Chip Errata
DSP56303 Digital Signal Processor
Mask: 2J22A

General remark: In order to prevent the usage of instructions or sequences of instructions that do not operate correctly, the user is encouraged to use the “lint563” program to identify such cases and use alternative sequences of instructions. This program is available as part of the Motorola DSP Tools CLAS package.

Silicon Errata

Errata Number	Errata Description	Applies to Mask
ES16	Description (added 7/7/1997): When the chip is powered up with PLL enabled (PINIT = 1), the skew between EXTAL and CLKOUT after the PLL locks cannot be guaranteed at high frequency (over 50 MHz, not 100% tested). Workaround: If skew between EXTAL and CLKOUT is needed, power up with PINIT = 0, and then enable the PLL by software.	2J22A

Errata Number	Errata Description	Applies to Mask
ES46	Description (added 7/7/1997): When a DMA controller is in a mode that clears $\overline{\text{DE}}$ (i.e., TM = 0xx), if the core performs an external access with wait states or there is a transfer stall (see Appendix B, Section B.3.4.2 in the DSP56300 Family Manual) or a conditional transfer interlock (see Appendix B, Section B.3.5.1) during the last DMA channel transfer, there will be one additional DMA word transfer. Workaround: There are three system-dependent workarounds for this problem. The user should test the system using these workarounds to determine which one to use in the particular system to overcome this problem. The workarounds are: Workaround 1: a. Prepare one additional memory word in the source and destinations buffers. This data should be ignored. b. Activate a DMA Interrupt Service Routine (ISR) or poll the DTD bit to ensure block transfer completeness. In the DMA ISR or the handler routine after status polling, reload the values of the address registers. Workaround 2: a. Use a DMA mode that does not clear DE (i.e., TM = 1xx) and activate the DMA interrupt. b. In the ISR, execute the following operations in the order listed: clear DE, update the address registers, and set DE. Workaround3: a. Use a DMA mode that does not clear DE (i.e., TM = 1xx). b. Change the address mode from linear addressing to 2D or from 2D to 3D and use an offset register to update the address automatically at the end of the block. **Note:** If the user can not use one of these workarounds, there may be other possible system-dependent workarounds.	2J22A

Errata Number	Errata Description	Applies to Mask

ES47	Description (added 7/7/1997):
	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`
		2J22A

Chip Errata
DSP56303 Digital Signal Processor
Mask:2J22A

Errata Number	Errata Description	Applies to Mask

ES48	Description (added 7/7/1997):
	Note: This is a subset of Errata # 46 (i.e., in every case that errata # 48 occurs, errata # 46 occurs, but not vice versa).
	When a DMA controller is in a mode that clears $\overline{DE}$ (i.e., TM = 0xx), and it transfers data to an external memory with two or more wait states, and the DSP core performs an external access with wait states or there is a transfer stall (see Appendix B, Section B.3.4.2 in the DSP56300 Family Manual) or a conditional transfer interlock (see Appendix B, Section B.3.5.1) during the last DMA channel transfer, the destination pointer for a subsequent DMA transfer may not be reprogrammed correctly. There are two defined workarounds to prevent the occurrence of this condition and one recovery code that should be used if the workarounds can not be used in a specific system:
	Workaround 1:
	a. Use a DMA mode that does not clear DE (i.e., TM = 1xx) and activate the DMA interrupt.
	b. In the DMA ISR, clear DE, update the address registers, and set DE.
	Workaround 2:
	a. Use a DMA mode that does not clear DE (i.e., TM = 1xx).
	b. Change the address mode from linear addressing to 2D or 2D to 3D and use an offset register to update the address automatically at the end of the block.
	Recovery (to recover if the condition occurs):
	a. Enable the DMA interrupt.
	b. Use the following code in the DMA ISR:
	`movep #dummy_source, x:M_DSRi`
	`movep #dummy_dest, x:M_DDRi`
	`movep #0, x:M_DEOi`
	`movep #9E0240, x:M_DCRi ; initiate one dummy transfer`
	`; if the bug occurred, the`
	`; transfer will be to the`
	`; old_block_last_dest + 1`
	`; and not to the dummy_dest`
	`nop`
		2J22A

Errata Number	Errata Description	Applies to Mask
ES53	Description (added 9/25/1997): Using the JTAG instruction code 1111 ($F) or 1101 ($D) for the BYPASS instruction may cause the chip to enter Debug mode (which then correctly sets the Status bits (OS[1:0]) in the OnCE Status and Control Register (OSCR[7:6]) and asserts the $\overline{\text{DE}}$ output to acknowledge the Debug mode status). Workaround: Use one of the following alternatives: a. If possible, do not use instruction code 1111 ($F) or 1101 ($D) for the BYPASS instruction. Use one of the other defined BYPASS instruction codes (i.e., any code from 1000–1100 ($8–$C) or 1110 ($E)). b. If you must use instruction code 1111 ($F) or 1101 ($D), use the following procedure: — While the $F or $D instruction code is in the Instruction Register, ensure that the JTAG Test Access Port (TAP) state machine does not pass through the JTAG Test-Logic-Reset state while accessing any JTAG registers (i.e., Instruction Register, Boundary Scan Register, or ID Register). — Before using any other JTAG instruction, load one of the other BYPASS instruction codes (i.e., any code from 1000–1100 ($8–$C) or 1110 ($E)) into the instruction register. Then, any other JTAG instruction may be used.	2J22A

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 manner: • 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.	2J22A

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, DSP56306, DSP56362 — `10101-11111`
	• DSP56305 — `11011`
	• DSP56301 — `10011-11011`
	• DSP56307 — `10111-11111`
		2J22A

Errata Number	Errata Description	Applies to Mask
ES89	Description (added 6/25/98): If the SCI Receiver is programmed to work with a different serial clock than the SCI Transmitter so that either the Receiver or Transmitter is using the external serial clock and the other is using the internally-generated serial clock—RCM and TCM in the SCCR are programmed differently)—then the internal serial clock generator will not operate and the SCI portion (Receiver or Transmitter) clocked by the internal clock will be stuck. Workaround: Do not use SCI with the two SCI portions (Receiver and Transmitter) clocked by different serial clocks; use either both externally or both internally clocked. Or: When using both portions of the SCI (Receiver & Transmitter), do not program different values on RCM and TCM in the SCCR.	2J22A

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.	2J22A

Errata Number	Errata Description	Applies to Mask

ES91	Description (added 7/22/98):
	If the Core reads data from the HRX while instructions are fetched from the memory Expansion Port (Port A) using 2 or more wait states, data may be lost.
	Workaround :
	There are three possible workarounds:
	1) The host should guarantee that there is no more than one word in the TXH:TXM:TXL-HRX data path at any time. This can be achieved if the host writes a word to the HI08 only when the TRDY flag is set (i.e. the data path is empty).
	2) Use a service routine running from fast (i.e. one wait state) external memory or internal memory to read the HRX read code; ensure that code that is fetched from slow (i.e. more than 1 wait state) external memory is located at least 4 instructions after the HRX register is read. For example:
	`READ_HRX_DATA`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	Note:
	a) Interrupt requests that fetch instructions from slow external memory should be masked during this service routine. Nonmaskable interrupt (NMI) request routines must not be in external memory.
	b) If running from fast external memory and if a DMA channel accessing external memory is used, then the DMA may cause extra wait states to the core. Thus, the DMA should have a lower priority than the core so that the core can access the external memory with no more than 1 wait state.
	3) Read the HRX using one of the channels of the on-chip DMA controller.
		2J22A

Errata Number	Errata Description	Applies to Mask
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.	2J22A
ES104	Description (added 11/20/98): An improper operation may occur when a DMA channel uses the following transfer modes: • DTM(2:0) = 100 • DTM(2:0) = 101 where the DE bit is not automatically cleared at the end of block and the DMA channel is disabled by software (DE bit is cleared) while it is triggered for a new transfer. Workaround: The DMA channel should be disabled only when it cannot be triggered by a new transfer. Use one of the following alternatives: 1. The system configuration must guarantee that no DMA trigger can occur while the DE bit is cleared. 2.The following sequence disables the DMA channel: a/ Wait until the DTD bit is cleared b/ Clear the DE bit c/ Wait until the DTD bit is set	2J22A

Errata Number	Errata Description	Applies to Mask

ES105	Description (added 11/25/98):
	If the core clears the HCIE bit on the HCR register while an interrupt is issued and the vector is read by the interrupt controller, then when the interrupt is serviced, HCP is not cleared, since the clear equation is conditioned by HCIE=1.
	Workaround :
	There are two possible workarounds:
	1) If only host commands are used as possible interrupt source to the core (i.e. HTIE and HRIE are both 0), then instead of bit-set and bit-clear to the HCIE, do bit-set and bit-clear instructions on the IPRP register for the HIE bit.
	2) If option "1" cannot be used, first turn off host interrupt requests (all the possible sources) by clearing the bits in the IPRP register that are used to enable/disable HDI08/HI08 interrupt requests (HPL on the DSP56307 and differs from chip to chip. Consult the user's manual for your DSP563xx chip to find the correct bit IPRP bit names.) Then issue 6 NOP instructions, clear the HCIE bit on the HCR, issue another 6 NOP instructions and finally re-enable the IPRP interrrupt request enable/disable bit as follows:
	`;; Clear the relevant bits on IPRP register acording to the`
	`;; current host interrupt priority settings`
	`BCLR    #M_HPL0,x:M_IPRP`
	`BCLR    #M_HPL1,x:M_IPRP`
	`;; Issue 6 NOP instructions`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`;; Clear the HCIE bit on HCR register to turn off host commands`
	`BCLR    #M_HCIE,x:M_HCR`
	`;; Issue 6 NOP instructions`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`NOP`
	`;; Restore the required host interrupt level`
	`BSET    #M_HPL0,x:M_IPRP`
	`BSET    #M_HPL1,x:M_IPRP`	

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.	2J22A

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:
	Avoid 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.`
		2J22A

Errata Number	Errata Description	Applies to Mask
ES115 cont.	movep #DDR_swflag, x:M_DDRC ;; DDR_swflag is an unused ;; X, Y or P memory ;; location, should be ;; initialized to $000000 ;; M_DDRC - ;; address of the channel C ;; DDR register . movep #TR_LENGTH, x:M_DCOC ;; see below the definition ;; of the TR_LENGTH value, ;; M_DCOC - address ;; of the channel C DCO register .movep #1f0240, x:M_DCRC ;; M_DCRB - address of the ;; channel C DCR register. ;; Set transfer mode - ;; block transfer, ;; triggered by ;; software highest ;; priority, continuous ;; mode on no-update ;; source and destination ;; address mode X memory ;; location for source ;; and destination (can be ;; chosen by ;; user accordingly to ;; DSR_swflag/DDR_swflag)	2J22A
ES115 cont.	;; disable DMA channel "A" ori #3, mr ;; mask all interrupts bset #23, x:M_DCRC ;; enable DMA channel C bclr #23,x:DDR_swflag,* ;; wait until DMA channel C ;; begin transfer bclr #23, x:M_DCRA ;; disable DMA channel A nop nop jclr #M_DTDA, x:M_DSTR,* ;; polling DTD bit of the ;; DMA channel A, The TR_LENGTH value can be defined as the maximum length of the external DMA transfer——from the length of the read DMA cycle and from the length of the write DMA cycle. The length of the external read/write DMA cycle can be defined as the length of the PORTA external access. The length of the internal read/write DMA cycle can be defined in the errata case as 2 DSP clock cycles. The TR_LENGTH can be found as sum of the lengths of the DMA read and DMA write cycles.	2J22A

Documentation Errata

Errata Number	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.
		2J22A

ED3	1. 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	2J22A

ED4	Description (added 7/7/1997):
	The following instructions should not start at address LA:
	MOVE to/from Program space {MOVEM, MOVEP (only the P space options)}
	This is not a bug but a documentation update (Appendix B, DSP56300 Family Manual).
		2J22A

ED7	Description (added 1/27/98): When activity is passed 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. This is not a bug, but a specification update.	2J22A
ED9	Description (added 1/27/98): When the SCI is configured in Synchronous mode, internal clock, and all the SCI pins are enabled simultaneously, an extra pulse of 1 DSP clock length is provided on the SCLK pin. Workaround: a. Enable an SCI pin other than SCLK. b. In the next instruction, enable the remaining SCI pins, including the SCLK pin. This is not a bug, but a specification update.	2J22A

Chip Errata
DSP56303 Digital Signal Processor
Mask:2J22A

ED14	The data sheets of the various DSP56300 host interfaces (HI32 excluded) must be modified to make the HI08/HDI08 compatible with PortA timing 114, which is included here as a reference. Timing 321 "Write data strobe deassertion width" should be split (similar to timing 319 "Read data strobe deassertion width"), as described here: Write data strobe deassertion width: • after HCTR, HCVR and "Last Data Register" writes 2.5*Tc+10.0 @66MHz 2.5*Tc+8.3 @80MHz 2.5*Tc+6.6 @100MHz • after TXH:TXM writes (with HBE=0), TXM:TXL writes (with HBE=1)25 @66MHz 20.6 @80MHz 16.5 @100MHz That is, a minimum of 4 WS for PortA is required for 100 MHz operation. Reference: Timing 114 @ 100MHz 114 WR_ deassertion time 0.5 x TC - 3.5 [WS = 1] 1.5ns TC - 3.5 [2 <= WS <= 3] 6.5ns 2.5 x TC - 3.5 [4 <= WS <= 7] 21.5ns 3.5 x TC - 3.5 [WS >= 8] 31.5ns	2J22A
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.	2J22A

ED17	Description (added 9/28/98): In all DSP563xx technical datasheets, a note is to be added under "AC Electrical Characteristics" that although the minimum value for "Frequency of Extal" is 0MHz, the device AC test conditions are 15MHz and rated speed. Workaround: N/A	2J22A
ED20	Description (added 11/24/98): In the Technical Datasheet Voh-TTL should be listed at 2.4 Volts, not as: `TTL = Vcc-0.4` Workaround: This is a documentation update.	2J22A
ED21	Description (added 11/24/98): In the Technical Datasheet Iol should be listed as 1.6 mA, not as 3.0 mA. Workaround: This is a documentation update.	2J22A
ED24	Description (added 11/24/98): The technical datasheet supplies a maximum value for internal supply current in Normal, Wait, and Stop modes. These values will be removed because we will specify only a "Typical" current. Workaround: This is a documentation update.	2J22A

ED26	Description (added 1/6/99): The specification DMA Chapter is wrong. “Due to the DSP56300 Core pipeline, after DE bit in DCRx is set, the corresponding DTDx bit in DSTR will be cleared only after two instruction cycles.” Should be replaced with: “Due to the DSP56300 Core pipeline, after DE bit in DCRx is set, the corresponding DTDx bit in DSTR will be cleared only after three instruction cycles.”	2J22A

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.”
		2J22A

ED29	Description (added 9/12/1997; identified as a Documentation errata 2/1/99): When the ESSI transmits data with the CRA Word Length Control bits (WL[2:0]) = 100, the ESSI is designed to duplicate the last bit of the 24-bit transmission eight times to fill the 32-bit shifter. Instead, after shifting the 24-bit word correctly, eight 0s are being shifted. Workaround: None at this time. **Pertains to:** UM, Section 7.4.1.7, “CRA Word Length Control.” The table number is 7-2.	2J22A
ED30	Description (added 9/12/1997; identified as a Documentation errata 2/1/99): When the ESSI transmits data in the On-Demand mode (i.e., MOD = 1 in CRB and DC[4:0] = $00000 in CRA) with WL[2:0] = 100, the transmission does not work properly. Workaround: To ensure correct operation, do not use the On-Demand mode with the WL[2:0] = 100 32-bit Word-Length mode. **Pertains to:** UM, Section 7.5.4.1, “Normal/On-Demand Mode Selection.”	2J22A

ED31	Description (added 9/12/1997; modified 9/15/1997; identified as a Documentation errata 2/1/99): Programming the ESSI to use an internal frame sync (i.e., SCD2 = 1 in CRB) causes the SC2 and SC1 signals to be programmed as outputs. If however, the corresponding multiplexed pins are programmed by the Port Control Register (PCR) to be GPIOs, then the GPIO Port Direction Register (PRR) chooses their direction, but this causes the ESSI to use an external frame sync if GPIO is selected. **Note:** This errata and workaround apply to both ESSI0 and ESSI1. Workaround: To assure correct operation, either program the GPIO pins as outputs or configure the pins in the PCR as ESSI signals. **Note:** The default selection for these signals after reset is GPIO. **Pertains to:** UM, Section 7.4.2.4, “CRB Serial Control Direction 2 (SCD2) Bit 4”	2J22A

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.”
		2J22A

ED33

2J22A

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
             .....
             .....
```

ED33 cont.

2J22A

```
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
```

ED33 cont.

2J22A

```
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
```

ED33 cont.

2J22A

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
       .....
       .....
```

ED33 cont.

2J22A

```
             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.”

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.	2J22A

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).	2J22A

ED39

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.”

2J22A

ED40

2J22A

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."

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).	2J22A

Motorola and Ⓜ are registered trademarks of Motorola, Inc.
OnCE is a trademark of Motorola, Inc.

NOTES

1. An over-bar (i.e., $\overline{xxxx}$) indicates an active-low signal.
2. The letters seen to the right of the errata tell which DSP56303 mask numbers apply.
3. The Motorola DSP website has additional documentation updates that can be accessed at the following URL:

   http://www.mot.com/SPS/DSP/home/eng/tec/doc_update.html

-end-