80960CA/CF
SPECIFICATION UPDATE

Release Date: July, 1996

Order Number: 272875-001

The 80960CA/CF may contain design defects or errors known as errata. Characterized errata that may cause the 80960CA/CF's behavior to deviate from published specifications are documented in this specification update.

CONTENTS

REVISION HISTORY

Date of Revision	Version	Description
07/01/96	001	This is the new Specification Update document. It contains all identified errata published prior to this date.

PREFACE

As of July, 1996, Intel's Semiconductor Products Group has consolidated available historical device and documentation errata into this new document type called the Specification Update. We have endeavored to include all documented errata in the consolidation process, however, we make no representations or warranties concerning the completeness of the Specification Update.

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This is the first release of the 80960CA/CF Specification Update. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Order
80960CA/CF 32-Bit High-Performance Embedded Processor datasheet	272886-001
i960® Cx Microprocessor User's Manual	270710-003
i960® Cx Microprocessor User's Manual - Instruction Set Quick Reference	272220-002

Nomenclature

Errata are design defects or errors. These may cause the published (component, board, system) behavior to deviate from published specifications. Hardware and software designed to be used with any component, board, and system must consider all errata documented.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in any new release of the specification.

NOTE:

Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

SUMMARY TABLE OF CHANGES

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the 80960CA/CF product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

New No

(#) The new assigned errata number.

Prev No

(#) The previously assigned errata number. These anomalies fall into three categories: (A) Anomalies that can cause serious problems in the users applications. These problems involve data corruption, crashing of programs etc. A hardware or software workaround must be employed to prevent these problem types. (B) Anomalies that may effect performance or interface requirements, but not function. (C) Anomalies that are largely definitional and may or may not be corrected or even need to be fixed.

Steps

X: Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.

(No mark) or (Blank box): This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Page

(Page): Page location of item in this document.

Status

Doc: Document change or update will be implemented.

Fix: This erratum is intended to be fixed in a future step of the component.

Fixed: This erratum has been previously fixed.

NoFix: There are no plans to fix this erratum.

Eval: Plans to fix this erratum are under evaluation.

Row

| Change bar to left of table row indicates erratum is either new or modified from previous version of the document.

Errata (1 of 3)

New #	Prev #	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
9600001	A2	X	X					8	Fixed	Bus Backoff and Ready
9600002	A5A	X	X					8	Fixed	DACK Timing Vs Ready Wait States
9600003	A5B	X	X					8	Fixed	DACK Timing with Multiple Accesses Per Bus Request
9600004	A18	X	X					9	Fixed	DMA SSDEM Mode Packing and EOP
9600005	A19	X	X					9	Fixed	Suspend DMA, EOP and SDMA
9600006	A20A	X	X					10	Fixed	EOP, SDMA and Multi-Channel DMA Operations
9600007	A20B	X	X					10	Fixed	EOP, SDMA and Multi-Channel DMA Operations
9600008	A21	X	X					11	Fixed	Disabling Branch Lookahead causes DMA Problems (See also 960013)
9600009	A22	X	X					11	Fixed	NMI is Level-Triggered after RESET
9600010	A25	X	X		X			11	Fixed	Interrupts during SYSCTL Instruction to Load and Lock the Instruction Cache
9600011	A26	X	X		X			12	Fixed	32-32 Bit Transfers in Source-Synchronized Demand Mode with Unaligned Destination Address
9600012	A27				X			12	Fixed	TC (Terminal Count) pins are non-functional when using any channel in source synchronized demand mode or block mode
9600013	A28	X	X		X			13	Fixed	DMA and Instruction Scheduler Interaction
9600014	A29	X	X		X			15	Fixed	Destination Synchronized 128- to 128-Bit Quad Word Transfer Mode
9600015	A30				X			15	Fixed	Disabling the Instruction Cache
9600016	A31				X			15	Fixed	Interrupt Can Cause User Process Stall when DMA Active Under Certain Complex Conditions
9600017	A32			X				16	NoFix	Register Cache
9600018	B1	X	X					16	Fixed	Cache Functionality/Re-fetching Cached Instructions
9600019	B3	X	X					17	Fixed	COBR Branch Trace

Errata (2 of 3)

New #	Prev #	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
9600020	B5	X	X					18	Fixed	EOP and DREQ Deassertion
9600021	B6	X	X					18	Fixed	DREQ Sampling
9600022	B8	X	X					18	Fixed	SYSCTL followed by RET May Incorrectly Fault
9600023	B10	X	X					18	Fixed	Terminal Count Operation
9600024	B11A	X	X					19	Fixed	Testif and Faultif Cause Incorrect Branch Trace
9600025	B11B	X	X					20	Fixed	Branch Tracing Missies Branch Instruction
9600026	B12	X	X					20	Fixed	Disabling Interrupts with the sysctl Instruction is Non-atomic
9600027	B13	X	X		X			20	Fixed	Unaligned DMA Transfer Modes When Using Incrementing Source and Destination Address
9600028	B14				X	X	X	21	NoFix	Data Cache Global Disable Bit is a 0 After a Device Reset
9600029	C1	X	X	X	X	X	X	21	NoFix	Instruction Cache
9600030	C2	X	X		X	X	X	21	NoFix	Register Cache
9600031	C3	X	X	X	X	X	X	21	NoFix	Data Address Breakpoint Fault on a CALLX
9600032	C4	X	X	X	X	X	X	22	NoFix	Data Address Breakpoints on Stacks and Tables
9600033	C5	X	X	X	X	X	X	22	NoFix	Pipelined Region Limitation
9600034	C6	X	X	X	X	X	X	23	NoFix	EOP and Buffer Complete Interrupt
9600035	C7	X	X	X	X	X	X	23	NoFix	MULI Fault Return
9600036	C8	X	X		X			23	NoFix	Fault Handler Executes before Interrupt Handler
9600037	C9	X	X	X	X	X	X	24	NoFix	Extra RIP Read
9600038	C10	X	X	X	X	X	X	24	NoFix	Disabling and Enabling Interrupts via Modifying the Interrupt Mask Register (sf1)
9600039	C10b	X	X	X	X	X	X	24	NoFix	Microcoded Instructions can be Corrupted after Writing to Interrupt Mask Register (sf1)
9600040	C11	X	X		X			25	Fixed	NMI During Built-In-Self-Test (BIST)
9600041	C12	X	X		X			25	Fixed	BTERM Functionality

Errata (3 of 3)

New #	Prev #	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
9600042	C13	X	X	X		X	X	26	NoFix	Modifying the Previous Frame Pointer (PFP) before Returning
9600043				X		X	X	27	NoFix	Erroneous TC Can Be Signaled When Using Multiple DMA Channels

Specification Changes

No.	Steppings			Page	Status	SPECIFICATION CHANGES
						None for this revision of this specification update.

Specification Clarifications

No.	Steppings			Page	Status	SPECIFICATION CLARIFICATIONS
	#	#	#			
						None for this revision of this specification update.

Documentation Changes

No.	Document Revision	Page	DOCUMENTATION CHANGES
001	270710-003	29	Chapter 4 - Instruction Set Summary
002	270710-003	29	Chapter 9 - Instruction Set Reference
003	270710-003	29	Chapter 11 - External Bus Description
004	270710-003	29	Chapter 12 - Interrupt Controller
005	270710-003	29	Chapter 12 - Page 12-22 Table 12-2
006	270710-003	30	Chapter 13 - DMA Controller
007	270710-003	30	Appendix A - Instruction Execution and Performance Optimization
008	270710-003	30	Appendix A - Page 1-42, Section A.2.6.10
009	270710-003	30	Appendix B - Bus Interface Examples
010	270710-003	30	Appendix F - Register and Data Structures
011	272220-002	30	Instruction Set Quick Reference - Page 9
012	272220-002	31	Instruction Set Quick Reference - Page 11

IDENTIFICATION INFORMATION

Markings

The various 80960CA steppings are identified by a topside mark as indicated below.

	C-2 Stepping	**C-3 Stepping**	**D-Stepping**
KU 80960CA -25	SV914	SW033	D2
KU 80960CA -16	SV913	SW032	D2
A 80960CA-33	SV908	SW031	D2
A 80960CA-25	SV907	SW030	D2
A 80960CA-16	SV906	SW029	D2
TA 80960CA-16		SW147	VA80960CA-16*

* TA80960CA16,S W147 will no longer be offered. It is replaced by VA80960CA16, which has a temperature range of -40 to +125C, and manufactured on the D-stepping. In addition, a 25 MHz extended temperature part will be offered as VA80960CA25.

ERRATA

9600001. A2 Bus Backoff and /READY

PROBLEM: If either /RDY or /BTERM is asserted between the deassertion of /BOFF and the completion of the /ADS strobe, the regenerated access will be lost or corrupted.

IMPLICATION: Improper operation and/or data corruption may result.

WORKAROUND: The work around is to ensure that /RDY or /BTERM is not asserted from the time /BOFF is asserted until the time that /ADS is regenerated.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600002. A5A DACK Timing Vs Ready Wait States

PROBLEM: The deassertion of DACK can be triggered by conditions other than the end of a Ready access. Because DACK is used as a chip select, simple connection of DACK to system DACK does not work for Ready controlled regions.

IMPLICATION: Additional glue logic is required.

WORKAROUND: External logic is necessary for proper operation. This logic should latch DACK during the address cycle.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600003. A5B DACK Timing with Multiple Accesses Per Bus Request

PROBLEM: DACK deasserts early on multiple bus accesses per bus request during the following conditions. If the DMA transfer type and external bus width are unequal such that the Bus Controller is required to issue multiple bus accesses per request, DACK deasserts on the rising edge of BLAST of the first access and stays deasserted for the duration of the request. For example, a DMA transfer type of 128 by 128 requires the Bus Controller to issue quad stores on a non-burst 32-bit external bus. Therefore DACK asserts with the first ADS and deasserts on the rising edge of BLAST of the first access.

IMPLICATION: Improper DMA transfers may occur.

WORKAROUND: Match the DMA mode transfer width with the external bus width. The correct functioning of DACK is as follows: DACK should assert at the falling edge of ADS and remain asserted throughout the entire bus request. DACK should deassert at the rising edge of BLAST of the last access.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600004. A18 DMA SSDEM Mode Packing and EOP

PROBLEM: Asserting EOP during a Source Synchronized Demand Mode DMA can corrupt the last word or byte of the transfer. The following table shows the packing modes and how the SAR (Source Address Register) and DAR (Destination Address Register) are aligned in order to manifest this condition. The EOP failure point denotes where a failure will occur with respect to EOP and individual transfers.

Packing type	SAR	DAR	EOP Fail Point
16-32	Aligned	Aligned	All Xfers
8-16	-	Aligned	1st Xfer Only
16-16	*Unaligned	Aligned	1st Xfer Only
32-32	Aligned	Unaligned	1st Xfer Only

* Block Mode

IMPLICATION: Data corruption may occur.

WORKAROUND: There are three workarounds. One is to not use EOP but rather an interrupt to signal an asynchronous termination of a transfer. Secondly, if it is known that the above condition is going to happen, throw away the last word or byte of the transfer and adjust the SAR and the BCR (Byte Count Register) accordingly. Finally, for those packing modes where EOP can only cause a failure during the first transfer, a dummy transfer can be inserted as the first transfer and no data will be corrupted.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600005. A19 Suspend DMA, EOP and SDMA

PROBLEM: An unrecoverable error can occur during the time that an interrupt is being posted if the Suspend DMA function is enabled (ICON bit 15) and either an EOP or SDMA event occurs. The following conditions must exist for the error to occur:

1. Suspend DMA function is enabled by setting ICON Register bit 15.
2. Two or more DMA channels are operating at the same time.
3. An interrupt is actually being posted.

4A. An EOP is received or,

4B. An SDMA is executed.

IMPLICATION: Unrecoverable error state may occur.

WORKAROUND: There are several workarounds. The Suspend DMA function should be enabled only if it can be guaranteed that an EOP or an SDMA will not occur. An SDMA can occur by first disabling the Suspend DMA function by resetting ICON Register bit 15. EOP, however, due to its asynchronous nature, cannot be used with the Suspend DMA feature enabled. Instead, a separate external hardware interrupt should be used to signal the end of a channel. An interrupt handler can then disable the channel by clearing the enable bit in the DMAC (DMA Command Register sf2) and wait until the active bit in the DMAC is cleared.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600006. A20A EOP, SDMA and Multi-Channel DMA Operations

PROBLEM: When a Setup DMA (SDMA) instruction is issued, transfer requests on all four channels are held off until SDMA completion. It is possible to enter a state in which a DMA Load request has completed but not the corresponding Store. Once the setup is done, the pending Store request will be serviced and completed. However, during the time that the SDMA instruction is executing, an EOP on the same channel may be accepted by the DMA controller. The EOP will be serviced BEFORE the DMA cycle has completed. The Store access will never be executed.

IMPLICATION: The transfer may be terminated prematurely and the data that is expected on the last Load may be lost.

WORKAROUND: The workaround for this anomaly is to make sure that an EOP does not occur during an SDMA instruction. An alternate method of setting up a DMA channel is to use the Chaining mode with wait. However, the user cannot change the DMA mode this way.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600007. A20B EOP, SDMA and Multi-Channel DMA Operations

PROBLEM: If, during the time that an SDMA is issued for a channel that was the last active channel, an EOP occurs on another channel, the Byte Count, Source Address, and Next Pointer Address pertaining to the SDMA may be corrupted.

IMPLICATION: Data corruption may occur.

WORKAROUND: The workaround for this anomaly is to make sure that an EOP does not occur during a SDMA instruction.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600008. A21 Disabling Branch Lookahead Causes DMA problems

PROBLEM: When using mulltiple DMA channels, bit 21 of the word at PRCB offset 0x20 must be zero (0). Otherwise, DMA data corruption occurs.

IMPLICATION: When using older debug monitors, DMA corruption can occur.

WORKAROUND: Make sure bit 21 of the PRCB offset 0x20 (instruction cache configuration word) is zero (0). This disables the branch lookahead logic, so some decrease in performance may be observed.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600009. A22 NMI is Level-Triggered after RESET

PROBLEM: NMI (Non-Maskable Interrupt) normally only occurs after a high-to-low (1-to-0) transition on the NMI pin. After RESET, however, NMI is level-triggered and a low (0) on the NMI pin will generate a Non-Maskable Interrupt.

IMPLICATION: Spurious NMI could occur.

WORKAROUND: Make sure the NMI pin is driven high during the RESET sequence. After RESET is deasserted, (driven high), the user must drive NMI high within 10 clocks. Otherwise, a Non-Maskable Interrupt will occur.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600010. A25 Interrupts During SYSCTL Instruction to Load and Lock the Instruction Cache

PROBLEM: Any hardware interrupts (maskable or unmaskable) which occur during the execution of a **SYSCTL** instruction to load and lock the instruction cache causes the processor to malfunction. **SYSCTL** instructions have the configure cache message and either the load and lock 1 Kbyte or load and lock 512 bytes cache mode configuration. The processor actually enters the interrupt handler, but enters in an unrecoverable state. The only way to completely recover from this malfunction is to reset the device.

IMPLICATION: Processor can enter an unrecoverable error state.

WORKAROUND: The workaround is to disable hardware interrupts while executing the **SYSCTL** instruction to load and lock the instruction cache. The user must also guarantee that his system never generates an NMI during the **SYSCTL** instruction.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600011. A26 32-32 Bit Transfers in Source-Synchronized Demand Mode with Unaligned Destination Address

PROBLEM: The DMA controller is optimized to perform unaligned 32-32 bit transfers. However, when in source-synchronized demand mode, the DMA controller requires an extra DREQ# to complete the transfer when the destination address is unaligned.

For example, assume an aligned source address and a destination address which is unaligned by one byte (has an address of xxxxxxxx1) and a byte count of 16. The DMA controller should transfer 3 bytes for the first DREQ#, 4 bytes the two middle DREQ#s, and 5 bytes for the last DREQ#. However, only 4 bytes are transferred for the 4th DREQ# and requires a 5th DREQ# to transfer the last byte.

All other source synchronized transfer modes with an unaligned destination address will work correctly (i.e., 16-16, 8-8, etc.).

IMPLICATION: Improper unaligned DMA behavior may result.

WORKAROUND: The workaround is to always ensure that your destination address is aligned on a 4 byte boundary, or have your hardware generate an extra DREQ#.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600012. A27 Terminal Count (TC) pins are non-functional when using any channel in source synchronized demand mode or block mode

PROBLEM: When any EOP/TC# pin is configured as an output (terminal count function), it may not ever be asserted if any of the four DMA channels is configured for either source synchronized demand mode, or block mode. The TC pin works correctly when all DMA channels are programmed only for destination synchronized demand mode or fly-by modes. This errata does not affect the operation of the EOP/TC# pin when programmed for the EOP function.

The TC pin will not be asserted when any unsynchronized DMA store is present on the external bus and the DMA controller issues the last load (the load in which TC should be asserted) of a source synchronized demand mode or block mode transfer. Note that even though TC is not asserted, the DACK# pin is asserted correctly for the load. An unsynchronized DMA store is defined to be a DMA store in which the DACK# pin is not asserted, as in source synchronized demand mode and block mode transfers.

IMPLICATION: For application purposes, this renders the TC pins non-functional when using these modes.

WORKAROUND: None

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600013. A28 DMA and Instruction Scheduler Interaction

PROBLEM: The DMA controller is implemented primarily in microcode. DMA operations are executed in microcode while providing core bandwidth for the user's program. Core resources are shared by implementing a separate hardware process for each DMA channel and one for the user code. This hardware mechanism allows the core to switch processes on clock boundaries to service either a DMA process or the user process. See Chapter 13 in the *i960® Cx Microprocessor User's Manual*.

Under very specific circumstances, a DMA process can cause a conditional branch instruction to branch along the incorrect path. The branch instruction unconditionally follows the path dictated by the branch prediction bit. The circumstances are:

1. The instruction scheduler must attempt to issue three instructions in parallel (a Reg-Mem-conditional branch triplet). For the device to execute three instructions in parallel, they must be in this order. For example:

```
CMPI    r4,r5          ;reg-side must affect condition codes
ST      r6,(r7)
BGE.f   some_where     ;must be conditional branch
                       ;which relies on condition codes
```

2. The reg-side instruction must begin on an odd word boundary.
3. The mem-side must be scoreboarded; i.e., the ST instruction cannot execute because of a resource limitation. The conditions in which the mem-side is scoreboarded are covered in the *i960® Cx Microprocessor User's Manual*.
4. A DMA event must occur while the mem-side is scoreboarded. A DMA event is defined to be the assertion of DREQ# on an active DMA channel and causing a process switch to a DMA task.
5. The branch prediction bit for the branch instruction must be incorrect for the result of the reg-side comparison.
6. Branch Lookahead must be enabled.

The instructions affected are listed below. Any combination of these instructions in the reg-mem-ctrl ordering are affected.

REG-side	**Mem side**	**Conditional**
ADDC	LDA offset	BE, BNE, BL, BLE
CMPI,CMP0	LDA (reg)	BG, BGE, BO, BNO
CMPDECI,CMPDECO	LDA offset(reg)	
CMPINCI,CMPINCO		
CONCMPI,CONCMPO	LD/ST offset	

CHKBIT LD/ST (reg)
SCANBYTE
SUBC

The LD/ST instructions include the byte, short, word, triple, and quad versions. The chances of this anomaly occurring in an application are exacerbated when the processor is in NIF mode, because the mem-side is always scoreboarded until the reg-side compare instruction is finished.

IMPLICATION: Incorrect program operation may occur.

WORKAROUND: For the 80960CA A through C-3 stepping, errata A21 prevents simply disabling Branch Lookahead for applications which have more than one DMA channel active at a time. Multichannel DMA applications will have to prevent this code sequence in their application to workaround this errata for the 80960CA. Intel will also provide switches for the development tools (C compilers) which will prevent this sequence of instructions from taking place, as well as a method for screening existing object code for the instruction triplet. Please contact your local Intel support person for information on obtaining these tools.

Code workaround - Because it is very difficult to predict when the mem-side will be scoreboarded, code workarounds should remove all cases of the code triplet. This guarantees that the errata will not occur in an application.

The simplest workaround is to insert a nop (mov g0,g0) between the reg-side and mem-side instructions. This prevents the triplet from being issued in parallel. The Intel tools will provide this kind of workaround as an assembler patch. An alternate workaround is to reorder the instructions so the triplet is not issued in parallel. A third workaround is to combine the separate reg-side and conditional branch instructions into a single compare and branch (cobr) instruction.

```
CMPI    r4,r5                ;insert a nop
MOV     g0,g0
ST      r6,(r7)
BGE.f   some_where

ST      r6,(r7)              ;reorder the instructions
CMPI    r4,r5
BGE.f   some_where

ST      r6,(r7)              ;Use a cobr instruction
CMPIBGE.f r4,r5,some_where  #.
```

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600014. A29 Destination Synchronized 128- to 128-Bit Quad-Word Transfer Mode

PROBLEM: The destination synchronized multi-cycle quad-word transfer mode will execute the final transfer (last sixteen bytes) immediately without waiting for the final DREQ#. A 32 byte transfer, for example, will only require a single DREQ# to transfer both quad words. The single DREQ# also causes two DACKS# to be asserted. When the DMA is programmed to transfer a single quad word (16 bytes), DMA will work correctly and require a single DREQ#. This erratum does not affect the fly-by, block, or source synchronized 128- to 128-bit quad word transfer modes.

IMPLICATION: Synchronous DMA operation may occur.

WORKAROUND: Do not use destination synchronized multi-cycle quad-word DMA.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600015. A30 Disabling the Instruction Cache

PROBLEM: The instruction cache cannot be disabled while the DMA is active. Prefetch algorithms operate differently with the instruction cache disabled. An interrupt, DMA event, and instruction prefetch all occurring around the same time can infinitely scoreboard the prefetch, causing user code to stop executing at the prefetch. This erratum cannot happen while the instruction cache is enabled.

IMPLICATION: No stepping of the 80960CA is effected by this errata.

WORKAROUND: The only way to recover from this condition is to reset the device.

STATUS: Fixed on the C-stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

9600016. A31 Interrupt Can Cause User Process Stall when DMA Active Under Certain Complex Conditions

PROBLEM: An interrupt request can cause the user program to stop execution when several specific events occur in sequence under strict timing requirements. The full explanation of this condition can be found in the paper titled "80960CF User Process Stall Bug" available through Intel application's FaxBack system; document number 2050. The FaxBack telephone number is 1-800-628-2283 or 916-356-3105 from outside the U.S. and Canada. The conditions required to cause the stall are:

1) A REG-format instruction followed by an "invalid" word in the cache. An "invalid" word is a word that was not written into the cache due to a change of program flow. This REG-format instruction must be on an even word (address ending in 0x0 or 0x8).
2) An instruction fetch issued in parallel with the execution of this REG-format instruction.
3) MEM side of CPU scoreboarded (e.g. due to bus queues full) causing fetch to be canceled.
4) DMA process switch occurring at least 3 clocks after the fetch, and MEM-side must still be scoreboarded.
5) Interrupt request occurs while MEM-side still scoreboarded.

This is most likely to occur under heavy bus traffic with 0 data to data wait states, heavy DMA activity, and heavy interrupt activity.

IMPLICATION: Only the 80960CF B step is affected.

WORKAROUND: DMA activity will continue until the DMA transfer is "done" (e.g. null chaining pointer, zero byte count, EOP). To recover from this condition, reset the device.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600017. A32 Register Cache

PROBLEM: Programming a register cache size of zero (0) causes the processor to generate an operation fault while performing a return. During proper operation, a value of zero (0) should yield one (1) set of register cache.

IMPLICATION: Improper program operation may occur.

WORKAROUND: Program the register cache size with a value of one (1). A value of zero (0) or one (1) yields one (1) set of register cache.

STATUS: There are no plans to correct this erratum. Refer to Summary Table of Changes to determine the affected stepping(s).

9600018. B1 Cache Functionality/Re-fetching Cached Instructions

PROBLEM: If loops exceed the cache size (256 instructions), even by one instruction, all instructions in that loop will be fetched again. This can be eliminated by setting bit 31 of the PRCB offset 0x20 to a one. However, setting this bit may impact performance. As a rule of thumb, it would be best to operate with bit 31 of the PRCB offset 0x20 cleared unless this cache re-fetching is clearly affecting the performance of the design.

Further analysis of how the 80960CA fetches instructions shows that other conditions exist where instructions are fetched from memory instead of being executed from the cache. This is a result of a change that improves the performance of the prefetch mechanism. Instructions are fetched into an instruction queue before they are executed. If an instruction is executed, it is loaded into the cache. If an instruction is not executed because of a branch or a call being taken, that instruction may not be loaded into the cache. This may leave instructions in the cache that are invalid, although the tag for the current cache line is valid. Extra fetches occur because of how the prefetch mechanism handles an invalid instruction when no cache miss occurs.

This erratum may affect performance as much if not greater than the setting of bit 31 mentioned above. Bit 31 enables or disables the original slow prefetching algorithm used in previous steppings.

Be aware that even with the above problem corrected, there is the possibility of seeing instructions being fetched that would appear to have been fetched previously. This extra fetching occurs because the cache is only guaranteed to be loaded if an instruction is executed, not just fetched.

IMPLICATION: Performance may be negatively impacted.

WORKAROUND: Setting bit 31 is considered the best workaround.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600019. B3 COBR Branch Trace

PROBLEM: A Branch Trace may be incorrectly reported on a cobr (Compare and Branch) instruction that is scoreboarded by a previous instruction. This occurs only when trace faults are enabled. Because the compare is scoreboarded, the branch never takes place; however, the trace fault is still reported.

IMPLICATION: Debugger may report branch trace when one did not occur.

WORKAROUND: Do not order a cobr instruction just after instructions such as multiply where one could possibly be using resources needed by the cobr instruction.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600020. B5 EOP and DREQ Deassertion

PROBLEM: A DMA cycle may terminate prematurely if the DMA controller accepts a DMA request and EOP occurs while DREQ is still asserted. Data may be buffered internally but never stored.

IMPLICATION: Data corruption may occur.

WORKAROUND: Do not assert EOP while DREQ is asserted.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600021. B6 DREQ Sampling

PROBLEM: In some DMA modes, the DREQ signal is sampled when the internal wait-state generator has timed out, regardless of the state of the external READY signal. This could lead to extra DMA cycles being generated erroneously by the incorrect sampling of DREQ during the time that READY is deasserted by an external memory system. Sampling of DREQ does not depend on the external system trying to complete a slow memory cycle; i.e., DREQ sampling does not depend on external READY.

IMPLICATION: Extra DMA transfers may occur.

WORKAROUND: Make sure DREQ is de-asserted before wait-state timer expires.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600022. B8 SYSCTL followed by RET May Incorrectly Fault

PROBLEM: A **sysctl** instruction followed by a **ret** instruction with no instructions in between may cause the false generation of an OPERATION.UNIMPLEMENTED fault.

IMPLICATION: User software operation may be negatively impacted.

WORKAROUND: Insert an instruction other than **ret** between sysctl and Ret instruction.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600023. B10 Terminal Count Operation

PROBLEM: On the 80960CA only (80960CF not affected): on steppings previous to the D-step the Terminal Count (TC) pin timing does not become active at the specified time. TC should have the same timing as DACK#, but does not. The following describes the condition:

Source Synchronized Demand Mode (SSDEM): The TC bit will not become active until all stores related to the last access are issued. For most aligned transfers in which there is no unpacking, the number of clock cycles between the deassertion of the last DACK and the active edge of TC will be 20. This assumes the 4:1 DMA cycle mode.

Block Mode: Same as SSDM with above.

Destination Synchronized Demand Mode (DSDEM): The last store is synchronized with the external bus. When the DACK deasserts, there are approximately 17 cycles to the active edge of TC.

The TC pin externally indicates when the DMA has fully completed its transfer and is no longer active. The TC bit in the DMA control register is set in parallel with pin activation.

Current Operation: On the D step -- which is not affected by this erratum, TC is active during the last DACK# of that channel and has the same timing as DACK#. For Source Synchronized transfers, TC is active during the last load. For Destination Synchronized transfers, TC is active during the last store.

If the last load/store bus request is executed as multiple bus accesses (such as a quad store to a non burst bus or unequal bus widths), DACK# and TC are active for the entire bus request (multiple accesses) with only one negative edge and one positive edge. An exception is if Nxda wait states are used. In this case, TC is only active for the first access. Note that TC is not a one clock cycle pulse but a pulse as long as the DACK.

IMPLICATION: Performance may be negatively impacted.

WORKAROUND: Rely on TC only after waiting the appropriate amount of time.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600024. B11A Testif and Faultif Cause Incorrect Branch Trace

PROBLEM: The testif and faultif instructions may cause branch traces.

IMPLICATION: Debugger may report branch trace when one did not occur.

WORKAROUND: None.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600025. B11B Branch Tracing Misses Branch Instruction

PROBLEM: When modpc is used to enable tracing and the Branch Tracing bit is set in the Process Control Register, a branch in parallel with the modpc instruction is not traced. If tracing is already enabled and modtc is used to turn on branch tracing, the same thing happens with a branch in parallel with the modtc.

IMPLICATION: Debugger may miss a branch breakpoint.

WORKAROUND: Separate the branch instruction from modtc or modpc by at least one instruction.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600026. B12 Disabling Interrupts with sysctl is non-atomic

PROBLEM: When the sysctl instruction is used to disable interrupts, there is a two-clock period window of time where interrupts can still be received and posted.

IMPLICATION: User software must take this period into account.

WORKAROUND: None.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600027. B13 Unaligned DMA Transfer Modes When Using Incrementing Source and Destination Address

PROBLEM: Existing DMA functionality limits the alignment address of the synchronizing device to the DMA transfer width; transfer width is equal to the synchronized side of the DMA transfer. For example, for a 32-32 source synchronized transfer, the source address must be aligned to word boundary (synchronizing side), while the destination can be aligned to a byte boundary. For a 16-32 destination synchronized transfer, the destination must be aligned to a word boundary, and the source address can be aligned to a byte boundary.

Byte count must also be aligned to the transfer width boundary or evenly divisible by the transfer.

IMPLICATION: Incorrect DMA operation will result.

WORKAROUND: Use only aligned source addresses.

STATUS: Fixed on the D stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

9600028. B14 Data Cache Global Disable Bit is 0 After Device Reset

PROBLEM: The data cache comes up globally enabled after device reset

IMPLICATION: Data cache operation may not function as expected.

WORKAROUND: Applications that need the data cache disabled after a device reset should disable the data cache by executing a setbit 30,sf2,sf2 before referencing any data that resides in a cacheable region.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600029. C1 Instruction Cache

PROBLEM: When the instruction cache is disabled, two cache lines (16 words) of the cache remain enabled. These two lines are not part of the 1024 byte cache.

IMPLICATION: Instructions of an extremely tight loop may be cached even when cache is disabled.

WORKAROUND: None.

STATUS: There are no plans to fix. . Refer to Summary Table of Changes to determine the affected stepping(s).

9600030. C2 Register Cache

PROBLEM: Local Register Cache Size - programming a register cache size of 0 causes 15 sets to be allocated. During proper operation, the register cache should be disabled by programming 0 frames.

IMPLICATION: User software must take this restriction into account.

WORKAROUND: Use data cache enable bit to disable data cache.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600031. C3 Data Address Breakpoint Fault on a CALLX

PROBLEM: When a Data Address breakpoint fault occurs on a CALLX, or any call with a frame flush, the return IP (RIP) reported will be that of the call. Per specification, the RIP should point to the first instruction of the called procedure.

IMPLICATION: Debugger must provide additional logic.

WORKAROUND: The trace fault handler must detect this condition and adjust the RIP before returning.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600032. C4 Data Address Breakpoints on Stacks and Tables

PROBLEM: If a data address breakpoint occurs on a memory access associated with the processor's interrupt or fault context switches or execution of a CALLS instruction, the fault may not be signaled. If it is signaled, the associated fault record may be incorrect and the Trace Controls register (TC) may be corrupted. During proper operation, the data address breakpoint fault would be signaled after completion of all operations associated with these microcoded sequences.

IMPLICATION: This may result in improper debugger behavior.

WORKAROUND: Data address breakpoints should not be set on the system procedure table, fault table, interrupt table, or stack locations which will contain interrupt records or fault records.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600033. C5 Pipelined Region Limitation

PROBLEM: Each pipelined region which has burst enabled must have Ready Control disabled in that region. During proper operation, the ready pins are ignored during reads in a pipelined region but can be used in a write to a pipelined region.

IMPLICATION: User software must take this restriction into account.

WORKAROUND: Disable READY for pipelined-burst memory regions.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600034. C6 EOP and Buffer Complete Interrupt

PROBLEM: There is no way to distinguish between an interrupt caused by an EOP and the interrupt that occurs at the end of a buffer in a DMA transfer when using source AND destination chaining.

IMPLICATION: User is not provided enough information about the interrupt.

WORKAROUND: To distinguish between the two interrupts, connect EOP to an interrupt pin.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600035. C7 MULI Fault Return

PROBLEM: A situation can occur where a MULI instruction generates a fault and the next instruction after the MULI is a complex memory instruction which is scoreboarded from some previous instruction. When the fault handler microcode executes, it may not return a correct RIP and, therefore, the complex memory instruction may never complete. All that is guaranteed is that the fault handler returns the proper IP for the MULI instruction.

IMPLICATION: Users must be aware of this fault handler restriction.

WORKAROUND: Fault handler for MULI should explicitly adjust RIP.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600036. C8 (B7) Fault Handler Executes before Interrupt Handler

PROBLEM: The *i960® Cx Microprocessor User's Manual* states that if an interrupt occurs at the same time that an instruction generates a fault, the interrupt will be serviced first and then the fault handler executes. Currently, the handler is executed first, then the interrupt service routine is executed.

IMPLICATION: This could result in abnormally long interrupt latencies and negatively impact performance.

WORKAROUND: None.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600037. C9 (A17) Extra RIP Read

PROBLEM: The return instruction (RET) does not perform as documented in the *i960® Cx Microprocessor User's Manual*. At the end of RET execution, an extra RIP (Return Instruction Pointer) read from the previous Stack Frame occurs. This may cause problems if the current PFP (Previous Frame Pointer) is not valid, such as during cold Reset. If the memory region that the PFP points to has Ready enabled and no memory exists in that region, the processor hangs.

IMPLICATION: User software must take this into account.

WORKAROUND: Make sure that the PFP is valid.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600038. C10 (B9) Disabling and Enabling Interrupts via Modifying the Interrupt Mask Register (sf1)

PROBLEM: One way to enable and disable interrupts is to mask interrupts by controlling certain bits of the Interrupt Mask Register (IMSK or sf1). Setting appropriate bits of sf1 to zero masks individual interrupts at the input pins. This occurs as soon as the IMSK register is modified by an instruction such as setbit or move. However, interrupts that occur at or just before the IMSK register is modified can still be serviced for up to eight clock cycles after the execution of the instruction that modifies the IMSK register.

IMPLICATION: User software must take this into consideration.

WORKAROUND: To ensure that critical code is not interrupted, insert eight no-op (mov g0, g0) instructions immediately after the instruction modifying the IMSK register. Conversely, after unmasking interrupts, it takes four clock cycles after modifying the IMSK register to generate interrupts.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600039. C10b Microcoded Instructions Can Be Corrupted after Writing to Interrupt Mask Register (sf1)

PROBLEM: One way to mask or disable interrupts is to clear certain bits in the Interrupt Mask Register (IMSK or sf1). If a hardware interrupt occurs during the same clock in which it was masked by a write to IMSK, any microcoded instruction executed within eight clock cycles following the write to IMSK will be corrupted.

IMPLICATION: User software must take this into consideration.

WORKAROUND: Make sure that no microcoded instructions are executed in eight clocks following any write to the IMSK register. The microcoded instructions are covered in Appendix A of the user's manual. It is probably best to simply insert 8 no-op (mov g0,g0) instructions after writing to the IMSK register, as this is also the workaround for errata C10.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600040. C11 NMI during Built-In-Self-Test (BIST)

PROBLEM: If an NMI occurs during BIST, the processor hangs and does not recover until the processor is reset.

IMPLICATION: Improper processor state may result.

WORKAROUND: Make sure NMI cannot occur during processor self test.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600041. C12 BTERM# Functionality

PROBLEM: This applies to 8-bit and 16-bit mode. This is best described by example:

```
lda 0x0, r3
ldq (r3), r4
```

Correct Operation

If BTERM# is asserted during data(0), ADS# will be asserted again for data(1) and the access will end by asserting BLAST# while reading data(3).

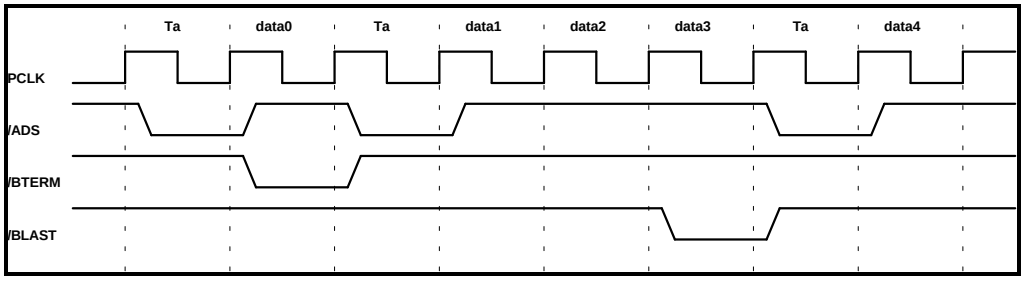


Figure 1: Timings for Correct Operation

Incorrect Operation

The access spans over data3. For example, BLAST# is asserted while data4 is read.

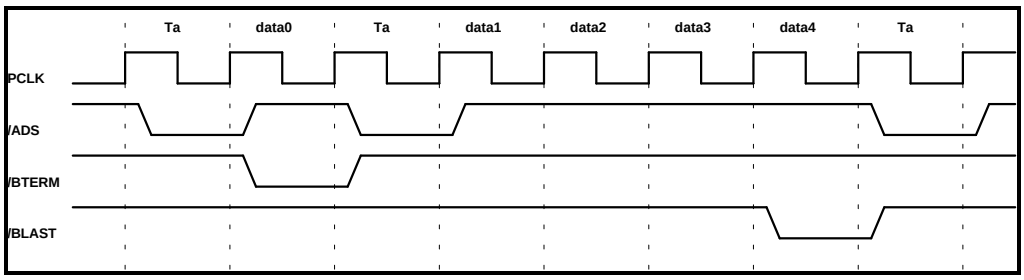


Figure 2: Timings for Incorrect Operation

IMPLICATION: Memory and I/O systems must accept these forms of access when using BTERM.

WORKAROUND: None.

STATUS: Refer to Summary Table of Changes to determine the affected stepping(s).

9600042. C13 Modifying the Previous Frame Pointer (PFP) before Returning

PROBLEM: If a return is performed immediately after the PFP's three least significant bits (Encoding of Return Status Field) is modified, the return instruction may operate improperly. This can occur because the return instruction may not see the new value in the PFP. This is most likely to occur when instructions are cached. See Example 1.

Example 1:

```
or g1, 7, pfp     /* modify return status in PFP */
ret
```

IMPLICATION: User software must take this into consideration.

WORKAROUND: PFP modification must occur at least three clock cycles before the return instruction is executed. One solution is to place three no-op (mov g0, g0) instructions between the instruction that modifies the PFP and the return instruction. See Example 2.

Example 2:

```
or g1, 7, pfp     /* modify return status in PFP */
mov g0, g0        /* no-op */
mov g0, g0        /* no-op */
mov g0, g0        /* no-op */
ret
```

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

9600043. Erroneous TC Can Be Signaled When Using Multiple DMA Channels

PROBLEM: A terminal count on one DMA channel causes the assertion of TC on another DMA channel under certain specific conditions. This only occurs when one or more channels are programmed as “source synchronized”. The problem is due to the lack of synchronization (by the CPU microcode) of the store operation when performing “Source Synchronized” DMA transfers. When an end of buffer condition occurs on a source-synchronized transfer, the DMA microcode asserts an internal signal used by the bus controller to cause the assertion of TC on the next DMA transfer (DACK asserted). The bus controller resets this handshaking bit after the access is completed on the bus. A DREQ for another channel which occurs near the last access of a source-synchronized transfer causes internal microcode process switching between the load and store of the source-synchronized transfer. When this occurs, the bus controller incorrectly fails to reset this internal TC request bit. This causes the next DMA transfer to incorrectly include an active TC.

WORKAROUND: Any of the following four workarounds may be used:

1) When using multiple DMA channels, do not use any source-synchronized transfers. Fly-by and destination-synchronized transfers do not exhibit this behavior.

2) Ensure that when performing source-synchronized transfers that no other DMA activity can occur on other channels.

3) Do not use the TC output signal, use interrupts instead. Program the transfer count to perform all but one transfer using the DMA. Enable interrupt on buffer complete bits. The interrupt handler should complete the access using an address similar to the DMA device, but change one address bit so it can be used as a pseudo TC. For example, if address bit 22 is not decoded in the system, it can be used as follows:

   0x00001000 Regular access to address 0x1000 (DMA access)

   0x00401000 Last access to address 0x1000 (From Interrupt handler)

4) Use a fixed high-priority DMA dummy channel (requires an unused DMA channel). Gate the TC pin of each source-synchronized DMA channel, with an “and” gate, to the DREQ pin of the highest priority channel which is programmed to perform a dummy fly-by transfer from memory. This will cause the erroneous TC to occur on an unused channel.

STATUS: There are no plans to fix. Refer to Summary Table of Changes to determine the affected stepping(s).

SPECIFICATION CHANGES

None for this revision of this specification update.

SPECIFICATION CLARIFICATIONS

None for this revision of this specification update.

DOCUMENTATION CHANGES

001. Chapter 4 - Instruction Set Summary

ITEM: Page 4-22, Table 4-4 For the CF, Mode 100 was incorrectly shown as locking 4 Kbytes; it now correctly shows 2 Kbytes.

In the third paragraph below the table, the second sentence was removed and replaced with bullets that more accurately describe the load and lock mechanism. This errata also occurs in Table 12-2 on page 12-22. The corrected pages are appended to this document.

002. Chapter 9 - Instruction Set Reference

ITEM: On page 9-78, in the sysctl description, the last paragraph on the page is incorrect and should be removed. It reads:

"When executing a sysctl instruction to load and lock either half or all of the cache, it is necessary to provide a cache load address. The last two bits of the cache load address must be 10base2 for the cache locking mechanism to work properly."

003. Chapter 11 - External Bus Description

ITEM: In Figure 11-2 (pg 11-8) WAIT# signal is incorrectly shown as transitioning; it now correctly shows that the signal is asserted high throughout. The corrected page is appended to this document.

004. Chapter 12 - Interrupt Controller

ITEM: Page 12-11 Figure 12-6. Vector Cache Enable bits (ICON.vce) are incorrectly defined.

Bit 0 was debounce; it now is correctly defined as Fetch From External Memory.

Bit 1 was Fast; is now correctly defined as Fetch From Internal RAM. The corrected page is appended to this document.

005. Chapter 12 - Page 12-22, Table 12-2

ITEM: For the CF, Mode 100 was incorrectly shown as locking 4 Kbytes; it now correctly shows 2 Kbytes. This errata also occurs on page 4-22. The corrected pages are appended to this document.

006. Chapter 13 - DMA Controller

ITEM: Page 13-22, Figure 13-9. DMA Command Register bits 30 (Data Cache Global Disable) and 31 (Data Cache Invalidate) are not defined in Figure 13-9 or in the text that follows the figure. These were correctly defined in the *i960® Cx Microprocessor User's Manual* supplement and unintentionally omitted from the latest revision of the user's manual. The corrected page is appended to this document.

007. Appendix A - Instruction Execution and Performance Optimization

ITEM: Page A-29, Table A-11. Mnemonic "bbe" is changed to "be." The corrected page is appended to this document.

008. Appendix A - Page A-42, Section A.2.6.10

ITEM: modpc definition changed to say, "requires 25 clocks."

modac definition changed to say, "requires 11 clocks."

The corrected page is appended to this document. See also the document change for the Instruction Set Quick Reference (Documentation Item #11).

009. Appendix B - Bus Interface Examples

ITEM: On pg B-5, Fig B-3, the ADS# signal incorrectly showed a deassertion in the 6th cycle and the 3rd deassertion in the 11th cycle. It now correctly shows NO deassertion in the 6th cycle and the last deassertion in the 10th cycle. (2nd deassertion removed; 3rd deassertion shifted left 1 cycle). The corrected page is appended to this document.

010. Appendix F - Register and Data Structures

ITEM: Appendix F is a compilation of all registers and data structures; therefore, this appendix has the same errata indicated in Figure 13-9 and Figure 12-6.

011. i960 ® Cx Microprocessor Users Guide Instruction Set Quick Reference (272220-002)

ITEM: Page 9: (changes indicated with heavier lines in the corrected page, which is appended to this document.): divi Instruction Issue changed to 13, emul Result Latency changed to 2,3,5,6

012. i960 ® Cx Microprocessor Users Guide Instruction Set Quick Reference (272220-002)

ITEM: Page 11: (changes indicated with heavier lines in the corrected page, which is appended to this document.): modac Instruction Issue changed to 11; Result Latency changed to 11. modpc Instruction Issue changed to 25; Result Latency changed to 25.

Table 4-4. Cache Configuration Modes

Mode Field	Mode Description	80960CA	80960CF
000_2	normal cache enabled	1 Kbyte	4 Kbytes
$XX1_2$	full cache disabled	1 Kbyte	4 Kbytes
100_2	Load and lock full cache (execute off-chip)	1 Kbyte[1]	2 Kbytes[2]
110_2	Load and lock half the cache; remainder is normal cache enabled	512 bytes	2 Kbytes
010_2	Reserved	1 Kbyte	4 Kbytes

NOTES:

1. On the CA, only interrupt procedures can execute in the locked portion of the cache.
2. On the CF, interrupt procedures and other code can operate in the locked portion of the cache.

ERRATA:
06/14/94:
Page 4-22, Table 4-4
For the CF, Mode 100_2 was incorrectly shown as locking 4 Kbytes; it now correctly shows 2 Kbytes.
In the third paragraph below the table, the second sentence was removed and replaced with bullets that more accurately describe the load and lock mechanism.
This errata also occurs on page 12-22.

Mode 000_2 configures the cache as two way set associative. Mode $XX1_2$ completely disables the cache. Either of these cache configurations can be specified when the processor initializes by programming the Cache Configuration Word in the PRCB. See section 14.2.6, “Process Control Block (PRCB)” (pg. 14-8). The modes allow the cache to be turned off temporarily to aid in debugging.

When the cache is disabled, the processor depends on a 16 word instruction buffer to provide decoding instructions. The instruction buffer operates as a small cache, organized as two sets of two way set associative cache, with a four word line size. When the main cache is disabled, small code loops may still execute entirely within the instruction buffer.

Modes 100_2 and 110_2 select cache load-and-lock options:

- On the CA: mode 100_2 loads and locks the full 1 Kbyte cache, 110_2 loads and locks half the cache.
- On the CF: either mode (100_2 and 110_2) loads and locks half the cache.

The **sysctl** instruction’s field 3 must contain an address; this address points to a quad-word aligned block of memory in the external address space. Instructions starting at this address are loaded into the cache. These instructions can only be accessed by selected interrupts which vector to these instructions’ addresses. The load-and-lock mechanism selectively optimizes latency and throughput for interrupts.

4.3.2.4 Reinitialize Processor

Executing **sysctl** with message type 03H reinitializes the processor. **sysctl** fields 3 and 4 must contain, respectively, the First Instruction Pointer and the PRCB Pointer. Reinitialization bypasses the i960 Cx processors’ built-in self-test. The PRCB is processed and the processor branches to the first instruction. See section 14.2, “INITIALIZATION” (pg. 14-2) for a complete description of the processor reinitialization steps.

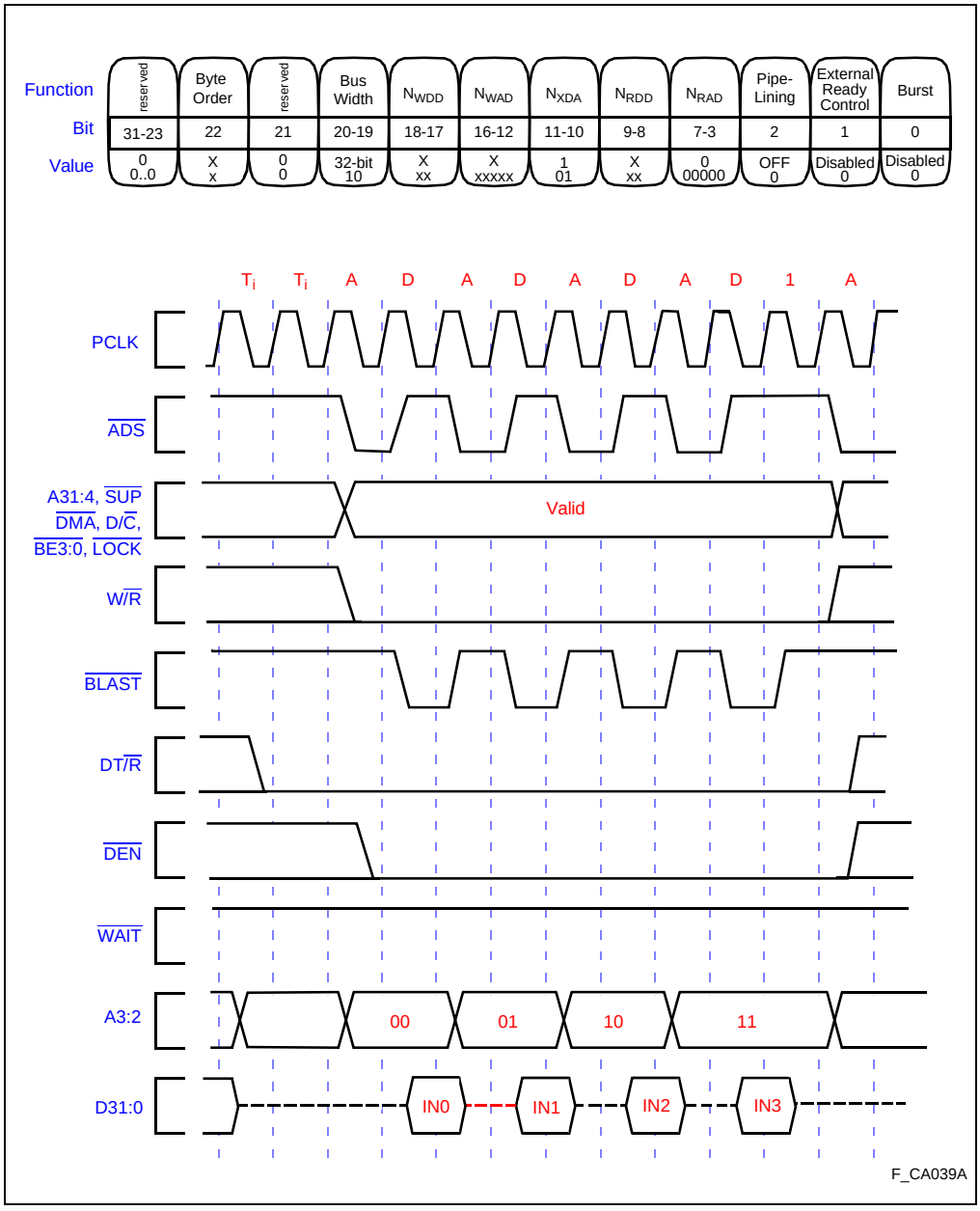


Figure 11-2. Quad-word Read from 32-bit Non-burst Memory

Errata 10/31/94 SRB.

Wait signal incorrectly shown as transitioning; it now correctly shows that the signal is asserted high throughout.

12.3.3 Programmer's Interface

The programmer's interface to the interrupt controller is through four control registers and two special function registers (all described in this section): ICON control register, IMAP0-IMAP2 control registers, IMSK special-function register (sf1) and IPND special function register(sf0).

12.3.4 Interrupt Control Register (ICON)

The ICON register (Figure 12-6) is a 32-bit control register that sets up the interrupt controller. Software can load this register using the **sysctl** instruction. The ICON register is also automatically loaded at initialization from the control table in external memory.

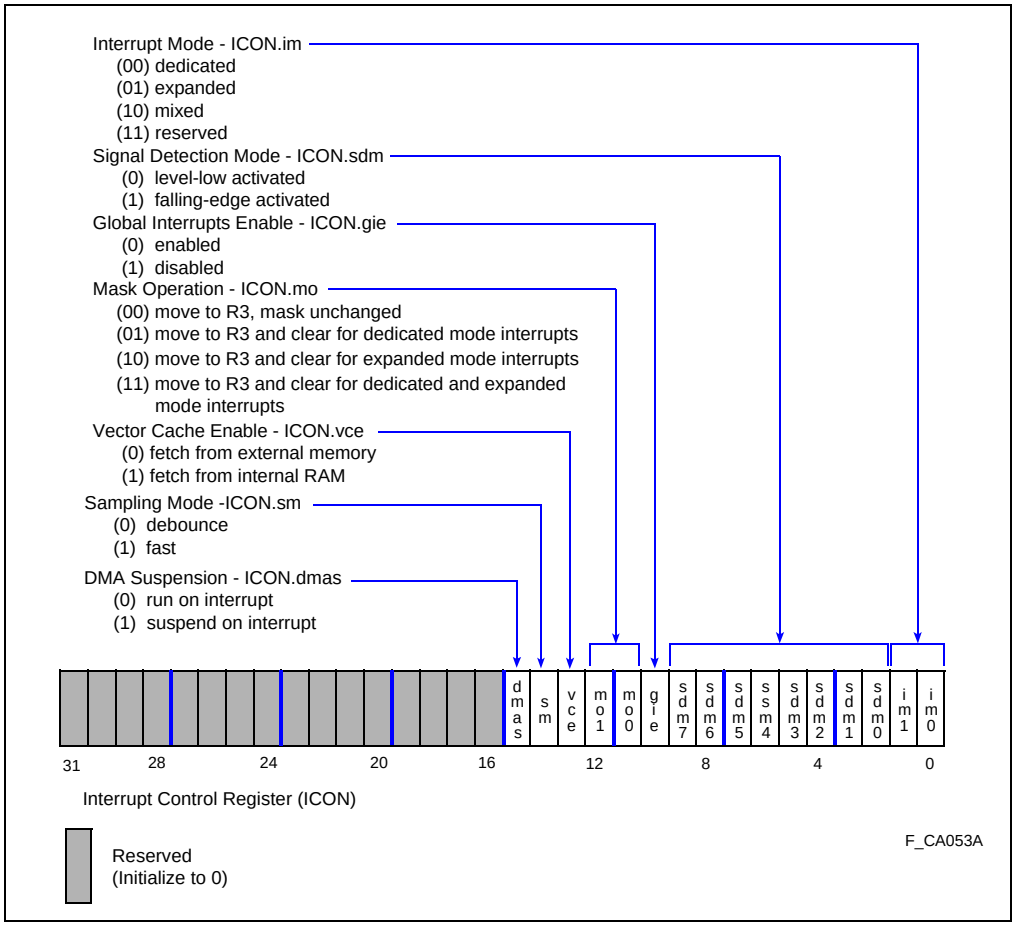


Figure 12-6. Interrupt Control (ICON) Register

Errata (12-06-94 SRB)
Vector Cache Enable bits (ICON.vce) incorrectly defined.
Bit 0 was "debounce"; it now is correctly defined as "Fetch From External Memory".
Bit 1 was "Fast"; is now correctly defined as "Fetch From Internal RAM".

The i960 CA processor supports 512 bytes or 1 Kbytes of locked cache. The i960 CF processor, with larger instruction cache, supports 2 Kbytes or 4 Kbytes of locked cache. As indicated in Table 12-2, the mode field of the **sysctl** instruction specifies the size of locked cache.

Table 12-2. Cache Configuration Modes

Mode Field	Mode Description	80960CA	80960CF
000_2	normal cache enabled	1 Kbyte	4 Kbytes
$XX1_2$	full cache disabled	1 Kbyte	4 Kbytes
100_2	Load and lock half cache (execute off-chip)	1 Kbyte[1]	2 Kbytes[2]
110_2	Load and lock half the cache; remainder is normal cache enabled	512 bytes	2 Kbytes
010_2	Reserved	1 Kbyte	4 Kbytes

NOTES:
1. On the CA, only interrupt procedures can execute in the locked portion of the cache.
2. On the CF, interrupt procedures and other code can operate in the locked portion of the cache.

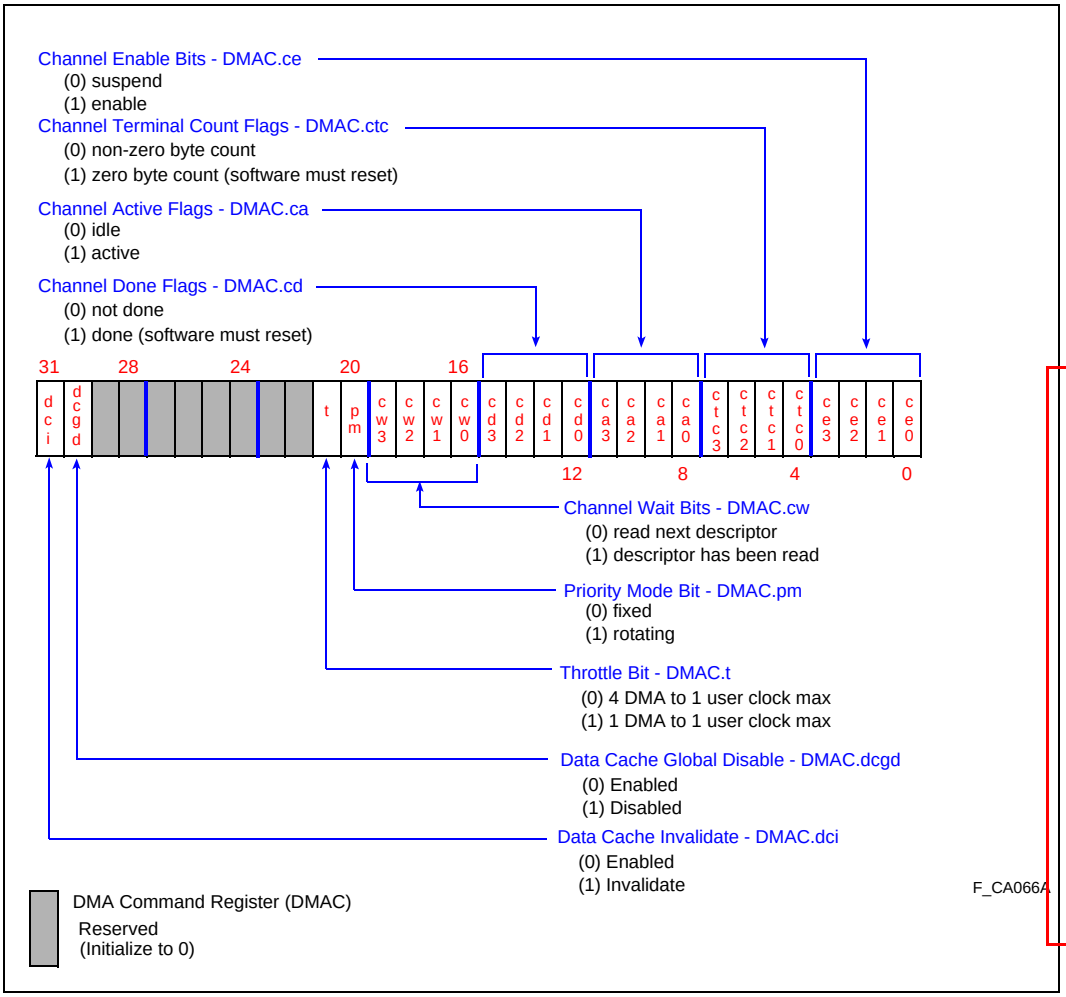
ERRATA:
06/14/94:
Page 12-22, Table 12-2
For the CF, Mode 100_2 was incorrectly shown as locking 4 Kbytes; it now correctly shows 2 Kbytes. This errata also occurs on page 4-22.

When **sysctl** executes (mode 110_2) with a command to lock half of the instruction cache, one way of the i960 CF processor's two-way set associative cache is preloaded and locked from the specified address. The other half of the instruction cache functions as a 2 Kbyte direct-mapped instruction cache. On the i960 CA processor, the instruction cache's unlocked portion functions as a 512 byte two-way set associative cache.

The i960 CF processor's instruction scheduler checks both ways of the cache for every instruction fetched. If an instruction is not found in either way, it is fetched from external memory and cached in the unlocked way.

The i960 CA processor only allows interrupt handlers to be locked in the cache. The interrupt vector's two least-significant bits must be set to 010_2 to cause the processor to fetch the interrupt procedure from locked cache rather than the normal memory/cache hierarchy. The interrupt procedure executes from the locked cache until a miss occurs in the locked section.

The cache remains locked until the cache mode is changed by the next sysctl instruction. The invalidate instruction cache **sysctl** message invalidates both the locked and unlocked halves of the cache. Refer to section 4.3, "SYSTEM CONTROL FUNCTIONS" (pg. 4-19) for details on using the **sysctl** instruction to configure the instruction cache.



ERRATA:

7/11/94

DMA Command Register bits 30 (Data Cache Global Disable) and 31 (Data Cache Invalidate) not defined in Figure 13-9 or in the text that follows the figure.

These were correctly defined in the *i960® CF Microprocessor Reference Manual Supplement* and unintentionally omitted from this manual.

Figure 13-9. DMA Command Register (DMAC)

The *channel enable bits* (bits 3-0) enable (1) or suspend (0) a DMA after a channel is set up. Bits 0 through 3 enable or disable channels 0 through 3, respectively. If an enable bit for a channel is cleared when a channel is active, the DMA is suspended after pending DMA requests for the channel are completed and all bus activity for the pending request is complete. The channel active bits indicate the channel is suspended. DMA operation resumes at the point it was suspended when the channel enable bit is set. To ensure that a DMA channel does not start immediately after it is set up, the enable bit for the channel must be cleared by software before **sdma** is issued. This is necessary because the DMA controller does not explicitly clear the enable bit after a DMA has completed.

Table A-11. CTRL Instructions

Mnemonic	Issue Clocks	Latency Clocks	Back-to-Back Throughput Clocks
be **bne** **bl** **ble** **bg** **bge** **bo** **bno**	1	2	2

ERRATA (12/15/95) SRB
In Appendix A, Table A-11, Mnemonic “bbe” is changed to “be”.

```
w: b      x
   ...
x: b      y
   ...
y: b      z
   ...
z: b      w
```

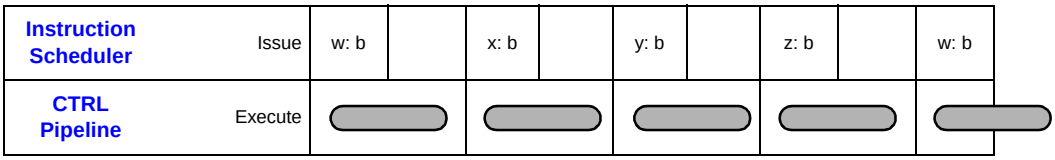


Figure A-14. CTRL Pipeline for Branches to Branches

Figures A-15, A-16 and A-17 show the IS issue stage and the CTRL pipeline for each case of possible IS branch lookahead detection. Assuming that the IS can see four instructions every clock from the instruction cache, the branch can be in the first, second or third group of instructions seen.

An *executable group* of instructions is a group of sequential instructions in the currently visible quad-word which can be issued in the same clock. See section A.2, “PARALLEL INSTRUCTION PROCESSING” (pg. A-14).

Figure A-15 shows the cases where a branch, when first seen by the IS, is in the first executable group of instructions. The IS issues the branch immediately, along with the first one (or two) instruction(s) ahead of it. Since the branch takes two clocks in the CTRL pipeline to execute, a one-clock break in the IS’s ability to issue instructions occurs. On the next clock, the IS issues a new group of instructions from the branch target.

A

A.2.6.7 Conditional Faults

fault* instructions are implemented with micro-flows and require one issue clock if the prediction bit is correct and no fault occurs. If the prediction bit is incorrect and no fault occurs, the instructions require two issue clocks. The time it takes to enter a fault handler varies greatly depending upon the state of the processor's parallel processing units.

A.2.6.8 Debug

mark and **fmark** are implemented with micro-flows. **mark** takes one issue clock if no trace fault is signaled. If a trace fault is signaled or **fmark** is executed, the processor performs an implicit call to the trace fault handler. As with conditional faults, the time required to enter a fault handler varies greatly.

A.2.6.9 Atomic

Atomic instructions are implemented with micro-flows. **atadd** takes seven issue clocks and **atmod** takes eight issue clocks to execute with an idle bus in a zero-wait state system. Memory wait states directly affect execution speed.

A.2.6.10 Processor Management

Processor management instructions implemented as micro-flows include: **modpc**, **modac**, **modtc**, **syncf**, **flushreg**, **sdma**, **udma** and **sysctl**.

ERRATA (12/15/95) SRB
modpc definition changed to say, "requires 25 clocks."
modac definition changed to say, "requires 11 clocks."

modpc requires 25 clocks.

modac requires 11 clocks.

modtc requires 15 clocks.

syncf takes 4 issue clocks if there are no possible outstanding faults. Otherwise, the instruction locks the IS until it is certain that no prior instruction will fault.

flushreg requires 24 clocks for each frame that is flushed. This translates to 120 cycles to flush five frames. Wait states in the memory being written affect this instruction's performance.

sdma executes in 22 clocks. In the case of back-to-back **sdma** instructions, 40 clocks are required.

udma requires 4 clocks.

sysctl Timings shown in Table A-19 assume a zero wait-state memory system.

B.1.4 Waveforms

Figure B-2 shows a Non-Pipelined SRAM Read Waveform; Figure B-3 shows a Non-Pipelined SRAM Write Waveform.

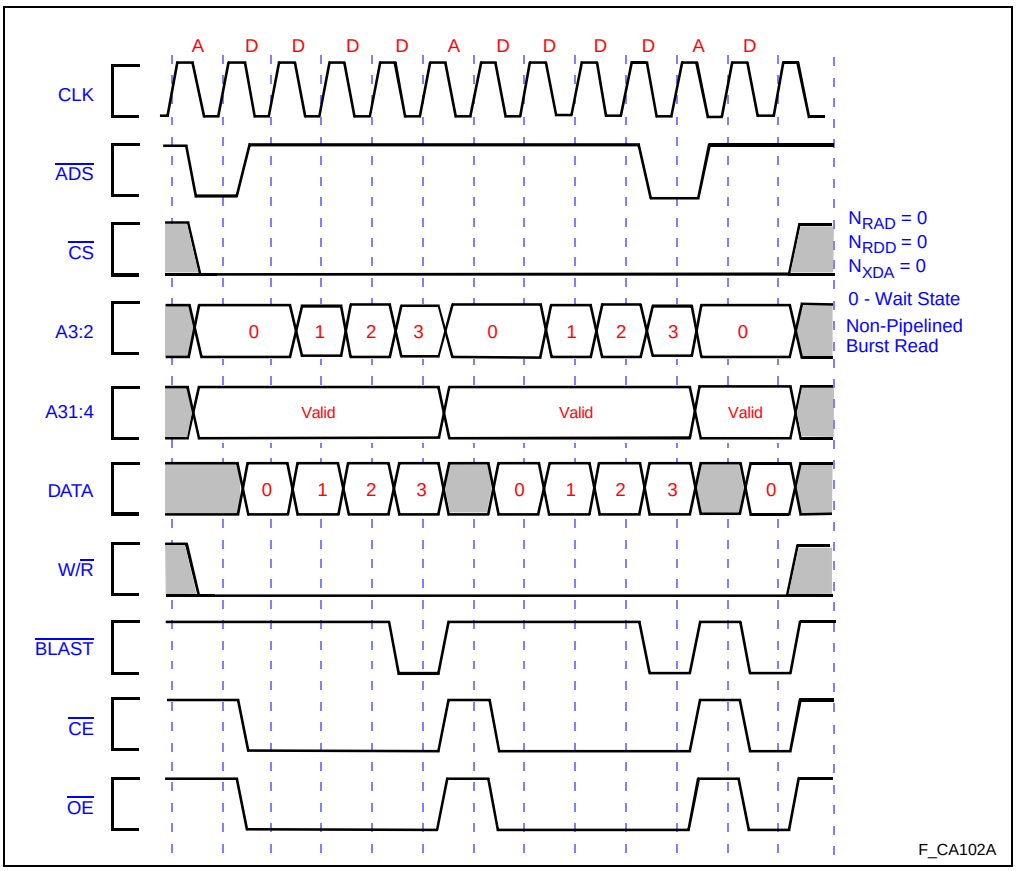
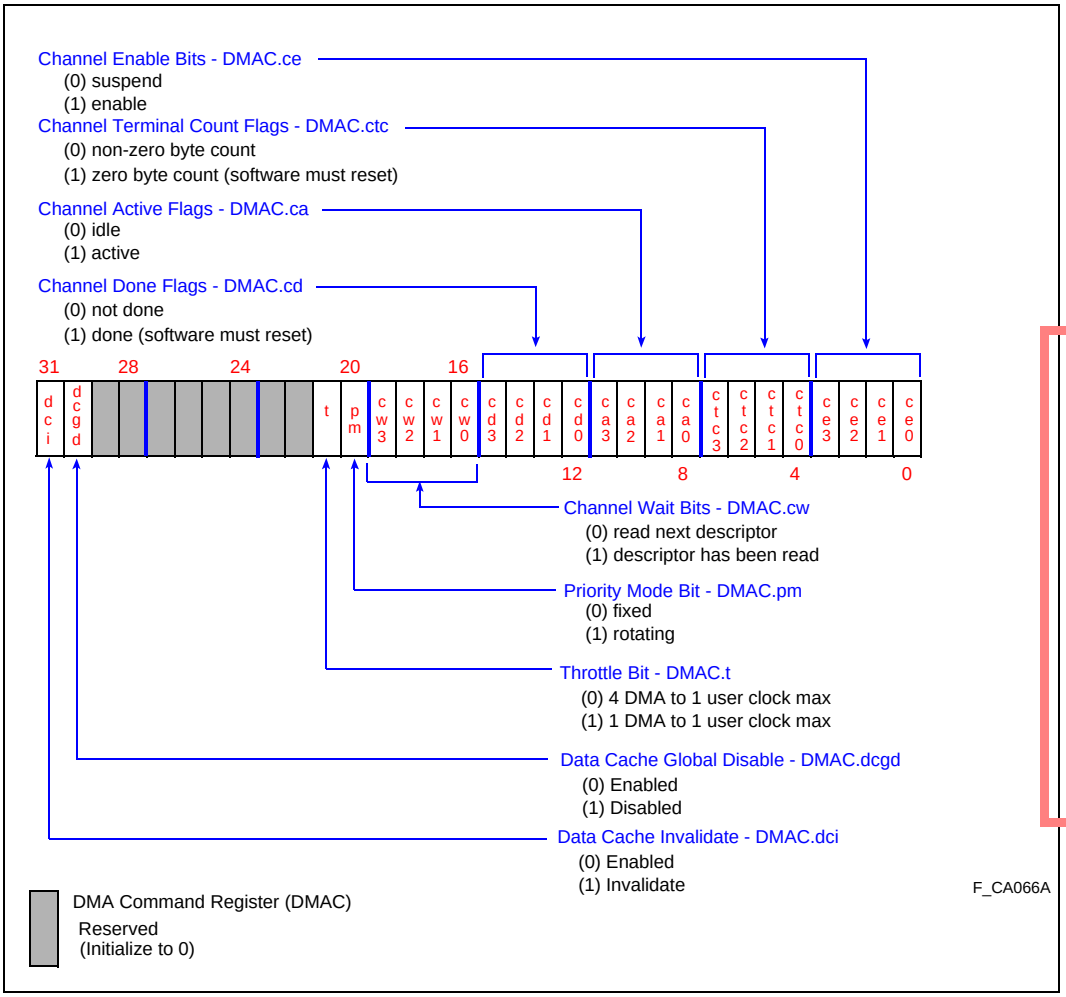


Figure B-2. Non-Pipelined SRAM Read Waveform

ERRATA (10-31-94) SRB
On pg B-5, Fig B-3, the ADS# signal incorrectly showed a deassertion in the 6th cycle and the 3rd deassertion in the 11th cycle.

It now correctly shows NO deassertion in the 6th cycle and the last deassertion in the 10th cycle. (2nd deassertion removed; 3rd deassertion shifted left 1 cycle).



ERRATA: 7/11/94

DMA Command Register bits 30 (Data Cache Global Disable) and 31 (Data Cache Invalidate) not defined in Figure 13-9 or in the text that follows the figure.

These were correctly defined in the i960® CF Microprocessor Reference Manual Supplement and unintentionally omitted from this manual.

Figure F-12. DMA Command Register (DMAC)

Section 13.10.1, “DMA Command Register (DMAC)” (pg. 13-21)

F

Errata (12-06-94 SRB)

Vector Cache Enable bits (ICON.vce) incorrectly defined.

Bit 0 was “debounce”; it now is correctly defined as “Fetch From External Memory”.

Bit 1 was “Fast”; is now correctly defined as “Fetch From Internal RAM”.

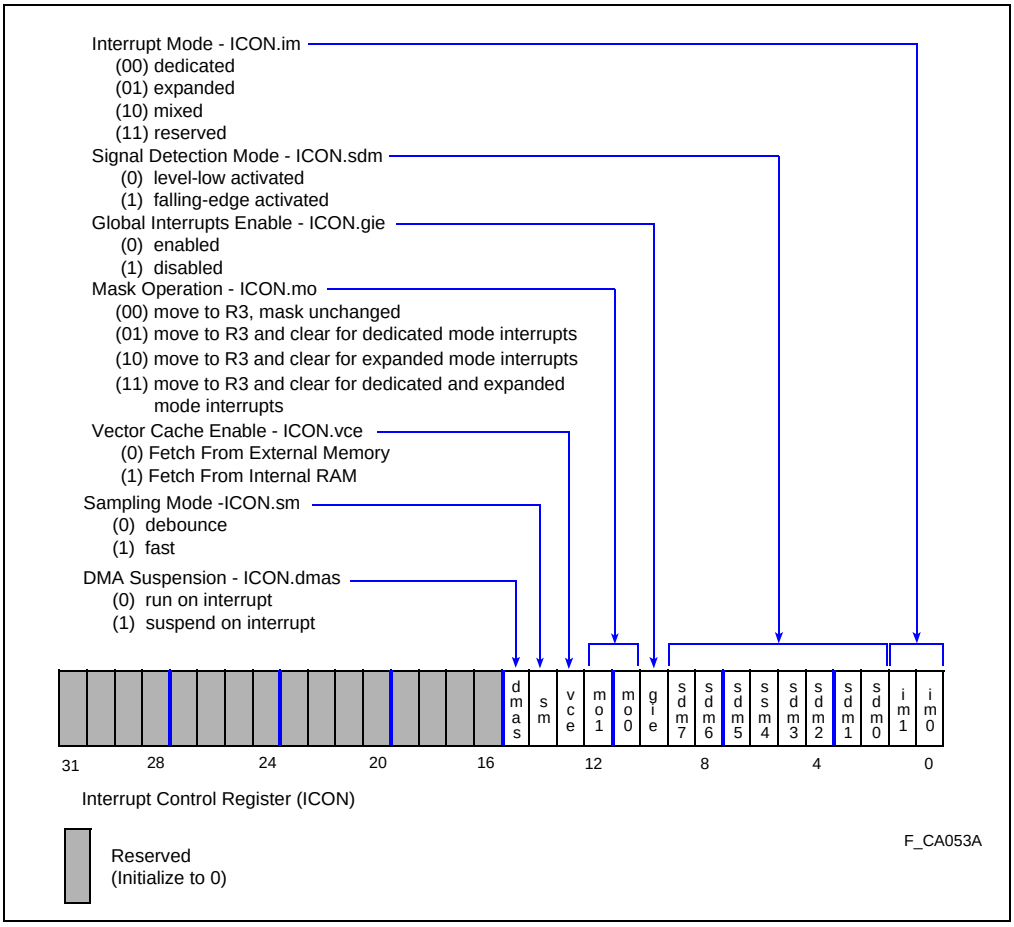


Figure F-16. Interrupt Control (ICON) Register

Section 12.3.4, “Interrupt Control Register (ICON)” (pg. 12-11)

Mnemonic	Description	Arithmetic Controls						Process Controls				Trace Controls		Faults						Opcode	Opcode Format	Instruction Execution		
		nif	om	of	cc 2	cc 1	cc 0	p	tfp	em	te	Events	Modes	T	O	A	C	P	Y			Mach. Type	Instruction Issue	Result Latency
concmpi	Conditional Compare Integer *src1*, reg/lit/sfr *src2*, reg/lit/sfr if (AC.cc2 = 0) { if (*src1* ≤ *src2*) AC.cc ← 010; else AC.cc ← 001; }	—	—	—	÷	÷	÷	—	÷	—	—	I	—	I	U	—	—	—	M	5A:3	REG	R	0.5 - 1	1
concmpo	Conditional Compare Ordinal *src1*, reg/lit/sfr *src2*, reg/lit/sfr if (AC.cc2 = 0) { if (*src1* ≤ *src2*) AC.cc ← 010; else AC.cc← 001; }	—	—	—	÷	÷	÷	—	÷	—	—	I	—	I	U	—	—	—	M	5A:2	REG	R	0.5 - 1	1
divi	Divide Integer *src1*, reg/lit/sfr *src2*, reg/lit/sfr *dst* reg/sfr if (*src1* =0) Arithmetic Zero Divide fault *dst* ← quotient (*src2*/*src1*) /* *src2*, *src1* and *dst* are 32 bits*/	—	—	¦	—	—	—	—	÷	—	—	I	—	I	U	IO ZD	—	—	M	74:B	REG	m	13	37
divo	Divide Ordinal *src1*, reg/lit/sfr *src2*, reg/lit/sfr *dst* reg/sfr if (*src1* = 0) Arithmetic Zero Divide fault *dst* ← quotient (*src2*/*src1*) /* *src2*, *src1* and *dst* are 32 bits */	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	ZD	—	—	M	70:B	REG	m	3	35,36
ediv	Extended Divide *src1*, reg/lit/sfr *src2*, reg/lit/sfr *dst* reg/sfr if (*src1* = 0) Arithmetic Zero Divide fault *dst* ← remainder (*src2*/*src1*) *dst + 1* ← quotient (*src2*/*src1*) /* *src2* is 64 bits; *src1*, *dst* and *dst + 1* are 32 bits */	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	ZD	—	—	M	67:1	REG	R	3	35,36
emul	Extended Multiply *src1*, reg/lit/sfr *src2*, reg/lit/sfr *dst* reg/sfr *dst* ← *src2* * *src1* /* *src2* and *src1* are 32 bits; *dst* is 64 bits */	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	—	—	—	M	67:0	REG	R	0.5 - 1	2,3,5,6
eshro	Extended Shift Right Ordinal *src1*, reg/lit/sfr *src2*, reg/lit/sfr *dst* reg/sfr *dst* ← *src2* >> (*src1* mod 32) /**src2* is 64 bits */	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	—	—	—	M	5D:8	REG	R	0.5 - 1	1
extract	Extract *bitpos*, reg/lit/sfr *len*, reg/lit/sfr *src/dst* reg *src/dst* ← (*src/dst* >> (*bitpos* mod 32)) and (2^(*len* mod 32) -1)	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	—	—	—	M	65:1	REG	μ	4	4
faulte	Fault If Equal if ((AC.cc and 010) ≠ 0) Constraint Range fault	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	—	R	—	—	1A	CTRL	μ	1 - 2	99 if fault taken

Mnemonic	Description	Arithmetic Controls						Process Controls				Trace Controls		Faults						Opcode	Opcode Format	Instruction Execution		
		nif	om	of	cc 2	cc 1	cc 0	p	tfp	em	te	Events	Modes	T	O	A	C	P	Y			Mach. Type	Instruction Issue	Result Latency
ldl	Load Long *src*, *dst* mem reg *dst*, *dst*+1 ← memory_long(*src*)	—	—	—	—	—	—	—	÷	—	—	I	—	I	OP U OC	—	—	—	—	98	MEM	M or µ	1 + efa	1 + efa + bus
ldob	Load Ordinal Byte *src*, *dst* mem reg *dst* ← memory_byte(*src*) zero-extended	—	—	—	—	—	—	—	÷	—	—	I	—	I	OP U OC	—	—	—	—	80	MEM	M or µ	1 + efa	1 + efa + bus
ldos	Load Ordinal Short *src*, *dst* mem reg *dst* ← memory_short(*src*) zero-extended	—	—	—	—	—	—	—	÷	—	—	I	—	I	OP U OC	—	—	—	—	88	MEM	M or µ	1 + efa	1 + efa + bus
ldq	Load Quad *src*, *dst* mem reg *dst*, *dst*+1, *dst*+2, *dst*+3 ← memory_quad(*src*)	—	—	—	—	—	—	—	÷	—	—	I	—	I	OP U OC	—	—	—	—	B0	MEM	M or µ	1 + efa	1 + efa + bus
ldt	Load Triple *src*, *dst* mem reg *dst*, *dst*+1, *dst*+2 ← memory_triple(*src*)	—	—	—	—	—	—	—	÷	—	—	I	—	I	OP U OC	—	—	—	—	A0	MEM	M or µ	1 + efa	1 + efa + bus
mark	Mark if ((PC.te = 1) and (TC.btm = 1)) { PC.tfp ← 1; TC.bte ← 1; Trace Breakpoint fault }	—	—	—	—	—	—	—	÷	—	—	IBR	—	IBR	U	—	—	—	—	66:B	REG	µ	17	17
modac	Modify AC *mask*, *src*, *dst* reg/lit/sfr reg/lit/sfr reg/sfr temp ← AC AC ← (*src* and *mask*) or (AC and not (*mask*)) *dst* ← temp	÷	÷	÷	÷	÷	÷	—	÷	—	—	I	—	I	U	—	—	—	M	64:5	REG	µ	11	11
modi	Modulo Integer *src1*, *src2*, *dst* reg/lit/sfr reg/lit/sfr reg/sfr if (*src2* = 0) Arithmetic Zero Divide fault *dst* ← *src2* mod *src* /* *src2*, *src1* and *dst* are 32 bits*/	—	—	¦	—	—	—	—	÷	—	—	I	—	I	U	IO ZD	—	—	M	74:9	REG	R	3	36
modify	Modify *mask*, *src*, *src/dst* reg/lit/sfr reg/lit/sfr reg *src/dst* ← (*src* and *mask*) or (*src/dst* and not(*mask*))	—	—	—	—	—	—	—	÷	—	—	I	—	I	U	—	—	—	M	65:0	REG	µ	3	3
modpc	Modify PC *src*, *mask*, *src/dst* reg/lit/sfr reg/lit/sfr reg if ((*mask* ≠ 0) and (PC.em ≠ Supervisor)) Type Mismatch fault temp ← PC PC ← (*mask* and *src/dst*) or (PC and not(*mask*)) *src/dst* ← temp	—	—	—	—	—	—	÷	÷	÷	÷	I	—	I	U	—	—	—	M	65:5	REG	µ	25	25