Special Environment Intel386^{TM} EX Embedded Processor B-Stepping Device Errata and Feature Enhancements

Presented in this document are known errata for the B-1 step production version of the Special Environment 80386EX embedded processor. (Any processor marked with the DESC number, 5962-9557001xxx, is a B-1 step production part.) A workaround, if known, is also presented. This document will be updated as time progresses to either indicate new errata or to indicate additional workarounds.

The table below summarizes the errata and identifies which errata applies to the B-1 stepping version.

Errata #	Description	Stepping Fix
3	WDT lockout sequence may not properly reload down counter	C
5	Timer generated DMA requests may not initiate properly after reset	C
6	DMA temporary register not flushed when software aborts transfer	C
8	Write sequence to enable expanded I/O space does not need to be sequential	C
9	32-bit WDT register is oriented as High word, Low word in I/O space	No Change
10	Remap configuration register at 22H and 23H have 10-bit address decode	C
11	CAS lines do not remain active during idle states between INTA cycles	No Change
12	Cannot pipeline DMA bus cycles	C
13	DSR1#/STXCLK pin has a permanent weak pull-up	C
14	READY# floats in T1 after internal bus cycle	C
17	TDO pin may float in user mode causing excessive current consumption	C
18	Overflow enable bits (TOV0, TOV1, ROV0, ROV1) in DMAOVFE register do not function as specified	C
19	Upper DMA byte count registers (DMA0BYC1, DMA1BYC1) change to 0F0H when the lower byte count registers underflow	No Change
20	At high baud rates, there is not enough time to turn the transmitter off by resetting the TEN bit	C
21	Write sequence for WDTCLR lockout does not need to be sequential	C
24	TEN bit in SSIOCON1 register turns off transmit channel too soon	C
27	Internal HOLD request synchronization may cause data/code fetches to be corrupted	C
29	Insufficient Address Hold Time after WR# goes high	C

The format of the document is as follows:

Errata Number, Module Affected
new errata will be added in sequence
Introduction
brief description of the errata
Report Date
First time errata was reported
Stepping Fix:
Indicates which stepping of the device the errata has been or will befixed (if known)
Description:
Detailed description of the errata, how it occurs and the effects it has on device/system operation.
Application:
Indicates which applications are most likely to be affected by the errata.
Workaround:
Hardware and/or software steps to restore the system to proper operation.

Errata #3, WDT
WDT lockout sequence may not properly reload down counter
Reported 2.9.94
Stepping Fix: C
Description:
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).
Application:
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
```

Errata #5, DMA
Timer generated DMA requests may not initiate properly after reset
Reported 2.9.94
Stepping Fix: C
Description:
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.
Application:
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.

Errata #6, DMA
DMA temporary register not flushed when software aborts transfer
Reported 2.9.94
Stepping Fix: C
Description:
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.
Application:

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.

Errata #8, CPU
Write sequence to enable expanded I/O space does not need to be sequential.
Reported 2.9.94
Stepping Fix: C
Description:
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 intent that if the sequence is not followed, 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).
Application:
This errata should not be a concern for most system software designs.
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 assure proper operation if the errata is fixed in a future stepping.

Errata #9, WDT
32-bit WDT register is oriented as High word, Low word in I/O space
Reported 4.27.94
Stepping Fix: no change
Description:
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.
Application:
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.

Errata #10, CPU
Remap configuration register at 22H and 23H have 10-bit address decode
Reported 4.27.94
Stepping Fix: C
Description:
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 (ex., 22H, 1022H, 2022H, etc.).
Application:
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.

Errata #11, ICU
CAS lines do not remain active during idle states between cascaded INTA cycles
Reported 4.27.94
Stepping Fix: no change
Description:

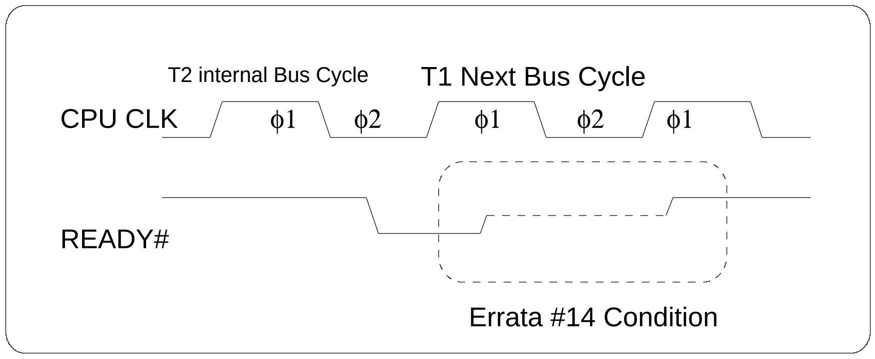
The Interrupt controller CAS lines do not maintain a valid value during the four idle states between the first INTA cycle and the second INTA cycle (assumes an externally cascaded device is present). The three CAS lines (CAS0-CAS2) are OR'd with address lines A16-A18. During the first INTA bus cycle, 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.
Application:
This errata only applies to those applications that expand interrupts using an external 8259A PIC device as slave interrupt controllers.
Workaround:
The CAS lines must be latched and preserved throughout both of the INTA bus cycles.

Errata #12, DMA
Cannot pipeline DMA bus cycles
Reported 4.27.94
Stepping Fix: C
Description:
During a DMA bus cycle, NA# does not function properly with the DMA unit.
Application:
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.

Errata #13, I/O Ports
DSR1#/STXCLK pin has a permanent weak pull-up
Reported 4.27.94
Stepping Fix: C
Description:
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.
Application: ALL
Workaround: None. The external driver must be strong enough to pull down the pin. The weak pullup is approximately 5K ohms.

Errata #14, BIU
READY# floats in T1 after internal bus cycle
Reported 4.27.94

Stepping Fix: C
Description:
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 CSU, where READY# is driven inactive during T1 of the next bus cycle.



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

Errata #17, Pin Circuitry
TDO pin may float in user mode causing excessive current consumption
Reported 10.7.94
Stepping Fix: C
Description:
The TDO pin usually floats in user mode. This could turn on the unused input buffer at the pin resulting in high leakage current.
Application:
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 pullup resistor on the TDO pin.

Errata #18 DMA
Overflow enable bits (TOV0, TOV1, ROV0, ROV1) in DMAOVFE register do not function as specified
Reported 10.7.94
Stepping Fix: C
Description:
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.

Application:
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.

Errata #19 DMA
Upper DMA byte count registers (DMA0BYC1, DMA1BYC1) change to 0F0h when the lower byte count registers underflow
Reported 10.7.94
Stepping Fix: no change
Description:
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, DMA0BYC1 or DMA1BYC1, is changed to 0F0H whenever the lower register underflows.
Application:
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 it's 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.

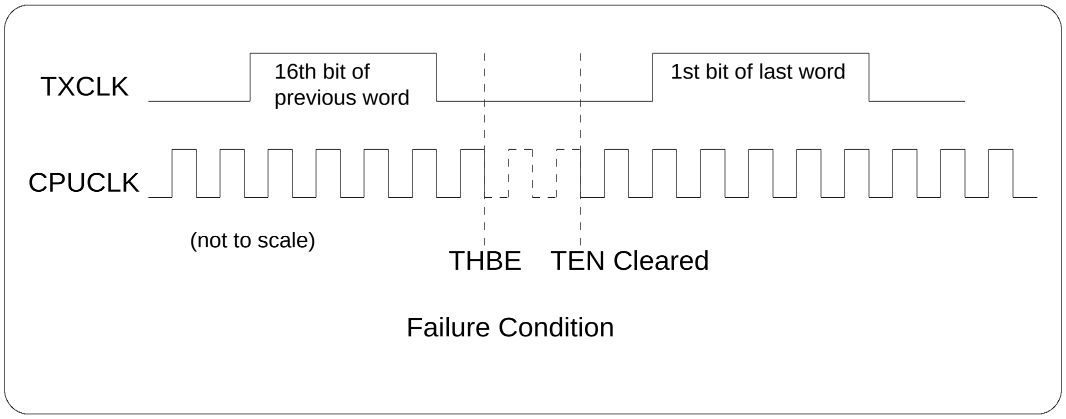
Errata #20 SSIO
At high baud rates, there is not enough time to turn the transmitter off by resetting the TEN bit
Reported 10.7.94
Stepping Fix: C
Description:
The SSIO transmitter is turned off by clearing the TEN bit in the SSIOCON1 control register. In the case where the SSIO is transmitting at high baud rates (i.e. the DMA is being used to service the transmitter) it is possible that there is not enough time between when a THBE (transmit holding buffer empty) interrupt occurs and the previous word is completely shifted out of the shift register for the processor to clear the TEN bit. 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 one time. 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 at, and any other factors which may affect interrupt latency.
Application:
Applications that require non-continuous high speed synchronous serial transfers.

Workaround:
This errata will simply 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 Programmers Reference Manual or Hardware Reference Manual.

Errata #21 WDT
Write sequence for WDTCLR lockout does not need to be sequential
Reported 10.7.94
Stepping Fix: C
Description:
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 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.
Application:
This errata should not be a concern for most system software designs.
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 assure proper operation if the errata is fixed in a future stepping.

Errata #24, SSIO
TEN bit in SSIOCON1 register turns off transmit channel too soon
Reported 10.7.94
Stepping Fix: C
Description:
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, then 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.

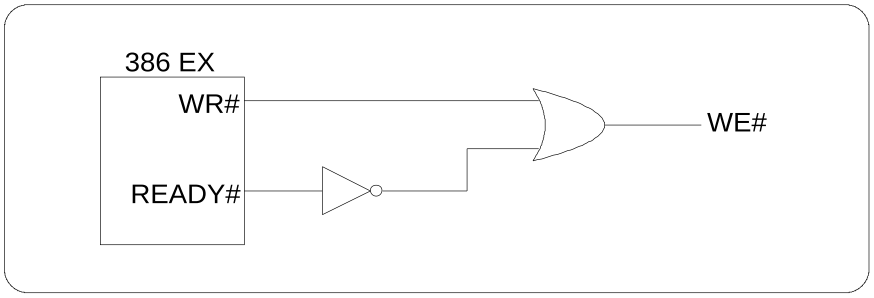


Application:
All applications that require non-continuous low speed synchronous serial transfers.
Workaround:
In order to assure that the last word is transmitted, the TEN bit must not be cleared until after the TXCLK transitions from low to high when the 1st bit of the last word is being transmitted. 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.

Errata #27, DMA
Internal HOLD request synchronization may cause data/code fetches to be corrupted.
Reported 2.1.95
Stepping Fix: C
Description:
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 just the wrong time, then the bus arbiter will continue to give the bus to the requestor, 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.
Application:
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 requestor cycle generates DACK, the target cycle doesn’t.

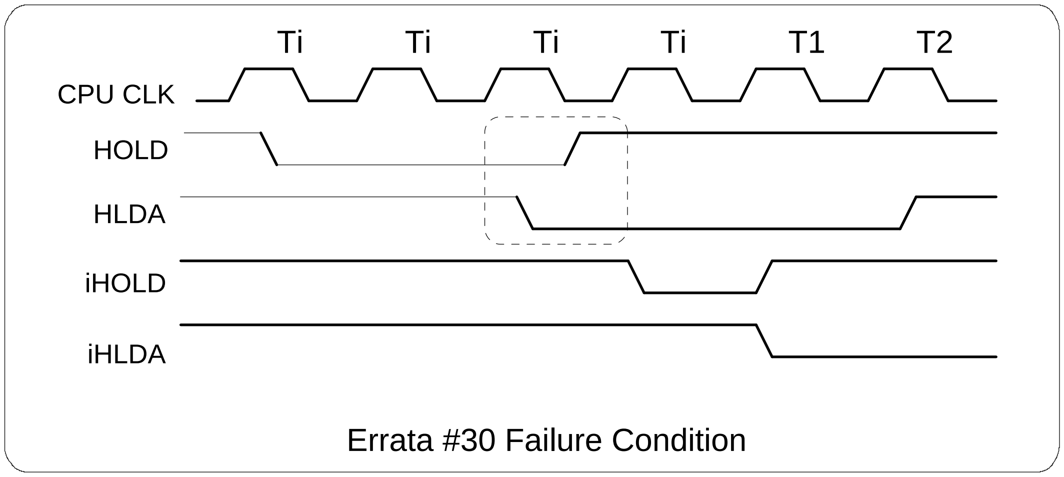
Consequently, the logic must be able to ensure that DREQ is left active until the channel is serviced.

Errata #29, BIU
Insufficient Address Hold Time after WR# goes high.
Reported 2.1.95
Stepping Fix: C
Description:
The address hold time after WR# goes high does not meet the data sheet spec of 5nS.
Application: 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 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 BBS or through literature.

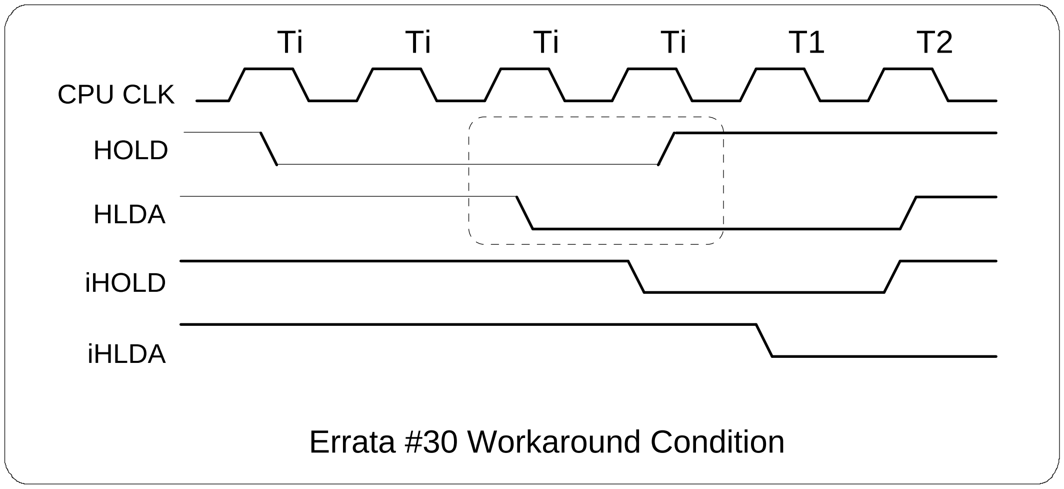


Errata #30, BA
HLDA inactive to HOLD active arbitration could improperly float the bus.
Reported 2.1.95
Stepping Fix: C
Description:
In a HOLD/HLDA cycle the HOLD signal must not be re-asserted until after HLDA is returned inactive. If, however, HOLD is asserted in the same T-state that HLDA goes inactive 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 HOLDand HLDA signals on a 386SX 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 it's iHOLD, and then the core removes iHLDA. As soon as the iHLDA is release 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 it's bus cycle and, at the same time, the bus arbiter has floated the external bus.



Application: 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 example waveform below.



Errata #31 Pin Circuitry
FLT# does not float all outputs
Reported:4.1.95
Stepping Fix: C
Description:
Asserting the FLT# pin does not float any of the I/O pins that have permanent weak pullups/pulldowns or I/O pins that have temporary weak pullups/pulldowns that are turned

on. In these cases, the actual pin drivers will be tri-stated, but the pullups/pulldowns will force them to the given state.
Application:
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 pullup/down has been turned off then asserting FLT# will float that pin. If, however, the target board is reset, then the reset action takes priority over the FLT# and turns the temporary pullup/down on again. If the application uses an external pullup/down that pulls the pin to the opposite value as the internal pullup/pulldown, then the pin voltage will be somewhere between 0V and Vcc.
Workaround:
Make certain that external pullups/downs pull each pin to the same level as any internal pullups/downs. See appendix A table A-4 in the Users manual for a description of which pins have weak pullups/downs.

Errata #32 SIO
SIO Break could be missed
Reported:4.1.95
Stepping Fix: C
Description:
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 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 dufferent 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.
Application:
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.

Errata #33 Bus Arbiter
Arbitration between Powerdown and Refresh cycles may leave control signals in wrong state.
Reported:7.12.95
Stepping Fix: C
Description:
Entering powerdown mode requires a HALT cycle to be executed after the powerdown bit in the PCON 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.
Application:
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.

Feature Enhancements

The following features have been added to the B-1 stepping:

Enhancement:
New configuration Mux to route Slave IR4 and IR5 interrupts to either INT6 or DMAINT
Description:
Multiplexors are being added to the slave IR4 and slave IR5 interrupt inputs. This will allow the DMAINT and INT6 interrupts to be routed to either the slave IR4 input or the slave IR5 input.
Impact to System:
The slave IR4 interrupt input is traditionally used to support mouse interrupts. Adding this mux allows the EX to take full advantage of existing mouse drivers.

Enhancement:
BS8# is don’t care for internal bus cycles.
Description:
Bus cycles that access the internal registers of the integrated peripherals will ignore the state of the BS8# pin.
Impact to System:
It is now possible to logically tie BS8# to ground for system designs that implement a total 8-bit system design. Previously, BS8# could not be driven low during bus cycles that accessed internal peripheral registers.

Enhancement:

Independent DMA/Interrupt control for SIO unit.

Description:

The transmit buffer empty interrupt enable (TBE) and receive buffer full interrupt enable (RBF) bits in both SIO channel Interrupt Enable registers will only control the enabling and disabling of interrupts. DMA requests are now always connected to the DMA input mux and will not be disabled if interrupts are disabled.

Impact to System:

It is now possible to disable the receive and/or transmit interrupts and still allow the generation of DMA requests. Previously, DMA requests were also controlled by the TBE and RBF bits, meaning that you potentially had to respond to an interrupt for every DMA transfer.

— End —