Intel386™ EX Embedded Processor

Specification Update

July 1998

Notice: The Intel386™ EX Embedded Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Order Number: 272873-009

Contents

Revision History 5
Preface 6
Summary Table of Changes 8
Identification Information 13
Errata 15
Specification Changes 40
Specification Clarifications 43
Documentation Changes 44

Revision History

The Intel386™ EX embedded processor has three production stepping versions. The first stepping, known as ‘A’ was made available in August, 1994. The second stepping, ‘B’ has been in production since the beginning of 1995. The B stepping was an interim version to fix the pipelining and some other errata problems found on the A stepping version. C-stepping was made available in early 1996, with C-1 production starting in June, 1996.

The Special Environment Intel386 EX has one production stepping version. The Special Environment ‘B’ stepping version began production in 1995; production ceased in January, 1998.

Table 1. Revision History

Revision Date	Version	Description
7/7/98	009	Corrected Documentation Change #1 and added Documentation Change 37.
2/23/98	008	Converted to new template. Clarified Errata #39 to apply to autotransmit mode only.
9/24/97	007	Added Errata 39 and 40. Added Documentation Changes 33–36.
8/12/97	006	Corrected code sequence in item #5 of Documentation Changes section (page 46). Added Documentation change 32.
7/21/97	005	Added Specification Change 8 and Documentation changes 30-31. Corrected code sequence in item #5 of Documentation Changes section (page 46).
12/23/96	004	Specification Changes IDIV instruction interrupt0 due to overflow
11/11/96	003	Corrected Device and Stepping identifier numbers and added a table to indicate IDCODE identifiers. Changed Documentation Errata #20 to read “...active” in first sentence. Added Documentation Errata 27, 28 and 29.
10/08/96	002	Added Special Environment Errata. Revised Errata 37. Added Specification Changes 5 and 6 (timing errata for EXTB and EXTC devices).
9/10/96	002	Added Documentation Changes #5 to #26
07/23/96	001	Revised timing errata for t34 SMM. Added additional information for errata #37 to clarify boundary conditions. Added note to errata 38 on HOLD signal boundary condition.
06/24/96	001	Added timing errata for t34 SMM. Added errata #35, 36, 37, and 38. Included additional documentation changes for Autotransmit.
05/01/96	001	This is the new Specification Update document. It contains all identified errata published prior to this date. Added errata #34
07/31/95	1.00	Added errata #1 to #33

Preface

As of July, 1996, Intel's Computing Enhancement 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 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

Table 2. Affected Documents/Related Documents

Title	Order
Intel386™ EX Embedded Microprocessor datasheet	272420-004, -006
Intel386™ EX Embedded Microprocessor User's Manual	272485-001, -002
Programming Flash Memory through the Intel386™ EX Embedded Microprocessor JTAG Port (AP-720)	272753-001
Intel Processor Identification With the CPUID Instruction (AP-485)	241618-003
Special Environment Intel386™ EX Embedded Microprocessor datasheet	271325-003
Intel386™ EX Embedded Microprocessor Evaluation Board Manual	272525-005
EXPLR1 Embedded PC Evaluation Platform Application Note	272777-001

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 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 Intel386 EX 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

Stepping

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 this erratum is either new or modified from the previous version of the document.

Errata

Errata Number	Rev. Date	Steppings			Page	Status	ERRATA
		A0	B1	C1			
1	07/31/95	X			15	Fixed	Watchdog Timer (WDT) Bus Monitor Failure During Bus Pipeline Operation
2	07/31/95	X			16	Fixed	Spurious Slave IR7 Interrupts When Using Internal WDT Interrupt
3	07/31/95	X	X		17	Fixed	WDT Lockout Sequence May Not Properly Reload Down Counter
4	07/31/95	X			17	Fixed	WDT Will Time Out During CPU Self-Test
5	07/31/95	X	X		18	Fixed	Timer Generated DMA Requests May Not Initiate Properly After RESET
6	07/31/95	X	X		18	Fixed	DMA Temporary Register Not Flushed When Software Aborts Transfer
7	07/31/95	X			19	Fixed	Chip Selects Remain Active During Bus HOLD And DMA Cascade Cycles
8	07/31/95	X	X		19	Fixed	Write Sequence To Enable Expanded I/O Space Is Not Sequential
9	07/31/95	X	X	X	21	NoFix	32-Bit WDT Register Is Oriented As High Word, Low Word In I/O Space
10	07/31/95	X	X		21	Fixed	Remap Configuration Register At 22H And 23H Have 10-Bit Address Decode
11	07/31/95	X	X	X	21	NoFix	CAS Lines Do Not Remain Active During Idle States Between Cascaded INTA Cycles
12	07/31/95	X	X		21	Fixed	Cannot Pipeline DMA Bus Cycles
13	07/31/95	X	X		22	Fixed	DSR1#/STXCLK Pin Has A Permanent Weak Pull-Up
14	07/31/95	X	X		22	Fixed	READY# Floats In T1 After Internal Bus Cycle Due to LBA#
15	07/31/95	X			22	Fixed	Pipelined Bus Operation Does Not Work Reliably, Causing System Problems
16	07/31/95	X			23	Fixed	DMA Registers At Addresses 08H, 09H, And 01AH Do Not Decode 16-Bit Addresses Correctly
17	07/31/95	X	X		23	Fixed	TDO Pin May Float in User Mode Causing Excessive Current Consumption
18	07/31/95	X	X		24	Fixed	Overflow Enable Bits (TOV0, TOV1, ROV0, ROV1) In DMAOVFE Register Do Not Function as Specified
19	07/31/95	X	X	X	24	NoFix	Upper DMA Byte Count Registers (DMA0BYC2, DMA1BYC2) Change to 0F0H When the Lower Byte Count Registers Underflow
20	07/31/95	X	X		24	Fixed	At High Baud Rates, There Is Not Enough Time to Turn the Transmitter Off By Resetting the TEN Bit
21	07/31/95	X	X		25	Fixed	Write Sequence for WDTCLR Lockout is Not Sequential

Errata

Errata Number	Rev. Date	Steppings			Page	Status	ERRATA
		A0	B1	C1			
22	07/31/95	X			25	Fixed	Setting the Channel Enable Bit (CE) in DMACMD1 Registers also Disables HOLD Requests
23	07/31/95	X			25	Fixed	DMA Synchronization Problem During Handoffs of Low Priority Channel to High Priority Channel, Resulting in Loss of Data
24	07/31/95	X	X		26	Fixed	TEN Bit In SSIOCON1 Register Turns Off Transmit Channel Too Soon
25	07/31/95	X			26	Fixed	Back-To-Back Writes To Interrupt Controller May Cause Failure In Second Write
26	07/31/95	X			27	Fixed	DMA And Bus Arbiter Reset May Cause Bus Contention
27	07/31/95	X	X		28	Fixed	Internal HOLD Request Synchronization May Cause Data/Code Fetches To Be Corrupted
28	07/31/95	X	X	X	28	NoFix	Writing To One DMA Channel's Register When A Request From The Other Channel Occurs May Cause An Improper Transfer.
29	07/31/95	X	X		28	Fixed	Insufficient Address Hold Time After WR# Goes High
30	07/31/95	X	X		29	Fixed	HLDA Inactive To HOLD Active Arbitration Could Improperly Float The Bus
31	07/31/95	X	X		30	Fixed	FLT# Does Not Float All Outputs
32	07/31/95	X	X		31	Fixed	SIO Break Could Be Missed
33	07/31/95	X	X		31	Fixed	Arbitration Between Powerdown And Refresh Cycles May Leave Control Signals In Wrong State
34	05/01/96		X		31	Fixed	Refresh Control With More Than 2 Wait States Could Cause An Erroneous Bus Cycle
35	06/24/96	X	X	X	32	No Fix	Internal READY# Generation Switching from Non-pipeline to Pipeline Can Be Temporarily Asserted
36	06/24/96			X	32	No Fix	NA# Must Not Be Asserted In A T2P State for DMA Pipelining
37	07/23/96			X	33	No Fix	Arbitration for HOLD During Refresh Cycle May Not Return an HLDA Signal
38	07/23/96			X	37	No Fix	READY# Source Conflicts May Cause Bus Cycle Failure
39	09/25/97 Updated: 2/26/98			X	38	No Fix	Not All Baud Rates from Baud-rate Generator Operate Correctly
40	09/25/97			X	39	No Fix	With Autotransmit Enabled, the First Bit Is Transmitted at the Wrong Baud Rate

Specification Changes

Change Number	Revision Date	Steppings			Page	Status	SPECIFICATION CHANGES
		A0	B1	C1			
1	06/24/96		X		40	Doc	AC Timings Modified for 5V +/-0.5V
2	06/24/96		X		40	Doc	AC Timings for modified for 3.3V +/-0.3V
3	06/24/96		X		41	Doc	AC Timings for 3.0V +/-0.3V
4	07/23/96			X	41	Doc	AC Timings for T34 on EXTB and EXTC devices is different when in SMM.
5	10/08/96			X	41	Doc	AC Timings Modified for EXTB Device
6	10/08/96			X	42	Doc	AC Timings Modified for EXTC Device
7	12/23/96	X	X	X	42	Doc	IDIV Instruction Interrupt0 Due To Overflow
8	7/21/97			X	42	Doc	FLT# Pin Should be Pulled Up to VCC

Specification Clarifications

Rev. Date	Steppings			Page	Status	SPECIFICATION CLARIFICATIONS
	A0	B1	C1			
						None for this revision of this specification update.

Documentation Changes

Change Number	Revision Date	Page	DOCUMENTATION CHANGES
1		44	272485-001, Correct Latching Of BS8# In Reference To 8 Bit Vs. 16 Bit Device Access (Modified on 7/7/98)
2		45	272485-001, Programming One Or Both Channels of the DMA Requires Both Channels To Be Masked. DMA Channels Must Also Be Masked While Reading/Writing To DMA Related Registers
3		45	272420-006, Values for ICC Represent the Maximum Supply Currents with the Device Held In Reset and Inputs Pulled to Their Inactive Levels
4		45	272485-002, Autotransmit Mode for SSIO Requires Baud Rates Above 3.2 MHz
5		46	272485-002, Section 4.5.2.1, Page 4-8, Figure 4-4.
6		46	272485-002, Section 4.6.4, Page 4-13, Figure 4-7.
7		46	272485-002, Section 5.3, Page 5-27, Figure 5-18.
8		46	272485-002, Section 6.2.4, Page 6-11.
9		47	272485-002, Section 6.3.1, Page 6-13.
10		47	272485-002, Section 6.3.6, Page 6-29, Figure 6-11.
11		47	272485-002, Section 6.3.6, Page 6-30, Figure 6-12.
12		48	272485-002, Section 6.6.1.2, Page 6-40, code example.
13		48	272485-002, Section 6.6.4, Page 6-43.

Documentation Changes

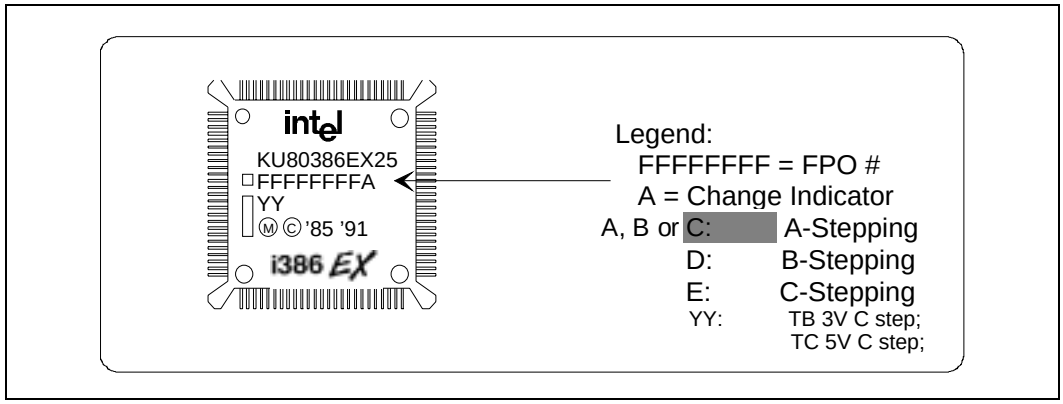
14		48	272485-002, Section 9.3.4, Page 9-21.
15		48	272485-002, Section 12.3, Page 12-28, Table 12-3.
16		48	272485-002, Section 12.3.11, Page 12-44.
17		48	272485-002, Section 12.3.13, Page 12-47.
18		48	272485-002, Section 13.2.2.2, Page 13-12.
19		48	272485-002, Section 15.5, Page 15-12.
20		48	272420-006, Section 7.0, Page 25.
21		49	272753-001, Figure 2, Page 7.
22		49	271325-003, Section 1.0, Pages 2-3.
23		49	271325-003, Section 1.0, Pages 5-6.
24		49	272485-002, Appendix A, Page A-10.
25		49	272485-002, Section 5.2.6, Page 5-20, Figure 5-12.
26		49	272525-005, Appendix C, Page C-1
27		49	272485-002, Section 13.2.2.1, Page 13-9
28		51	272485-002, Section 13.2.2.2, Page 13-12
29		51	272485-002, Section 14.6, Page 14-22
30		51	272420-006, Page 9, Table 4
31		52	272420-006, Page 12, Table 4
32		52	272485-002, Page 14-3, Note
33		52	272485-002, Page 9-3, Figure 9-1
34		52	272485-002, Section 13.2.1, Page 13-5
35		52	272753-001, Table 3, Page 6.
36		53	272485-002, Section 13.2.1, Page 13-12
37		53	272777-001, Table 9, Page 20

Identification Information

Markings

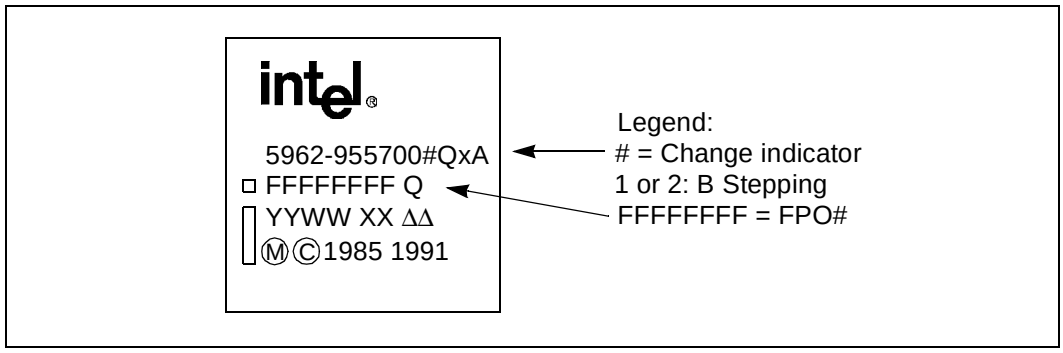
C-stepping production devices are marked with a *change indicator* ‘E’. The B-stepping production devices are designated with a ‘D’ and the A-stepping production version of the Intel386 EX is designated with a change indicator of ‘A’, ‘B’, or ‘C’. Figure 1 is an illustration indicating where the change indicator marking is located on the package markings. The change indicator is used to relate information found here, and in other documents like datasheets, to a specific device.

Figure 1. Commercial Markings



The B-stepping production version of the Special Environment Intel386 EX is designated with a change indicator of ‘1’ or ‘2’. Figure 2 is an illustration indicating where the change indicator marking is located on the package markings. The change indicator is used to relate information found here, and in other documents like datasheets, to a specific device.

Figure 2. Special Environment Markings



If there is not an FPO# line on the device — hence, no change indicator — but is marked with a Qxxxx (where x=any digit) on the second line, it is likely a pre-production sample part. Consult your Intel representative if you are unclear on the device stepping.

Intel386 EX processors may be identified electrically according to device type and stepping. Refer to the datasheet for instructions on how to obtain the identifier number.

Device and Stepping	Identifier in DX
80386EX A0	2309H
80386EX B1	2309H
80386EX C1	2309H

Refer to the datasheet for instructions on how to obtain identifier numbers. Another source is the *Intel Processor Identification With the CPUID Instruction* Application Note number AP-485.

JTAG registers

Intel provides a copy of the Boundary-Scan-Description-Language file for the A stepping of the Intel386™ EX Embedded Processor on the BBS, at (916) 356-3600. This file lists:

- The physical pin layout of all pins in the Boundary Scan Register
- The valid and reserved JTAG unit opcodes
- The expected contents of the IDCODE register for the Intel386 EX Embedded Processor
- A description of the BSR contents

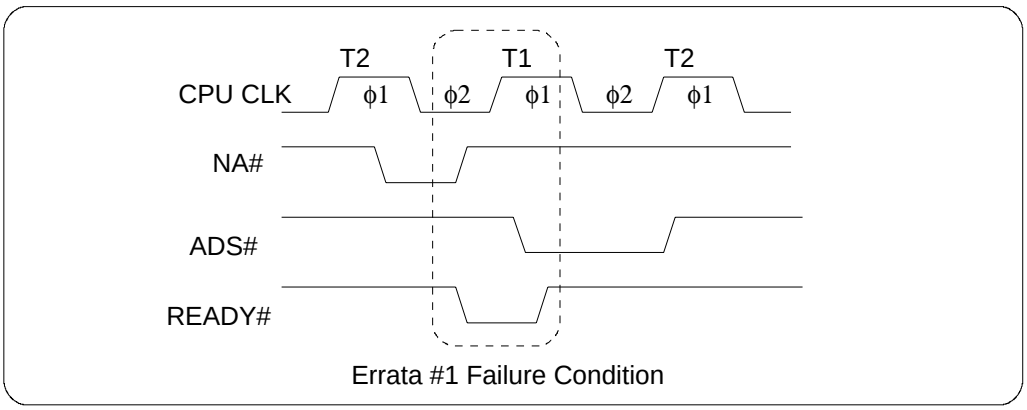
The BSDL file may be incorporated into software which uses the JTAG port for testing or programming functions.

Device and Stepping	Identifier in IDCODE
80386EX A0	00270013H
80386EX B1	00270013H
80386EX C1	3V: 20270013H
	5V: 28270013H

Errata

1. Watchdog Timer (WDT) Bus Monitor Failure During Bus Pipeline Operation

Problem: The errata applies to the WDT unit when using the Bus Monitor Mode and Bus Pipeline operation. If a pipelined bus cycle is requested at the same time the current bus cycle is completed, the WDT down counter is not reloaded or stopped. This condition means that the WDT could time-out and erroneously report a system failure. The waveform below shows the errant condition.

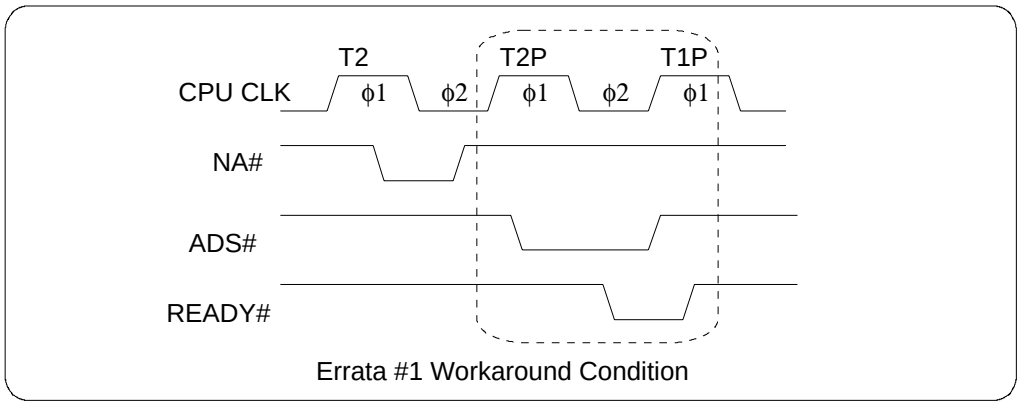


Errata #1 Failure Condition

The errata condition is caused by the system generating READY# immediately after NA# has been requested, and a T1 cycle is generated (rather than a T2P cycle). Essentially, the system is requesting a pipelined access, but does not need it because READY# has been returned. Such a condition is not typically expected when using the pipeline feature.

Implication: Applications that use the bus monitor mode of the WDT need to be aware of this condition. If bus pipelining is not used, the errata can be ignored. If bus pipelining is used, look to the workaround description below to ensure the errata condition is avoided.

Workaround: Ensure that READY# is not generated after a pipelined bus operation is requested. See the example waveform below.



Errata #1 Workaround Condition

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

2. Spurious Slave IR7 Interrupts When Using Internal WDT Interrupt

Problem: The WDT interrupt signal (WDTOUT) to the ICU does not remain active until the interrupt is acknowledged by the CPU (valid INTA cycle), resulting in the generation of a spurious IR7 request on the slave interrupt controller.

An interrupt request must remain active until the CPU has acknowledged the interrupt, at which time the request is latched, the appropriate in-service bit is set, and the vector type is read. If an interrupt is generated, but removed before the CPU has acknowledged it, the interrupt controller returns a default IR7 vector type and does not set any in-service bits. It is then the responsibility of the IR7 interrupt service routine to check the IR7 in-service bit and determine if a valid (in-service bit set) or spurious (in-service bit cleared) interrupt has occurred. However, since the WDTOUT is connected to the IR7 input of the slave interrupt controller, it is not possible to determine if the WDT or another interrupt generated the spurious interrupt.

Implication: This errata applies to those applications that typically use the WDTOUT as a general purpose counter and wish to generate interrupts when the counter times out. The application software needs to handle the case that both WDT interrupts and spurious interrupts will be reported the same way.

Workaround: The WDT interrupt to the slave interrupt controller is an internal signal and cannot be modified by the user. Therefore it is not possible to remove the error condition from a hardware perspective. The slave IR7 interrupt service routine must be written to handle both WDT interrupts and spurious interrupts without the assistance of the in-service bit.

One possible way to determine if the WDT generated the interrupt would be to test the WDT counter value. Since the WDT counter reloads when the WDT count expires (i.e., reaches zero), then comparing the current WDT counter value to the reload value may indicate that the WDT expired and generated the interrupt. Of course, since the WDT counter decrements every two CLK2 periods, the WDT counter and the reload counter will never be the same. Thus, such a test as mentioned here would have to account for a possible range of values (i.e., the WDT counter is less than some percentage value of the reload counter).

Additionally, if an external interrupt pin is available the WDTOUT pin can always be connected to it through an external inverter to provide the WDT interrupt.

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

3. WDT Lockout Sequence May Not Properly Reload Down Counter

Problem: If the WDT is in general timer mode or bus monitor mode, and the CLKDIS bit is set (i.e., counter is stopped), enabling the system watchdog mode by executing the lockout sequence does not reload the down counter. This means the counter will start counting down from whatever the previous value was in the counter (rather than starting from the value in the reload counter).

Implication: Since the WDT is reset to general timer mode with the CLKDIS bit cleared, this errata applies to an application that disables the WDT before it enables the system watchdog mode.

Workaround: There are two possible workarounds. Executing two lockout sequences will reload the counter correctly. Essentially the first lockout sequence enables the WDT, while the second sequence reloads the counter. Another workaround that requires less code is to enable the counter first by clearing the CLKDIS bit, and then execute the lockout sequence (see code example below).

```
MOV   DX, WDTSTATUS
IN    AL, DX          ; Read WDT status register
AND   AX, 0FCH        ; Clear clock dis/bus mon bits
OUT   DX, AL          ; Update register
MOV   DX, WDTCLR
MOV   AX, 0F01EH
OUT   DX, AX
MOV   AX, 0FE1H
OUT   DX, AX          ; Lockout Sequence
```

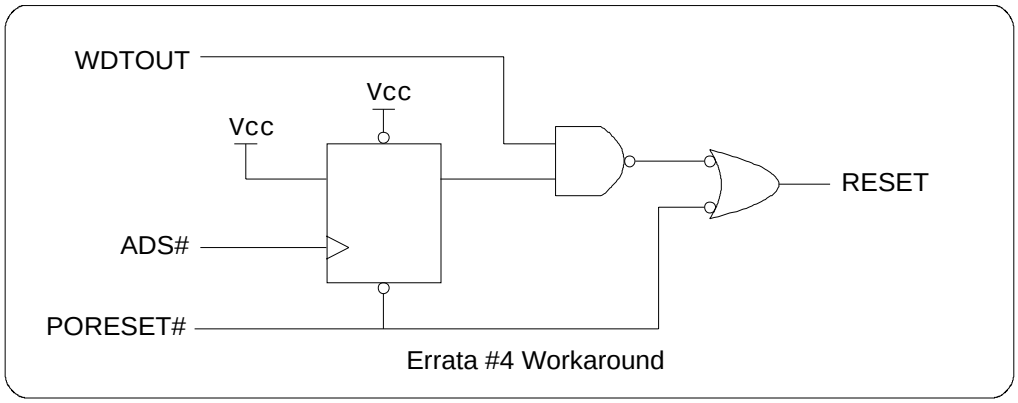
Status: Corrected on C-stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

4. WDT Will Time Out During CPU Self-Test

Problem: When entering CPU self-test mode (BUSY# low when RESET goes low), the WDT counter is loaded with the value 0000FFFFH (65,535). However, it takes over 1,048,575 (000FFFFFH) clocks to complete an internal self-test, meaning the WDT will have timed out prior to completing a CPU self-test.

Implication: This errata only applies to a system design that uses the WDTOUT signal to generate a device reset. The WDT will time out and reset the device before the CPU self-test can complete.

Workaround: The WDTOUT signal cannot be directly tied to the RESET input during self-test since there is no mechanism to increase the default value loaded into the WDT counter. The circuit shown below can be used to block WDTOUT until the first ADS# is generated. The signal PORESET# is a power-on reset signal generated by the system. WDTOUT is an output of the Intel386 EX processor, and RESET is an input to the processor. Please note that the circuit below has not been tested in a real design and is provided as an example only.



Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

5. Timer Generated DMA Requests May Not Initiate Properly After RESET

Problem: The internal signal paths between the Timer Control Unit (TCU) outputs and the DMA Control Unit request inputs contain a flip-flop that may not be properly initialized after a device reset (i.e., the output of the flip-flop may be high or low). This means that a TCU DMA request may be present prior to the timer actually generating the request. The errata applies to both DMA timer requests.

Implication: This errata only affects DMA requests generated by the TCU. If the TCU is configured to generated DMA requests, the initialization software must ensure the TCU DMA request bit is cleared.

Workaround: Before programming the TCU, and after programming the DMA configuration register, the DMA status register can be read to determine if a timer request is already pending (errata condition). If the request is pending, a dummy timer DMA operation must be initiated prior to using the TCU for DMA requests (after one DMA cycle, the flip-flop will be properly initialized). If the request is not pending, the errata condition does not apply and the dummy DMA cycle does not need to be run.

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

6. DMA Temporary Register Not Flushed When Software Aborts Transfer

Problem: During two-cycle DMA operation, should the channel's transfer be aborted before terminal count is reached or EOP# is generated, the data in the temporary register is not flushed. When the channel is reprogrammed, the old data in the temporary register will be transferred to the new destination address.

Implication: This errata applies to two-cycle transfers only, and not fly-by transfers (since the temporary buffer is not used in fly-by transfers). If the application terminates a DMA transfer by disabling the DMA channel, it is possible that not all of the data has been transferred from the temporary buffer and will be unexpectedly inserted into the next DMA transfer.

Workaround: DMA transfers must terminate using TC or EOP# only. It is not possible to flush the temporary registers in software.

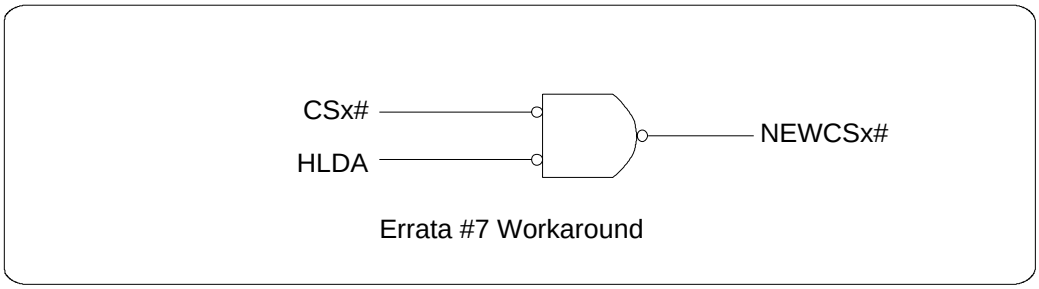
Status: Corrected on the C-stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

7. Chip Selects Remain Active During Bus HOLD And DMA Cascade Cycles

Problem: When the device enters a bus HOLD state because of a HOLD or DMA Cascade operation, the Chip Select that was valid in the previous cycle can remain active during the bus hold. This can cause bus contention when another bus master gains control of the bus.

Implication: This errata only applies to those applications which cause the bus to enter a bus hold state and the assertion of a chip-select can cause bus contention.

Workaround: Qualify a chip-select with HLDA such as shown in the circuit below. The chip-select will go inactive if HLDA goes active. If using a PLD device, then HLDA should be used to qualify the chip-select as shown in the following diagram.



Errata #7 Workaround

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

8. Write Sequence To Enable Expanded I/O Space Is Not Sequential

Problem: The I/O write sequence to enable expanded I/O space does not have to be executed back-to-back. Any other I/O address or I/O address sequence can occur during the sequence and will not affect the enabling of expanded I/O space. This departs from the original design intent that if the sequence is not followed exactly, it must be repeated from the beginning.

Consider the sequence to enable expanded I/O space as a sequence of states shown below:

```
; Start at State A (expanded I/O space Disabled)

    MOV   AX, 8000H   ;Writing 00 to byte I/O
    OUT   23, AL      ;address 23H gets to state B

    XCHG  AL, AH      ;Writing 80H to byte I/O
    OUT   22H, AL     ;address 22H gets to state C

    OUT   22H, AX     ;Writing 0080H to word I/O
                      ;address 22H gets to state D,
                      ;which enables Expanded I/O space.
```

Each write to I/O space with the appropriate data value in AL moves the internal state machine from one state to the next. Thus the sequence of going from State A to State B, State B to State C, and State C to State D enables expanded I/O space. Anything can happen between the states shown above, and the sequence is not interrupted (i.e., reset back to State A).

To make the device more secure and avoid inadvertently enabling the ESE bit, the C-step device state machine requires a sequential back-to-back write. State machine security will not allow software to enable I/O out of sequence. If the sequence is not followed, the internal state machine must be started in the Reset state A to assure proper enabling of the ESE bit. The code sequence below assures the proper advancement of states:

```
; Disable interrupts to avoid unexpected state machine
; reset due to I/O within an interrupt handler

    CLI              ;Disables interrupts

; Start at Reset state (expanded I/O space Disabled)
    IN    AL, 23H    ;Resets state machine (Reset=State A)


    MOV   AX, 8000H  ;Clears ESE bit
    OUT   23H, AL    ;First unlock sequence, moves
                     ;state machine to state B

    XCHG  AL, AH     ;Writing 80H to byte I/O
    OUT   22H, AL    ;Second unlock seequence, moves
                     ;state machine to state C

    OUT   22H, AX    ;Third unlock sequence, moves
                     ;state machine to state D, sets ESE
                     ;bit, which enables Expanded I/O space

    STI              ;Will re-enable interrupts, if desired
```

Implication: The software writer must assume that the writes have to be back-to-back and disable interrupts during the successive writes. This should not impact B step designs if the recommended programming techniques were used. It may be necessary to add instructions for C-step designs, if the back-to-back sequence was not followed in the software. Disabling interrupts and resetting the internal state machine will assure the write sequence enables the ESE bit. Software instructions used to clear the ESE bit can advance the state machine to State B if the instruction match is OUT 23H, AL. To properly enable expanded I/O space as desired, insert the line IN AL, 23H before beginning the unlock sequence.

Workaround: None. The application software must be aware that the I/O write sequence is not reset if any other I/O writes occur during the sequence. Note however, that code should be written assuming the sequence does need to be back-to-back. This will ensure proper operation if the errata is fixed in a future stepping.

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

9. 32-Bit WDT Register Is Oriented As High Word, Low Word In I/O Space

Problem: The 32-bit counter register and reload register are located with the high word located at a lower I/O address space than low word. As a result, 32-bit I/O accesses to these registers will have high and low words swapped, making it impossible to write a 32-bit value directly to the counter and reload registers.

Implication: This errata only applies to system software that maintains a 32-bit image of the WDTRLD and WDTCNT registers in memory and wishes to perform a 32-bit register move to I/O address space.

Workaround: The memory or register image must swap the low-word and high-word values to perform a 32-bit register move to the I/O registers.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

10. Remap Configuration Register At 22H And 23H Have 10-Bit Address Decode

Problem: When expanded I/O space is not enabled, only the lower 10-bits of I/O addresses are used to decode the internal peripherals and the configuration registers at 22H and 23H. This means that the configuration registers will appear in I/O space at every 1K address boundary (for example, 22H, 1022H, 2022H, etc.).

Implication: This errata affects those applications which may place system peripherals in I/O address space that didn't expect registers 22H and 23H to repeat every 1K block. The access to I/O locations such as 1022H will be decoded internally and generate a 1 wait internal access. Depending on the value written, it could possibly enable expanded I/O address space (see errata #8).

Workaround: Enable expanded I/O space to turn on 16-bit decode when communicating with I/O devices that overlap 22H and 23H at I/O addresses above 1K. This means that the internal peripherals are decoded using all 16-bit of the I/O address.

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

11. CAS Lines Do Not Remain Active During Idle States Between Cascaded INTA Cycles

Problem: The interrupt controller CAS lines do not maintain a valid value during the four idle states between the first INTA cycle and the second INTA cycle (assumes an externally cascaded device is present). The three CAS lines (CAS0-CAS2) are OR'd with address lines A16-A18. During the first INTA bus cycle, these line drive out the cascade address which is decoded by the slave interrupt controller to determine if it must place the vector type on the bus during the second INTA. There are four idle states automatically inserted between the end of the first INTA bus cycle and the start of the second INTA bus cycle. During these idle states, the CAS lines revert back to address mode and are driven high (logic 1), which will cause problems for the external interrupt controller.

Implication: This errata only applies to those applications that expand interrupts using an external 8259A Programmable Interrupt Controller device as slave interrupt controllers.

Workaround: The CAS lines must be latched and preserved throughout both of the INTA bus cycles.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

12. Cannot Pipeline DMA Bus Cycles

Problem: During a DMA bus cycle, NA# does not function properly with the DMA unit.

Implication: This errata affects systems that make use of pipeline bus operation and the DMA controller.

Workaround: None. Do not use address pipelining when using the integrated DMA controller.

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

13. DSR1#/STXCLK Pin Has A Permanent Weak Pull-Up

Problem: The weak pull-up device used to hold the state of the DSR1#/STXCLK pin at a high level until device configuration is complete does not turn off. This condition results in a higher than normal leakage current for the pin.

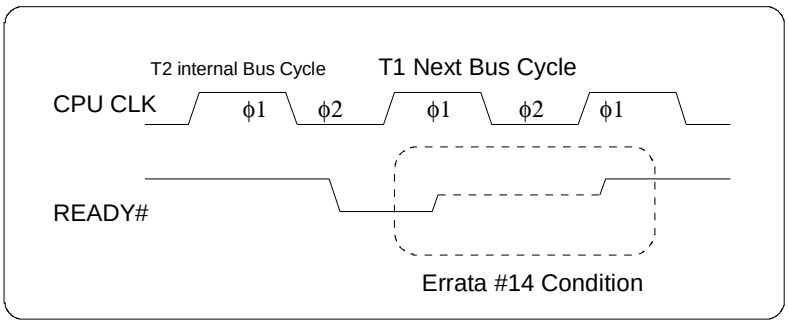
Implication: This errata applies to all applications.

Workaround: None. The external driver must be strong enough to pull down the pin. The weak pull-up is approximately 5 KOhms.

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

14. READY# Floats In T1 After Internal Bus Cycle Due to LBA#

Problem: During internal bus cycles, the LBA# signal does not remain active into the next T1 bus state, causing READY# to float during T1. This is unlike bus cycles controlled by the Chip Select Unit (CSU), where READY# is driven inactive during T1 of the next bus cycle.



Implication: This errata applies to all applications.

Workaround: Pull-up READY# if the float condition will cause system logic to operate improperly.

Status: Corrected on C-stepping. LBA# description has been documented in the User's Manual to reflect proper operation. Refer to Summary Table of Changes to determine the affected stepping(s).

15. Pipelined Bus Operation Does Not Work Reliably, Causing System Problems

Problem: Some of the internal peripherals (such as the Interrupt Control unit) are responsible for decoding the various bus states (e.g., T1, T2) and bus cycles (e.g., I/O read, Halt) to determine if some action on their part is necessary. For example, the Interrupt Control unit decodes INTA bus cycles and determines if it should drive a vector type on the bus during the T2 bus state. Bus states consist of T1, T2, Ti, Th, T1P, T2P, and T2i. Refer to any of the Intel386™ processor manuals for information on bus states. Bus cycles are decoded from the three control lines M/IO#, W/R#, and D/C#.

These same peripherals, however, are blocked from responding from bus cycles during special bus states, such as IDLE (Ti) and HOLD (Th), where bus cycle information is unknown or indeterminate. Unfortunately, some of the peripherals were not blocked from pipeline idle, or T2i, states. This means it is possible for a peripheral unit, the Interrupt Controller for example, to

incorrectly decode bus cycles during a T2i state and perform an operation that is unintentional. T2i bus state occurs during pipelined bus cycles when NA# has been asserted and there is no internal CPU request pending.

All of the peripherals correctly respond to Ti (idle) and Th (hold) bus states, meaning that normal (non-pipelined) operation works correctly. If pipeline bus operation is utilized, then it is possible for peripherals like the Interrupt control unit to begin to behave incorrectly should a T2i state occur. For example, if the three control lines (M/IO#, W/R#, D/C#) all go to a low state during a T2i state, the Interrupt Controller decodes this to mean that an INTA cycle is in progress and responds accordingly. This may cause the processor to be in a state where pending interrupts are seemingly ignored.

The Interrupt Control unit is just one example of where a system failure can result due to the execution of a T2i bus state. There is no way to guarantee, or even tell what state the three control lines will be driven to in a T2i bus state, thus it is not possible to define all conditions (sequences of bus states) that will cause one problem or another, or no problem at all.

Implication: All applications that make use of pipelined bus operation are affected.

Workaround: Tie NA# high and do not use pipelined bus operation.

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

16. DMA Registers At Addresses 08H, 09H, And 01AH Do Not Decode 16-Bit Addresses Correctly

Problem: The I/O address decoders used for the DMA registers at I/O addresses 08H, 09H, and 01AH do not decode 16-bit addresses correctly. Accesses to registers or I/O peripherals mapped into 1 kilobyte multiples (0408H, 0808H, etc.) of the DMA register addresses will cause unexpected accesses to the DMA registers. Note that in the chip select unit CS1# and CS3# have registers which are mapped into addresses F408H, F409H, and F41AH. These chip selects will be impacted by this errata.

Implication: This errata applies to all applications that make use of the DMA unit.

Workaround: Do not map external I/O peripherals into addresses that are at 1K multiples of the DMA register addresses. For the two chip selects that are affected by this errata it is important to initialize the chip selects and then immediately set up the DMA channels, in that sequence. Very few applications will need to modify the contents of the chip select registers after initialization, but for those that do it will be necessary to re-initialize the affected DMA registers.

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

17. TDO Pin May Float in User Mode Causing Excessive Current Consumption

Problem: The TDO pin usually floats in user mode. This could turn on the unused input buffer at the pin resulting in high leakage current.

Implication: This errata applies to all applications that utilize the idle or power down modes, or that are sensitive to small increases in Icc due to leakage current.

Workaround: Put a weak pull-up resistor on the TDO pin.

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

18. Overflow Enable Bits (TOV0, TOV1, ROV0, ROV1) In DMAOVFE Register Do Not Function as Specified

Problem: The overflow enable bits in DMA register DMAOVFE do not match the definition. TOV1 and TOV0 control Requester Address and Byte Count registers. They should control Target Address and Byte Counter registers. ROV1 and ROV0 control Target Address registers. They should control Requester Address registers.

Implication: This errata applies to all applications that utilize the DMA controller. Note that it will only affect those applications that require the TOV bit to be set to a different value than the ROV bit.

Workaround: The application must be aware that the register bit functions are reversed from the definition given and account for that in software.

Status: Corrected on C-stepping. With this workaround, the application must be changed when a conversion to C-step is initiated. However, this will only impact applications requiring the TOV bit to be set to a different value than the ROV bit. Refer to Summary Table of Changes to determine the affected stepping(s).

19. Upper DMA Byte Count Registers (DMA0BYC2, DMA1BYC2) Change to 0F0H When the Lower Byte Count Registers Underflow

Problem: In the DMA when the overflow enable bit is not set for the Byte counter (only 16-bits of byte counter are being used), the upper byte count register, DMA0BYC2 or DMA1BYC2, is changed to 0F0H whenever the lower register underflows.

Implication: This errata applies to all applications that utilize the DMA controller.

Workaround: This errata will not impact the operation of the DMA controller. The user should be aware, however that if the application requires the DMA to switch from 8237A mode to its enhanced mode (using all 24 bits of byte count) then the high byte count registers may not contain the reset value of 00H and should be re-initialized.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

20. At High Baud Rates, There Is Not Enough Time to Turn the Transmitter Off By Resetting the TEN Bit

Problem: The SSIO transmitter is turned off by clearing the TEN bit in the SSIOCON1 control register. In the case where the SSIO is transmitting at high baud rates (i.e., the DMA is being used to service the transmitter) it is possible that there is not enough time for the processor to clear the TEN bit between when a THBE (transmit holding buffer empty) interrupt occurs and when the previous word is completely shifted out of the shift register. This is due to interrupt latency time. The result is that it is possible for the last word of a transmission to be shifted out of the shift register more than once. The baud rate at which this starts to become a problem depends on the frequency the processor is running at, the mode (real or protected) it is running in, and any other factors which may affect interrupt latency.

Implication: This errata applies to all applications that require non-continuous high speed synchronous serial transfers.

Workaround: This errata will limit how fast the SSIO transmitter can operate. Since the baud rate will be limited by interrupt latency, this can be calculated using the instruction execution/interrupt latency information given in the *Intel386™ SX Processor Programmer's Reference Manual* or *Intel386™ SX Processor Hardware Reference Manual.*

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

21. Write Sequence for WDTCLR Lockout is Not Sequential

Problem: The I/O write sequence to WDTCLR to do the lockout does not have to be executed back-to-back. An access to any I/O address or I/O address sequence except for another WDT register can occur during the sequence and will not affect the lockout of the watchdog timer. This departs from the original design intent that if the sequence is not followed, it must be repeated from the beginning. The sequence will abort only if a write to another WDT register is made before the second write to WDTCLR.

Implication: The software writer must assume that the I/O writes have to be back-to-back. This should not impact B-step designs if the recommended programming techniques were used.

Workaround: None, as the application software must be aware that the I/O write sequence is not reset if any other I/O writes occur during the sequence. Note however, that code should be written assuming the sequence does need to be back-to-back. This will assure proper operation if the errata is fixed in a future stepping.

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

22. Setting the Channel Enable Bit (CE) in DMACMD1 Registers also Disables HOLD Requests

Problem: The Channel Enable (CE) bit in the DMACMD1 control register is used to globally disable requests from both channels of the DMA controller to the bus arbiter. However, it not only disables requests from the DMA channels, but also from the HOLD pin.

Implication: This errata applies to all applications that require disabling of the DMA unit.

Workaround: When both channels of the DMA controller are not needed they should be disabled using either the Individual Channel Mask Register - DMAMSK, or the Group Channel Mask Register - DMAGRPMSK.

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

23. DMA Synchronization Problem During Handoffs of Low Priority Channel to High Priority Channel, Resulting in Loss of Data

Problem: Between the DMA logic and the DMA state machine is a problem that manifests itself if the bus arbiter changes which channel with an active request has the highest priority, and at the same instant asserts XFRREQ (transfer request). If the bus arbiter receives a DMA request on a channel that has a lower priority, it will activate an enable channel signal to the DMA, and request the bus from the core. If a request comes in on a higher channel, the bus arbiter will then deselect the channel with a lower priority, and select the channel with the higher priority. If conditions are right, the bus arbiter can deselect the lower priority channel, select the higher priority channel, and give the DMA XFRREQ (transfer request) simultaneously.

Implication: This errata applies to all applications that use both DMA channels. This problem appeared specifically when the DMA priority was set to its default value (i.e., DMA0 has higher priority over DMA1).

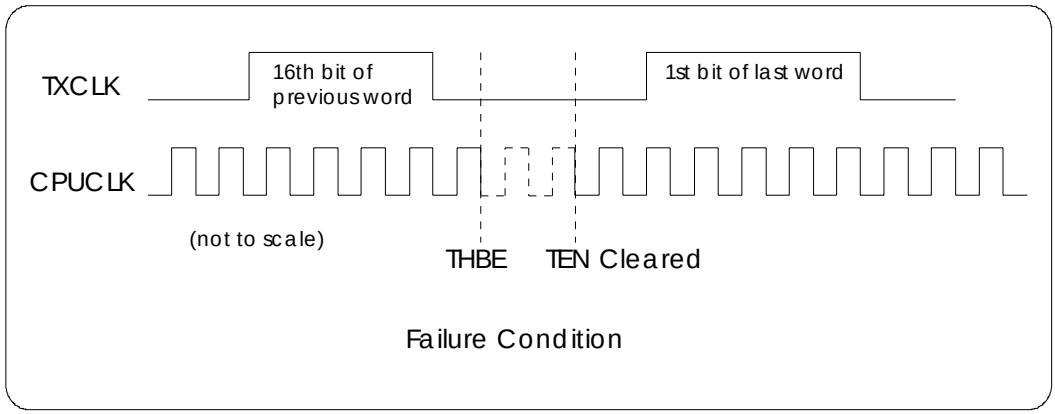
Workaround: Change the default DMA priority by writing 00h to DMACMD2 register, and always use DMA1 as the higher priority channel.

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

24. TEN Bit In SSIOCON1 Register Turns Off Transmit Channel Too Soon

Problem: When the SSIO is transmitting data (master or slave mode) the transmitter is turned off by clearing the TEN bit of the SSIOCON1 register. One way of indicating to the software to clear this bit is by waiting for the Transmit Holding Buffer Empty (THBE) signal to go active, indicating that the word written to the holding register is ready for transmission from the shift register. Even if the software waits for this signal and then writes to SSIOCON1 to disable TEN, depending on when TEN is cleared, the last word transmit may or may not take place.

If the TXCLK is significantly slower than the CPUCLK, it is possible for the THBE bit to be set and the TEN bit to be cleared all during the low time of the TXCLK (see diagram). In this case, even though the last word to be transmitted has been loaded into the shift register, it will not be shifted out.



Implication: This errata applies to all applications that require non-continuous low speed synchronous serial transfers.

Workaround: In order to assure that the last word is transmitted, the TEN bit must not be cleared until after the TXCLK transitions from low to high when the 1st bit of the last word is being transmitted. Consequently, the way to assure that this happens is to either: 1) utilize a timer to assure that enough time elapses from when the THBE bit goes high until the TXCLK goes high, or 2) execute a fixed number of NOPs after THBE goes high before clearing TEN. In either case, the length of the delay is dependent upon the frequency of TXCLK and the CPUCLK.

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

25. Back-To-Back Writes To Interrupt Controller May Cause Failure In Second Write

Problem: When back-to-back write I/O operations are performed to the OCW2 register in either the master or the slave in the Interrupt Control Unit an error can occur in the second write operation. One flip-flop in the asynchronous logic of the ICU does not get set correctly. Back to back writes to the ICU are defined as two writes to the ICU without a read of the ICU in between them. It doesn't matter if there are other operations that occur between the writes.

When a data byte is written, bits 3 and 4 of the byte determine which internal register (OCW2, OCW3, or ICW1) is selected. The data bus retains the last value that was put on it. Because the flip-flop is in the wrong state for the second write, a race condition can occur if the data bus

changes too slowly with respect to the write signal inside the interrupt controller. It is possible for the old values of bits 3 and 4 to cause a short clock pulse to the wrong register before the new values of bits 3 and 4 are set.

The problem shows up most noticeably when the internal peripheral data bus has a value of XXX011XX on it from the last I/O operation and you write an end of interrupt command to OCW2. The data byte would be XXX00XXX and should select OCW2. Instead, the 01 that was on the bus temporarily selects OCW3 and latches a 1 into bit 2, setting the POLL bit. When the POLL bit is set, no further interrupt requests are passed from the interrupt controller to the CPU.

Implication: This errata applies to all applications that use the interrupt control unit.

Workaround: Doing a read operation at either the master or slave ICU after a write to that same address will always put the flip-flop to the correct value. The next write to the same address will then work correctly and not show the race condition. Note that register ICW1 is not affected by the flip-flop value, so the dummy read only needs to be done when writing to OCW2. See the sample end-of-interrupt code below:

```
MOV   AL, eiodata
MOV   DX, MPICP0    ;Master ICU I/O address
OUT   DX, AL
IN    AL, DX        ;Dummy read to set flip flop
```

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

26. DMA And Bus Arbiter Reset May Cause Bus Contention

Problem: The DMA and bus arbiter can be reset by writing to address 000DH (or F00DH) which is the DMACLR register. DMACLR is not actually a physical register, but rather the reset is caused by an address decode of an I/O write to this location. Even though this is the case, there can be local bus DMA data bus contention if the “data” written to DMACLR is not 00H. The result of this contention is that some of the DMA registers may not properly reset when DMACLR is written.

Implication: This errata applies to all applications that use the DMA or bus arbiter.

Workaround: Always write 00H to DMACLR when the DMA or bus arbiter needs to be reset. See sample code below.

```
MOV   DX, 000DH     ;DMACLR Address
MOV   AL, 00H       ;Must be 00H to avoid contention
OUT   DX, AL        ;Resets the DMA and Bus Arbiter
```

Status: Corrected on B stepping. Refer to Summary Table of Changes to determine the affected stepping(s).

27. Internal HOLD Request Synchronization May Cause Data/Code Fetches To Be Corrupted

Problem: Any operation that asserts the internal hold request signal (iHOLD) may cause bus contention if that request is removed before the iHLDA is returned valid. The requests that may cause this problem are DREQ0, DREQ1, HOLD, and the internal refresh request. The issue is that when the request is asserted the bus arbitration sequence starts in order to transfer control of the bus. If the request is pulled away at the wrong time, then the bus arbiter will continue to give the bus to the requester, but the core will recognize that the request has been removed and attempt to take control of the bus also. This will result in contention on the bus and corrupted data/code.

Implication: This errata applies to all applications that use multiple bus masters.

Workaround: Make certain in the application that the DREQs or HOLD requests are not removed until either DACK or HLDA respectively are returned by the processor. Note that for a two cycle DMA transfer only the requester cycle generates DACK, the target cycle doesn't. Consequently, the logic must be able to ensure that DREQ is left active until the channel is serviced.

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

28. Writing To One DMA Channel's Register When A Request From The Other Channel Occurs May Cause An Improper Transfer.

Problem: If a DMA request occurs on one DMA channel while the processor is writing to the other channel's registers (requester address, target address, or byte count) then it is possible for the transfer to be corrupted on the channel that made the request. The symptoms from this errata have occurred in two different ways under specific conditions. The first condition is if the transfer is a write (data transferred from requester to target), the requester is an 8-bit device and the target is a 16 bit device and the channel is programmed for two cycle, single transfer, single buffer mode. Under these conditions the last byte of the buffer is read from the requester but never written to the target. The second condition is the same as the first, except that the transfer is a read (data transferred from target to requester) instead of a write. In this condition there is one (or possibly more) too many bytes written to the requester.

Implication: This errata applies to all applications that use both DMA channels and allow one to be enabled while the other is being programmed.

Workaround: Do not attempt to reprogram one channel of the DMA while the other is enabled.

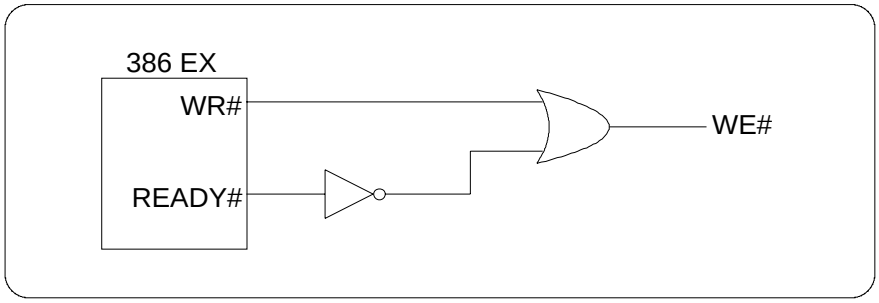
Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

29. Insufficient Address Hold Time After WR# Goes High

Problem: The address hold time after WR# goes high does not meet the data sheet spec of 5 ns.

Implication: This errata applies to all applications that use the WR# signal as the write strobe to memory or I/O.

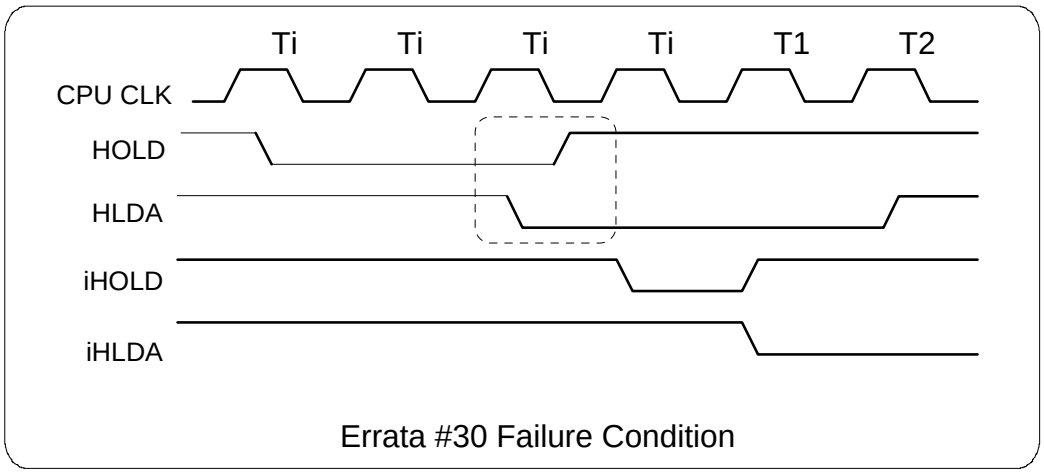
Workaround: The circuit shown below may be used to generate a write strobe if the READY# signal is not generated too early in the T2 cycle. Please note this circuit has not been tested in a real design and is provided as an example only. For applications which cannot use the circuit below it is necessary to implement an external WE# signal using a simple state machine design. An example of what that state machine could look like is available in the EV386EX evaluation board design, or in the Point of Sales Terminal reference design available from the WEB, BBS or through literature.



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

30. HLDA Inactive To HOLD Active Arbitration Could Improperly Float The Bus

Problem: In a HOLD/HLDA cycle the HOLD signal must not be re-asserted until after HLDA is returned inactive. If, however, HOLD is asserted in the same T-state that HLDA goes inactive a problem may occur where the bus arbiter relinquishes control to the core and at the same time floats the bus by asserting the HLDA pin. The following waveform shows the conditions that cause this error. The signals iHOLD and iHLDA are internal signals and represent the HOLD input to the core and the HLDA output from the core (equivalent to the HOLD and HLDA signals on a Intel386™ SX processor). In this diagram a T1 occurs because a request was pending by the core during the previous HOLD cycle. One T-state after the arbiter releases the bus (HLDA = 0) it removes its iHOLD, and then the core removes iHLDA. As soon as the iHLDA is released, the T1 state starts. However, because of the timing of the next HOLD request, the bus arbiter re-asserts iHOLD and HLDA even though the core has not asserted iHLDA. The result is that the core continues to run its bus cycle and, at the same time, the bus arbiter has floated the external bus. It is also possible for the following to occur when HOLD/HLDA arbitration is at issue: HOLD is driven active during the T1 or Tw of a valid bus cycle. The processor begins the next bus cycle and generates HLDA. HLDA is active for only one T-state. ADS# is extended through the bus cycle. The alternate bus master sees the HLDA and performs its bus activity. HOLD is then driven inactive and the bus is returned to the Intel386™ EX Processor. At this point there is bus contention.

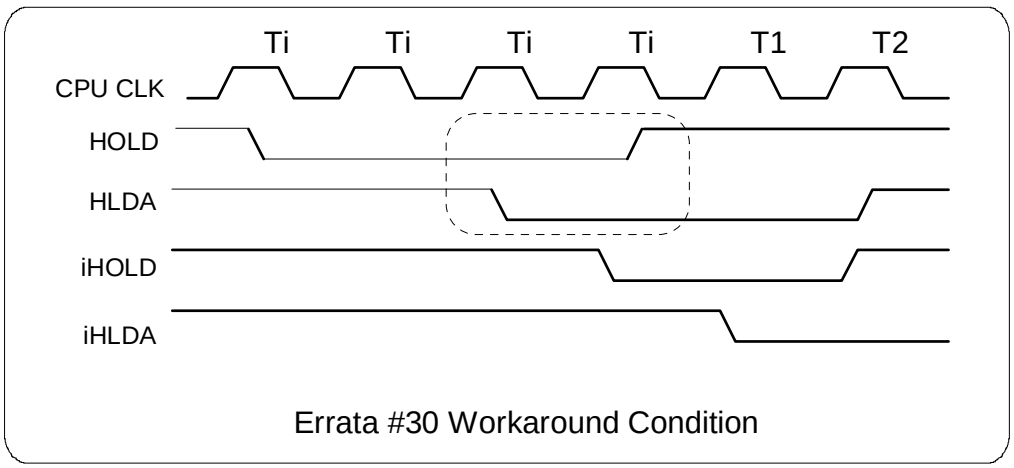


Errata #30 Failure Condition

It is also possible for the following to occur when HOLD/HLDA arbitration is an issue: HOLD is driven active during the T1 or Tw of a valid bus cycle. The processor begins the next bus cycle and generates HLDA. HLDA is active for only one T-state. ADS# is extended through the bus cycle. The alternate bus master sees the HLDA and performs its bus activity. HOLD is then driven inactive and the bus is returned to the Intel386™ EX Processor. At this point there is bus contention.

Implication: This errata applies to all applications that use external bus masters with HOLD/HLDA arbitration.

Workaround: Synchronize the external HOLD request such that it is does not go active until at least one T-state after HLDA is inactive. See the following example waveform.



Errata #30 Workaround Condition

Another workaround is to delay the generation of HOLD until the last T2-state of a valid bus cycle (excluding refresh cycles or INTA cycles).

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

31. FLT# Does Not Float All Outputs

Problem: Asserting the FLT# pin does not float any of the I/O pins that have permanent weak pull-ups/pull-downs or I/O pins that have temporary weak pull-ups/pulldowns that are turned on. In these cases, the actual pin drivers will be three-stated, but the pull-ups/pulldowns will force them to the given state.

Implication: This errata is really only a problem when an In-Circuit-Emulator is being used. Many ICE systems utilize a clamp on header that uses the FLT# pin to float the actual processor in order to "take over" the system. In this case, if a temporary pull-up/down has been turned off then asserting FLT# will float that pin. If, however, the target board is reset, the reset action takes priority over the FLT# and turns the temporary pull-up/down on again. If the application uses an external pull-up/down that pulls the pin to the opposite value as the internal pull-up/pull-down, the pin voltage will be somewhere between 0V and Vcc.

Workaround: Make certain that external pull-ups/downs pull each pin to the same level as any internal pull-ups/downs. See Appendix A table A-4 in the *Intel386™ EX Embedded Microprocessor User's Manual* for a description of which pins have weak pull-ups/downs.

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

32. SIO Break Could Be Missed

Problem: The SIO Break Interrupt in the LSR register indicates when a break condition has occurred. This bit is cleared on reading the LSR. If the application is polling the line status register to detect a break interrupt it is possible for the read action to occur at the same time the SIO is setting the BI bit. If this occurs, the read action will prevent the BI bit from being set causing the program to miss the break. This occurs because the SIO uses different clock signals to set and clear the BI bit. Since these clock signals are not synchronous they could happen at the same time, and the clearing action will take precedence.

Implication: This errata applies to any application that uses a polling routine to detect a break interrupt.

Workaround: A possible workaround is to send multiple breaks from the transmitting end of the SIO. In this case, the SIO may miss the first, but detect the second.

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

33. Arbitration Between Powerdown And Refresh Cycles May Leave Control Signals In Wrong State

Problem: Entering powerdown mode requires a HALT cycle to be executed after the powerdown bit in the PWRCON register has been set. If a refresh request occurs at about the same time a condition may occur where the bus arbiter grants the bus to the RCU, and at the same time puts the device into powerdown mode. The symptom of this problem is that the ADS# and REFRESH# signals will latch in their active states, and power consumption is higher than it should be.

Implication: This errata applies to any application that uses powerdown mode and the refresh control unit.

Workaround: Prior to executing the HALT instruction that puts the device into powerdown the RCU should be disabled by clearing bit 15 of the RFSCON register. This refresh is not supported in powerdown mode this should not create a problem.

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

34. Refresh Control With More Than 2 Wait States Could Cause An Erroneous Bus Cycle

Problem: An arbitration fault occurs between the refresh control unit (RCU) and the bus interface unit (BIU) under certain refresh conditions. If a refresh of DRAM occurs immediately following an I/O access of any type (this includes interrupts), an erroneous bus cycle may occur if the DRAM is configured for more than two wait states. This will cause the refresh cycle to be terminated after two wait states and may cause corruption of data if the DRAM does require more than two wait states.

Implication: This errata impacts those applications using DRAM with more than 2 wait states and also utilizes the internal Refresh Control Unit of the Intel386™ EX Processor.

Workaround: None. The only alternative at this point is to use DRAM that does not require more than two wait states or to use an external DRAM controller.

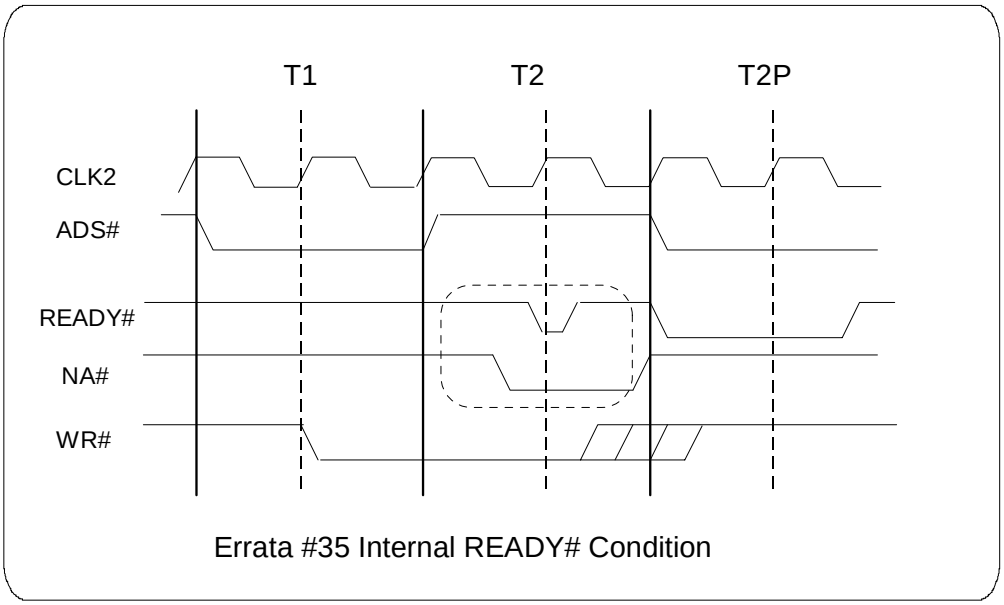
Status: This errata applies to B1 stepping only. Refer to Summary Table of Changes to determine the affected stepping(s).

35. Internal READY# Generation Switching from Non-pipeline to Pipeline Can Be Temporarily Asserted

Problem: If the processor chip select is configured with zero wait states and is switching from non-pipeline to pipeline mode, READY# can be temporarily asserted. This occurs in T2 only if READY# is being generated internally to the processor. Normally, READY# would be asserted in T2. However, if NA# is also asserted in T2, READY# can temporarily be asserted and then return high due to the capture of NA#. READY# will then return active during the next state, T2P. This will not impact processor operation.

Implication: This errata applies to all applications programming zero wait states on chip select and generating an internal READY#. The temporary transition of READY# will only occur when switching from non-pipeline to pipeline mode.

Workaround: None. Do not trigger any external state machine on asserted READY# signal in T2 unless it occurs at the end of the T2 state where expected.



Errata #35 Internal READY# Condition

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

36. NA# Must Not Be Asserted In A T2P State for DMA Pipelining

Problem: If NA# is asserted (Low) in a T2P state, data transfer can be corrupted. This will only affect Pipelined DMA transfers and could result in the loss of data on the last transfer.

Implication: This errata affects those applications which desire to pipeline DMA cycles. Due to this limitation, NA# must not be asserted during T2P for DMA pipelining. The ability to pipeline DMA cycles was not an option on A or B step devices.

Workaround: None. DMA pipelining was not an option on B step. For C-step applications, the user must assure that NA# is not asserted in T2P DMA cycles.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

37. Arbitration for HOLD During Refresh Cycle May Not Return an HLDA Signal

Problem: Any operation that causes the internal hold signal iHOLD, to be asserted, deasserted, and then reasserted within one bus cycle can cause internal conflicts of the bus arbiter. The first iHOLD assertion can be caused by HOLD, DMA or a REFRESH request. The external hold signal, HOLD, must be responsible for the reassertion of the second iHOLD. If DMA or REFRESH are responsible for the second iHOLD assertion, the problem will not occur since DMA or REFRESH do not toggle the HLDA signal.

This errata will not occur if external HOLD or the internal Refresh Control Unit are not used. However, if a combination of HOLD asserted asynchronously in any cycle and internal refresh are used, this problem will only manifest itself as shown in conditions shown below:

1. For system designs that use both HOLD and the internal Refresh Control Unit, the conditions are as listed in a, b, and c.
   a. A small RFSCIR register value can cause the current refresh cycle and the next refresh cycle to almost overlap. Refer to Figure 3. When such a condition exists, there is a possibility that the refresh latency will cause a gap between the current refresh terminating (t3) and the next refresh (t6). This will create a problematic window if HOLD is asserted during that time. The minimum RFSCIR count that can be used without causing this problem depends on HOLD latency which is related to bus activity. HOLD latency is similar to refresh latency since they both generate an iHOLD request to the core. HOLD latency is basically a factor of wait states, locked cycles and special cycles such as interrupts. Worst case conditions should be used and can be calculated based on the information on page 6-37 (Hold Signal Latency) of the Intel386™ EX Embedded Microprocessor User's Manual (order number 272485-002). If the RFSCIR is a few cycles longer than the maximum calculated HOLD latency plus refresh duration, the problematic window will not exist. This condition should not happen in a typical customer design, since the RFSCIR value is usually larger than the worst case minimum value. To avoid this error, no workaround is needed as long as the RFSCIR value is greater than the recommended minimum value. The minimum RFSCIR can be roughly calculated as follows:

      Hold Latency => 9 * (wait_state + 2) + 17

      Minimum RFSCIR => (Hold Latency) + (Refresh wait state +2) + 5

      Example: For systems with 2 wait states for all cycles.

         Hold Latency = 9 * 4 + 17 = 53

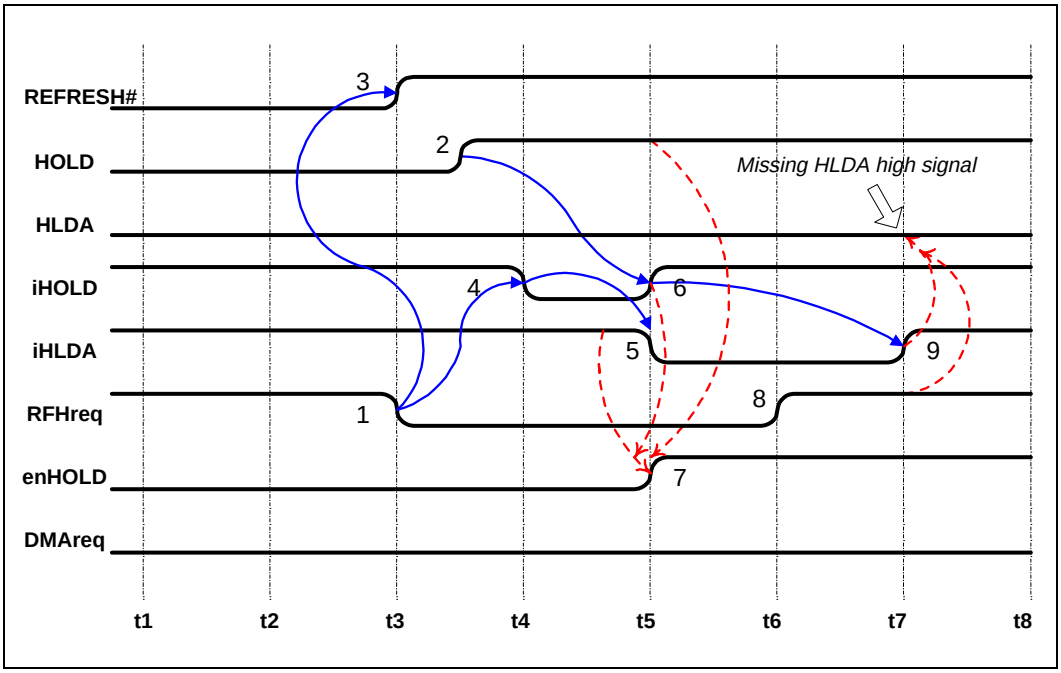
         Minimum RFSCIR = 53 + 4 + 5 = 62 (3E hex)

      If 8 bit bus is used (BS8), hold latency will be twice as long.

Figure 3 shows Condition #1a where an external HOLD and refresh request conflict can occur. The table below describes each signal name used:

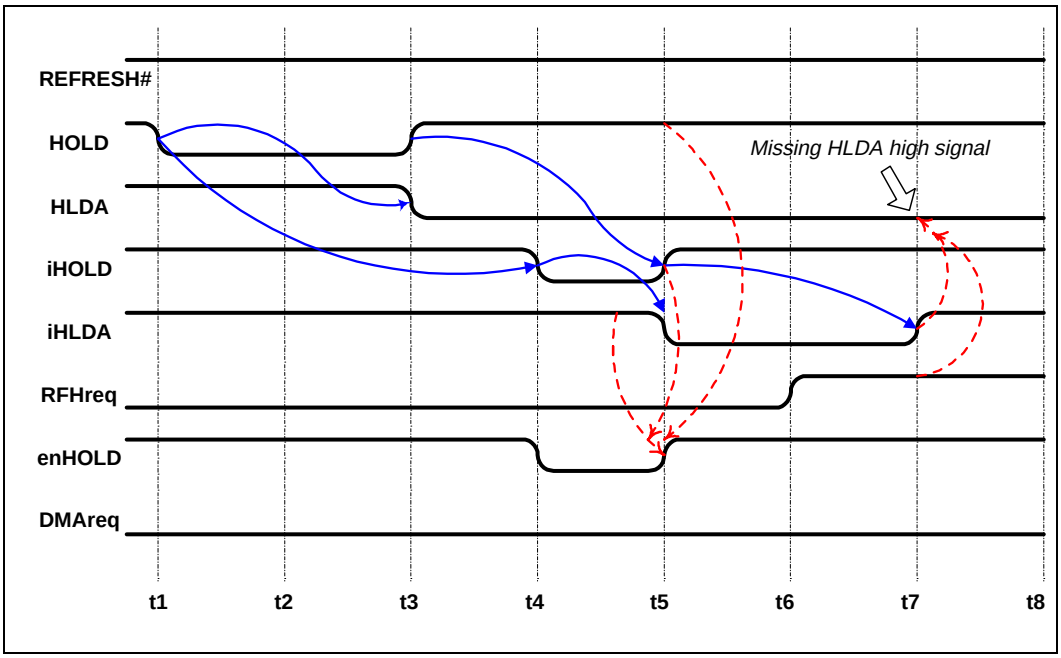
Signal	Description
REFRESH#	External refresh output pin. Refresh is in process when this signal is low.
HOLD	External hold input pin. Active high.
HLDA	External hold acknowledge output pin. CPU is in hold state when active.
iHOLD	Internal hold request to the i386 CPU core. Active high.
iHLDA	Internal hold acknowledge output from i386 CPU core. Active high.
RFHreq	Internal refresh bus request from refresh unit to bus arbiter. Active high.
enHOLD	Internal bus arbiter state signal indicating HOLD is granted. Active high.
DMAreq	Internal DMA bus request signal to bus arbiter. Active high.

Figure 3. HOLD with Short REFRESH Interval



b. HOLD is removed and reasserted within one bus cycle. Refer to Figure 4. This condition will cause the problematic window with fast iHOLD deassertion (t4) and assertion (t5). Note that if HOLD is deasserted because of pending REFRESH and reasserted again within one bus cycle, the problem will not occur since the iHOLD signal is never deasserted due to refresh. The reassertion of the HOLD would then be serviced upon completion of the refresh.

Figure 4. Quick HOLD Assert and Deassert with Slow REFRESH Interval



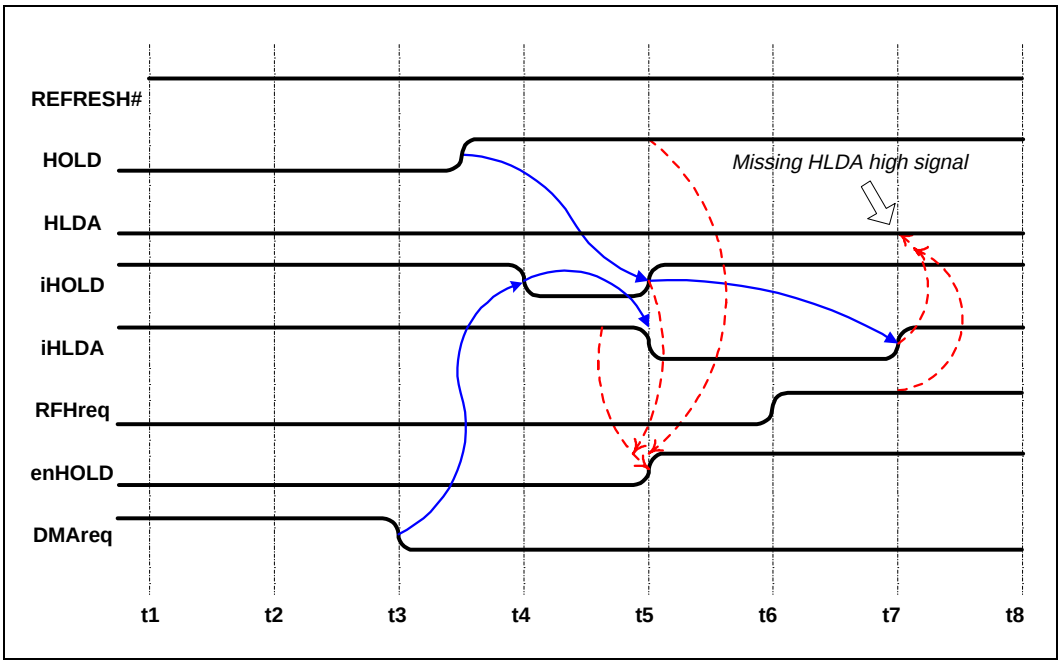
c. Slow deassertion of HOLD when internal refresh is pending can cause the current refresh to almost overlap with the next refresh cycle. In this case, the problem is similar to what is described in (a). Ensure that extra Refresh latency is not intentionally induced because of late HOLD deassertion when refresh is pending.

2. For system designs that use DMA, HOLD and REFRESH.

For the errata to occur, HOLD must be asserted asynchronously at any cycle such as in an idle cycle. Refer to Figure 5. Termination of DMAREQ at time t3 with assertion of HOLD at time t4 creates the problematic window. If there is a refresh request at time t6, the HLDA is blocked and will not go active at time t7. If HOLD is synchronized at the beginning of a *new* valid bus cycle (except for REFRESH), HLDA will always be asserted since the problematic window will not occur.

Note: DMAREQ is a signal internal to the processor. This signal is not always within the user's control and could be internally deasserted, such as in autoinitialize mode. To avoid issues with this type of situation, it is recommended that the user deassert the external request DREQ after the last transfer has completed. It is further recommended that DREQ only be deasserted after a valid DMA acknowledge (DACK# goes low) has been returned. If a DREQ is held active and then deasserted during or shortly after the autoinitialize sequence, the problematic window can be induced and cause the errata to occur. See errata #28 for additional details on DMA programming restrictions.

Figure 5. DMA, HOLD and REFRESH with Normal Interval



Implication: This errata applies only to those C-step applications that use the internal Refresh Control Unit of the Intel386™ EX Microprocessor along with the external bus master HOLD/HLDA arbitration.

Workaround: If the situations listed in conditions #1 and #2 cannot be avoided, the user needs to synchronize the external HOLD request such that it is does not go active in an idle bus cycle or a refresh cycle. Alternately, HOLD can be deasserted if a HLDA is not received within a time-out period, as described in workaround 2. A third listing describes another recommended workaround when using HOLD, DMA, and REFRESH.

Workaround 1: This involves asserting HOLD to the processor only during a new valid bus cycle, other than a refresh cycle. This workaround applies to those applications which use HOLD and the Refresh Control Unit but not the DMA. The logic required to implement this workaround is straightforward:

Assert HOLD only after a valid ADS# has been asserted signifying a new, valid bus cycle (use ADS# to qualify HOLD). See the Intel386 EX Embedded Microprocessor User's Manual for explanation of valid bus cycles.

This will work because the bus arbiter can then properly determine bus ownership. This avoids potential for conflict between the Refresh Control Unit and an external bus master. This type of workaround may need additional logic to be implemented externally and will prevent the problem of missing HLDA from occurring.

Workaround 2: Deassert HOLD if HLDA is not asserted within the worst case HOLD latency. This can be implemented with external logic that times-out if a HLDA signal is not asserted after the hold latency interval. Note: Slow deassertion of HOLD can increase refresh latency. The user should make sure that the errata is not induced because of late HOLD deassertion when refresh is pending. Added refresh latency can increase the potential for overlapping refresh requests (causing the error). In this workaround, HOLD should be qualified with the ADS# signal. This entails deasserting HOLD if a HLDA has not been returned and no valid ADS# signals have occurred within the time-out period for worst case HOLD latency.

Workaround 3: In certain instances Workarounds 1 and 2 will not suffice. In those cases, the following workaround is suggested. Institute a 10 “T” state time period (two CLK2s per “T” state) starting with:

a.) Deassertion of a valid DACK# signal;

b.) Removal of a HOLD request;

c.) A REFRESH# pulse is detected.

During this 10 “T” state time period, new DMA or HOLD requests should be blocked. This allows enough time for the bus arbiter to avoid the problematic window described in the problem section above. If items a, b, or c occur within the 10 “T” state time period, the cycle counter must be restarted and new requests blocked.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

38. READY# Source Conflicts May Cause Bus Cycle Failure

Problem: When the external HOLD signal is used, the C-step processor may experience bus cycle failures when any chip select, including UCS#, is set for the upper region of memory and programmed to generate READY# internally. The problem originates internal to the processor. The actual issue occurs just prior to the core asserting hold acknowledge (iHLDA). The HOLD or a cascaded DMA request can cause the processor to function incorrectly, if the READY# source changes from external to internal. This happens in the idle cycle just before iHLDA is asserted. A variety of bus cycle types can insert an idle cycle prior to an iHLDA. Once this type of cycle completes, the iHLDA signal will be asserted after one idle cycle. During this idle cycle, the address lines all go to a logic “1” state and will activate any chip select signal programmed for the upper region of memory. If this chip select is programmed for an internal READY#, the problem can occur if the previous cycle used an external READY# signal.

On the next fetch of data, the bus cycle will appear to be correct. Following the second access after iHLDA has gone inactive, bus cycles do not work correctly. In particular, the RD# or WR# signals do not go active. When the above condition occurs, internal circuitry does not respond to further bus cycles and the processor will remain in the T2 state. A condition found to cause this is a chip select match during the idle cycle just before an iHLDA, but only if a chip select is programmed for internal READY# generation.

Implication: This errata affects all applications which use a mix of internal and external READY# generation and HOLD or cascaded DMA channel requests.

Workaround: The suggested workaround is to avoid a direct Chip Select Unit match of address lines during an idle cycle. During the idle cycle just prior to an iHLDA, all address lines will become a logic 1. To avoid this problem, follow one of the recommendations given below:

Workaround 1: The chip select programming must ensure that no region exactly matches address bits A25:A11 as all 1’s. One method is to ensure that at least one address bit in both the CS*n*MSK and CS*n*AD registers is a zero. This will ensure that the active chip select region contains at least one zero in the A25:A11 region. This will prevent a 3FFFFFFH match during an idle cycle and minimize the impact to available memory usage. The limitations with this workaround is that Port92 reset cannot be easily used. To use the Port92 reset, block all HOLD and cascaded DMA requests and enable a chip select region that contains the upper part of memory. This will ensure that there is memory at the reset entry point.

Workaround 2: (NOTE: This workaround has not been fully validated). If an unused chip select channel is available, enable it for the upper 2 Kbytes of memory such as:

CS*n*#, memory area 3FF:F800H to 3FF:FFFFH (Uppermost 2K of memory)
1 WS + EXTERNAL ready
16 bit
No SMM
Output of channel not connected to external pin
Leave UCS# programmed as usual

The impact is that there are a few more programming steps and you lose 2 Kbytes of memory addressing. Considerations for this workaround: 1.) To utilize Port 92 RESET: Deconfigure CS*n*#, mask HOLD; do a Port 92 reset; then reprogram CS*n*# and unmask HOLD. NOTE: Instead of a Port 92 reset for changing from Protected to Real mode, it is possible to clear the PE bit in CR0. This avoids having to do a Port 92 reset; 2.) Make sure the program does not access the upper 2 Kbytes of memory after a software RESET sequence

Workaround 3: Generate external READY# for the all cycles if you are utilizing HOLD or cascaded DMA. This will keep the internal logic signals stable before the transition of the iHLDA signal. The user logic must still block any HOLD and cascaded DMA requests from being asserted after reset, until the chip selects are initialized. Another alternative is to program all chip selects to generate internal READY#. These two suggestions of all internal or all external are mutually exclusive. A combination of internal and external READY# signals cannot be used in workaround 3.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

39. Not All Baud Rates from Baud-rate Generator Operate Correctly

Problem: When Autotransmit mode is enabled, not all baud rates set for the baud-rate generator work properly under all conditions.

Implication: In Autotransmit mode, using arbitrary values in the SSIOBAUD register and CLKPRS register may result in baud rates that do not operate correctly. Selection and ensuring correct operation of values other than those tested is the responsibility of the customer.

Workaround: Intel recommends using one of the baud rates shown in the table below.

Baud Rate† (bps)	SSIOBAUD Register	CLKPRS Register
2400	CFH	29H
9600	A0H	18H
57600	11H	6H

† The first bit is transmitted at the maximum baud generation rate. All subsequent bits are transmitted at the correct clock rate for the selected baud rate.

The discussion in section 13.2.1 in the *Intel386™ EX Microprocessor User's Manual* is for users who want to select their own baud rate. Intel does not guarantee that all values and baud rates will operate as discussed in this section. Baud rates other than those shown in the table above are not tested.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

40. With Autotransmit Enabled, the First Bit Is Transmitted at the Wrong Baud Rate

Problem: With autotransmit enabled, the first bit in the output is transmitted at the maximum clock baud rate. All subsequent bits then follow at the selected clock baud rate. When the maximum clock baud rate is chosen, all bits transmit correctly.

Implication: The first bit transmitted can be much narrower than subsequent bits, causing the receiving device to miss the first bit, or miscount bits in a serial stream.

Workaround: There is no known workaround for this problem.

Status: No fix is planned. Refer to Summary Table of Changes to determine the affected stepping(s).

Specification Changes

1. AC Timings Modified for 5V +/-0.5V

Issue: The timings in datasheet 272420-004 should be changed as shown in the table below. The following AC timings have been modified for 5V +/-0.5V:

Parameter	Current Data Sheet Spec.		Revised Spec.	
	min. (ns)	max. (ns)	min. (ns)	max. (ns)
t12	4	31	4	28
t21	7		5	
t24	3		5	
t30	5		6	
t31	4	36	4	32
t34	4	39	7	32

Figure 6. AC-001

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-004.

2. AC Timings for modified for 3.3V +/-0.3V

Issue: The timings in datasheet 272420-004 should be changed as shown in the table below. The following AC timings have been modified for 3.3V +/-0.3V:

Parameter	Current Data Sheet Spec.		Revised Spec.	
	min. (ns)	max. (ns)	min. (ns)	max. (ns)
t20	4		5	
t24	5		6	
t30	5		7	

Figure 7. AC-002

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-004.

3. AC Timings for 3.0V +/-0.3V

Issue: The timings in datasheet 272420-004 should be changed as shown in the table below. The following AC timings have been modified for 3.0V +/-0.3V:

Parameter	Current Data Sheet Spec.		Revised Spec.	
	min. (ns)	max. (ns)	min. (ns)	max. (ns)
t20	4		5	
t24	5		6	
t30	5		7	

Figure 8. AC-003

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-004.

4. AC Timings for T34 on EXTB and EXTC devices is different when in SMM.

Issue: The t34 timing in datasheet 272420-006 has an errata associated with SMM on the EXTB and EXTC devices. If SMM is being used, the t34 timing for 5V at 33 MHz is actually 25 ns, not 24 ns as published. For 3V at 25 Mhz, the t34 timing is actually 34 ns, not 33 ns as published. If SMM is not used the published timings apply.

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

5. AC Timings Modified for EXTB Device

Issue: The timings in datasheet 272420-006 should be changed as shown in the table below. The following AC timings have been modified for 3 V, 20 MHz:

Parameter	Current Data Sheet Spec.		Revised Spec.	
	min. (ns)	max. (ns)	min. (ns)	max. (ns)
t6	4	34	4	36
t36	4	26	4	29
t113		48		52
t114		48		52
t115	9		11	
t117		42		46

Figure 9. AC-004

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

6. AC Timings Modified for EXTC Device

Issue: The timings in datasheet 272420-006 should be changed as shown in the table below. The following AC timings have been modified for 5 V, 25 MHz and 33 MHz:

Parameter	Current Data Sheet Spec.		Revised Spec.	
	min. (ns)	max. (ns)	min. (ns)	max. (ns)
t115	6		7	

Figure 10. AC-005

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

7. IDIV Instruction Interrupt0 Due To Overflow

Issue: The IDIV instruction operands need to be sign-extended to avoid incurring an overflow condition (interrupt0 exception).

An extremely limited subset of IDIV calculations may not properly generate an overflow interrupt0 exception subsequent to the actual occurrence of an overflow calculation. This inhibited interrupt0 exception condition results in an incorrect IDIV quotient calculation. The IDIV instruction does not incur the overflow condition when sign-extension is conducted before the operation. Direct assembly language programming should also perform sign-extension prior to executing IDIV to preclude the occurrence of the overflow condition (sign-extension of the divisor is used to prevent the overflow condition, sign-extension of the dividend is used to satisfy the IDIV instruction format). The Intel386 processor instruction set incorporates several commands to facilitate sign-extension. The CBW, CWD(CWDE), and CDQ instructions are often utilized to automatically sign-extend byte, word, and double-word data.

Fortran and C require that arithmetic operations be performed on operands of equal length; some compilers may already either extend the shorter operand or extend both operands to a common larger size. To accommodate both positive and negative numbers, these compilers typically establish this equal length by performing sign-extension.

Affected Docs: *Intel386™ EX Embedded Microprocessor* User's Manual, order number 272485-002.

8. FLT# Pin Should be Pulled Up to VCC

Issue: For the Intel386 EXTB, and for the EXTC running below 3.6 V, Intel recommends that the customer add a 3 to 7 KOhm pull-up resistor to the FLT# pin. Not all systems running below 3.6 V require this additional resistor; the customer must determine when this pull-up is required.

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

Specification Clarifications

None for this revision of this specification update.

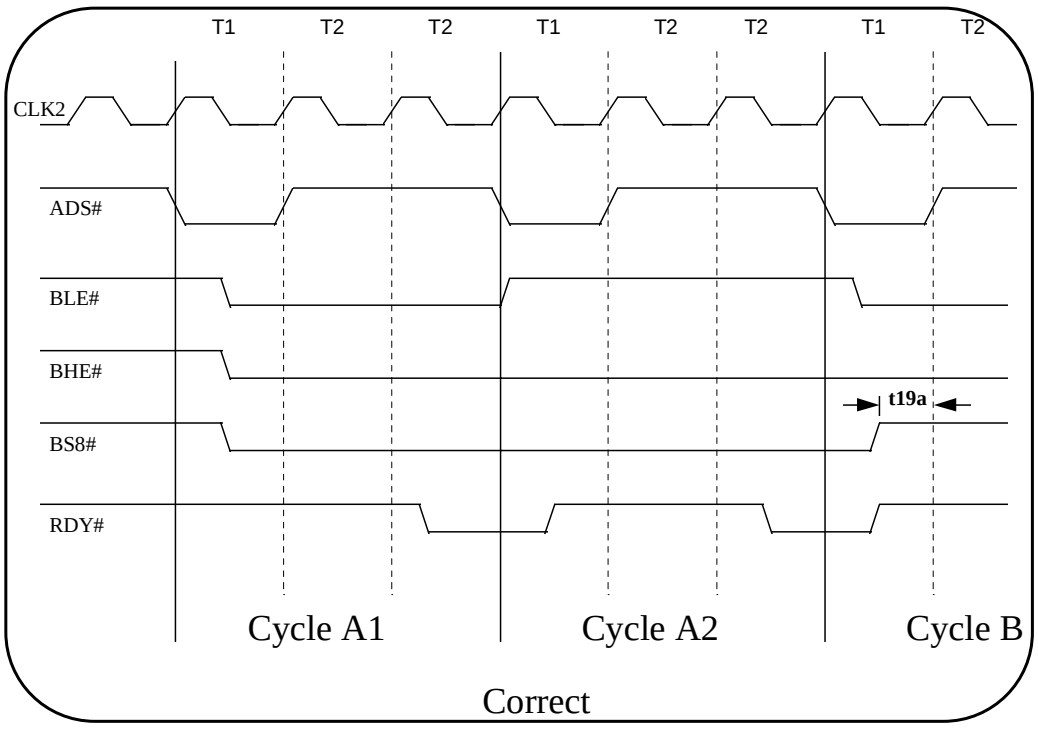
Documentation Changes

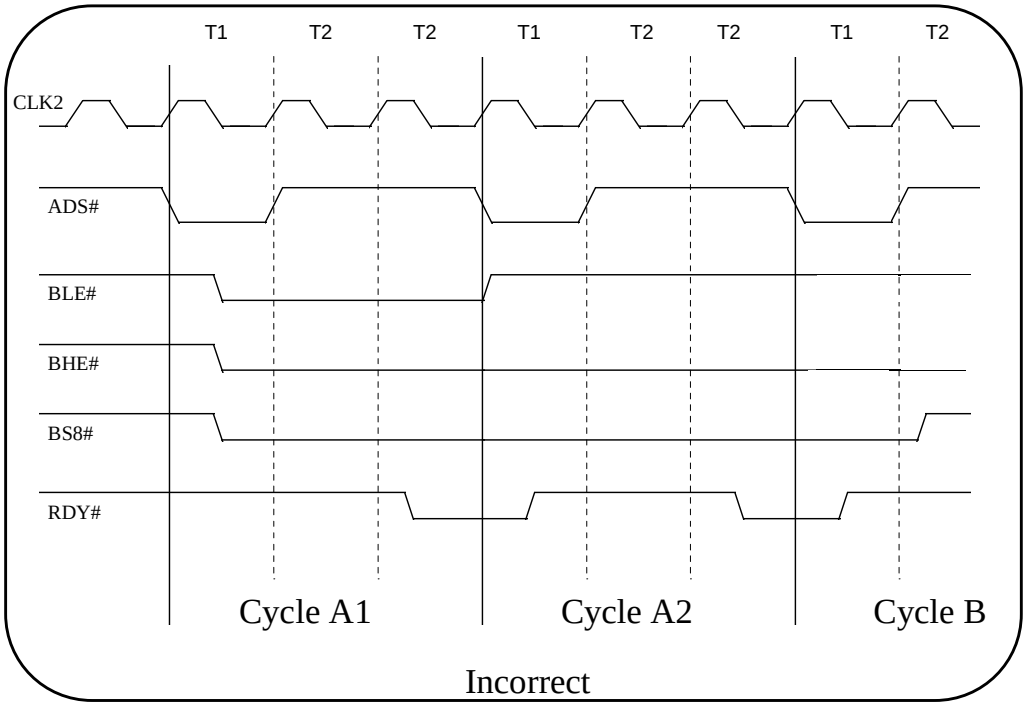
1. 272485-001, Correct Latching Of BS8# In Reference To 8 Bit Vs. 16 Bit Device Access

Issue: All related A and B step documentation indicates that the state of the BS8# pin is ignored until the end of the T2 state when READY# is active. This is incorrect. Assume the processor does a cycle to an 8-bit device (CYCLE A) followed by a cycle to a 16-bit device (CYCLE B). If CYCLE A is a 16-bit read or write cycle, and external logic asserts BS8#, the processor breaks the single cycle into two 8-bit cycles (CYCLE A1 and CYCLE A2). If BS8# is active in CYCLE A2, it does not matter. However, if BS8# is not driven inactive before the end of T1 (end of T1 - setup time) of CYCLE B, then the errata will occur. Note that the problem is only present if CYCLE B is to the high byte of the 16-bit memory (BHE# low and BLE# high).

The reason this occurs is as follows: the processor latches the state of BS8# at the beginning of the first T2 of CYCLE B (to perform byte-steering within the processor) and resets this latch only in the next Ti or T1 state. Therefore, if BS8# is kept active into the first T2 of CYCLE B, then the byte-steering logic of the processor incorrectly assumes that CYCLE B is also a cycle to an 8-bit device. This causes the bytes to end up on the wrong data path.

Here are example waveforms demonstrating incorrect vs. proper latching of BS8#:





Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-001.

2. 272485-001, Programming One Or Both Channels of the DMA Requires Both Channels To Be Masked. DMA Channels Must Also Be Masked While Reading/Writing To DMA Related Registers

Issue: Improper DMA function or corruption of data can occur during DMA channel programming unless both DMA channels are masked. If only one channel of the DMA is being used, you need only mask that channel while reprogramming. In addition, DMA channels should be masked while reading or writing to any of the DMA registers, with the exception of DMAIS.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-001.

3. 272420-006, Values for I_{CC} Represent the Maximum Supply Currents with the Device Held In Reset and Inputs Pulled to Their Inactive Levels

Issue: In Table 6 (DC Characteristics) of the datasheet, the values for I_{CC} represent the maximum supply currents with the device held in reset and inputs pulled to their inactive levels.

Affected Docs: *Intel386™ EX Embedded Microprocessor* Datasheet, order number 272420-006.

4. 272485-002, Autotransmit Mode for SSIO Requires Baud Rates Above 3.2 MHz

Issue: In the B step device where the SSIO is transmitting at high baud rates (i.e., the DMA is being used to service the transmitter) it is possible that there is not enough time for the processor to clear the TEN bit between when a THBE (transmit holding buffer empty) interrupt occurs and when the previous word is completely shifted out of the shift register. This is due to interrupt latency time. The result is that it is possible for the last word of a transmission to be shifted out of the shift register more than once. The baud rate at which this starts to become a problem depends on the

frequency the processor is running at, the mode (real or protected) it is running in, and any other factors which may affect interrupt latency.

As a result, a feature called Autotransmit Mode was added to the C-step device. If this mode is enabled, the transmitter will stop once data is shifted out, if no new data has been written into the buffer. Since this mode was designed to be utilized for high speed transfers, it should only be used at baud rates about 3.2MHz. Otherwise, the first bit to be shifted out could have a pulse smaller than normal which could cause incorrect data transfers. This information supplements the explanation in the *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

5. 272485-002, Section 4.5.2.1, Page 4-8, Figure 4-4.

Issue: The code in the ESE Bit Code Example has changed:

```
;;Disable interrupts to avoid unexpected state machine
;;reset due to I/O within an interrupt handler
    CLI                         ;Disables interrupts

;;Start at Reset state (expanded I/O space Disabled)
    IN    AL, 23H               ;Resets state machine (Reset=State A)

    MOV   AX, 8000H             ;Clears ESE bit
    OUT   23H, AL               ;First unlock sequence, moves
                                ;state machine to state B

    XCHG  AL, AH                ;Writing 80H to byte I/O

    OUT   22H, AL               ;Second unlock sequence, moves
                                ;state machine to state C

    OUT   22H, AX               ;Third unlock sequence, moves
                                ;state machine to state D, sets ESE
                                ;bit, which enables Expanded I/O space
    STI                         ;Will re-enable interrupts, if desired
```

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

6. 272485-002, Section 4.6.4, Page 4-13, Figure 4-7.

Issue: Changed On-Chip UART-2 to On-Chip UART-0.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

7. 272485-002, Section 5.3, Page 5-27, Figure 5-18.

Issue: The bit 0 function description referred to Table 5-1; it now refers to Table 5-2.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

8. 272485-002, Section 6.2.4, Page 6-11.

Issue: Added "If READY# is generated externally, the input must remain stable during setup and hold times." to the end of the first paragraph.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

9. 272485-002, Section 6.3.1, Page 6-13.

Issue: In Step1, first paragraph, changed “PH2” to “PH2C clock”. In Steps 4 and 5, changed “PH2” to “PH2C clock”. Added note: “See Chapter 8 for definitions of PH1C, PH2C, PH1P and PH2P.”

Affected Docs: *Intel386™ EX Embedded Microprocessor User’s Manual*, order number 272485-002.

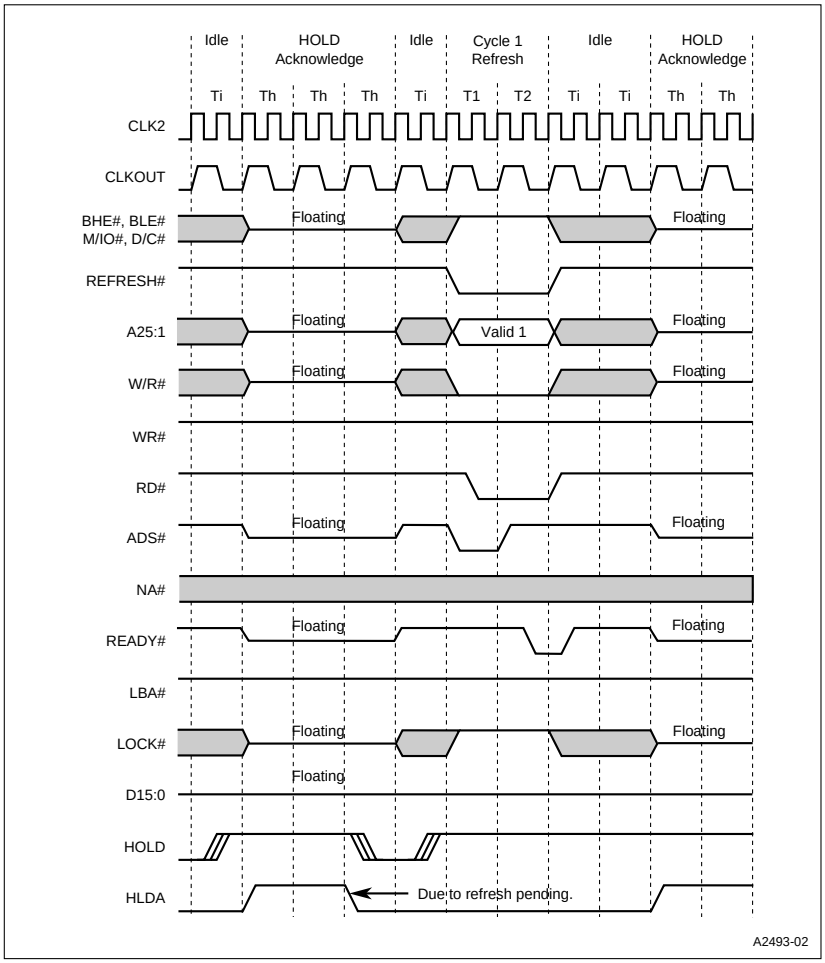
10. 272485-002, Section 6.3.6, Page 6-29, Figure 6-11.

Issue: Changed “BHE#, BLE#, M/IO#, D/C#” to read “UCS#, CS6:0, BHE#, BLE#, M/IO#, D/C#”.

Affected Docs: *Intel386™ EX Embedded Microprocessor User’s Manual*, order number 272485-002.

11. 272485-002, Section 6.3.6, Page 6-30, Figure 6-12.

Issue: Modified the ADS# and READY# waveforms to indicate floating. The corrected figure is shown below.



Affected Docs: *Intel386™ EX Embedded Microprocessor User’s Manual*, order number 272485-002.

12. 272485-002, Section 6.6.1.2, Page 6-40, code example.

Issue: In third line of code example, changed 0fffbh to 0fff3h.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

13. 272485-002, Section 6.6.4, Page 6-43.

Issue: Second paragraph, changed reference from "Chapter 6, Bus Interface Unit" to "Chapter 15, Refresh Control Unit".

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

14. 272485-002, Section 9.3.4, Page 9-21.

Issue: In the first paragraph, changed "32 + n x 8" to "32 + (n x 8)".

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

15. 272485-002, Section 12.3, Page 12-28, Table 12-3.

Issue: At the end of the description of Channel 0 and 1 Target Address, added "These registers cannot be read when operating in chaining mode."

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

16. 272485-002, Section 12.3.11, Page 12-44.

Issue: In the note, change "from initiating" to "to initiate".

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

17. 272485-002, Section 12.3.13, Page 12-47.

Issue: In step 7, change "setting bit 2" to "clearing bit 2".

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

18. 272485-002, Section 13.2.2.2, Page 13-12.

Issue: At the end of the first paragraph, add the line "When transmission has stopped, the SSIOTX retains the state of the last bit shifted out."

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

19. 272485-002, Section 15.5, Page 15-12.

Issue: In the bullet at the top of the page, change "<8" to "<50" and change "RCU always has bus control" to "RCU may always have bus control".

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

20. 272420-006, Section 7.0, Page 25.

Issue: In the first paragraph at the top of the second column, replace the last sentence with the following:

"RD# and WR# change to their active states at the beginning of phase two. RD# changes to its inactive state (end of cycle) at the beginning of phase one. See the *Intel386™ EX Embedded Microprocessor User's Manual* for a detailed explanation of early READY# vs. late READY#."

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

21. 272753-001, Figure 2, Page 7.

Issue: In Note 2 of Figure 2, the sentence beginning with “Pin 4 of U7...” should read “Pin 4 of JP7....”

Affected Docs: *AP-720 Programming Flash Memory through the Intel386™ EX Embedded Microprocessor JTAG Port*, order number 272753-001.

22. 271325-003, Section 1.0, Pages 2-3.

Issue: Change Captions to read:

Figure 2. 168-Lead Pin Grid Array Pinout (Bottom View — Pin Side)

Figure 3. 168-Lead Pin Grid Array Pinout (Top View — Component Side)

Affected Docs: *Special Environment Intel386™ EX Embedded Microprocessor Datasheet*, order number 271325-003.

23. 271325-003, Section 1.0, Pages 5-6.

Issue: In Figure 4 and Table 2, change the name of pin 151 from “SMIACT#/EXCSIG” to “SMIACT#”.

Affected Docs: *Special Environment Intel386™ EX Embedded Microprocessor Datasheet*, order number 271325-003.

24. 272485-002, Appendix A, Page A-10.

Issue: Added note: “Any unused input pins should be pulled to their inactive state.”

Affected Docs: *Intel386™ EX Embedded Microprocessor User’s Manual*, order number 272485-002.

25. 272485-002, Section 5.2.6, Page 5-20, Figure 5-12.

Issue: The diagram should indicate a value of 0 in PINCFG.6 selects CS6# and a 1 selects REFRESH# at the package pin.

Affected Docs: *Intel386™ EX Embedded Microprocessor User’s Manual*, order number 272485-002.

26. 272525-005, Appendix C, Page C-1

Issue: Line item #17 has a part number of “S19953, SMT”. It should read “Si9953, SMT”.

Affected Docs: *Intel386™ EX Embedded Microprocessor Evaluation Board User’s Manual*, order number 272485-002.

27. 272485-002, Section 13.2.2.1, Page 13-9

Issue: Replace figure 13-8 with Figures 13-8a and 13-8b as shown below.

Figure 11. Transmit Data by Polling

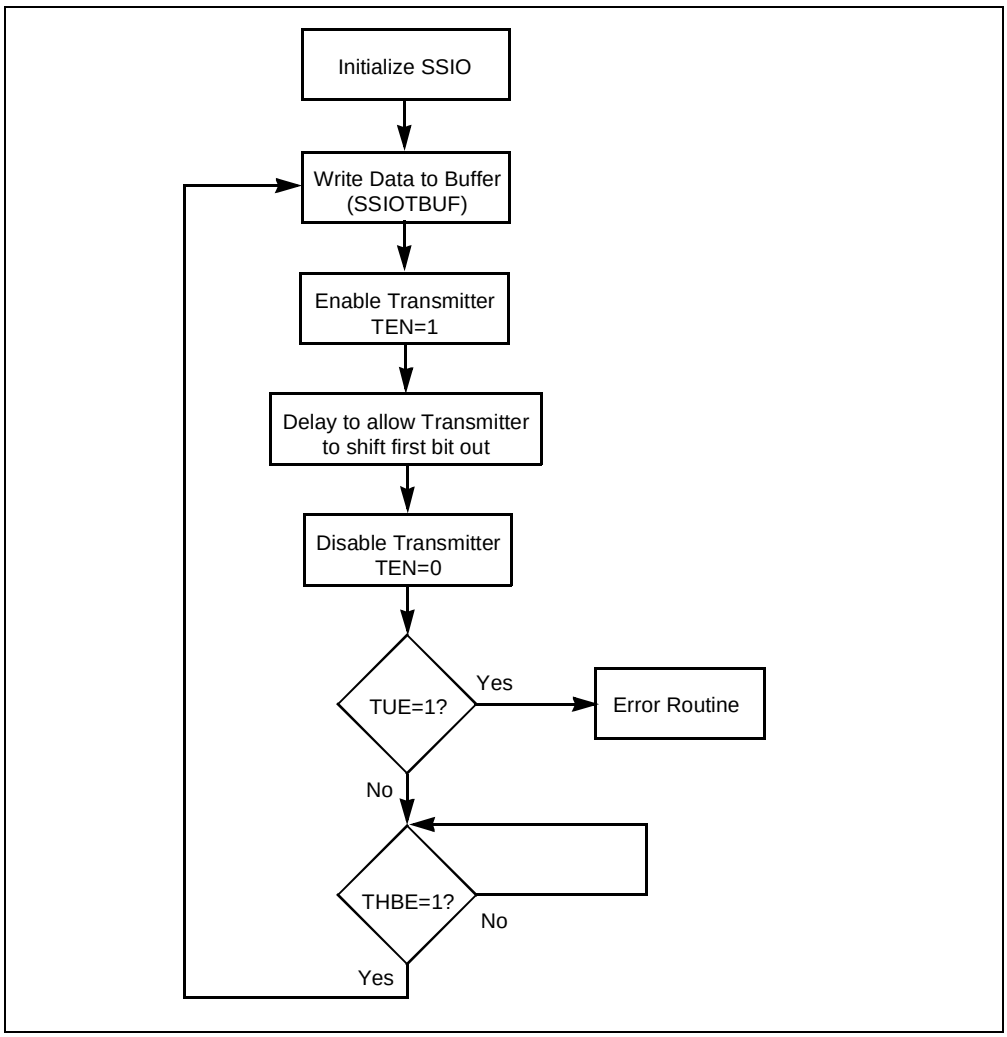
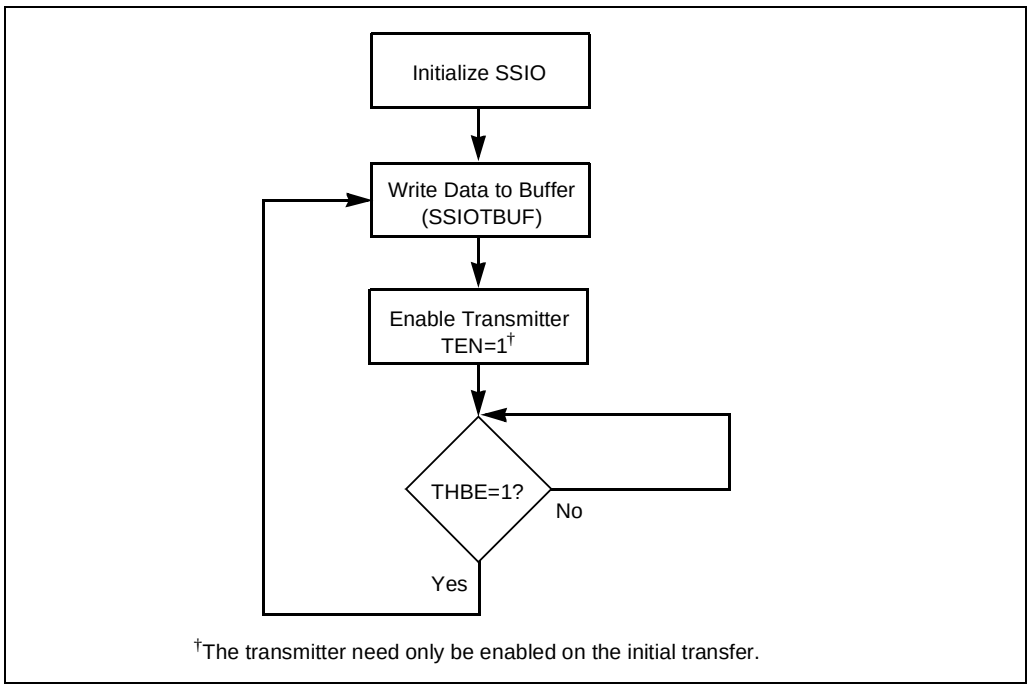


Figure 12. Transmit Data by Polling (Autotransmit Enabled)



28. 272485-002, Section 13.2.2.2, Page 13-12

Issue: In the first paragraph of section 13.2.2.2, delete the line "In this mode the TEN bit is ignored." At the end of the second paragraph, add this line: "The TEN bit must be set initially to enable the transmitter."

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

29. 272485-002, Section 14.6, Page 14-22

Issue: Change the first bullet in section 14.6 to read: "When programming a chip-select channel, it is recommended that you program the Low Mask Register last. This ensures that all other bits are properly programmed before the region is enabled. When reprogramming the channel, disable the channel before changing anything else. If the channel must remain active, ensure that the correct memory region is still enabled after each program instruction. Further care should be taken that no other memory transfers are enabled at this time (i.e., DMA)."

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

30. 272420-006, Page 9, Table 4

Issue: For the Intel386 EXTB, and for the EXTC running below 3.6 V, Intel recommends that the customer add a 3 to 7 KOhm pull-up resistor to the FLT# pin. Not all systems running below 3.6 V require this additional resistor; it is up to the customer to determine when this pull-up is required.

At the end of the description of the FLT# pin in the Name and Function column of Table 4: Delete "This pin should be a no-connect if not used." and add "This pin should be tied to V_{CC} through a 3 to 7 KOhm pull-up resistor."

Affected Docs: *Intel386™ EX Embedded Microprocessor Datasheet*, order number 272420-006.

31. 272420-006, Page 12, Table 4

Issue: Please add the following statement to the end of the description of the SSIOTX pin in the Name and Function column of Table 4: Intel does not specify a data hold time for SSIOTX. Slower external devices may require additional hardware to properly interface the SSIO unit.

Affected Docs: *Intel386™ EX Embedded Microprocessor* Datasheet, order number 272420-006.

32. 272485-002, Page 14-3, Note

Issue: The address examples shown in the note are incorrect. Change the last sentence of the note to read: "For example, a 256-Kbyte block can only start at an address that is a multiple of 256 Kbytes (0H, 80000H, 100000H, etc.).

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

33. 272485-002, Page 9-3, Figure 9-1

Issue: Figure 9-1 incorrectly shows MCR0.3 as the control for the mux into the Master IR4, and MCR1.3 as the control for the mux into the Master IR3. MCR1.3 controls the mux into IR4. MCR0.3 controls the mux into IR3. This is described correctly in Table 9-1 on page 9-5 of the manual.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

34. 272485-002, Section 13.2.1, Page 13-5

Issue: Add the following to the end of section 13.2.1:

Do not assume that all baud rates work under all conditions. Intel recommends using one of the baud rates shown in the table below.

Figure 13. Tested Baud Rates and Setup Values

Baud Rate† (bps)	SSIOBAUD Register	CLKPRS Register
2400	CFH	29H
9600	A0H	18H
57600	11H	6H

† The first bit is transmitted at the maximum clock baud rate. All subsequent bits are transmitted at the correct clock rate for the selected value.

Note: The discussion in this section is for users who want to use their own baud rate. Intel does not guarantee that all values and baud rates will operate as discussed in this section. Baud rates other than those shown in Table 13-3 are not tested. Selection and correct operation of values other than those tested is the responsibility of the customer.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

35. 272753-001, Table 3, Page 6.

Issue: The IDCODEs provided for the C- step 5V and 3V devices are reversed. They should read, 5V 2827 0013H and 3V 2027 0013H.

Affected Docs: *AP-720 Programming Flash Memory through the Intel386(TM) EX Embedded Microprocessor JTAG Port*, order number 272753-001.

36. 272485-002, Section 13.2.1, Page 13-12

Issue: At the end of this section, add this note:

Note: With autotransmit enabled, the first bit in the output is transmitted at the maximum baud rate. All subsequent bits then follow at the selected baud rate. This means that the first bit can be much narrower than subsequent bits, causing the receiver to miss the first bit, or miscount bits. When the maximum baud rate is chosen, all bits transmit correctly.

Affected Docs: *Intel386™ EX Embedded Microprocessor User's Manual*, order number 272485-002.

37. 272777-001, Table 9, Page 20

Issue: Incorrect Low Byte listed in the table for the CS5 Low Address. The correct Low Byte is C0 (was shown as 80).

Affected Docs: *EXPLR1 Embedded PC Evaluation Platform Application Note*, order number 272777-001.