AMD 186CC/CU/CH

Revision B2 and B3 (PRL 4003h) Errata and Documentation Defects/Enhancements

Note: **All Errata in this errata sheet apply to Silicon Revision B2 and Silicon Revision B3 of the Am186CC . There are no functional differences between Rev B2 and Rev B3. The Rev level was changed to support production enhancements.** *All production packages were marked B2 before production enhancements . All production packages were marked B3 after production enhancements. This errata sheet applies to all parts marked B2 and it also applies to all parts marked B3.*

Note: Am186CC/CH/CU User's Manual Amendment (PID # 21914/1) and Register Set Manual Amendment (PID# 21916/1) are here: **http://www.amd.com/products/epd/techdocs/index.html**

Revision History:
01/21/99 - Created
02/02/99 - Added Erratum #B2-06
03/03/99 - Merged Rev B2 and Rev B3 Errata Sheets
03/03/99 - Renamed all errata prefixes from #B2... to #B3...
03/03/99 - Added Errata #B3-07, #B3-08, and #B3-09
03/03/99 - Converted document format from FRAME to WORD
03/16/99 - Moved Erratum B3-00 to Doc Section. Did not renumber remaining errata.
09/16/99 - Changed the name and other contents of erratum #9.
09/15/99 - Added Errata 10 through 25.
09/15/99 - Added Doc Defect #5 (AM186/AM188 Instruction Set Manual)
11/29/99 - Added Doc Defect #6 (Output pin drive capability)
11/29/99 - Added Doc Defect #4 (Output pin drive capability)
11/29/99 - Added Doc Enhancement #7 (PIO Supply Current Limit:)
01/05/00 - Changed Revision Status on Errata #B3-10 through #B3-25
02/21/01 - Added Documentation Defect #8 (MOV Instruction Clock Cycles)
02/21/01 - Added #B3-26(SmartDMA Corrupted Transmit Buf (CC/CH) when using HOLD)
02/21/01 - Added #B3-27(/PCM_TSC_A signal generated incorrectly when toggling EN of TSACON)
02/21/01 - Added Documentation Updates #9 and #10.
02/21/01 - Added Doc Defect #11
04/16/01 - Removed Doc Defect #2, #6 and #7, since they are included in the updated version of document
Am186CC Microcontroller Data Sheet, PID # 21915B
Am186CH Microcontroller Data Sheet, PID # 22024B
Am186CU Microcontroller Data Sheet, PID # 22025B
Am186CC/CH/CU Microcontrollers User's Manual, PID # 21914B
Am186CC/CH/CU Microcontrollers Register Set Manual, PID # 21916B
Removed Doc Defect #1, #3, #9 and #10, since they are included in
Am186CC/CH/CU Microcontrollers User's Manual Amendment, PID # 21914B/1
Am186CC/CH/CU Microcontrollers Register Set Manual Amendment, PID # 21916B/1
Added Doc Defect #12 and #13

Table of Contents:

#B3-01 - USB Controller May not Recognize an Extended EOP Signal

Errata #B3- 01	*USB Controller May not Recognize an Extended EOP Signal which occurs when Multiple Hubs are Attached*			
Revision: **Status:**	**B2/B3**	**C0**		
	Affected	**Fix Planned**		
System Symptom:	Setup: There are more than three Hubs between The USB Host and the 186CC. Symptom: The 186CC does not recognize messages from the USB Host.			
Errata Description:	The Single Ended '0' which signals end of packet (EOP) could get extended so long that the 186CC UDC categorizes the message as bad and throws it away. This means that the USB Host and the 186CC can not communicate.			
HW / SW Work Around:	None Work-around with System Limitations: 1) Do not place more than 3 Hubs between the USB Host and the 186CC.			

#B3-02 - UART/HSUART Data Write to Shift Register

Errata #B3- 02	*Data Write to Shift Register is based off the Divided clock in UART/HSUART*			
Revision: **Status:**	**B2/B3** **Affected**	**C0** **No fix planned**		
System Symptom:	Two back to back data byte writes cannot safely be written to the LS/HS Uart hold ing register when the Transmitter Empty bit (TEMT) is set. This breaks compatibil ity with previous 186 parts where it has always been safe to write two bytes of data to the transmitter when TEMT bit is set. Note: This applies to HS UART only when the txt FIFO is disabled.			
Errata Description:	Data written to the transmit holding register for the UART/HSUART is moved into the shift register based on the divided clock rather than on the processor clock. This means that, when TEMT is active, data written to the holding register does not move into the shift register for one full bit time.			
HW / SW Work Around:	Do not perform back to back writes to the Txt holding reg when the TEMT bit is set. Wait for the THRE (transmit holding register empty) bit to be set before writing the next data byte.			

#B3-03 - Cycle Immediately Proceeding Bus Hold does not Terminate Properly

Errata #B3- 03	*Cycle Immediately Proceeding a Bus Hold does not Terminate Properly*			
Revision: **Status:**	**B2/B3**	**C0**		
	Affected	Fix Planned		
System Symptom:	Potential for multiple devices driving the 186 bus concurrently when using external Bus Mastering.			
Errata Description:	The Middle and Peripheral Chip Select pins (MCS[0.3], PCS[0.7]) will NOT de-assert before three-stating when entering a bus hold cycle where the immediate prior cycle had either a MCS[0.3] or PCS[0.7] pin active. NOTE: MCS pins can be muxed for DRAM use (RAS1 and CASx signals) and exhibit the same behavior.			
HW / SW Work Around:	H/W Work Around: If External Bus Mastering is required, add pull-up resistors (1 Kohm) to all PCS[0.7] and MCS[0.3] pins used as Chip Select or DRAM signals.			

#B3-04 - HSUART and UART Transmitters Insert Extra Stop Bits

Errata #B3- 04	*HSUART and UART Transmitters Insert Extra Stop Bits with H/W Flow Control Enabled*			
Revision:	**B2/B3**	**C0**		
Status:	**Affected**	**Fix Planned**		
System Symptom:	Setup: System is using High Speed and/or Low Speed UART Transmitter. H/W flow con trol is enabled. Symptom: Delays are noticed between (HS)UART Transmit Frames			
Errata Description:	This problem only occurs when H/W flow control is enabled, and affects both High Speed and Low Speed UART Transmitters: 1) If the CTS pin is asserted (high to low transition) during the stop bit time or after, the transmitter will wait an additional two bit times before beginning transmission of the next frame. Under normal operation (without a receive fifo on the receiving device) this would result in two extra stop bits. 2) If the CTS pin is asserted during the transmission of the last data bit, a single extra stop bit is inserted. 3) If CTS is asserted before the last data bit begins transmission or if it is never de-asserted, then no extra stop bits are inserted			
HW / SW Work Around:	None S/W Work Around with System Limitation: 1) Minimize CTS toggling by utilizing a UART FIFO on the Receiving device 2) Don’t use H/W flow control			

#B3-05 - Unable to Exit AutoBaud Mode

Errata #B3- 05	*Unable to Exit AutoBaud Mode*			
Revision: **Status:**	**B2/B3**	**C0**		
	Affected	**No fix planned**		
System Symptom:	Setup: HSUART is setup for AutoBaud mode. Before reception of an AutoBaud character (to initiate and complete the AutoBaud sequence), AutoBaud mode is DISABLED by clearing the Auto Baud enable bit. Symptom: Errors occur during reception of HSUART frames			
Errata Description:	If the abaud bit in the HSPCON1 register has been set (enabling autobaud), simply clearing the enable bit will NOT properly Exit autobaud mode. If the bit is cleared, the start bit for the first frame received will be ignored and the next low bit will be interpreted as the start bit for the next frame.			
HW / SW Work Around:	S/W Work Around needed to properly Exit AutoBaud mode 1) Set the Baud Divisor register (HSPBDV) to 1 (this is for efficiency only) 2) Clear the abaud bit in HSPCON1 3) Wait for two Rx clocks - the HSUART state machine needs to actually see two Rx clocks so it would be safest to wait for a period of at LEAST THREE Rx clocks (3 * 16 * HSPDDV Processor or UCLKs) 4) Turn off the rmode bit - needed to reset the state machine 5) Restore the value of the HSPBDV register 6) Turn the rmode bit back on.			

#B3-06 - USB Interrupt Endpoint ACT_REQ Bit NOT Set

Errata #B3- 06	*USB Interrupt Endpoint ACT_REQ bit is NOT Set after a USB Reset or Suspend Condition*			
Revision: **Status:**	**B2/B3** **Affected**	**C0** **Fix Planned**		
System Symptom:	Setup: Using the USB Interrupt Endpoint Symptom: Following a USB Reset or Suspend Condition, the Interrupt Endpoint FIFO is not flushed and the ACT_REQ bit is not set in the Interrupt Endpoint Control/Status register (IEPCTL). When a subsequent IN command is issued from the Host - stale data is returned.			
Errata Description:	Following a USB Reset or Suspend condition, All USB Endpoints are supposed to have their FIFO's flushed and the endpoint buffer action required (ACT_REQ) bits set active. This action is NOT occurring on the Interrupt endpoint.			
HW / SW Work Around:	S/W Work Around If Interrupt Endpoint is required for System: 1) Setup an interrupt to occur on a Reset AND a Suspend Condition (via USB UIMASK2 register) 2) In the Int Handlers for Reset and Suspend - CLEAR the NOT_FLUSHED bit in the Interrupt Endpoint Control/Status register (IEPCTL). This will flush the FIFO of old data and will set ownership of the data buffer to the device S/W (i.e. set the ACT_REQ bit) **NOTE:** Any of the programmable Data Endpoints (A - D) can be used as an Interrupt end point. If used for that purpose the S/W work-around above is not required.			

#B3–07 - UART/HSUART Incorrectly Report Receive Data Ready

Errata #B3- 07	*UART/HSUART Incorrectly Report Receive Data Ready after being Disabled then Re-enabled*			
Revision:	**B2/B3**	**C0**		
Status:	**Affected**	**No fix planned**		
System Symptom:	Setup: Using the UART or HSUART Receiver (Enabled). The Receiver is disabled, then re-enabled (via RMODE bit). Symptom: An unexpected character is found in the Receive Data Register/FIFO.			
Errata Description:	There are TWO similar issues that may cause the behavior listed above: 1) If the (HS)UART receiver is disabled while the RDR (receive data ready) status bit is SET and the receiver is then re-enabled, it will report the unread character again. 2) If the (HS)UART receiver is disabled while the receiver is in-frame, i.e. after a start bit has been detected but before the RDR bit has been set for this frame, the receiver will continue reception of the frame when it is re-enabled. This will result in the RDR bit being set before one frame time has elapsed after the receiver is re-enabled. The received character's least significant bit(s) will reflect the RxD line before the receiver was disabled while the most significant bit(s) will reflect the RxD line after the receiver is re-enabled. This behavior is not dependent on how long the receiver is disabled. **NOTE:** During the time the receiver is disabled it will correctly report no data pending.			
HW / SW Work Around:	S/W Work Around This work around requires status checking BEFORE disabling the (HS)UART receiver and AFTER enabling it as well. <u>Before Disabling (via RMODE bit):</u> Check the RDR bit state. If it is clear (no data available), disable the Receiver. If it is set (data available), read the data then check RDR again. Continue until RDR is clear after a data read. <u>After Enabling (via RMODE bit):</u> Enable the Receiver. Check the IDLE bit state. If set (receiver NOT in frame) then NO other action is required. If clear, the (HS)UART believes there is an incoming frame. Wait until RDR is set and then read and discard the frame. **NOTE:** It may take up to a full frame time before the frame is received and reported.			

#B3-08 - Smart DMA Ring Buffer Reads/Writes do not Function Correctly

Errata #B3- 08	*Smart DMA Ring Buffer Reads/Writes do not Function Correctly when Rings are Located in 8 bit SRAM*			
Revision:	**B2/B3**	**C0**		
Status:	**Affected**	**No Fix Planned**		
System Symptom:	Setup Using 8 bit SRAM for Smart DMA Ring Buffer memory. Symptom: Smart DMA data transfers do not occur at all OR they are transferred to/from the wrong addresses.			
Errata Description:	When the Smart DMA Ring Buffers are located in 8 bit memory, the Descriptor Reads and/or Writes to this buffer area are not done correctly. Instead of two 8 bit transfers only one 16 bit transfer occurs which results in garbage being read or written into the upper 8 bits of the descriptor word. The Smart DMA controller was architected to ONLY perform 16 bit Ring buffer reads/writes. Memory size is not accounted for and assumed to be 16 bit. **NOTE:** If ONLY the Data buffers (not the Ring buffers) are located in 8 bit memory the transfers work fine. This ONLY affects the Ring memory access.			
HW / SW Work Around:	Work Around Do not Use 8 bit SRAM for Smart DMA. Use 16 bit DRAM or SRAM.			

#B3-09 - UART/HSUART May Lose Data (Extended Reads only)

Errata #B3- 09	*UART/HSUART may Lose Data using Extended Reads with or without FIFO Enabled and with or without Polled Mode*			
Revision: **Status:**	**B2/B3**	**C0**		
	Affected	**Fix Planned**		
System Symptom:	<u>Setup:</u> Using UART/HSUART. Extended Reads are used to check for Receive Data Ready (RDR bit set). <u>Symptom:</u> An expected incoming character is lost. S/W may hang in a polling loop waiting for the lost character.			
Errata Description:	When S/W is using Extended Reads for Receive Data Ready status checking (polling the receive data register (H)SPRXD), it is possible for the (H)SPRXD register to be read just as RDR bit is being set. If this occurs the data returned on the read does NOT indicate that the bit was set - it is cleared in the register causing the data to be lost (i.e never reported). No other status bits are affected. The other status bits are considered invalid when RDR is not set.			
HW / SW Work Around:	<u>S/W Work Arounds:</u> 1) Do not use Extended Reads.			

#B3-10 - USB Clear Feature Command Incorrectly Clears Stall

Errata #B3- 10	*USB CLEAR_ FEATURE Command Incorrectly Clears a STALL Condition*			
Revision: **Status:**	**B1** **Affected**	**B2/B3** **Fixed**	**C0** **Fixed**	
System Symptom:	<u>Setup:</u> A USB Endpoint (sans Control Endpoint0) is STALLED. It has NOT been Configured via SET_CONFIGURATION. <u>Symptom:</u> A CLEAR_FEATURE command clears the STALL condition. An Invalid Command indication is the expected behavior.			
Errata Description:	USB 1.1 specifically states that a CLEAR_FEATURE command used to clear a STALL condition is ONLY valid when the device is in a "Configured State". If the device is in an "Address State", it is ONLY valid in reference to Control Endpoint0. USB 1.0 and 1.1 state that for a device's function to be used, it MUST be configured. Once it is configured (via a Set_Configuration), the device is in the "Configured State". Therefore, clearing the STALL condition while NOT in the "Configured State" is incorrect behavior (unless Control Endpoint0). The correct behavior is to indicate an Invalid Command.			
HW / SW Work Around:	None			

#B3-11 - Violation of GCI and PCM Frame Sync Hold Times

Errata #B3- 11	*Violation of GCI and PCM Frame Sync Hold Times*			
Revision: **Status:**	**B1**	**B2/B3**	**C0**	
	Affected	**Affected**	**Fixed**	
System Symptom:	Two separate but similar issues - see below: Setup: 1) Using GCI Interface with external Clock and Frame Sync inputs. 2) Using PCM Interface with external Clock and Frame Sync inputs. Symptom: 1) Violation of the GCI Frame Delay to Clock Specification (t_{FD}). The Frame Sync pulse may not be synchronized with its associated GCI Clock edge. It may be recognized one GCI clock EARLY. If this occurs the1st data bit would be lost (i.e. the 1st actual data bit would be written / read one clock early). *2)* Violation of the PCM Hold Time from Clock low to FSC Valid specification (t_{HCF}). The Frame Sync pulse may not be synchronized with its associated PCM Clock edge. It may be recognized one PCM clock EARLY. If this occurs the1st data bit would be lost (i.e. the 1st actual data bit would be written / read one clock early).			
Errata Description:	1) Frame Sync (GCI_FSC_A) is not meeting the 0 nS required hold time. The Frame Sync signal requires a small amount of hold time (2 nS). See Specification t_{FD} under GCI Bus Timing in the 186CC Data Sheet. 2) Frame Sync (PCM_FSC_x) is not meeting the 0 nS required hold time. The Frame Sync signal requires a small amount of hold time (2 nS). See Specification t_{HCF} under PCM Highway TIming (Timing Slave) in the 186CC Data Sheet.			
HW / SW Work Around:	H/W Work-Around: 1) Add a delay of 2nS to the Frame Sync input pin (GCI_FSC_A) by adding a 510 Ohm Series Resistor. 2) Add a delay of 2nS to the Frame Sync input pin (PCM_FSC_x) by adding a 510 Ohm Series Resistor.			

#B3-12 - HSUART Baud Rate Divisor Register Erroneously Reads 0

Errata #B3- 12	*High Speed UART Baud Rate Divisor Register may Erroneously be read as ’0’ immediately after AutoBaud Detection is complete.*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: Performing AutoBaud detection via the High Speed UART Interface. Symptom: When AutoBaud detection is complete, the Baud Rate Divisor register (HSPBDV) is updated with the detected value. If this register is polled during the AutoBaud process, it is possible that a value of ’0’ (an illegal divisor) may be read before it is updated with the proper value.			
Errata Description:	The AutoBaud logic clears the Baud Rate Divisor register prior to setting it with the calculated value. If this register is polled continuously by software, it is possible that a value of ’0’ may be read.			
HW / SW Work Around:	S/W Work-Around: Do not attempt to read the calculated value in the High Speed Baud Rate Divisor Register (HSPBDV) until the Auto Baud enable bit (ABAUD) in the High Speed Serial Port Control1 Register (HSPCON1) is cleared by H/W. When this bit is cleared, AutoBaud detection is complete.			

#B3-13 - HSUART AutoBaud Logic Incorrectly Uses Divided Clock

Errata #B3- 13	*High Speed UART AutoBaud Logic Incorrectly Uses the Divided Clock*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: Performing AutoBaud detection via the High Speed UART Interface. The High Speed Serial Baud Rate Divisor register (HSPBDV) is programmed with a large divisor value BEFORE AutoBaud mode is enabled. Symptom: The calculated (and written) Baud divisor value from the AutoBaud process may be erroneous.			
Errata Description:	The High Speed UART incorrectly uses the divided (Baud) clock during AutoBaud mode. Therefore the behavior of the AutoBaud logic is affected by the current setting of the Baud Divisor register.			
HW / SW Work Around:	S/W Work-Around: BEFORE enabling AutoBaud mode, set the High Speed Serial Baud Rate Divisor register (HSPBDV) to a value of '1'.			

#B3-14 - HSUART RTR# Erroneously Asserted during Autobaud mode

Errata #B3- 14	*RTR# Erroneously Asserted during Autobaud mode*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: AutoBaud mode enabled for the High Speed UART. Hardware flow control is enabled. Symptom: A frame sent to the 186CC is lost.			
Errata Description:	RTR# is de-asserted (goes high) briefly at the end of the start bit in autobaud mode, but doesn't stay de-asserted as expected. It erroneously gets asserted (goes low). This could cause a frame to be sent to the 186CC when it is unable to accept it, and therefore the frame would be lost.			
HW / SW Work Around:	S/W Work-Around: BEFORE enabling AutoBaud mode: 1) Set the High Speed Serial Baud Rate Divisor register (HSPBDV) to a value of '1' (see Erratum B1-17) AND 2) Wait for Several Rx clocks to elapse (suggested delay is one bit time)			

#B3-15 - HSUART Data Used in Character Match not Synchronized

Errata #B3- 15	*Data Used in Character Match not Synchronized*			
Revision: **Status:**	**B1**	**B2/B3**	**C0**	
	Affected	**Affected**	**Fixed**	
System Symptom:	Setup: Using an External Clock for the HSUART (via UCLK pin). Character Matching is Enabled for the High Speed UART. Symptom: A character match is missed or an erroneous match is reported.			
Errata Description:	When using external clocking via the UCLK pin, the clocks used by the character matching logic are not synchronized. Therefore a possibility exists where a match may be missed or an erroneous match may be reported.			
HW / SW Work Around:	None			

#B3-16 - HSUART False Start During AutoBaud Mode (Divisor Set to Zero)

Errata #B3- 16	*False Start during AutoBaud Mode Results in Baud Divisor set to illegal value*			
Revision: **Status:**	**B1**	**B2/B3**	**C0**	
	Affected	**Affected**	**Not Affected**	
System Symptom:	Setup: AutoBaud mode enabled for the High Speed UART Symptom: After AutoBaud completes, the High Speed UART Baud divisor register (HSPBDV) is set to ’0’ - an illegal value.			
Errata Description:	A False Start is a glitch of value ’0’ on the High Speed UART receive line. Currently the 186CC interprets ANY low on the receive line as valid - including a False Start. Glitches less than 8 clocks will cause AutoBaud mode to exit and a value of zero placed into the High Speed Baud Divisor register (HSPBDV).			
HW / SW Work Around:	None			

#B3-17 - HSUART TFLUSH Bit Not Flushing All Data from Transmit Pipeline

Errata #B3- 17	*High Speed UART TFLUSH Bit does NOT flush all Data out of the Transmit Pipeline*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: Using the High Speed UART with FIFO enabled. The TFLUSH bit (in HSPCON1 register) has been used to flush out the Transmit data. Symptom: An unexpected frame is transmitted.			
Errata Description:	NOTE: This issue will only occur when the Transmitter is completely full (18 bytes in all: 1 shift register byte plus 16 FIFO bytes plus 1 holding register byte) The TFLUSH bit ONLY flushes the contents of the Transmit FIFO (16 bytes). It does NOT clear the transmit holding register nor does it affect the shift register which is currently transmitting a byte. If a TFLUSH occurs when the transmitter is full (see Note above), the byte in the shift register (byte 1) will transmit to completion FOLLOWED by the byte in the holding register (the18th byte). This could lead to an unexpected frame being transmitted as generally a flush is performed to clear out ALL queued data.			
HW / SW Work Around:	S/W Work Around: BEFORE setting the TFLUSH bit, be sure that the Transmit Holding register is empty by polling the THRE bit (in HSPSTAT reg). When this bit is ‘1’ it is safe to flush the FIFO.			

#B3-18 - HSUART Receiver may get FIFO Overrun with Flow Control Enabled

Errata #B3- 18	*High Speed UART Receiver may get FIFO Overrun with Flow Control enabled*			
Revision: **Status:**	**B1**	**B2/B3**	**C0**	
	Affected	**Affected**	**Fixed**	
System Symptom:	<u>Setup:</u> Using the High Speed UART. Receive FIFO is enabled. H/W flow control is enabled. The BAUD divisor (HSPBDV) is set to 1. <u>Symptom:</u> A Receiver Overrun error occurs (OER bit set in HSPSTAT reg)			
Errata Description:	The High Speed UART receiver (with FIFO enabled) can hold up to 33 bytes (32 FIFO bytes plus 1 data register). With H/W flow control enabled, RTR# should remain de-asserted (high) after the 33rd byte is received to prevent any more data from being sent (assuming no data is being read from the FIFO). The 186CC currently has an issue where it will re-assert RTR# after the 33rd byte is received (and the receiver is full) and potentially cause an overrun error. NOTE: This behavior only occurs when the BAUD divisor register (HSPBDV) is set to 1.			
HW / SW Work Around:	None <u>Work-around with System Limitations:</u> 1) Do not use H/W flow control when the Receive FIFO is enabled and the BAUD Divisor is set to 1 2) Don't use the Receive FIFO when a BAUD divisor of 1 is used.			

#B3-19 - HSUART Overrun Immediate Bit (OERIM) Erroneously Active

Errata #B3- 19	*The Overrun Immediate bit (OERIM) is Erroneously Active wh en the High Speed UART Receive FIFO is Disabled*			
Revision: **Status:**	**B1**	**B2/B3**	**C0**	
	Affected	**Affected**	**Fixed**	
System Symptom:	Setup: Using the High Speed UART. The Receive FIFO is DISABLED. An Overrun has occurred. Symptom: The Overrun Error detected bit (OER) is set in the High Speed Serial Status Register (HSPSTAT) as expected. The Overrun immediate bit (OERIM) is ALSO set. This status bit should NOT be active when the Receive FIFO is disabled (OERIM is intended to signal that an overrun has occurred somewhere in the FIFO)			
Errata Description:	The OERIM bit remains active when the Receive FIFO is disabled. Since this bit only has meaning when the FIFO is enabled - this could cause confusion (or unexpected interrupts if enabled) to the user.			
HW / SW WORK AROUND:	S/W Work Around: Ignore the OERIM bit when the Receive FIFO is disabled (be sure to mask it off as an interrupt source).			

#B3-20 - HSUART Data Not Completely Flushed when RFLUSH is Set

Errata #B3- 20	*High Speed UART Data is Not Completely Flushed Out When the Receiver Flush bit (RFLUSH) is Set*			
Revision: **Status:**	**B1** **Affected**	**B2/B3** **Affected**	**C0** **Fixed**	
System Symptom:	Setup: Using High Speed UART. FIFO is enabled. Symptom: A receive FIFO flush is performed via the RFLUSH bit. All of the data in the Receiver is NOT cleared - indicated by the Receive Data Ready bit (RDR) in the HSPSTAT register remaining set.			
Errata Description:	When the RFLUSH bit is set, the Receive Data ready (RDR) bit remains high and a subsequent read of the High Speed UART Data Register will retrieve the first data byte. This occurs because the Data Register does NOT get flushed, only (up to) 32 bytes in the FIFO. The flush is supposed to clear ALL the Data in the Receiver.			
HW / SW Work Around:	S/W Work Around: After performing a flush via the RFLUSH bit, read the Receive Data Register (RDR) one time.			

#B3-21 - HSUART Takes Longer Than Expected to Detect Break

Errata #B3- 21	*(HS)UART Takes Longer than Expected to Detect a Break Condition*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: Using UART or HSUART to receive an incoming Break condition. Symptom: A Break condition is being missed. The Break condition (RXD line held low) must be held longer than expected before the (HS)UART detects the condition.			
Errata Description:	The Break condition Must be held for three (3) frame times instead of the expected two (2) frame times if the RXD line is NOT Idle. For Idle conditions a break time of one (1) frame time is required.			
HW / SW Work Around:	S/W Work Around: Hold the Break condition for 3 frame times to ensure detection by the (HS)UART Receiver for ALL line conditions.			

#B3-22 - HSUART Bit Issues

Errata #B3- 22	*(HS)UART Idle Bit issues*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Two separate issues - see below: <u>Setup:</u> 1) Using HSUART, AutoBaud is enabled. 2) Using UART or HSUART, The Receiver is disabled then re-enabled. <u>Symptom:</u> 1) When AutoBaud is enabled, the IDLE bit de-asserts IMMEDIATELY (i.e. shows a non idle state), regardless of the state of the RXD line. 2) When (HS)UART is re-enabled, the IDLE bit does NOT get asserted immediately as expected, it is taking about 2 bit times.			
Errata Description:	1) HSUART logic incorrectly de-asserts the IDLE bit as soon as AutoBaud mode is enabled. 2) When the HS(UART) is disabled, the IDLE counter is not reset. In addition when the receiver is re-enabled the IDLE bit should be asserted until the first low is seen on the RXD line - this behavior is NOT occurring.			
HW / SW Work Around:	None			

#B3-23 - HSUART Idled Bit may be cleared Erroneously

Errata #B3- 23	*(HS)UART Idled Bit may be cleared Erroneously*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: The Idled Bit in the Serial Port Status Register (H)SPSTAT has been set. Symptom: S/W is able to clear the Idled bit although the condition for setting this bit is still present, i.e. a high detected on RXD line for 40 bit times while receive data is available.			
Errata Description:	Current implementation of the IDLED bit allows S/W to clear this bit WITHOUT changing the conditions which caused it to be set - i.e. either reading all of the data or starting reception of a new frame.			
HW / SW Work Around:	S/W Work Around: Avoid clearing the IDLED bit until all of the receive data has been read.			

#B3-24 - HSUART THRE Bit Extended Reads Dependent on RDR Bit

Errata #B3- 24	*(HS)UART THRE bit via Extended Reads is Dependent on the RDR bit.*			
Revision:	**B1**	**B2/B3**	**C0**	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: Using Extended Reads with the UART or HSUART. Using the THRE bit in the Extended Read status register to determine when to write to the transmitter. Symptom: The THRE bit remains cleared even though the Transmit Holding Register is empty.			
Errata Description:	The THRE bit is readable in the RXD extended status. This bit indicates when it is valid to write data to the transmitter. In a typical case, software can be expected to use the extended read of the RXD register and, if there is no pending receive data, to perform a write to the transmitter if theTHRE is set. However in the current implementation, this copy of THRE will always read as cleared (0) if RDR is NOT set. The THRE in the RXD extended reads needs to be a simple copy of the THRE from the STAT register.			
HW / SW Work Around:	S/W Work Around: Do not use an Extended Read of the (H)SPSTAT register to determine when to write to the Transmit Holding register.			

#B3-25 - HSUART Receive FIFO is not Read Only

Errata #B3- 25	*HSUART Receive FIFO is not Read Only*			
Revision: **Status:**	**B1** **Affected**	**B2/B3** **Affected**	**C0** **Fixed**	
System Symptom:	Setup: Using HSUART with Receive FIFO enabled. Symptom: Able to write to the Read Only Receive FIFO.			
Errata Description:	The HSUART Receive FIFO should never be writable via S/W. The Current implementation erroneously allows these writes to occur, therefore the FIFO can accidently be written with data.			
HW / SW Work Around:	S/W Work Around: Do not write to the HSUART Receive FIFO			

#B3-26 - SmartDMA Corrupted Transmit Buffer (Am186CC/CH) when using HOLD

Errata #B3- 26	*SmartDMA Corrupted Transmit Buffer Errata (AM186CC/CH) when using HOLD*			
Revision:	B1	B2/B3	C0	
Status:	**Affected**	**Affected**	**Fixed**	
System Symptom:	Setup: The HOLD input signal is asserted long enough to cause a transmit FIFO that is being filled by the SDMA controller to underrun. Symptom: When using the Smart DMA(SDMA) controller, a corrupt 0xFF gets written to the buffer pointed to by the current SDMA descriptor.			
Errata Description:	When the processor is placed into a hold state by asserting the HOLD input signal long enough to cause a transmit FIFO that is being filled by the SDMA controller to underrun, the SDMA controller sometimes writes a 0xFF to the buffer pointed to by the current SDMA descriptor. The 0xFF is written to a location which is one byte prior to the location pointed to by the SDMA Channel Pair Current Receive Address Register for the affected SDMA channel. The corrupt 0xFF does not get inserted into the FIFO being used unless the current buffer is retransmitted without correcting the area of the buffer corrupted by the 0xFF. This errata has been observed when using SDMA with an HDLC channel and may possibly manifest itself in systems using SDMA with USB where the system is frequently put in hold.			
HW / SW Work Around:	S/W Work Around: When an underrun condition is detected, take the following steps. 1) Disable the SDMA transmitter by clearing the TXST bit in the appropriate SmartDMA Channel Pair Control Register. 2) Clear the ownership bit of the current SDMA descriptor. 3) Clear the appropriate status bits that are set by the underrun condition. 4) Reenable the SDMA transmitter by setting the TXST bit. 5) Rewrite the data to be transmitted by the SDMA to the current buffer which was being used when the underrun occurred. This will overwrite any possible 0xFF that was written into the buffer. 6) Set the ownership bit for the current SDMA descriptor to retransmit the current buffer. The above procedure may have to be adjusted depending on your Smart DMA descriptor ring configuration and application.			

#B3-27 - /PCM_TSC_A signal generated incorrectly when toggling EN of TSACON

Errata #B3-27	*/PCM_TSC_A signal generated incorrectly when toggling EN of TSACON*			
Revision:	**All**			
Status:	**Affected**			
System Symptom:	**<u>Setup:</u>** HDLC Channel A is configured as Raw PCM Highway and HDLC B is configured as multiplexed PCM mode. The EN bit for channel A (in TSACON) is set to a zero (channel A disabled) while the /PCM_TSC_A signal is low. The /PCM_TSC_A signal will remain low (instead of going high). It will continue to drive the pin low until TSA channel A is re-enabled. The same problem can occur if HDLC channels C or D are configured as multiplexed PCM mode. **<u>Symptom:</u>** The /PCM_TSC_A signal will remain low until TSA Channel A is enabled again. When the EN bit for Channel A (in TSACON) is set to one (channel A enabled), the /PCM_TSC_A signal will go high.			
Errata Description:	When the TSA for HDLC Channel A is disabled, the /PCM_FSC_A clock is disabled, preventing the signal on /PCM_TSC_A from being updated to allow it to drive high.			
HW / SW Work Around:	**1.** Disable the TSA for channel x (where x is B, C, or D) in the TSxCON register, by setting the EN bit to a zero. **2.** Program the TSA channel x BPSTART field in the TSxSTART register with a value that exceeds the maximum bit position possible for the time division multiplexed (TDM) frame. **3.** Enable the TSA for channel x in the TSxCON register by setting the EN bit. **4.** This procedure will allow /PCM_TSA_A to go high. There will be no data coming from this channel x even though it is enabled.			

Documentation Defects/Enhancements:

4. Status change for USB GET_ CONFIGURATION/GET_INTERFACE Erratum

This Erratum will be converted to a documentation update and will not be fixed in any future Revisions of this part. This erratum and related workarounds will be included in a documentation ammendment.

This previous erratum, #B3- 00 is now a documentation update.	*USB GET_ CONFIGURATION and GET_INTERFACE Commands may return an Un-Supported Value*			
Revision:	**B2/B3**	**C0**		
Status:	**Affected**	**Affected, no fix planned.**		
System Symptom:	An unsupported Configuration and/or Interface may be returned via the USB GET_CONFIGURATION and GET_ INTERFACE commands. Host Commands addressing a disabled endpoint or an endpoint whose alternate setting/interface values to not match the UDC will NOT be passed to the USB S/W.			
Errata Description:	*Condition 1 and 2:* USB is currently not configured (configuration = 0), and the software supports a maximum configuration which is less than the 186CC HW maximum (3) SET_CONFIGURATION command is issued by the Host with a value that is larger than the software maximum(i.e. unsupported), but not larger than the hardware maximum(3). The software will force a stall of the endpoint (in conformance with the USB spec) and will take no action with the request (i.e the device is still not configured) (condition 1) *Result:* If the host then issues a GET_CONFIGURATION command (a command which is handled solely by the USB circuitry without software intervention), the returned response is the unsupported configuration, and NOT the true current configuration. A similar situation exists (condition 2) with the SET_INTERFACE and GET_INTERFACE commands with an alternate setting that is within the range supported by the hardware, but larger than the maximum alternate setting supported by the underlying software.			

	Condition 3 and 4: The Host attempts to address a disabled endpoint (condition 3), or to an endpoint whose alternate setting and/or interface values (condition 4) do not match the 186CC UDC. *Result:* The Host commands will NOT be passed from the 186CC UDC to the S/W resulting in a time out situation on the Host.
HW / SW Work Around:	*Condition 1 and 2:* The problem with SET_CONFIGURATION and SET_INTERFACE is that the 186CC UDC has no knowledge of valid Configurations and/or Interface/Alt set tings outside the hard coded constraints. The workaround is to make EVERY con figuration (0-3) a valid configuration. This requires duplicating functionality. There is no requirement that configurations 0-3 have to be unique* *Condition 3 and 4:* The workaround for condition 3 is to enable ALL endpoints and let S/W STALL packets for requests to non existent endpoints. The workaround for condition 4 is, after receiving a SET_CONFIGURATION com mand, configure all endpoints for the new configuration. This will ensure that a host command that attempts to address a device endpoint (regardless of whether the endpoint exists) will be passed from the 186CC UDC to the S/W. * There is no need to report descriptors for each configuration permutation. This is only a means of covering illegal host behavior.
STATUS:	*No fix planned for any future revision of this part. A description of this erratum and related workaround information is included in a current documentation addendum.*

5. AM186/AM188 Instruction Set Manual PID # 21267A

Page 4-17 change the SF Flag bit settings as follows:

SF=1 if result is 0 or positive	- (should be SF = 0)
SF=0 if result is negative	- (should be SF = 1)

8. Instruction Set Manual - MOV Instruction Clock Count Incorrect

On page 4-153 of the "Am186 and Am188 Family Instruction Set Manual", the MOV instruction is discussed. The first 4 line items with the MOV instruction, list incorrect clock cycles to execute the instruction. The correct # of cycles for specific MOV instruction types is listed below.

Form :	**Opcode**	**Description:**	**Am186**	**Am188**
MOV r/m8,r8	88/r	Copy register to r/m byte	2/12	2/12
MOV r/m16,r16	89/r	Copy register to r/m word	2/12	2/16
MOV r8,r/m8	8A/r	Copy r/m byte to register	2/9	2/9
MOV r16,r/m16	8B/r	Copy r/m word to register	2/9	2/13

11. User’s Manual (PID # 21914B) – Page 7-1 – Number of Interrupt sources
Change the following statements in the 5th paragraph on page 7-1 of the User’s Manual.
1. In the 2nd Sentence regarding the CH:
From: 14 internal maskable interrupt sources To: 13 internal maskable interrupt sources
2. In the 3rd sentence regarding the CU:
From: 13 internal maskable interrupt sources To: 12 internal maskable interrupt sources

12. Data Sheet (186CC PID # 21915B) – page 22 – TMROUT0 and TMROUT1 pin descriptions correction
Page 21 for Am186CH Data Sheet (PID # 22024B) and Am186CU Data Sheet (PID # 22025B)

Page	**Item**	**Original Text**	**Change To**	**Comment**
22	[TMROUT1] [TMROUT0] Last sentence.	[TMROUT1]-[TMROUT0] are three-stated during bus-hold or reset conditions.		Remove

13. Data Sheet (186CC PID # 21915B) – page 14 and 16 – ARDY and SRDY pin description clarification
Page 13 and 16 for Am186CH Data Sheet (PID # 22024B) and Am186CU Data Sheet (PID # 22025B)

Page	**Item**	**Original Text**	**Change To**	**Comment**
14	ARDY Last paragraph	If the system does not use ARDY, tie the pin low to yield control to SRDY.	If the system does not use ARDY, tie the pin low to yield control to SRDY. **When ARDY is configured as PIO8, the internal SRDY signal is driven low.**	Clarification
16	SRDY Last paragraph	If the system does not use SRDY, tie the pin Low to yield control to ARDY.	If the system does not use SRDY, tie the pin Low to yield control to ARDY. **When SRDY is configured as PIO35, the internal SRDY signal is driven low.**	Clarification