = <X,X+1,X+2,X+3>

(where y is not equal to z)

5.1.3 Conditions

The Index for a linefill after the CP15 Invalidate All operation must differ from that for the preceding linefill, but the Index subset must be the same in each case.

5.1.4 Implications

The absence of the round-robin count reset does not have any measurable impact on any software routines, as it is not a functional requirement.

5.1.5 Workaround

No workaround suggested, as the erratum does not have any measurable impact on software routines.

5.2 Destination register update in bounced coprocessor 14 MRC operations (4) (Revisions r0p0, r0p2)

5.2.1 Summary

A bounced coprocessor 14 MRC will update the destination register. This is not a serious issue, see implications for details.

5.2.2 Description

The Jazelle hardware is configured by writing to an internal coprocessor, CP14. Some of these instructions may be bounced e.g. a non-privileged process trying to read the Jazelle ID register. A bounced CP14 MRC instruction incorrectly updates the destination register as though the instruction had completed; however the instruction is properly bounced to the undefined instruction handler.

5.2.3 Conditions

The erratum applies to all bounced CP14 instructions. The only CP14 instructions that can bounce on an ARM9EJ-S are Jazelle related instructions.

5.2.4 Implications

Once a CP14 MRC has been bounced control is correctly passed to the undefined instruction handler. The handler would normally do one of two things.

1. Create a value to go in the destination register and return control to the original process.
2. Terminate the process.

In case 2 the fact that the destination register has been updated does not matter because all register values will be thrown away. In case 1 it should also not matter because the handler will be re-writing the destination register with the value that it wants the process to see, over-writing the value put there by the MRC instruction.

If the handler requires that the destination register is not corrupted by the bounced MRC then this erratum is fatal. This scenario is not considered likely or sensible.

5.2.5 Workaround

There should not be a need for a workaround.

5.3 DHBL byte lane strobes endianness inconsistent for buffered writes (7) (Revisions r0p0, r0p2, r0p3)

5.3.1 Summary

The DHBL byte lane strobes may not be correct for AHB writes corresponding to byte and half-word stores which are buffered via the main write-buffer, if the endianness of the system is changed without prior draining of the write-buffer.

5.3.2 Description

For correct operation the DHBL byte lane strobe signals for AHB transactions corresponding to byte and half word stores should always reflect the endianness of the system, when the store is issued to the ARM926EJ-S memory sub-system by the ARM9EJ-S processor. If the stores are buffered via the main write-buffer then it is possible for the endianness of the system to be changed before the corresponding write transactions take place on the data side AHB interface. The ARM926EJ-S BIU uses the current endian setting of the system to produce DHBL, and in the event of this having changed since the stores were placed in the write-buffer, the endianness setting used to produce DHBL will be incorrect.

5.3.3 Implication

For systems which utilize dynamic endian switching, the ARM926EJ-S write-buffer must be drained prior to changing the endianness of the system.

5.3.4 Workaround

A CP15 MCR instruction can be used to drain the write-buffer prior to changing endianness. An example code sequence is show below:

```
MCR     p15, 0, r0, c7, c10, 4     ; drain write buffer
MRC     p15, 0, r0, c1, c0, 0      ; read control register
ORR      r0, r0, #1 << 7           ; set B bit for big-endian operation
MCR     p15, 0, r0, c1, c0, 0      ; write control register
```


5.4 **Inefficient fill of instruction prefetch buffer (8)** (Revisions r0p0, r0p2, r0p3)

5.4.1 Summary

The ARM926EJ-S contains a 4 entry buffer used for prefetching non-cacheable instructions. If prefetching is not disabled then non-cacheable instruction fetches will appear on the AHB as either SINGLE, or INCR4 transfers. INCR4 transfers should occur if the prefetch buffer is empty. The erratum is such that for certain cases when the prefetch buffer is empty, SINGLE transfers are performed instead of INCR4, leading to inefficient filling of the buffer.

5.4.2 Description

INCR4 transfers are not performed if the prefetch buffer is empty and a sequential instruction fetch is issued by the ARM9EJ-S core. For further information about instruction fetching, see ref. 2. In this case instruction fetches appear on the AHB as SINGLE transfers. INCR4 transfers are however still made for non-sequential instruction fetches, and when the prefetch buffer has been flushed due to a potential system hazard (for example a CP15 instruction has been executed).

5.4.3 Implication

This erratum does not cause incorrect operation of the device, but performance degradation may occur in systems which take advantage of IHBURST.

5.4.4 Workaround

There is no known workaround.

5.5 Writes unbuffered in bufferable region with data cache disabled (10) (Revisions r0p0, r0p2, r0p3)

5.5.1 Summary

If the data cache is disabled then the main write buffer is not used, and all data side accesses will be treated as NCNB (non-cacheable, non-bufferable), rather than NCB (non-cacheable, bufferable).

5.5.2 Description

If the MMU is enabled and the corresponding page-table descriptor for a given memory region indicates that the region is bufferable, and then the region will be treated as a non-bufferable region if the data-cache is disabled.

The following table shows the resultant cacheable and bufferable attributes for a given memory region, with the MMU enabled. Incorrect behavior due to the errata is shown using gray shading.

DCache Enabled	CB bits in PTE	Cacheable	Bufferable
No	11 (WB)	No	No
No	10 (WT)	No	No
No	01 (NCB)	No	No
No	00 (NCNB)	No	No
Yes	11 (WB)	Yes	Yes
Yes	10 (WT)	Yes	Yes
Yes	01 (NCB)	No	Yes
Yes	00 (NCNB)	No	No

WB = writeback

WT = write-through

NCB = non cacheable, bufferable

NCNB = non cacheable, non-bufferable

5.5.3 Implication

If the errata occurs then the main write-buffer will not be used, and the corresponding AHB HPROT attributes will also indicate non-cacheable, non-bufferable. This may result in some performance degradation.

5.5.4 Workaround

Enable the data cache.

5.6 Setting CP15 Disable Write Back bit can cause data loss for dirty cachelines (14) (Revisions r0p0, r0p2, r0p3, r0p4, r0p5)

5.6.1 Summary

Setting the Disable Write Back bit of the Cache Debug Control Register can cause stale data to be written to main memory on a cache line eviction.

5.6.2 Description

The Disable Write Back (DWB) bit of the Cache Debug Control Register forces Write-Through behavior for areas of memory identified as Write-Back. Stores to Write-Back locations will also update main memory when the DWB bit is set.

If a Write-Back cache entry is dirty before the DWB bit is set, subsequent writes to that cache entry will also update main memory. If the cache entry is evicted, it is possible for stale data to be written back to main memory.

For example

```
 STR Rx,[Ry]   ; WB store hits D$ - 1/2 cache line corresponding to Ry marked as
dirty
    .
    .
 MRC p15, 7, Rd, c15, c0, 0   ; Read Cache Debug Control Register
 ORR Rd, Rd, #0x4
 MCR p15, 7, Rd, c15, c0, 0   ; value of Rd sets the DWB bit
    .
    .
 STR Rz,[Ry]   ; store to [Ry] with DWB bit set
 LDR Ra,[Rb]   ; load causes cache-line corresponding to Ry is evicted
```

In this case the value in main memory corresponding to the PA of Ry is that of Rx, rather than Rz.

5.6.3 Conditions

1. A write back entry is allocated to the cache
2. The write back entry is made dirty
3. The DWB bit (bit[2] of the Cache Debug Control Register) is then set
4. A subsequent write hits the cache entry and also updates external memory
5. The cache entry is later evicted.

In additional there are timing requirements that reduce the possibility of observing this erratum.

5.6.4 Implications

The DWB bit is normally only set during debug sessions. The effect of this erratum is that occasionally Write-Back memory locations will be corrupted.

5.6.5 Workaround

This errata can be worked around by setting the Disable DCache Linefill (DDL) bit (bit[0]) of the Cache Debug Control Register, whenever the DWB bit is set.

```
MRC p15, 7, Rd, c15, c0, 0   ; Read Cache Debug Control Register
ORR Rd, Rd, #0x5             ; Set DWB and DDL
MCR p15, 7, Rd, c15, c0, 0   ; value of Rd sets the DWB bit
```


5.7 Shift-IR produces incorrect output on DBGTDO when Scan chain 15, INTEST selected (15) (Revisions r0p0, r0p2, r0p3, r0p4, r0p5)

5.7.1 Summary

Incorrect data is output on DBGTDO during Shift-IR when scan chain 15 is selected.

5.7.2 Description

If scan chain 15 is selected with INTEST and the TAP state machine is in the Shift-IR state, the least significant bit of the Scan chain 15 data register is output on DBGTDO rather than the expected instruction register data.

5.7.3 Implications

If the DBGTDO output is daisy chained to another device, it is possible for devices after the ARM926EJ-S to be incorrectly programmed.

5.7.4 Workaround

To workaround this erratum another scan chain should be selected after accessing scan chain 15.

