MOTOROLA

Chip Errata
DSP56366 Digital Signal Processor
Mask: 2J26D

General: To prevent the use of instructions or sequences of instructions that do not operate correctly, use the "lint563" program to identify such cases and use alternative sequences of instructions. This program is available in the Motorola DSP Tools CLAS package.

Silicon Errata

Errata Number	Errata Description	Applies to Mask
ES14	Description (added before 2/18/1996): When the DMA performs external memory accesses with priority higher than the core and both continuous mode and interrupt enable bits are set in the channel's control register, then the DMA interrupt might not occur if the core performs external memory access immediately after the enabling ($\overline{DE}$ = 1) of the DMA channel. Workaround: In this scenario any of the following alternatives can be used: a. Do not set continuous mode. b. Use dynamic DMA-core priority. c. Guarantee that the core performs at least two instructions fetched from internal memory immediately after setting of the $\overline{DE}$.	2J26D

ES21	Description (added 4/16/1996):
	
	If the DMA channel performs non-zero wait state data accesses to/from external memory and the DMA interrupt is enabled, a false interrupt may occur in addition to the correct one.
	
	Workaround: Ensure that the channel's DTD status bit in the DSTR register is set before jumping to the interrupt service routine (i.e., the interrupt is correct only when DTD is set).
	
	Example:
	
	`ORG P:I_DMA2`
	`JSSET #M_DTD2,X:M_DSTR,ISR_ ; ISR_ is interrupt service routine`
	`; label for DMA channel 2`
		2J26D

Errata Number	Errata Description	Applies to Mask
ES42	Description (added 3/3/ 98): When a Direct Memory Access (DMA) channel is in Line mode (i.e., the DMA Transfer Mode is DTM = 010) with address modes defined by DMA Three Dimensional mode D3D = 0 and DMA = 10010x (i.e., the DMA Counter (DCO) is in mode A), and the DCO value is greater than $FFF, then the DMA does not function properly. This address mode implies “no update” at the destination and “no update” or “post increment by 1” mode at the source. Workaround: Use Block Transfer mode (i.e., DTM = 000). For the DCO and DMA Address Mode (DAM) settings described in this erratum, the Line Transfer mode of DMA is identical to its Block Transfer mode, so this combination is redundant. In fact, a block containing only one line is still a block.	2J26D

ES47	Description (added 3/3/1997):
	
	If the DMA channel and the core access the same 1/4 K internal X data, Y data, or program memory page, and the DMA interrupt is enabled, a false interrupt may occur in addition to the correct one.
	
	Workaround: Ensure that the channel’s DTD status bit in the DSTR is set before jumping to the Interrupt Service Routine (i.e., the interrupt is correct only when DTD is set).
	
	Example:
	
	`ORG P:I_DMA0`
	`JSSET #M_DTD0,X:M_DSTR,ISR_; ISR_ is the Interrupt Service`
	`; Routine label for DMA channel 0`
		2J26D

Errata Number	Errata Description	Applies to Mask
ES54	Description (added 1/27/98): When a DMA channel is configured using its DMA Control Register (DCR) in the following way: • Line Transfer mode is selected (DTM[2:0] = 010) • Non-Three-Dimensional Address mode is selected (D3D = 0) • Destination Address Offset Register DOR1 or DOR3 is selected (DAM[5:3] = 001 or 011) • No Source Address Offset is selected (DAM[2:0] = 100 or 101) The DMA transfer does not function as intended. Workaround: Select Destination Address Offset Register DOR0 or DOR2 by setting DAM[5:3] = 000 or 010.	2J26D

Errata Number	Errata Description	Applies to Mask

ES84	Description (added 5/13/98):
	
	When software disables a DMA channel (by clearing the DE bit of the DCR) , the DTD status bit of the channel may not be set if any of the following events occur:
	
	a. Software disables the DMA channel just before a conditional transfer stall (Described by App B-3.5.1,UM).
	
	b. Software disables the DMA channel at the end of the block transfer (that is after the counter is loaded with its initial value and transfer of the last word of the block is completed).
	
	As a result, the Transfer Done interrupt might not be generated.
	
	Workaround:
	Avoid using the instruction sequence causing the conditional transfer stall (See DSP56300 UM, App B-3.5.1 for description) in fast interrupt service routines. Every time the DMA channel needs to be disabled by software, the following sequence must be used :
	
	`bclr #DIE,x:M_DCR ; not needed if DIE is cleared`
	`bclr #DE,x:M_DCR`
	`; instead of two instructions above, one 'movep' instruction may be used`
	`; to clear DIE and DE bits`
	`movep #DCR_Dummy_Value,x:M_DCR`
	`bclr #DE,x:M_DCR`
	`nop`
	`nop`
	
	Here, the `DCR_Dummy_value` is any value of the DCR register that complies with the following requirements:
	
	• DE is set;
	• DIE is set if Transfer Done interrupt request should be generated and cleared otherwise;
	• DRS[4:0] bits must encode a reserved DMA request source (see the following list of reserved DRS values);
	
	List of reserved DRS[4:0] values (per device):
	
	• DSP56302, DSP56309, DSP56303, DSP56304, DSP56362 — `10101-11111`
	• DSP56305 — `11011`
	• DSP56301 — `10011-11011`
	• DSP56307 — `10111-11111`
		2J26D

Errata Number	Errata Description	Applies to Mask
ES90	Description (added 6/25/98)/Modified 4/19/99: A deadlock occurs during DMA transfers if all the following conditions exist: 1. DMA transfers data between internal memory and external memory through port A. 2. DMA and the core access the same internal 0.25K memory module. 3. One of the following occurs: a. The bus arbitration system is active, i.e., $\overline{BG}$ is changing, not tied to ground. b. Packing mode (bit 7 in the AAR[3 - 0] registers) is active for DMA transfers on Port A. Workaround: One of the following, but workarounds 2, and 3 are valid ONLY to section 3 a of the errata - i.e. not valid if packing mode is used, and workaround 4 is valid only to section 3 b of the errata - i.e., not valid if bus arbitration is active. 1. Use intermediate internal memory on which there is no contention with the core. 2. Tie $\overline{BG}$ to ground, or have an external arbiter that asserts $\overline{BG}$ even if BR is not asserted. 3. Set the BCR[BRH] bit, whenever BR must be active. 4. Avoid using packing mode.	2J26D

Errata Number	Errata Description	Applies to Mask

ES94	Description (added 8/10/98):
	
	Enabling any DMA channel by software for transferring a block of data (TM=011 in the channel control register) might not work properly.
	
	Workaround:
	
	Triggering of a channel for block transfer by software can be replaced by triggering of the DMA channel for block transfer by a peripheral (e.g. Timer, SCI etc.) that is not used while the block of data should be transferred by DMA. This can be done as follows:
	
	1. Set the DSR, DDR and DCO registers of the DMA channel according to the application case.
	
	2. Transfer mode of the DMA channel (in the DCR register) should be set to TM = 000 or TM = 100 (See Section 8.1.5.3, 563xx UM).
	
	3. DMA Request Source of the DMA channel should be set according to the chosen peripheral, which should trigger the DMA channel (see Section 8.1.5.6 56300 UM and "DMA Request Sources" Table in the CORE CONFIGURATION item of the 563xx UM).
	
	4. All others fields of the DCR register, except the DE bit, should be set according to the application case.
	
	5. Configure the peripheral to assert its DMA request line;
	
	6. Set DE bit of the DCR register.
	
	Example 1:
	
	Assuming that the SCI is not used while the block of #DCO3 words is transferred by DMA channel 3, the SCI Transmit Data (TDRE = 1, DRS[4:0] = 01111) trigger can be used instead of a software trigger for channel 3.
	
	1. Initialize DMA channel registers
	
	`movep #DSR3,x:M_DSR3`
	`movep #DDR3,x:M_DDR3`
	`movep #DCO3,x:M_DCO3`
	`bset #0,x:M_PCRE`
	
	Now when the DMA channel is enabled, a transfer of the block begins.
		2J26D

Errata Number	Errata Description	Applies to Mask

	2. Enable DMA channel
	`movep #$867a40,x:M_DCR3 ;; enable DMA channel 3,`
	`;; block transfer mode,`
	`;; DRS[4:0] = 01111`
	Example 2 :
	Assuming that Timer 0 is not used while a block of #DCO3 words is to be transferred by DMA channel 3, the Timer0 (TCF0 = 1, DRS[4:0] = 10000) trigger can be used instead of a software trigger for channel 3.
	1. Initialize DMA channel registers
	`movep #DSR3,x:M_DSR3`
	`movep #DDR3,x:M_DDR3`
	`movep #DCO3,x:M_DCO3`
	2. Initialize Timer 0
	`movep #$0,x:M_TCSR0 ;;no prescaling, inv=0, mode 0,`
	`;; no interrupt, reload is disabled`
	`movep #$0,x:M_TLR0 ;;initialize load reg.`
	`movep #$0,x:M_TCPR0 ;;initialize compare reg.`
		2J26D

ES94 cont.	3. Generate DMA channel trigger
	`- option 1`
	`movep #$A48254,x:M_DCR3 ;; enable DMA channel 3, block transfer`
	`;; mode, DE isn't disabled at end of`
	`;; transfer, triggered by Timer0 (TCF0=1)`
	`bset #M_TE,x:M_TCSR0 ;; enable timer`
	`nop`
	`nop`
	`bclr #M_TE,x:M_TCSR0 ;; disable timer`
	`- option 2`
	`movep #$848254,x:M_DCR3 ;; enable DMA channel 3, block transfer`
	`;; mode, DE is disabled at the end of`
	`;; the transfer, triggered by Timer0`
	`;; (TCF0=1)`
	`bset #M_TE,x:M_TCSR0 ;; enable timer`
	`nop`
	`nop`
	`bclr #M_TE,x:M_TCSR0 ;; disable timer`	

Errata Number	Errata Description	Applies to Mask

ES94 cont.	- option 3
	
	`bset #M_TE,x:M_TCSR0 ;; enable timer`
	`movep #$848254,x:M_DCR3 ;; enable DMA channel 3, block transfer`
	`;; mode, DE is disabled at the end of`
	`;; the transfer, triggered by Timer0`
	`;; (TCF0=1)`
	`bclr #M_TE,x:M_TCSR0 ;; disable timer`
	
	Following are the differences between these three options:
	
	option 1 : The DMA channel should be enabled only for the first block transfer. An additional block transfers can be triggered by the following sequence:
	
	`bset #M_TE,x:M_TCSR0 ;; enable timer`
	`nop`
	`nop`
	`bclr #M_TE,x:M_TCSR0 ;; disable timer`
	
	Note:
	
	Execution of this sequence can be interrupted because the DMA channel is triggered after the TE bit of the TCSR0 register is set. The TE bit must be cleared no later than 33554430 DSP clock cycles after it is set. Otherwise, an undesirable trigger for the DMA channel is generated. The DMA channel must be disabled every time when Timer 0 is used for another purpose.
	
	option 2 : The whole sequence must be used every time when a block transfer is to be triggered.
	
	Note:
	
	Execution of this sequence may be interrupted because the DMA channel is triggered after the TE bit of the TCSR0 register is set. The TE bit must be cleared no later than 33554430 DSP clock cycles after is is set. Otherwise, an undesirable trigger for the DMA channel is generated.
	
	option 3: The whole sequence must be used every time a block transfer is to be triggered.
		2J26D

Errata Number	Errata Description	Applies to Mask
ES94 cont.	**Note:** Execution of the first two instructions of the sequence must be uninterruptable. The TE bit of the TCSR0 register must be cleared no later than 33554430 DSP clock cycles after it is set. Otherwise, an undesirable trigger for the DMA channel is generated. Another peripheral can be used for this purpose, but taking into consideration its specific features.	2J26D
ES95	Description (added 8/15/98): If more than a single DMA channel is enabled while the DSP stays in the WAIT processing state, and triggering one of the DMA channels causes an exit from the WAIT state (See A-6.115, UM), triggering another DMA channel might cause improper DMA operation. Workaround: Assure that only a single DMA channel can be triggered during DSP WAIT state. If the application cannot guarantee this, other DMA channels should be disabled before the WAIT processing state is entered and then reenabled after WAIT state is exited.	2J26D
ES104	Description: (added 11/24/98): An improper operation may occur when all the following conditions apply: • The DMA channel is in a mode that does not automatically clear the DE bit at the end of the block (DTM[2:0] = 1xx in DCR). • This channel is disabled by software (by clearing DE in DCR) while it is triggered for a new transfer. • The previous operation is not yet completed. Workaround: The DMA channel should be disabled only when it is not triggered for a new transfer, i.e. when the DACT bit in the DSTR register is cleared. **Note:** To perform this operation most efficiently, all other DMA channels should be disabled.	2J26D

Errata Number	Errata Description	Applies to Mask
ES114	Description (added 4/19/99, revised 4/30/99): A DMA channel may operate improperly when the address mode of this channel is defined as three-dimensional (D3D=1) and DAM[5:0] = 1xx 1 10 or DAM[5:0] = 01xx 10 (i.e., triple counter mode is E). Workaround: Use the triple counter modes C(DAM[1:0]=00) or D(DAM[1:0]=01) instead of the E(DAM[1:0]=10) mode.	2J26D

ES115	Description (added 4/19/99):
	
	When a DMA channel (called channel A) is disabled by software clearing the channel's DCR[DE] bit, the DTD bit may not get set, and the DMA end of the block interrupt may not happen if one of the following occurs:
	
	1. There is another channel (channel B) executing EXTERNAL accesses, and the DE bit of channel A is being cleared by software at the end of the channel B word transfer - if channel B is in Word transfer mode, or at the end of the channel B line transfer - if channel B is in Line Transfer mode, or at the end of the channel B block transfer - if channel B is in Block transfer mode.
	
	2. This channel (A) is executing EXTERNAL accesses, and the DE bit of this channel (A) is being cleared by software at the end of the channel B word transfer - if channel B is in Word transfer mode, or at the end of the channel B line transfer - if channel B is in Line transfer mode.
	
	Workaround:
	
	kAvoid executing a DMA external access when any DMA channel should be disabled. This can be done as follows. Every time the DMA channel needs to be disabled by software, the following sequence must be used:
	
	`;; initialize an unused DMA channel "C"`
	`movep #DSR_swflag, x:M_DSRC ;; here DSR_swflag is an`
	`;; unused X, Y or P memory`
	`;; location, should`
	`;; be initialized to`
	`;; $800000`
	`;; M_DSRC - address of the`
	`;; channel C DSR register.`
		2J26D

Errata Number	Errata Description	Applies to Mask

ES119	Description (added 7/27/99):
	
	When disabling DAX interrupts in the XCTR register (or disabling the DAX) exactly when the DAX interrupt is pending at the DSP core, an unknown interrupt vector may be generated.
	
	Workaround:
	
	The user should first turn off DAX interrupt requests by clearing DAL0 and DAL1 bits in the IPRP register, then issue 6 NOP instructions, then clear interrupt enable bits in the XCTR register (or disable DAX), issue another 6 NOP instructions, and finally re-enable the DAL0 and DAL1 bits on IPRP as shown in the following example:
	
	`;; Clear the relevant bits in the IPRP register`
	`BCLR #M_DAL0,x:M_IPRP`
	`BCLR #M_DAL1,x:M_IPRP`
	`;; Issue 6 NOP instructions`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`;; Clear the interrupt enable bits in the XCTR register`
	`BCLR #M_XBIE,x:M_XCTR`
	`BCLR #M_XUIE,x:M_XCTR`
	`BCLR #M_XDIE,x:M_XCTR`
	`;; Issue 6 NOP instructions`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`;; Restore the required DAX interrupt level`
	`BSET #M_DAL0,x:M_IPRP`
	`BSET #M_DAL1,x:M_IPRP`
		2J26D

Errata Number	Errata Description	Applies to Mask
ES129	Description (added 7/5/2000) If either HDM1 or HDM0 in the HCR register is set, the value of bits 6 and 5 of the ICR register should reflect the value of HDM[1–0], respectively. Instead, both ICR bits reflect DM1. The Host DMA functionality is correct, only the value read from the ICR register is wrong. Workaround: To determine the Host DMA operating mode, read the HDM[2-0] bits in the HCR register.	2J26D
ES130	Description (added 7/5/2000): If the DMA writes to HTX, the HTDE status flag is not set immediately after the data is transferred to the RXH:RXM:RXL registers. HTDE will be set only when one of the following conditions occurs: 1) The external host accesses one of the HDI08 registers. 2) The 56300 core accesses one of the on-chip peripherals. 3) The DMA reads HRX. No data is lost. Workaround: There is none. This bug introduces more lattency in the assertion of the next DMA transfer request.	2J26D
ES132	Description (added 6/18/2001): For operating frequencies above 90 MHz, if both the DMA channel and the core simultaneously access the same 1/4K page of internal memory (X, Y or program), improper DMA channel operation may occur. Workaround: Avoid simultaneous DMA and core accesses to the same 1/4 K page of internal memory when operating at frequencies above 90 MHz.	2J26D

Documentation Errata

	Errata Description	Applies to Mask

ED1	Description (revised 11/9/98):
	
	XY memory data move does not work properly under one of the following two situations:
	
	1. The X-memory move destination is internal I/O and the Y-memory move source is a register used as destination in the previous adjacent move from non Y-memory
	
	2. The Y-memory move destination is a register used as source in the next adjacent move to non Y-memory.
	
	Here are examples of the two cases (where x:(r1) is a peripheral):
	
	Example 1:
	
	`move #$12,y0`
	`move x0,x:(r7) y0,y:(r3) (while x:(r7) is a peripheral).`
	
	Example 2:
	
	`mac x1,y0,a x1,x:(r1)+ y:(r6)+,y0`
	`move y0,y1`
	
	Any of the following alternatives can be used:
	
	a. Separate these two consecutive moves by any other instruction.
	
	b. Split XY Data Move to two moves.
	
	Pertains to: DSP56300 Family Manual, Section B-5 "Peripheral pipeline restrictions.
		2J26D

ED3	Description (added 5/7/1996): A one-word conditional branch instruction at LA-1 is not allowed. **Pertains to:** DSP56300 Family Manual, Appendix B, Section B.4.1.3	2J26D
ED4	Description (added 10/13/1997): The following instructions should not start at address LA: MOVE to/from Program space {MOVEM, MOVEP (only the P space options)} This is a documentation update to the Appendix B, DSP56300 Family Manual.	2J26D

	Errata Description	Applies to Mask
ED7	Description (added 1/27/98): When activity passes from one DMA channel to another and the DMA interface accesses external memory (which requires one or more wait states), the DACT and DCH status bits in the DMA Status Register (DSTR) may indicate improper activity status for DMA Channel 0 (DACT = 1 and DCH[2:0] = 000). Workaround: None.	2J26D
ED15	Description (added 7/21/98): The DRAM Control Register (DCR) should not be changed while refresh is enabled. If refresh is enabled only a write operation that disables refresh is allowed. Workaround: First disable refresh by clearing the BREN bit, than change other bits in the DCR register, and finally enable refresh by setting the BREN bit.	2J26D

	Errata Description	Applies to Mask

ED28	Description (added 1/7/1997; identified as Documentation Errata 2/1/99):
	
	When two consecutive LAs have a conditional branch instruction at LA-1 of the internal loop, the part does not operate properly. For example, the following sequence may generate incorrect results:
	
	`DO #5, LABEL1`
	`NOP`
	`DO #4, LABEL2`
	`NOP`
	`MOVE (R0) +`
	`BSCC _DEST ; conditional branch at LA-1 of internal loop`
	`NOP ; internal LA LABEL2`
	`NOP ; external LA LABEL1`
	`NOP`
	`NOP`
	`_DEST NOP`
	`NOP`
	`RTS`
	
	Workaround: Put an additional NOP between LABEL2 and LABEL1.
	
	Pertains to: DSP56300 Family Manual, Appendix B, Section B-4.1.3, “At LA-1.”
		2J26D

ED32	Description (added 11/9/98; identified as a Documentation errata 2/1/99):
	
	When returning from a long interrupt (by RTI instruction), and the first instruction after the RTI is a move to a DALU register (A, B, X, Y), the move may not be correct, if the 16-bit arithmetic mode bit (bit 17 of SR) is changed due to the restoring of SR after RTI.
	
	Workaround:
	
	Replace the RTI with the following sequence:
	
	`movec ssl,sr`
	`nop`
	`rti`
	
	Pertains to: DSP56300 Family Manual. Add a new section to Appendix B that is entitled “Sixteen-Bit Compatibility Mode Restrictions.”
		2J26D

	Errata Description	Applies to Mask

ED33	Description (added 12/16/98; identified as a Documentation errata 2/1/99):
	
	When Stack Extension mode is enabled, a use of the instructions BRKcc or ENDDO inside do loops might cause an improper operation.
	
	If the loop is non nested and has no nested loop inside it, the erratais relevant only if LA or LC values are being used outside the loop.
	
	Workaround:
	
	If Stack Extension is used, emulate the BRKcc or ENDDO as in the following examples. We split between two cases, finite loops and do forever loops.
	
	1) Finite DO loops (i.e. not DO FOREVER loops)
	
	==========
	
	BRKcc
	
	Original code:
	
	`do #N,label1`
	`.....`
	`.....`
	`do #M,label2`
	`.....`
	`.....`
	`BRKcc`
	`.....`
	`.....`
	`label2`
	`.....`
	`.....`
	`label1`
	
	Will be replaced by:
	
	`do #N, label1`
	`.....`
	`.....`
	`do #M, label2`
	`.....`
	`.....`
	`Jcc fix_brk_routine`
	`.....`
	`.....`
		2J26D

	Errata Description	Applies to Mask
ED33 cont.	nop_before_label2 nop ; This instruction must be NOP. label2 .... label1 .... fix_brk_routine move #1,lc jmp nop_before_label2 ENDDO ------ Original code: do #M,label1 .... do #N,label2 .... ENDDO .... label2 .... label1 Will be replaced by: do #M, label1 .... do #N, label2 .... JMP fix_enddo_routine	2J26D

	Errata Description	Applies to Mask
ED33 cont.	nop_after_jmp NOP ; This instruction must be NOP. .... label2 .... label1 .... fix_enddo_routine move #1,lc move #nop_after_jmp,la jmp nop_after_jmp 2) DO FOREVER loops ========== BRKcc ----- Original code: do #M,label1 .... do forever,label2 .... BRKcc .... label2 .... label1	2J26D

	Errata Description	Applies to Mask

ED33 cont.	Will be replaced by:
	`do #M,label1`
	`.....`
	`.....`
	`do forever,label2`
	`.....`
	`.....`
	`JScc fix_brk_forever_routine ; <---`
	`note: JScc and not Jcc`
	`.....`
	`.....`
	`nop_before_label2`
	`nop ; This instruction must be NOP.`
	`label2`
	`.....`
	`.....`
	`label1`
	`....`
	`....`
	`fix_brk_forever_routine`
	`move ssh,x:<..> ; <..> is some reserved not used address (for temporary data)`
	`move #nop_before_label2,ssh`
	`bclr #16,ssl ;`
	`move #1,lc`
	`rti ; <---- note: "rti" and not "rts" !`
	`ENDDO`
	`------`
	`Original code:`
	`do #M,label1`
	`.....`
	`.....`
		2J26D

	Errata Description	Applies to Mask
ED33 cont.	do forever,label2 .... ENDDO .... label2 .... label1 Will be replaced by: do #M,label1 .... do forever,label2 .... JSR fix_enddo_routine ; <--- note: JSR and not JMP nop_after_jmp NOP ; This instruction should be NOP .... label2 .... label1 .... fix_enddo_routine nop move #1,lc bclr #16,ssl move #nop_after_jmp,la rti ; <--- note: "rti" and not "rts" **Pertains to:** DSP56300 Family Manual, Section B-4.2, “General Do Restrictions.”	2J26D

	Errata Description	Applies to Mask
ED34	Description (added 1/5/99; identified as a Documentation errata 2/1/99): When stack extansion is enabled, the read result from stack may be improper if two previous executed instructions cause sequential read and write operations with SSH. Two cases are possible: Case 1: For the first executed instruction: move from SSH or bit manipulation on SSH (i.e. jclr, brclr, jset, brset, btst, bsset, jsset, bsclr, jsclr). For the second executed instruction: move to SSH or bit manipulation on SSH (i.e. jsr, bsr, jscc, bscc). For the third executed instruction: an SSL or SSH read from the stack result may be improper - move from SSH or SSL or bit manipulation on SSH or SSL (i.e., bset, bclr, bchg, jclr, brclr, jset, brset, btst, bsset, jsset, bsclr, jsclr). Workaround: Add two NOP instructions before the third executed instruction. Case 2: For the first executed instruction: bit manipulation on SSH (i.e. bset, bclr, bchg). For the second executed instruction: an SSL or SSH read from the stack result may be improper - move from SSH or SSL or bit manipulation on SSH or SSL (i.e., bset, bclr, bchg, jclr, brclr, jset, brset, btst, bsset, jsset, bsclr, jsclr). Workaround: Add two NOP instructions before the second executed instruction. **Pertains to:** DSP56300 Family Manual, Appendix B, add a new section called “Stack Extension Enable Restrictions.” Cover all cases. Also, in Section 6.3.11.15, add a cross reference to this new section.	2J26D

	Errata Description	Applies to Mask
ED38	Description (added 7/14/99): If Port A is used for external accesses, the BAT bits in the AAR3-0 registers must be initialized to the SRAM access type (i.e. BAT = 01) or to the DRAM access type (i.e. BAT = 10). To ensure proper operation of Port A, this initialization must occur even for an AAR register that is not used during any Port A access. Note that at reset, the BAT bits are initialized to 00. **Pertains to:** *DSP56300 Family Manual*, Port A Chapter (Chapter 9 in Revision 2), description of the BAT[1 –0] bits in the AAR3 - AAR0 registers. Also pertains to the core chapter in device-specific user's manuals that include a description of the AAR3 - AAR0 registers with bit definitions (usually Chapter 4).	2J26D

	Errata Description	Applies to Mask

ED40	Description (added 11/11/99):
	When an instruction with all the following conditions follows a repeat instruction, then the last move will be corrupted.:
	1. The repeated instruction is from external memory.
	2. The repeated instruction is a DALU instruction that includes 2 DAL registers, one as a source, and one as destination (e.g. tfr, add).
	3. The repeated instruction has a double move in parallel to the DALU instruction: one move's source is the destination of the DALU instruction (causing a DALU interlock); the other move's destination is the source of the DALU instruction.
	Example:
	rep #number
	tfr x0,a x(r0)+,x0 a,y0 ; This instruction is from external memory
	|__|_____|------|----------> This is condition 3 second part.
	|_____|----------> This is condition 3, first part - DALU interlock
	In this example, the second iteration before the last, the "x(r0)+,x0" doesn't happen. On the first iteration before the last, the X0 register is fixed with the "x(r0)+,x0", but the "tfr x0,a" gets the wrong value from the previous iteration's X0. Thus, at the last iteration the A register is fixed with "tfr x0,a", but the "a,y0" transfers the wrong value from the previous iteration's A register to Y0.
	Workaround:
	1. Use the DO instruction instead; mask any necessary interrupts before the DO.
	2. Run the REP instructions from internal memory.
	3. Don't make DALU interlocks in the repeated instruction. After the repeat make the move. In the example above, all the "move a,y0" are redundant so it can be done in the next instruction:
	`rep #number`
	`tfr x0,a x(r0)+,x0`
	`move a,y0`
	If no interrupts before the move is a must, mask the interrupts before the REP. **Pertains to:** *DSP56300 Family Manual*, Rev. 2, Section A.3, "Instruction Sequence Restrictions."
		2J26D

	Errata Description	Applies to Mask
ED41	Description (added 10/09/1997): If the stack extension is enabled, the instructions listed below should not be placed as the next-to-last or as the last instruction of a DO loop (i.e., should not appear at LA-1 or LA). The instructions are: XY Memory Data Move (A-6.76) X Memory Move (A-6.71) Y Memory Move (A-6.73) Long Memory Data Move (A-6.75) Immediate Short Data Move (A-6.68) Register to Register Data Move (A-6.69) Address Register Update (A-6.70) X Memory and Register Data Move (A-6.72) Y Memory and Register Data Move (A-6.74) *Arithmetic Instructions that allow Parallel Moves listed above* IFcc and IFcc.U (A-6.41) Workaround: Insert a NOP or other instruction not listed above as the next-to-last and last instructions in the DO loop.	2J26D
ED42	Description (added on 3/22/2000) The DMA End-of-Block-Transfer interrupt cannot be used if DMA is operating in the mode in which DE is not cleared at the end of the block transfer (DTM = 100 or 101). **Pertains to:** *DSP56300 Family Manual,* Rev. 2, Section 10.4.1.2, "End-of-Block-Transfer Interrupt." Also, Section 10.5.3.5, "DMA Control Registers (DCR[5–0]," discussion of bits 21 – 19 (DTM bits).	2J26D

NOTES

1. An over-bar (i.e., $\overline{xxxx}$) indicates an active-low signal.

2. The letters in the right column tell which DSP56366 mask numbers apply.
3. The Motorola DSP website has additional documentation updates that can be accessed at the following URL:

   http://www.motorola-dsp.com/

-end-