

AMD Élan™ SC520

Rev. B0 and B0 8A Errata and Documentation Defects/Enhancements

Update History:

10/27/00 Created
12/06/00 Added Erratum #B0-03
04/26/02 Removed all previous documentation defects. All previous defects have been incorporated in updated documentation since March 2001.
04/26/02 Added 'B0 8A' to the title
06/07/02 Added documentation defect DS-1
06/07/02 Added Erratum #B0-04
05/08/05 Added Erratum #B0-05

Table of Contents:

Erratum #B0-01 - UART Receive DMA Request Stuck Asserted in 16550 Mode.....	2
Erratum #B0-02 - JTAG Port Stops Working After Processor Executes HLT Instruction.....	3
Erratum #B0-03 - Unexpected UART Receive FIFO Time-out Interrupt or DMA Request	4
Erratum #B0-04 - Unexpected Entry to AMDebug Mode when DEBUG_ENTER is Latched LOW at POR	5
Erratum #B0-05 - SWTMRILLI Count Failure	6
Documentation Defects and Enhancements	10

Erratum #B0-01 - UART Receive DMA Request Stuck Asserted in 16550 Mode

Revision:	A1	B0		
Status:	Affected	Affected		
Date Entered:	09/09/99			
System Symptom:	<p>Setup: Receive a data byte from UART in 16550-compatible mode using polled or interrupt method. Switch to DMA receive method by initializing DMA controller.</p> <p>Symptom: DMA will read receive data from UART even though receiver FIFO is empty.</p>			
Erratum Description:	<p>The UART's Receiver DMA Request line (DRQ) asserts when a data byte is deposited into the receive FIFO. However, the DRQ line sometimes fails to de-assert if reads of the Receive Buffer Register (RBR) are used to empty the FIFO. The DRQ line becomes stuck in the asserted state. If the DMA method is subsequently used to read UART receiver data, DMA reads will occur even though the receive FIFO is empty.</p> <p>The UART must be in 16550-compatible mode for this problem to appear. Varying UART DMA_MODE and receive FIFO trigger levels can affect the frequency of occurrence. Both UARTs are affected.</p>			
HW / SW Work Around:	<p>A stuck receiver DRQ line may be cleared by changing the UART to 16450-compatible mode, reading the RBR register, and then changing back to 16550-compatible mode. DMA operation will then proceed normally. The following Assembler code fragment demonstrates:</p> <pre> LES BX,UART1FCRSHAD ; FCR Shadow @ MMCR+0CC2h (UART2: +0CC6h) MOV AL,ES:[BX] ; Read initial FCR value MOV AH,AL ; Save it for later restoration AND AL,0FEh ; Mask FCR's FIFO_ENB to enter 450 mode MOV DX,UART1FCR ; FCR @ I/O 3FAh (UART2: 2FAh) OUT DX,AL ; Write new 450 mode FCR value MOV DX,UART1RBR ; RBR @ I/O 3F8h (UART2: 2F8h) IN AL,DX ; Read RBR to clear stuck DRQ MOV DX,UART1FCR ; FCR @ I/O 3FAh (UART2: 2FAh) MOV AL,AH ; FCR initial value saved earlier OUT DX,AL ; Write FCR to restore to initial value </pre> <p>This work-around should be compatible with future silicon revisions.</p>			

Erratum #B0-02 - JTAG Port Stops Working After Processor Executes HLT Instruction

Revision:	A1	B0		
Status:	Affected	Affected		
Date Entered:	07/13/00			
System Symptom:	Setup: Issue the HLT instruction through the JTAG port. Symptom: The JTAG port stops working.			
Erratum Description:	Under normal operation, the AMDebug circuitry and the JTAG port provide data about the processor state - registers, memory, trace history - to the debugger. This data can be collected by the run-control tool and passed up to a host computer for the debugger to interpret. After a HLT instruction, the run-control tool no long receives valid data, but instead receives sequences of 0x00000200 or 0x00000290 for all data requests. In this state the JTAG port will receive a software RESET command, which provides a mechanism for returning the SC520 to a proper state. This can be observed with the following run-control tools: AMC CodeTAP, AMC SuperTAP, Macraigor Raven			
HW / SW Work Around:	1) Replace all HLT instructions with the NOP instruction. 2) It is possible to use the JTAG trace command to step past a HLT instruction. If this is done, the run-control tool continues to receive valid data from the processor.			

Erratum #B0-03 - Unexpected UART Receive FIFO Time-out Interrupt or DMA Request

Revision:	A1	B0		
Status:	Affected	Affected		
Date Entered:	12/06/00			
System Symptom:	<p>Setup: Enable either of the internal UARTs for 16550-compatible mode and program Receive FIFO Register Trigger, RFRT, to a value greater than 1 byte (i.e. 4, 8, or 14).</p> <p>Symptom: A received data available interrupt is generated when the first byte of a continuous data stream is placed in FIFO. If DMA Mode is enabled, a DMA request is generated early instead of an interrupt.</p> <p>There is no data loss or corruption; an interrupt / DMA request is delivered early.</p>			
Erratum Description:	<p>Table 18-4 (page 18-3) of the Elan SC520 Register Set Manual states:</p> <p>"A FIFO time-out occurs when the receive FIFO is not empty, and more than four continuous character times have elapsed without more data being placed into or read out of the receive FIFO. Reading a character from the receive FIFO clears the time-out interrupt."</p> <p>The description of the Enable Received Data Available Interrupt bit, ERDAI, (page 18-11) in the Elan SC520 Register Set Manual states:</p> <p>"1 = Enable received data available interrupt in 16550-compatible mode, this bit also enables FIFO trigger level reached interrupt and time-out interrupt."</p> <p>In the Elan SC520 the FIFO Time-out counter starts counting the four character times as soon as the UART clocks are started, and does not require any data to be in the FIFO to qualify latching the time-out condition. Only reading a character from the FIFO will clear this latched condition. Every time the counter reaches the four character limit before the serial data can start filling the empty receive FIFO, a receive FIFO time-out state will be latched in the UART, and a FIFO time-out interrupt will be generated as soon as the FIFO is not empty.</p> <p>This erratum only affects the first character of a continuous data stream received by the UART. Following the FIFO Time-out interrupt for the first character received, the remainder of the data stream will be indicated according to the trigger value set in the RFRT bits of the UART 1 (or 2) FIFO Control registers.</p>			
HW / SW Work Around:	User's code must be able to handle a received data available time-out interrupt (or DMA request if DMA enabled) when the first byte of a continuous data stream is placed in FIFO.			

Erratum #B0-04 - Unexpected Entry to AMDebug Mode when DEBUG_ENTER is Latched LOW at POR

Revision:	A1	B0		
Status:	Affected	Affected		
Date Entered:	06/07/02			
System Symptom:	<p>Setup: Set DEBUG_ENTER and AMDEBUG_DIS LOW, using either the internal pulldowns or providing your own external pulldowns.</p> <p>Symptom: Entry to AMDebug is allowed even though DEBUG_ENTER was latched LOW at power-on reset (POR).</p>			
Erratum Description:	<p>After the POR, if REMON is started on the host PC while the RAVEN was connected to the AMDebug port, the JTAG port will lockup and the target SC520 will continue to execute code. REMON will show characteristic register data values of '00000010' until an external reset with power applied occurs (PWRGOOD reset, typically a pushbutton on the system board).</p> <p>A PWRGOOD reset will clear the TAP lockup, leaving the TAP in AMDebug mode (Run-Test/Idle). The system will not stop immediately after subsequent PWRGOOD resets unless DEBUG_ENTER is latched HIGH, but AMDebug will provide valid responses to REMON.</p> <p>A POR will also clear the TAP lockup, but will disable AMDebug mode.</p> <p>REMON would show characteristic register data values of '00000000' if AMDEBUG_DIS were set to HIGH.</p>			
HW / SW Work Around:	<p>If you wish to ensure that JTAG / AMDebug is prevented from being activated, DEBUG_DIS must be latched HIGH at POR (pullup to VCC). A jumper or a switch would allow the TAP to be activated when needed.</p>			

Erratum #B0-05 - SWTMRMILLI Count Failure

Revision:	A1	B0		
Status:	Affected	Affected		
Date Entered:	05/08/05			
System Symptom:	<p>Setup: Read the SWTMRMILLI Timer when a carry is occurring.</p> <p>Symptom: SWTMRMILLI can fail to increment properly if it is read when the SWTMRMICRO counter rolls over from 999 to 0.</p>			
Erratum Description:	<p>Proper Function:</p> <p>The software timer is intended to provide a millisecond time base with microsecond resolution. Ideal applications for this function include providing a system wide software time base, code profiling, and precise measurement of the time between events. Features of the software timer include:</p> <ul style="list-style-type: none"> * One 16-bit millisecond counter that increments with a period of one millisecond. This yields a maximum duration of 65.5 seconds. Note that this timer is accurate to the precision of the 33-MHz crystal used in the system. * A microsecond latch register that provides the number of microseconds since the last time that the millisecond register was read. * The 16-bit millisecond counter is reset to zero when it is read. * The software timer can be configured to maintain an accurate time when either a 33.000-MHz or 33.333-MHz crystal is used in the system. <p>Erratum Description:</p> <p>The cumulative research in this area, indicates that there is a Logic Design Race Condition based fault in the SWTMRMILLI/MICRO counters.</p> <p>"The Read-Only SWT Microsecond Count bit field holds the latched microsecond count value from a free running microsecond counter."</p> <p>Each read of this bit field returns the currently latched microsecond count value. On each read of the SWTMRMILLI register, the value in the internal microsecond counter is latched into this bit field (US_CNT).</p> <p>The carry pulse from the SWTMRMICRO counter rolling over from 999 to 0, can increment the SWTMRMILLI counter during a small timing window between Latching SWTMRMICRO and Clearing the count that essentially discards the updated count value. The design should have held off the read latching until the carry was completed, or at least hold off the carry action until the Read-Latch-Clear operation had completed.</p>			

	<p>About 10% of the reads returning a 999 or a zero US_CNT lose the milli-second carry. We have looked for a way to avoid reading SWTMRMILLI during the carry cycle. But, since subsequent reads of SWTMRMICRO only repeat the last LATCHED value (short of an independent timer), there is no way to determine if the next SWTMRMILLI read will occur during an update in progress.</p> <p>The odds of reading SWTMRMILLI while an update is in progress is ~0.1% and the odds of the update failing due to the read is less than 10%, or losing one millisecond in about 10,000 reads. Any timed events that return a US_CNT of 999 or zero, should be re-run to verify the value.</p> <p>The MICRO timer counts to 0x3f7 (999) and on the next count asserts a CARRY to SWTMRMILLI and a CLEAR to SWTMRMICRO.</p> <p>Reading SWTMRMILLI at this point can have four (4) possible results:</p> <ol style="list-style-type: none"> 1. SWTMRMICRO gets latched and SWTMRMILLI is read before the CARRY asserts. :: No Lost Time. 2. SWTMRMICRO gets latched as the CARRY asserts, before SWTMRMICRO is CLEARED, and SWTMRMILLI is read before the CARRY is counted then CLEARED shortly after. :: SWTMRMICRO is 0x3f7 and One Millisecond Lost with CARRY. 3. SWTMRMICRO gets latched after the CARRY asserts, SWTMRMICRO is CLEARED, and SWTMRMILLI is read before the CARRY is counted then CLEARED shortly after. :: SWTMRMICRO is 0x000 and One Millisecond Lost with CARRY. 4. SWTMRMICRO gets latched after the CARRY asserts, SWTMRMICRO is CLEARED, and SWTMRMILLI is read after the CARRY is counted then CLEARED shortly after. :: No Lost Time. <p>This is a design bug in the Software Millisecond timer. If a random read of SWTMRMILLI with SWTMRMICRO latches a value of 999 or 0, there is no way on the Sc520 to determine if it lost the Millisecond CARRY.</p>
<p>HW / SW Work Around:</p>	<p>Workaround: None with SWTMRMILLI.</p> <p>Please see previous Erratum Description for possible effects on your system.</p> <p>Depending on the customer's required use for SWMTRMILLI, and if the general purpose timers described in chapter 17 of the Elan Sc520 Microcontroller User's Manual are available then a similar function could be achieved.</p> <p>Using a 33-MHz/4 input clock to the cascaded GP timer 0 and timer 2, timer 2 could be set to a maximum count of 8,250. This would produce 1-millisecond counts to GP timer 0, and GP</p>

timer 2 counts of 121-ns. (e.g. 8.25 counts per microsecond).

The rest of the chapter explains the software implementation to read the cascaded GP timers.

Note: ==My code segment to test the SWTMRMILLI register=====

```
main:
    DB      66                // Clear counter for test
    XOR     CX,CX

    PUSH   ES
    PUSH   DS

    MOV     AX,ES              //Set up buffer in near heap
    ADD     AX,0080
    MOV     ES,AX
    XOR     DI,DI

    MOV     AX,DF00           // Point to MSCR_BASE_SC520_CDP
    MOV     DS,AX

rd_tmr:
    DB      66
    MOV     AX,[0C60]         //Read SWTMRMILLI & SWTMRMICRO as one
32-bit request
    MOV     DX,BX

    DB      66                // Push two 16-bit counter values
    PUSH   AX
    POP     BX                // Get SWTMRMILLI
    POP     BX                // Get SWTMRMICRO
    TEST   BX,BX             // Q: Is SWTMRMICRO doing a carry?
    JNZ    rd_tmr           // N: Keep Reading

    TEST   DX,DX            // Y: Q: Was the previous read also
Zero?
    JZ     rd_tmr           // Y: already counted Carry, go
back to reading timer

    DB      66                // N: Count this Carry event
    INC    CX
    TEST   CX,CX            // Q: Back to Zero yet?
    JZ     exit             // Y: Limit to 64K events for now

    TEST   AX,AX            // N: Q: Did SWTMRMILLI get the
Carry?
    JZ     rd_tmr           // N: Keep Reading

    DB      66                // Y: Log the SWTMRMILLI value
in case greater than one
    STOSW
    TEST   DI,8000          // Q: Have we filled the (32KB) 8K-
Entry buffer?
    JZ     rd_tmr           // N: Keep Reading

exit:                          // Y: Exit test and show results=>
```

```

//      CS= Total # Carries (HEX),
//      DI= 4 times # of Successful
Carries
POP      DS      // Restore Environment
POP      ES
INT      3      // Return to Debug

-e.g. 0x2381 Carries detected,
      0x2000 (0x8000/4) successful
      0x381 Failed , or ~9.87% Failures ( 0x381/0x2381 )
=====
```

Documentation Defects and Enhancements

ElanSC520 Microcontroller Datasheet, PID# 22003B

DS-1. page 50; Charaterization is complete for Icc_RTC values. The typical value remains 5 μ A (no change from simulation data). The previously unpopulated max value is 10 μ A. The minimum value was not characterized.

ElanSC520 Microcontroller User's Manual, PID# 22004B

-none

ElanSC520 Microcontroller Register Set Manual, PID# 22005B

-none