

**FUNCTIONAL DIFFERENCES BETWEEN
DSP56301 REV. A (MASK F92R)
AND DSP56301 REV. B (MASK F48S) AND
SUBSEQUENT REVISIONS**

26 June 1997

MOTOROLA INC.
6501 William Cannon Dr. West
Austin, Texas 78735

CONTENTS

1.	PURPOSE OF THIS DOCUMENT	3
2.	GENERAL DESCRIPTION OF SPECIFICATION CHANGES. . . .	4
3.	MEMORY ALLOCATION CHANGES	5
4.	ENHANCED EXTERNAL ADDRESSING CAPABILITY	6
5.	NEW INSTRUCTION—VITERBI SHIFT LEFT	7
6.	PCI FUNCTIONALITY	7
6.1	PCI Activity While in Personal Software Reset State	7
6.2	New PCI Bus Command—Memory Write and Invalidate (1111) . . .	8
6.3	Addition of New PCI Status Bits—MDT and RDCQ	8
6.4	PCI Configuration Address Space	10
7.	BCLK SIGNAL	12
8.	INCREASED PULL-UP RESISTOR SIZE	13
9.	POWER-ON RESET (POR) CIRCUIT	14
10.	BOOTSTRAP PROGRAM CHANGES	14

APPENDICES

A	MEMORY MAPS	15
B	AA CONTROL REGISTERS (ONE FOR EACH AA PIN)	23
C	VITERBI SHIFT LEFT (VSL) INSTRUCTION	25
D	BOOTSTRAP PROGRAM	26
E	DSP56301 CHIP ERRATA FIXED IN REVISION B	37

1. PURPOSE OF THIS DOCUMENT

Early evaluation of the DSP56301 indicated the need for changes to enhance and expand chip functionality. The purpose of this document is to describe the specification changes incorporated in Revision B of this chip and to identify the functional differences between Revision A and Revision B.

Note: In addition to the specification changes, Revision B also fixed some of the errata identified in earlier mask sets. **Appendix E** provides a detailed list of the chip errata fixed in Revision B.

2. GENERAL DESCRIPTION OF SPECIFICATION CHANGES

The DSP56301 was modified in the following ways:

- **Memory Allocation**—A Memory Switch mode was added to provide more flexibility in allocation of memory space by allowing a switch of 2 K of RAM between Program RAM and the X data and Y data RAM spaces. As before, a 1 K program cache space can be allocated within the Program RAM space.
- **Enhanced External Addressing Capability**—An Address Attribute Disable option has been added. This provides a glueless support for the Long Move (MOVE L:) instruction and allows using multiple AA signals as additional address lines.
- **New Instruction**—The Viterbi Shift Left (VSL) instruction has been added to the instruction set to enhance Viterbi algorithm performance.
- **PCI Functionality**—PCI functionality has been enhanced by improving some functions and by adding certain optional functions described in the PCI Specification, including the following:
 - In the Personal Software Reset state, responding to configuration space transactions with a retry event to enable the user to modify the sub-vendor information before the PCI BIOS reads the register
 - Implementing the Memory Write and Invalidate (C3–C0 = 1111) PCI Bus Command
 - Defining Bits 14 (MDT) and 15 (RDCQ) of the DSP PCI Status Register (DPSR)
 - Adding the Cache Line Size Configuration (CCLS) register and the Subsystem ID and Subsystem Vendor ID Configuration (CCLS) within the PCI configuration address space
- **Added Signal**—The new design adds a $\overline{\text{BCLK}}$ signal in order to provide greater system design flexibility. A customer can use the clock rising edge or falling edge by selecting the BCLK or $\overline{\text{BCLK}}$, or both, without additional hardware logic.
- **Increased Pull-up Resistor Size**—The internal pull-resistor has been increased in size by a factor of ten for each of the JTAG/OnCE Port input signals $\overline{\text{DE}}$, $\overline{\text{TRST}}$, TDI, and TMS.
- **Power-On Reset (POR)**—This circuit has been added to the internal PLL circuitry to protect the PLL.
- **Updated Bootstrap Program**—A new bootstrap mode is added to load the Program RAM with the Host Interface (HI32) programmed to operate in the 16-bit Universal Bus Mode supporting a glueless DSP56301–DSP563xx interface.

3. MEMORY ALLOCATION CHANGES

The DSP56301 contains 8 K of RAM, originally divided by default into:

- Program RAM (4 K)
- X data RAM (2 K)
- Y data RAM (2 K)

The enhanced functionality adds a Memory Switch (MS) bit in the Operating Mode Register (OMR) that works in conjunction with the previously defined Cache Enable (CE) bit in the Status Register (SR) to reallocate the 8 K of on-chip RAM. **Table 1** describes these bits and their function. **Table 2** provides a summary of the possible memory configurations using the two bits in conjunction.

Table 1 RAM Configuration Bit Descriptions

Bit Name	Bit Location	Function Description
Cache Enable (CE)	SR 19	This bit programs whether the instruction cache is enabled (CE = 1) or disabled (CE = 0).
Memory Switch (MS)	OMR 7	This bit programs whether 2 K of RAM is switched from Program RAM to data RAM (1 K in X data RAM and 1 K in Y data RAM) (MS = 1) or not (MS = 0).

Table 2 Summary of Available RAM Configurations

Program RAM Size	Instruction Cache Size	X Data RAM Size	Y Data RAM Size	Instruction Cache	Switch Mode
4096 × 24-bit	0	2048 × 24-bit	2048 × 24-bit	disabled (CE = 0)	disabled (MS = 0)
3072 × 24-bit	1024 × 24-bit	2048 × 24-bit	2048 × 24-bit	enabled (CE = 1)	disabled (MS = 0)
2048 × 24-bit	0	3072 × 24-bit	3072 × 24-bit	disabled (CE = 0)	enabled (MS = 1)
1024 × 24-bit	1024 × 24-bit	3072 × 24-bit	3072 × 24-bit	enabled (CE = 1)	enabled (MS = 1)

The actual memory locations for Program RAM and the Instruction Cache in the Program memory space are determined by the MS and CE bits, and their addresses are given in **Table 3**.

Table 3 Memory Addressing for Program RAM and Instruction Cache

MS	CE	Program RAM Address	Cache
0	0	\$000–\$FFF	does not exist
0	1	\$000–\$BFF	exists
1	0	\$000–\$7FF	does not exist
1	1	\$000–\$3FF	exists

The actual memory locations for both X and Y data RAM in their own memory space are determined by the MS bit, and their addresses are given **Table 4**.

Table 4 Memory Locations for Data RAM

MS	Data RAM Location
0	\$000–\$7FF
1	\$000–\$BFF

Detailed memory maps are provide in **Appendix A** on page 15 of this document.

4. ENHANCED EXTERNAL ADDRESSING CAPABILITY

The DSP56301 specification has been changed to add an Address Priority Disable (APD) bit as Bit 14 of the Operating Mode Register (OMR). In the original specification, the priority of the four Address Attribute signals was fixed: AA3 had the highest priority and AA0 had the lowest priority. This priority mechanism did not allow more than one AA to be active at the same time. This prioritization meant the user had to include additional interface hardware to implement the Long Move (MOVE L:) instruction or to use a portion of these signals as additional address lines. The addition of the APD bit allows the user to disable the default priority and perform these functions with no additional hardware. **Appendix B** on page 23 of this document provides a detailed explanation of the new functionality with corrections to **Section 2.5.1** of the *DSP56300 Family Manual* indicated in *italics*.

5. NEW INSTRUCTION—VITERBI SHIFT LEFT

Because of the increasing use of convolutional encoders/decoders (also known as trellis codes or the Viterbi algorithm) in digital communication applications, the core specification has been modified to add a Viterbi Shift Left (VSL) instruction to enhance performance when processing code containing the Viterbi algorithm. Basically, this instruction operates on a 48-bit operand, storing the most significant 24 bits in an X data memory location, while performing a left shift on the least significant 24 bits, inserting a zero or one (determined by operand i) as the new least significant bit, and storing the result in a Y data memory location. **Appendix C** on page 25 of this document provides a detailed description of the new instruction using the format of **Appendix A** of the *DSP56300 Family Manual* and as shown in *Using the DSP56300/600 VSL Instruction for Viterbi Decoding (APR32/D)*.

6. PCI FUNCTIONALITY

The PCI specification includes items which are required for compliance and optional items that enhance performance. The DSP56301 has been modified to provide improved performance of some functions along with the addition of some option components.

6.1 PCI Activity While in Personal Software Reset State

The specification of the DSP56301 was changed so that in the Personal Software Reset state, the HI32 responds to configuration space transactions with a retry event. This change is an addition to the response to memory space transactions during the Personal Software Reset state. Currently, Section 6.1.1.13 of the DSP56301 User's Manual states:

“In the personal software reset state all data paths are cleared, and the HI32 will respond to all memory space transactions with a retry event.”

For Revision B, this section should be changed to read:

“In the Personal Software Reset state all data paths are cleared, and the HI32 will respond to all memory *and configuration* space transactions with a retry event. “

6.2 New PCI Bus Command—Memory Write and Invalidate (1111)

When the HI32 is configured as a PCI Bus Master, there is a set of PCI bus commands that can be invoked by programming the values of the Command (C3–C0) Bits 11–8 in the DPAR Register. The DSP56301 specification has been changed in Rev. B to define a Memory Write and Invalidate command as C3–C0 = 1111. **Table 5** defines the new list of PCI Master Bus Commands.

Table 5 PCI Bus Commands Supported by the HI32 as PCI Master

C3–C0	Command Type
0000	illegal
0001	illegal
0010	I/O read
0011	I/O Write
0100	illegal
0101	illegal
0110	Memory Read
0111	Memory Write
1000	illegal
1001	illegal
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	illegal
1110	Memory Read Line
1111	Memory Write and Invalidate
Note: When using the Memory Write and Invalidate command, a minimum transfer of one complete cacheline should be guaranteed. This should be reflected by the Burst Length value used (BL[5:0] in the DMPC). The cacheline size is set by the PCI configurator in the Cache Line Size Configuration Register (CCLS) but the DSP56300 Core cannot access this value. The system should provide the CCLS value to the DSP56300 Core in some another (user defined) way.	

6.3 Addition of New PCI Status Bits—MDT and RDCQ

To provide better monitoring of PCI Master data transfers, two bits have been implemented in the DSP PCI Status Register (DPSR):

- Master Data Transferred (MDT)—Bit 14
- Remaining Data Count Qualifier (RDCQ)—Bit 15

6.3.1 PCI Master Data Transferred (MDT) Bit 14

The MDT indicates the status of the latest completed PCI transaction, where the HI32 was involved as a PCI master. MDT is set at the end of a transaction (MARQ = 1) if all the master data (as defined by BL[5:0] in the DPMC) was transferred successfully by the HI32. Otherwise, the MDT bit is cleared.

If MARQ is set it is sufficient to check MDT to know whether the HI32 succeeded in transferring all master data to the designated target. If MARQ is set and MDT is cleared the user can discover why the transaction was terminated before all the data was transferred by checking the TO, TRTY, TDIS, TAB and MAB status bits in the DPSR.

Note: If the Master Access Counter is disabled (MACE = 0 in DPCR), the MDT bit is not valid. Hardware and software resets clear MDT.

6.3.2 Remaining Data Count Qualifier (RDCQ) Bit 15

The RDCQ qualifies the RDC[5:0] value in DPSR. If the MDT bit is cleared (i.e. not all the master data was transferred) at the end of a transaction (MARQ = 1) initiated by the HI32, the burst length for the next transaction to the same target required to complete the data transfer should be calculated using the following formula:

$$BL[5:0] = RDC[5:0] + RDCQ.$$

Note: If either TAB, TRTY or MAB status bit is set in the DPSR, the transaction may be initiated again with the same address and burst length by writing the DPAR with its previous value.

Note: If the Master Access Counter is disabled (MACE is cleared in DPCR), the RDC[5:0] and RDCQ bits are not valid.

6.4 PCI Configuration Address Space

The following two configuration space registers are added to the HI32:

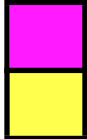
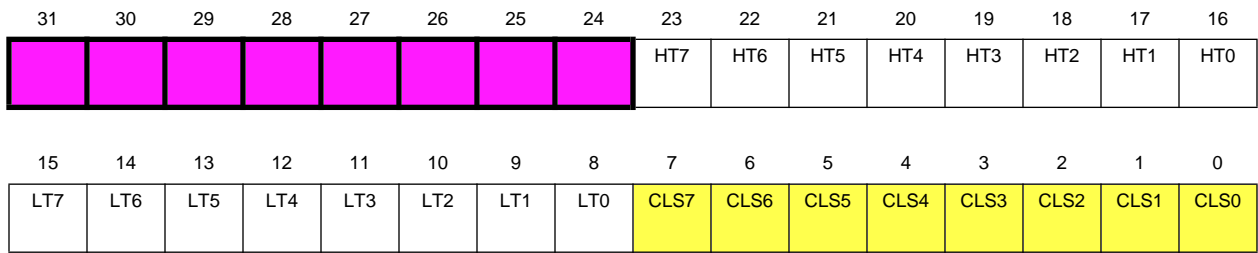
- Cache Line Size Configuration Register (CCLS)
- Subsystem ID and Subsystem Vendor ID Configuration Register (CSID).

Figure 1 shows the location of the new registers within the PCI configuration address space. A detailed description of each register is provided in the following subsections.

\$00	CDID/CVID	Device ID (CDID)		Vendor ID (CVID)	
\$04	CSTR/CCMR	Status (CSTR)		Command (CCMR)	
\$08	CCCR/CRID	Class Code (CCCR)			Revision ID (CRID)
\$0C	CHTY/CLAT/ CCLS		Header Type (CHTY)	Latency Timer (CLAT)	Cache Line (CCLS)
\$10	CBMA	Memory Space Base Address (CBMA)			
\$14		Reserved (6 Dwords)			
\$28					
\$2C	CSID	Subsystem ID and Subsystem Vendor ID (CSID)			
\$30		Reserved (3 Dwords)			
\$38					
\$3C	CILP	MAX_LAT	MIN_GNT	Interrupt Line	Interrupt Pin
\$40		Reserved (48 Dwords)			
\$FC					
Note: Addresses shown are in bytes.					

Figure 1 Host Side Registers (PCI Configuration Address Space)

6.4.1 Cache Line Size (CLS7–CLS0) Register (Bits 7–0)



Not implemented, read as zero and should be written zero

Location of new Cache Line Size Register

Bit		Name	Function
CCLS	7–0	CLS7–CLS0	Cache Line Size
CLAT	15–8	LT7–LT0	Latency Timer (High)
CHTY	23–16	HT7–HT0	Header Type (hardwired to \$00)
CCLS	31–24	not implemented	

The read/write bits CLS7–CLS0 specify the system cache line size in units of 32-bit words. The personal hardware reset clears LT7–LT0.

Note: When using some PCI commands (e.g., the Memory Write and Invalidate command), a minimum transfer of one complete cacheline should be guaranteed. This should be reflected by the Burst Length value used (BL[5:0] in the DMPC). The cacheline size is set by the PCI configurator in the Cache Line Size Configuration Register (CCLS) but the DSP56300 Core cannot read this value. The system should provide the CCLS value to the DSP56300 Core in some another (user defined) way.

6.4.2 Subsystem ID and Subsystem Vendor ID Configuration Register (CSID)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID15	SID14	SID13	SID12	SID11	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	SID2	SID1	SID0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVID15	SVID14	SVID13	SVID12	SVID11	SVID10	SVID9	SVID8	SVID7	SVID6	SVID5	SVID4	SVID3	SVID2	SVID1	SVID0

Bit	Name	Function
15–8	SVID15–SVID0	Subsystem Vendor ID
31–16	SID15–SID0	Subsystem ID

The CSID is a PCI standard 32-bit read/write register mapped into the PCI configuration space, when in the PCI mode (HM=\$1). The CSID register is read if a configuration read command is in progress and the PCI address is \$2C. In the Self Configuration mode (HM = \$5): the DSP56300 Core can indirectly write the CSID (See “SELF CONFIGURATION MODE” on page 72 of the DSP56301 User’s Manual).

The CSID register cannot be accessed by the host when not in the PCI mode (HM≠\$1). This register is used to uniquely identify the add-in board or subsystem where the DSP56301 resides. It provides a mechanism for add-in card vendors to distinguish their cards from one another even though the cards may have the same DSP56301 on them (and, therefore, the same Vendor ID and Device ID). Implementation of this register is optional and an all zero value indicates that the device (e.g. add-in board) does not support subsystem identification. The CSID bits are cleared after power-up. Any reset does not affect the value written to the CSID.

The following procedure should be used to write the CSID:

1. Power-up the DSP56301.
The default CSID value is \$00000000. The HI32 is in the Personal Software Reset state (HM = \$0) and responds to memory and configuration space PCI transactions with a retry event.
2. Boot the DSP56301 through EPROM or SCI.
 - a. The program downloaded to the DSP56301 should:
 - Enter the Self Configuration mode (HM = \$5) and write CSID.
The HI32 will still respond to memory and configuration space PCI transactions with a retry event.

- Optional: set PCTL value.
This enables to run the DS56301 from low frequency external clock.
- Enter the Personal Software Reset state (HM = \$0).
- Enter PCI mode (HM = \$1).
Now the DSP56301's PCI configuration space may be accessed by a PCI master.
- Optional: Set the mode bits in the OMR to MC:MB:MA = 100 and jump to the DSP56301 bootstrap ROM start address \$FF0000 for further program download from the HI32 in the PCI mode.

Example 6-1 Code for CSID setting

```

move    #0,x0                ; set constant
movep   #>$500000,x:M_DCTR    ; Set Self Configuration Mode
rep     #4
movep   x0,x:M_DPAR           ; set register pointer to SIDR/SVID
movep   #>$012345,x:M_DPMC     ; set SIDR value to $2345
movep   #>$6789ab,x:M_DPAR     ; set SVID value to $89ab and write SIDR/SVID
movep   x0,x:M_DCTR           ; personal software reset
movep   #>$100000,x:M_DCTR     ; set PCI mode

```

7. $\overline{\text{BCLK}}$ SIGNAL

The DSP56301 specification has been changed to add a new signal Bus Clock Not ($\overline{\text{BCLK}}$). When the DSP is the bus master, $\overline{\text{BCLK}}$ is an active-low output and is the inverse of the BCLK signal. Otherwise, the signal is tri-stated. The signal has been added as pin 28 (formerly a No Connect) on the 208-pin TQFP package. The $\overline{\text{BCLK}}$ signal is pin R9 on the 252-pin PBGA package.

8. INCREASED PULL-UP RESISTOR SIZE

The internal pull-resistor has been increased in size by a factor of ten for each of the JTAG/OnCE Port input signals $\overline{\text{DE}}$, $\overline{\text{TRST}}$, TDI, and TMS. Because some system designs gang inputs for many DSP devices, the effective value of the pull-up could become very low. Therefore, the internal pull-up resistance was increased to a nominal value of 900 Ω in the Rev. B design of the DSP56301 to compensate for use in such designs.

9. POWER-ON RESET (POR) CIRCUIT

The Power-On Reset (POR) circuit has been added to the internal PLL circuitry to protect the PLL in the event that power is applied to the part before a clocking signal is applied. This circuit is disabled whenever a normal clock is applied to the part. Because there may be conditions under which the input clock fails or is not allowed to be applied to the PLL input, the circuit was redesigned to prevent the part from drawing excessive current if the condition occurs. The data sheet will continue to specify that the PLL must have a normal clock applied before applying power to the part.

10. BOOTSTRAP PROGRAM CHANGES

The new bootstrap Mode 3 is added (MD:MC:MB:MA = x011) to load the program RAM from the Host Interface programmed to operate in the Universal Bus mode supporting DSP56301-to-DSP563xx glueless connection. **Table 6** summarizes the Operating Modes. The new Bootstrap Program is listed in **Appendix D** on page 26.

Table 6 DSP56301 Operating Modes

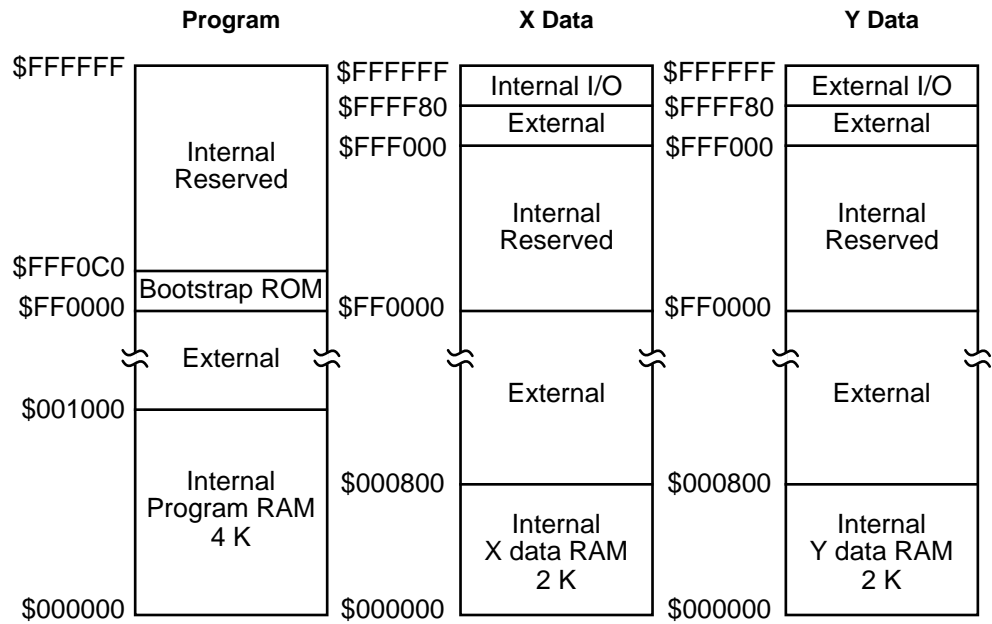
Mode	MOD D	MOD C	MOD B	MOD A	Reset Vector	Description
0	0	0	0	0	\$C00000	Expanded mode
1	0 or 1	0	0	1	\$FF0000	Bootstrap from byte-wide memory
2	0 or 1	0	1	0	\$FF0000	Bootstrap thru SCI
3	0 or 1	0	1	1	\$FF0000	Host Bootstrap 24-bit-wide UB Mode (DSP56301-to-DSP563xx)
4	0 or 1	1	0	0	\$FF0000	Host Bootstrap PCI Mode (32-bit-wide)
5	0 or 1	1	0	1	\$FF0000	Host Bootstrap 16-bit-wide UB Mode (ISA)
6	0 or 1	1	1	0	\$FF0000	Host Bootstrap 8-bit-wide UB Mode in double-strobe pin configuration
7	0 or 1	1	1	1	\$FF0000	Host Bootstrap 8-bit-wide UB Mode in single-strobe pin configuration
8	1	0	0	0	\$008000	Expanded mode

APPENDIX A

MEMORY MAPS

The following figures describe each of the memory space and RAM configurations defined by the settings of the SC, CE, and MS bits. The figures show the configuration, the table describes the bit settings, memory sizes, and memory locations.

Default

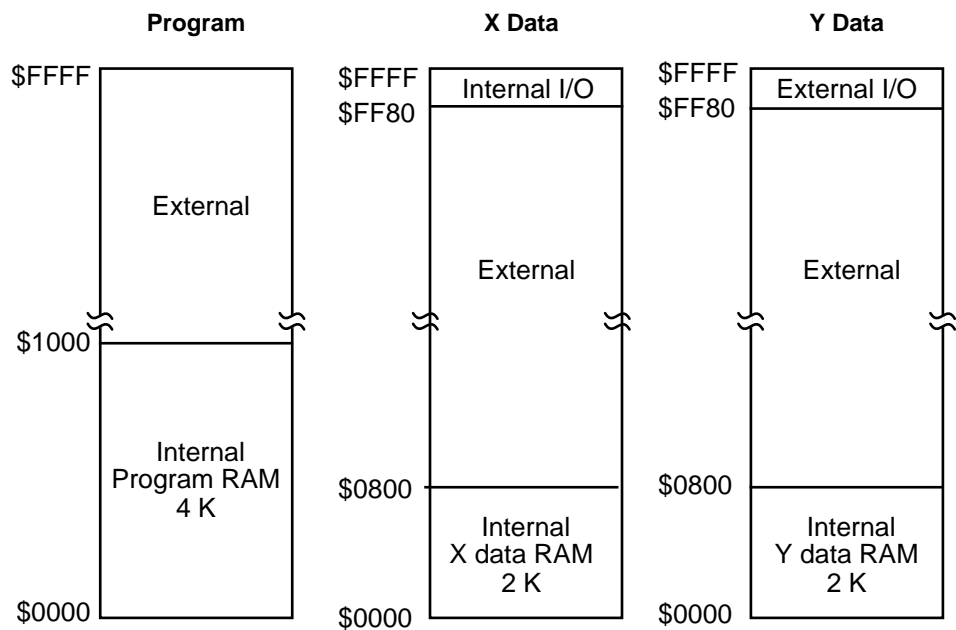


Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	0	4 K \$000- \$FFF	2 K \$000 - \$7FF	2 K \$000 - \$7FF	None	16 M

AA0557

Figure 2 Default Settings (0, 0, 0)

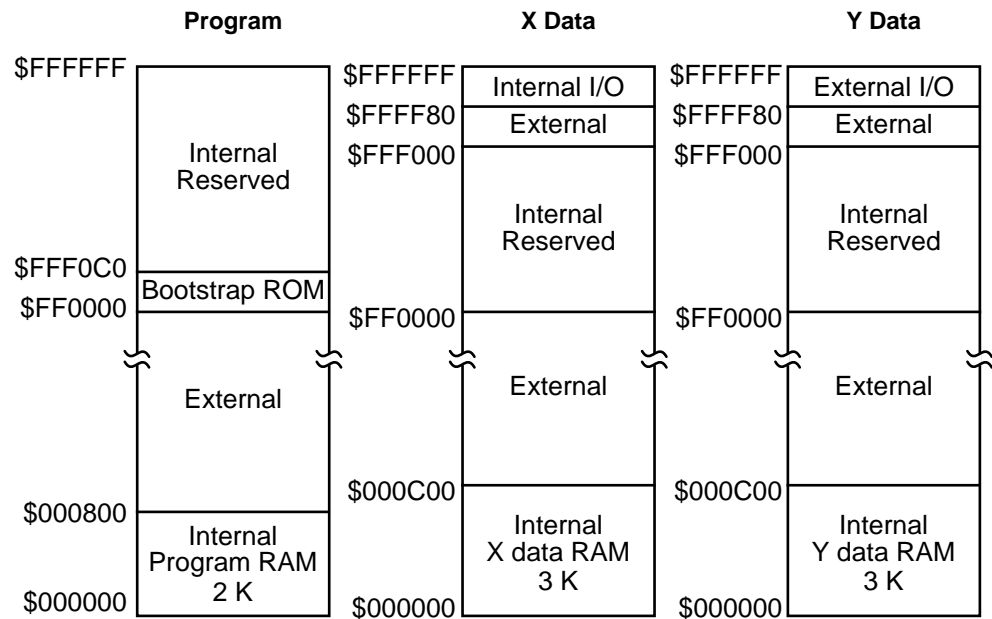
Memory Maps



Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	4 K \$000- \$FFF	2 K \$000 - \$7FF	2 K \$000 - \$7FF	None	64 K

AA0558

Figure 3 16-bit Space with Default RAM (0, 0, 1)

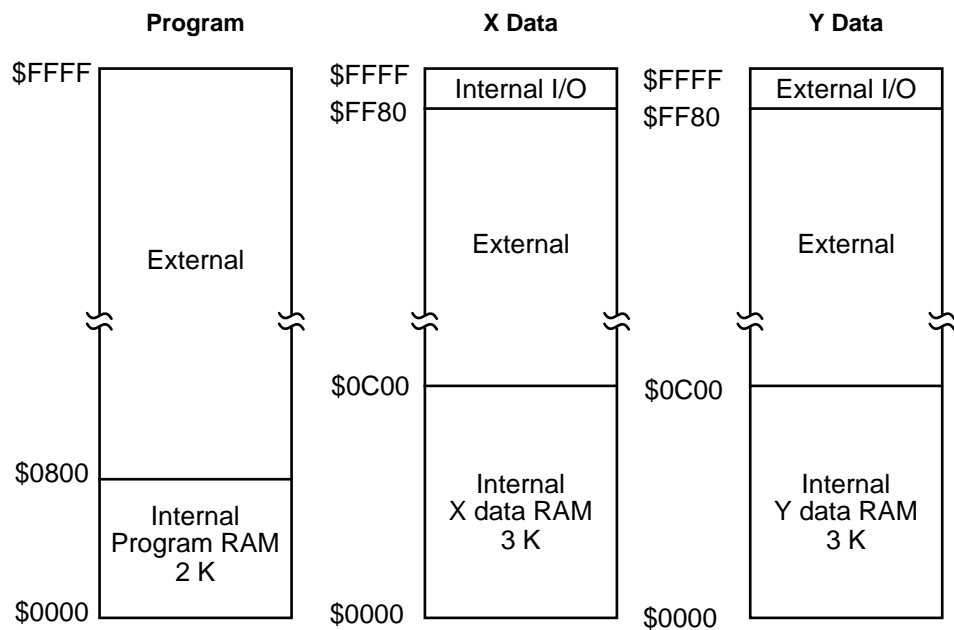


Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	2 K \$000- \$800	3 K \$000 - \$BFF	3 K \$000 - \$BFF	None	16 M

AA0559

Figure 4 Switched Program RAM (0, 1, 0)

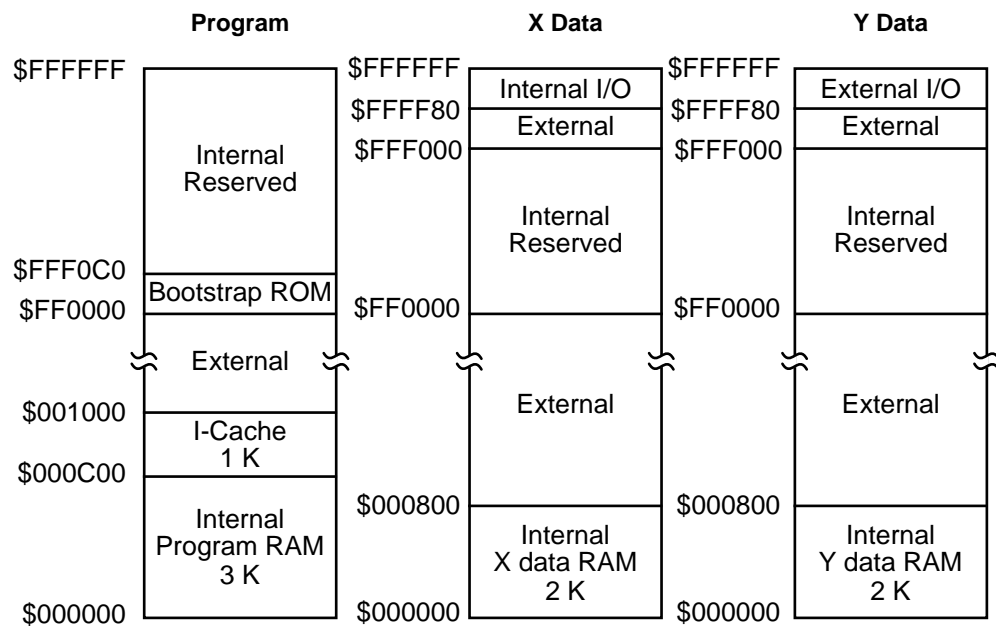
Memory Maps



Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	1	2 K \$000- \$7FF	3 K \$000 - \$BFF	3 K \$000 - \$BFF	None	64 K

AA0560

Figure 5 16-bit Space with Switched Program RAM (0, 1, 1)

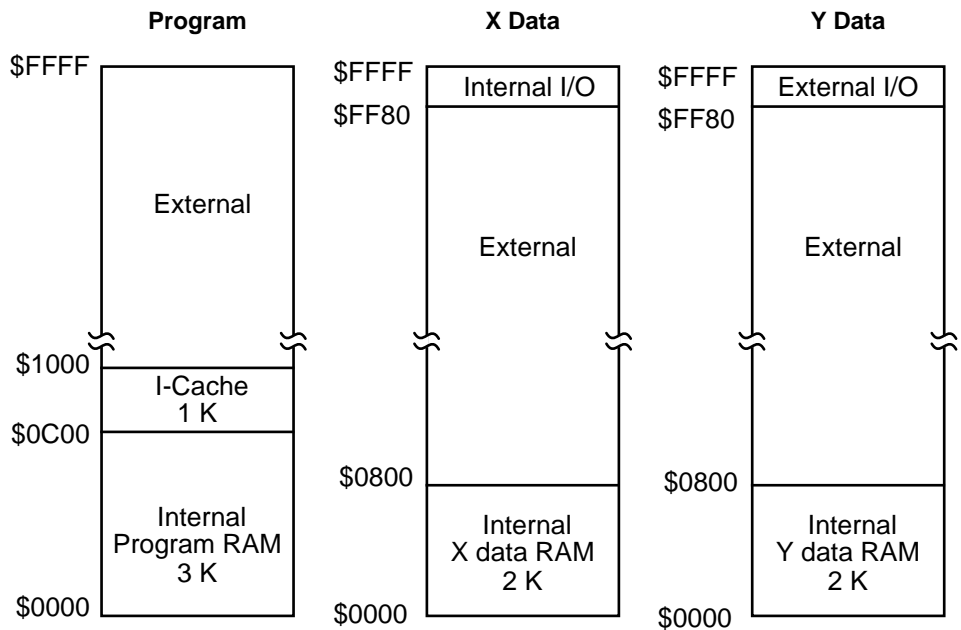


Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	3K \$000- \$BFF	2 K \$000 - \$7FF	2 K \$000 - \$7FF	1 K \$C00 - \$FFF	16 M

AA0561

Figure 6 Instruction Cache Enabled (1, 0, 0)

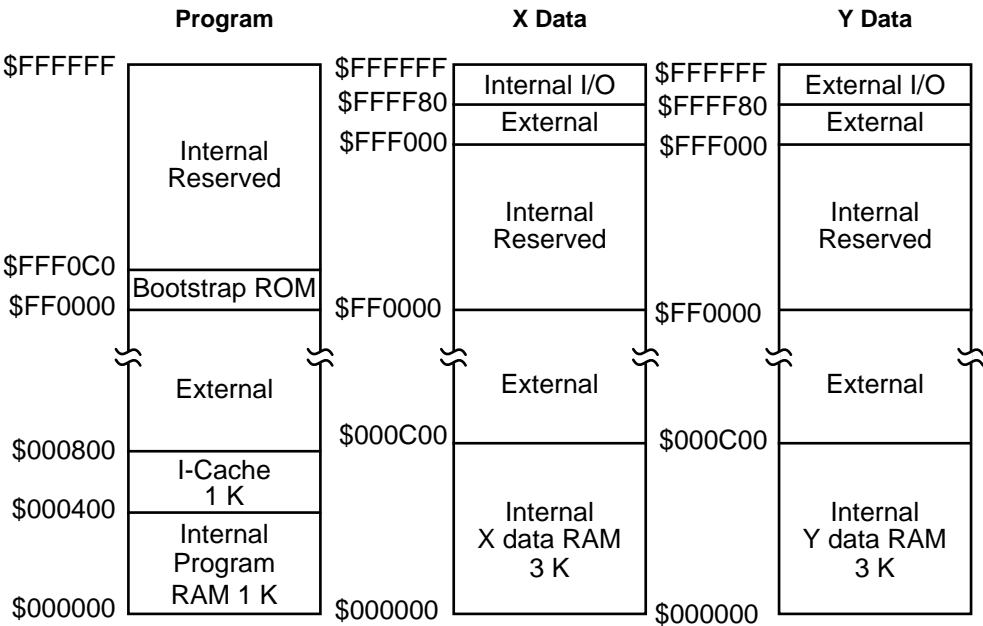
Memory Maps



Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	1	3 K \$000- \$BFF	2 K \$000 - \$7FF	2 K \$000 - \$7FF	1 K \$C00 - \$FFF	64 K

AA0562

Figure 7 16-bit Space with Instruction Cache Enabled (1, 0, 1)

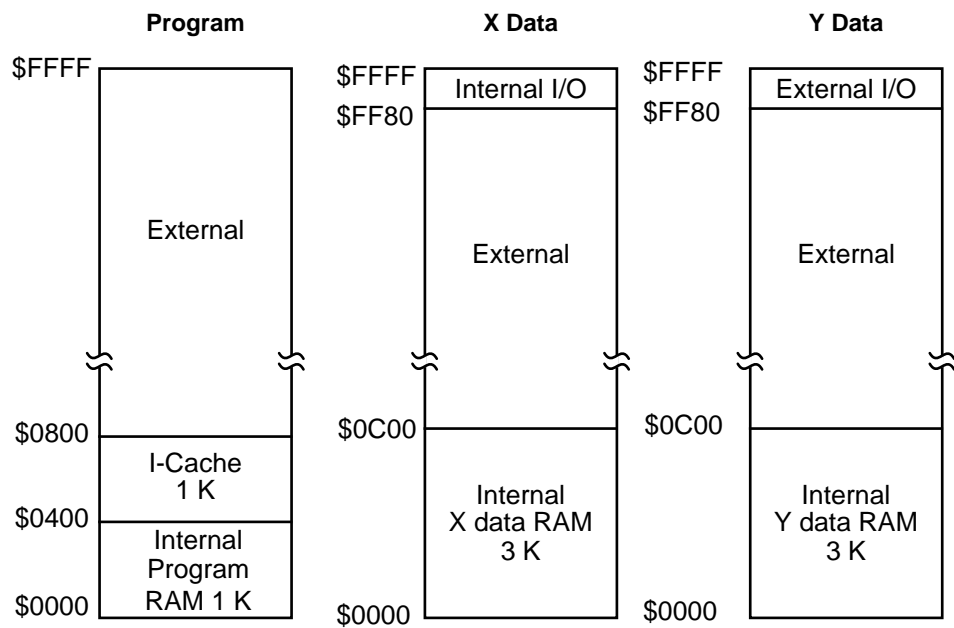


Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	1 K \$000- \$3FF	3K \$000 - \$BFF	3 K \$000 - \$BFF	1 K \$400 - \$7FF	16 M

AA0563

Figure 8 Switched Program RAM and Instruction Cache Enabled (1, 1, 0)

Memory Maps



Bit Settings			Memory Configuration				
CE	MS	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	1	1 K \$000- \$3FF	3 K \$000 - \$BFF	3 K \$000 - \$BFF	1 K \$400 - 7FF	64 K

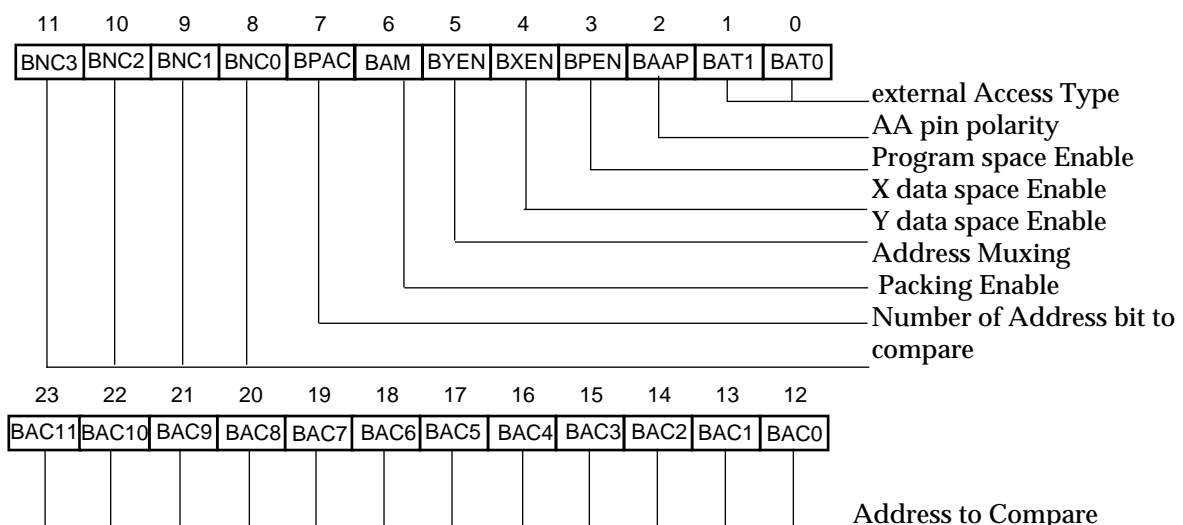
AA0564

Figure 9 16-bit Space, Switched Program RAM, Instruction Cache Enabled (1, 1, 1)

APPENDIX B

AA CONTROL REGISTERS (ONE FOR EACH AA PIN)

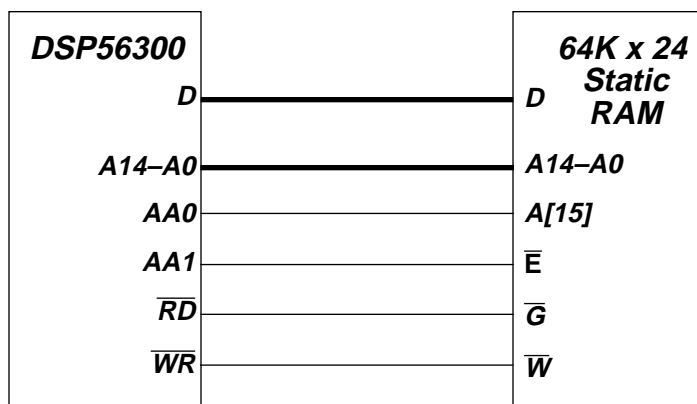
The **four** control registers (AAR3, AAR2, AAR1, AAR0) are 24 bit read write registers used to control the activity of the AA3-AA0/RAS3-RAS0 pins. An AA/RAS pin is asserted if the address in his appropriate AAR register (BAC bits) matches the external address (the exact number of address bits that are compared is determined by BNC bits) and if the external access is aimed to a space (X Y or P) that is enabled in the appropriate AAR register. All AAR registers are disabled (all the AAR bits are cleared) during hardware reset. The AAR bits are shown in the following figure and described in the following paragraphs.



- Note:
1. A priority mechanism exists among the four AAR control registers in order to resolve selection conflicts. AAR3 has the highest priority and AAR0 has the lowest priority, (e.g. if the external address matches the address and the space that is specified in both AAR1 and AAR2, the external access type will be selected according to the AAR2 register). The priority mechanism allows continuous partition of the external address space.
 2. When a selection conflict occurs, i.e. the external address matches the address and the space that is specified in more than one AAR, the assertion of the lower priority AA/RAS pin(s) is programmable. When the APD bit in OMR is cleared (see Chapter 6), only one AA/RAS pin of higher priority will be asserted. When the APD bit is set, the lower priority AA/RAS pin(s) will be asserted in addition to the higher priority AA/RAS pin. AAR of higher priority defines the external memory access type (memory type, wait states etc.) The lower priority AA/RAS pin(s) associated with DRAM memory type (BAT(1:0) = 10) will not be activated. This allows a glueless support of Long Move (move L:) instruction to/from external memory as shown in Figure 11.

Figure 10 Address Attribute Registers (AAR3-AAR0)

AA Control Registers (One for Each AA Pin)



- Note:
- In this example:
 - X space is mapped to SRAM addresses \$8000-\$FFFF.
 - Y space is mapped to SRAM addresses \$0000-\$7FFF.
 - AAR1 is programmed as SRAM in location X:Y:\$000000-\$007FFF, active low.
 - AAR0 is programmed as SRAM in location Y:\$000000-\$007FFF, active low.
 - The instruction "move X,L:\$1000" will generate the following two external accesses:
 - X space access to location \$9000. AA0 is high (not asserted).
 - Y space access to location \$1000. AA0 is low (asserted).
 - In both accesses AA1 is asserted and activates the SRAM.
 - In both accesses AAR1 defines the access characteristics, i.e. memory type, wait states etc.
 - When the AA/ \overline{RAS} pin functions as AA pin, it is negated at the start of the next clock cycle only if there is no external access that use the same AA pin (i.e. the AA pin will be kept asserted in a sequence of two consecutive external accesses that access the same memory bank). This method enables the use of low power standby mode in the external memories (these memories should be accessed first by a dummy access).
 - The programmer should guarantee an AAR register is not changed while accessing the memory selected by this AAR, otherwise improper operation may result.
 - A write operation to any AAR register will cause the DRAM controller to invalidate the page logic and will force the next DRAM access to be an out of page access.

Figure 11 Long Move (move L:) instruction support

APPENDIX C

VITERBI SHIFT LEFT (VSL) INSTRUCTION

VSL

VSL

Viterbi Shift Left

Operation:

$S[47:24] \rightarrow X:ea; \{S[22:0], i\} \rightarrow Y:ea$

Assembler Syntax:

VSL S,i,L:ea

Description: Store the most significant part (24 bits) of the source accumulator at X memory (at effective address location) while for the least significant part (24 bits) of the source accumulator shift one bit to the left and insert 0 or 1 at the least significant bit, according to operand i, and store the result at Y memory at the same address. This instruction enhance Viterbi algorithm performance.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
x	x	x	x	x	x	x	x
CCR							

x This bit is unchanged by the instruction

Instruction Formats and opcodes:

	23	16 15								8 7								0						
VSL S,i,L:ea	0	0	0	0	1	0	1	S	1	1	M	M	M	R	R	R	1	1	0	i	0	0	0	0
	OPTIONAL EFFECTIVE ADDRESS EXTENSION																							

Instruction Fields:

{S} **S** Source register A,B (see **Table A-10** on page A-239)
{i} **i** Bit value, 0 or 1 to be placed in the least significant bit of Y:<ea>
{ea} **MMMR** Effective address (see **Table A-16** on page A-241)

Note: All references in this appendix are to the *DSP56300 Family Manual*.

APPENDIX D

BOOTSTRAP PROGRAM

```
; BOOTSTRAP CODE FOR DSP56301 - (C) Copyright 1996 Motorola Inc.
; Revised June 18, 1996.
;
; Bootstrap through the Host Interface, External EPROM or SCI.
;
; This is the Bootstrap program contained in the DSP56301 192-word Boot
; ROM. This program can load any program RAM segment from an external
; EPROM, from the Host Interface or from the SCI serial interface.
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x000, then the Boot ROM is bypassed and the DSP56301
; will start fetching instructions beginning with address $C00000 (MD=0)
; or $008000 (MD=1) assuming that an external memory of SRAM type is
; used. The accesses will be performed using 31 wait states with no address
; attributes selected (default area).
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x001, then it loads a program RAM segment from
; consecutive byte-wide P memory locations, starting at P:$D00000 (bits
; 7-0). The memory is selected by the Address Attribute AAL and is
; accessed with 31 wait states.
;
; The EPROM bootstrap code expects first to read 3 bytes specifying the
; number of program words, afterwards 3 bytes specifying the address to
; start loading the program words and then 3 bytes for each program word
; to be loaded. The number of words, the starting address and the program
; words are read least significant byte first followed by the mid and
; then by the most significant byte.
;
; The program words will be condensed into 24-bit words and stored in
; contiguous PRAM memory locations starting at the specified starting
; address. After reading the program words, program execution starts
; from the same address where loading started.
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x010, then it loads the program RAM from the SCI interface.
;
; The SCI bootstrap code expects first to receive 3 bytes specifying the
; number of program words, afterwards 3 bytes specifying the address to
; start loading the program words and then 3 bytes for each program word
; to be loaded. The number of words, the starting address and the program
; words are received least significant byte first followed by the mid and
; then by the most significant byte.
```

```

;
; The program words will be condensed into 24-bit words and stored in
; contiguous PRAM memory locations starting at the specified starting
; address. After reading the program words, program execution starts
; from the same address where loading started.
;
; The SCI is programmed to work in asynchronous mode with 8 data bits, 1
; stop bit and no parity. The clock source is external and the clock
; frequency must be 16x the baud rate. After each byte is received, it
; is echoed back through the SCI transmitter.
;
;
;
; If MD:MC:MB:MA=x011, then it loads the program RAM from the Host Interface
; programmed to operate in the Universal Bus mode supporting DSP56301-to-DSP56301
; glueless connection.
;
; The HI32 bootstrap code expects first to read a 24-bit word specifying
; the number of program words, afterwards a 24-bit word specifying the
; address to start loading the program words and then 24-bit word for
; each program word to be loaded.
;
; The program words will be stored in contiguous PRAM memory
; locations starting at the specified starting address. After
; reading the program words, program execution starts from the same
; address where loading started.
;
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0) in HCTR register. This will start execution of the
; loaded program from the specified starting address.
;
; During the access, the HAEN and HA10-HA3 pins must be driven low; pins
; HA2-HA0 select the HI32 registers.
; Before booting through the Host Interface it is recommended that the
; Host boot program will verify that the HI32 is operational, by reading
; the status register (HSTR) and confirm that its value is $3.
;
; Suggested DSP56301-to-DSP56301 connection:
;
; slave                master
; DSP56301/HI32        DSP56301/PortA
;
; HA[10:3]  <-  A[10:3]      ; selects HI32 (base address 00000000)
; HA[2:0]   <-  A[2:0]      ; selects HTXR registers
; HD[24:0]  <-> D[24:0]      ; Data bus
; HBS_     <-  BS_          ; Bus Strobe (optional, see Note1)
; HAEN     <-  AAx          ; DMA cycle disable (AAx is active low)
; HTA      ->  TA_          ; Transfer Acknowledge (optional, see Note2)
; HIRQ_    ->  IRQx_        ; Interrupt Request (active low, open drain)
; HWR_     <-  WR_          ; Write strobe
; HRD_     <-  RD_          ; Read strobe
; HRST     <-  system reset ; Reset (active low)
;

```

Bootstrap Program

```
; Pins HP31, HP32 and HDAK_ must be tied to Vcc. Pins HP[22:20] may be
; used as GPIO pins. Pin HINTIA_ may be used as software driven interrupt
; request pin.
;
; Note1: If HBS_ to BS_ connection is used, the synchronous connection of
; the HI32 is used and therefore the DSP56301/master should access the
; DSP56301/slave as SRAM with 2 wait states. In addition the CLKOUT of
; DSP56301/master should be connected to EXTAL of DSP56301/slave, and both
; master and slave should enable the PLL while in the case of slave
; multiplication, division and pre-division factors should be one to
; guarantee synchronization between master and slave.
; In the case of asynchronous connection, HBS_ must be tied to Vcc.
;
; Note2: If HTA to HTA_ connection is not used, it is recommended that
; the HOST Processor's boot program will verify that the Host Interface
; is ready, by reading the status register (HSTR) and confirm that TRDY=1
; or HTRQ=1.
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x100, then it loads the program RAM from the Host
; Interface programmed to operate in the PCI target (slave) mode.
;
; The HI32 bootstrap code expects first to read a 24-bit word specifying
; the number of program words, afterwards a 24-bit word specifying the
; address to start loading the program words and then 24-bit word for
; each program word to be loaded.
;
; The program words will be stored in contiguous PRAM memory
; locations starting at the specified starting address. After
; reading the program words, program execution starts from the same
; address where loading started.
;
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0) in HCTR register. This will start execution of the
; loaded program from the specified starting address.
;
; The HOST Processor must first configure the Host Interface as PCI slave
; and then start writing data to the Host Interface. The HOST Processor
; must program the HCTR HTF1-HTF0 bits as 01, 10 or 11 and then
; correspondingly drive the 24-bit data mapped into 32-bit PCI bus word.
;
; Note that for the synchronization purposes, the DSP to PCI clock ratio
; should be more then 5/3.
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x101, then it loads the program RAM from the Host
; Interface programmed to operate in the Universal Bus mode supporting
; ISA (slave) glueless connection.
;
; Using self configuration mode, the base address in CBMA is initially
; written with $2f which corresponds to an ISA HTXR address of $2fe
```

```

; (Serial Port 2 Modem Status read only register).
;
; The HI32 bootstrap code expects to read 32 consecutive times the "magic
; number" $0037. Subsequently the bootstrap code expects to read a 16-bit word
; which is the designated ISA Port Address; this address is written into the
; CBMA. The HOST Processor must poll for the Host Interface to be re-configured.
; This must be done by reading the HSTR and verifying that the value $0013 is
; read. From this moment the HOST Processor may start writing data to the
; Host Interface.
;
; The HI32 bootstrap code expects first to read a 24-bit word (see
; Note below) specifying the number of program words, afterwards a
; 24-bit word specifying the address to start loading the program
; words and then 24-bit word for each program word to be loaded.
;
; The program words will be stored in contiguous PRAM memory
; locations starting at the specified starting address. After
; reading the program words, program execution starts from the same
; address where loading started.
;
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0) in HCTR register. This will start execution of the
; loaded program from the specified starting address.
;
; Note: This ISA connection implies 16 bit data width access only and
; that the number of 16-bit wide words that are transferred must be
; even.
;
; The 24-bit words has to be packed into 16-bit ISA words and then sent
; by the HOST Processor in the following sequence:
;
;   | M0 | L0 |
;   | L1 | H0 |
;   | H1 | M1 |
;
; The boot program will convert every three 16-bit wide host words to two
; 24-bit wide DSP56301 opcodes in the following format:
;
;   | H0 | M0 | L0 |
;   | H1 | M1 | L1 |
;
; The Host Processor must program the Host Interface to operate in the
; zero fill mode (HTF1-HTF0 = 01 in HCTR).
;
; Suggested DSP56301 to ISA connection:
;
; HA[10] <- SBHE_          ; selects HI32 (base address 10011111)
; HA[9] <- SA[0]           ; selects HI32 (base address 10011111)
; HA[8:3] <- SA[9:4]       ; selects HI32 (base address 10011111)
; HA[2:0] <- SA[3:1]       ; selects HIXR registers
; HD[15:0] - SD[15:0]      ; Data bus
; HD[23:16] - Not connected ; High Data Bus - Should be pulled up or down
; HDBEN_ -> OE_           ; Output enable of transceivers

```

Bootstrap Program

```

; HDBDR  -> DIR           ; Direction of transceivers
; HSAK_   -> IO16_        ; 16 bit data word
; HBS_    <- Vcc          ; Bus Strobe disabled
; HAEN    <- AEN          ; DMA cycle enable
; HTA     -> CHRDY        ; Channel ready
; HWR_    <- IOWC_        ; IO/DMA write strobe
; HRD_    <- IORC_        ; IO/DMA read strobe
; HRST    <- inverted RSTDRV ; invert ISA reset
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x110, then it loads the program RAM from the Host
; Interface programmed to operate in the Universal Bus (UB) mode, in
; double-strobe pin configuration.
;
; The HI32 bootstrap code expects first to receive 3 bytes specifying the
; number of program words, afterwards 3 bytes specifying the address to
; start loading the program words and then 3 bytes for each program word
; to be loaded. The number of words, the starting address and the program
; words are received least significant byte first followed by the mid and
; then by the most significant byte.
;
; The program words will be condensed into 24-bit words and stored in
; contiguous PRAM memory locations starting at the specified starting
; address. After reading the program words, program execution starts
; from the same address where loading started.
;
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0) in HCTR register. This will start execution of the
; loaded program from the specified starting address.
;
; The user must externally decode the port address with active low logic and
; connect the select line to HAEN; all the address lines shall be pulled down
; except for HA3, HA2 and HA1 that select the HOST Interface registers.
;
; When booting through the Host Interface it is recommended that the Host
; boot program will verify that the Host Interface is operational, by
; reading the status register (HSTR) and confirm that TRDY=1.
;
; When booting through the Host Interface, it is recommended that the
; HOST Processor's boot program will verify that the Host Interface is
; ready, by reading the status register (HSTR) and confirm that TRDY=1
; or HTRQ=1.
;
;
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; If MD:MC:MB:MA=x111, then it loads the program RAM from the Host
; Interface programmed to operate in the Universal Bus (UB) mode, in
; single-strobe pin configuration.
;
; Other than the single-strobe pin configuration, this mode is identical to
; the double-strobe pin configuration UB mode (MD:MC:MB:MA=x110).
;

```

```

;

BOOT    equ    $D00000    ; this is the location in P memory
                        ; on the external memory bus
                        ; where the external byte-wide
                        ; EPROM would be located
AARV    equ    $D00409    ; AAR1 selects the EPROM as CE~
                        ; mapped as P from $D00000 to
                        ; $DFFFFFF, active low

M_SSR    EQU    $FFFF93    ; SCI Status Register
M_STXL    EQU    $FFFF95    ; SCI Transmit Data Register (low)
M_SRXL    EQU    $FFFF98    ; SCI Receive Data Register (low)
M_SCCR    EQU    $FFFF9B    ; SCI Clock Control Register
M_SCR    EQU    $FFFF9C    ; SCI Control Register
M_PCRE    EQU    $FFFF9F    ; Port E Control register
M_DCTR    EQU    $FFFFC5    ; DSP Control Register (DCTR)
M_DPMC    EQU    $FFFFC7    ; DSP PCI Master Control Register (DPMC)
M_DPAR    EQU    $FFFFC8    ; DSP PCI Address Register (DPAR)
M_DSR    EQU    $FFFFC9    ; DSP Status Register (DSR)
M_DRXR    EQU    $FFFFCB    ; DSP Receive Data FIFO (DRXR)
M_AAR1    EQU    $FFFFFF8    ; Address Attribute Register 1

        ORG PL:$ff0000,PL:$ff0000    ; bootstrap code starts at $ff0000

START
        jclr #3,omr,CONT    ; If MD:MC:MB:MA=xxxx continue boot
CONT    clr a #$0a,X0    ; clear a and load X0 with constant $0a0000
        move #$3e,x1    ; X1=$3E0000 prepare for UB mode host programming
                        ; HM=$3 (UB)
                        ; HIRD=1 (HIRO_ pin - drive high enabled)
                        ; HIRH=1 (HIRO_ pin - handshake enabled)
                        ; HRSP=1 (HRST pin - active low)
                        ; HTAP=0 (HTA_ pin - active high)
                        ; HDSM=0 (Double-strobe pin mode enabled)
        jclr #2,omr,EPRSCILD    ; If MD:MC:MB:MA=x0xx,
                        ; go load from EPROM/SCI/DSP56301-DSP56301
        jclr #1,omr,IHOSTLD    ; If MD:MC:MB:MA=x10x, go load from PCI/ISA HOST
        jclr #0,omr,UB2HOSTLD    ; If MD:MC:MB:MA=x110, go load from
                        ; double-strobe UB Host
                        ; If MD:MC:MB:MA=x111, go load from
                        ; single-strobe UB Host

;=====
; This is the routine that loads from the Host Interface in UB (UNIVERSAL) mode,
; with single-strobe pin configuration (RD/WR,DS).
; MD:MC:MB:MA=x111 - Host UB

UB1HOSTLD
        bset #13,x1    ; HDSM=1 (Double-strobe pin mode disabled)

;=====

```

Bootstrap Program

```

; This is the routine that loads from the Host Interface in UB (UNIVERSAL) mode,
; with double-strobe pin configuration (RD,WR).
; MD:MC:MB:MA=x110 - Host UB

UB2HOSTLD
    movep x1,X:M_DCTR          ; Configure HI32 in UB mode Single or Double strobe
    do #6,_LOOP0              ; read # of words and start address
    jclr #2,X:M_DSR,*          ; Wait for SRRQ to go high (i.e. data ready)
    movep X:M_DRXR,a2          ;
    asr #8,a,a                 ; Shift 8 bit data into A1
_LOOP0
    move a1,r0                 ; starting address for load
    move a1,r1                 ; save it in r1
                                ; a0 holds the number of words
; Download P memory through UB

    do a0,_LOOP1              ; Load instruction words
    do #3,_LOOP2              ; for each byte
_LBLA
    jset #2,X:M_DSR,_LBLB      ; Wait for SRRQ to go high (i.e. data ready)
    jclr #3,X:M_DSR,_LBLA      ; If HF0=1, stop loading new data.
    enddo                     ; Must terminate the do loop
    bra <TERMINATE             ; Terminate loop (enddo) and finish
_LBLB
    movep X:M_DRXR,a2          ; Store 16-bit data in accumulator
    asr #8,a,a                 ; Shift 8 bit data into A1
_LOOP2
    movem a1,p:(r0)+           ; Store 24-bit data in P mem
    nop                        ; movem cannot be at LA.
_LOOP1
    bra <FINISH                ; finish bootstrap

;=====
IHOSTLD
    jclr #0,omr,PCIHOSTLD      ; If MD:MC:MB:MA=x100, go load from PCI HOST

;=====
; This routine loads from the Host Interface in ISA (UNIVERSAL) mode.
; MD:MC:MB:MA=x101 - Host ISA

; Using self configuration mode, the base address in CBMA is written with
; $2f which corresponds to an ISA HTXR address of $2fe (Serial Port 2 Modem
; Status read only register).

ISAHOSTLD
    move #$5a,b                ; b1=$5a0000
    movep b1,X:M_DCTR          ; Configure HI32 as Self-Config
    movep #$00002f,X:M_DPMC    ; write to DPMC
    rep #4
    movep X0,X:M_DPAR          ; write to DPAR (CSTR+CCMR,CCCR+CRID,CLAT,CBMA)
                                ; completing 32 bit write
; Switch to ISA mode

```



```

    movep X0,X:M_DCTR      ; Software personal reset
    move  #010020,y1       ; width 16, offset 32
                           ; (also used as replacement to needed NOP after
                           ;sw reset!)

    movep #03a0000,X:M_DCTR ; HM=$3 (UB)
                           ; HIRD=1 (HIRQ_ pin - drive high enabled)
                           ; HIRH=0 (HIRQ_ pin - handshake disabled)
                           ; HRSP=1 (HRST pin - active low)
                           ; HDRP=0 (HDRQ pin - active high)
                           ; HTAP=0 (HTA_ pin - active high)
                           ; HDSM=0 (Data-strobe pin mode enabled)

; read the "magic sequence" 32 consecutive words with value $37
_LBLC
    do #32,_LOOP3         ;
    jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
    movep X:M_DRXR,A1      ; Store 24-bit data into A1
    and  #00ffff,A         ; Mask upper byte
    cmp  #$37,A           ; Compare the 24-bit dat to $000037
    beq  <_LBLD           ; If data = $37 then go back to loop
    enddo                 ; else break the loop and retry
    bra  <_LBLC

_LBLD
    nop

_LOOP3

; read new CBMA value ("ISA base address")
    jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
    movep X:M_DRXR,A1      ; Store 24-bit data into A1

; Switch to Self Configuration mode
    movep X0,X:M_DCTR      ; Software personal reset
    movep A1,X:M_DPMC      ; write to DPMC
                           ; (also used as replacement to needed NOP after sw
reset!)
    movep b1,X:M_DCTR      ; Configure HI32 as Self-Config
    rep  #4
    movep X0,X:M_DPAR      ; write to DPAR (CSTR+CCMR,CCCR+CRID,CLAT,CBMA)

; Switch to ISA mode
    movep X0,X:M_DCTR      ; Software personal reset
    move  #010010,x1       ; width 16, offset 16
                           ; (also used as replacement to needed NOP after sw reset!)
    movep #03a0010,x:M_DCTR ; HM=$3 (UB)
                           ; HIRD=1 (HIRQ_ pin - drive high enabled)
                           ; HIRH=0 (HIRQ_ pin - handshake disabled)
                           ; HRSP=1 (HRST pin - active low)
                           ; HDRP=0 (HDRQ pin - active high)
                           ; HTAP=0 (HTA_ pin - active high)
                           ; HDSM=0 (Double-strobe pin mode enabled)
                           ; HF4 =1 (turn on flag 4 for handshake)

    jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)

```

Bootstrap Program

```

        movep X:M_DRXR,a0      ; Store number of words
        jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
        movep X:M_DRXR,x0      ; Store starting address
        jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
        movep X:M_DRXR,y0      ; Store starting address
        insert x1,x0,a         ; concatenate next 16-bit word
        insert y1,y0,a         ; concatenate next 16-bit word
        move a1,r0             ; start to p-mem
        move a0,a1             ; number of words to transfer

; Download P memory through UB
        lsr a    r0,r1         ; divide loop count by 2 and save r0

        do a1,_LOOP4           ; Load instruction words
_LBLE
        jset #2,X:M_DSR,_LBLEF ; Wait for SRRQ to go high (i.e. data ready)
        jclr #3,X:M_DSR,_LBLE  ; If HF0=1, stop loading new data.
        bra <TERMINATE         ; Terminate loop (enddo) and finish
_LBLEF
        movep X:M_DRXR,a0      ; Store 16-bit data in accumulator
_LBLG
        jset #2,X:M_DSR,_LBLH  ; Wait for SRRQ to go high (i.e. data ready)
        jclr #3,X:M_DSR,_BLG   ; If HF0=1, stop loading new data.
        bra <TERMINATE         ; Terminate loop (enddo) and finish
_LBLH
        movep X:M_DRXR,x0      ; Store 16-bit data in register
_LBLI
        jset #2,X:M_DSR,_LBLJ  ; Wait for SRRQ to go high (i.e. data ready)
        jclr #3,X:M_DSR,_BLI   ; If HF0=1, stop loading new data.
        bra <TERMINATE         ; Terminate loop (enddo) and finish
_LBLJ
        movep X:M_DRXR,y0      ; Store 16-bit data in register
        insert x1,x0,a         ; concatenate next 16-bit word
        insert y1,y0,a         ; concatenate next 16-bit word
        movem a0,p:(r0)+       ; Store 24-bit data in P mem.
        movem a1,p:(r0)+       ; Store 24-bit data in P mem.
        nop                    ; movem cannot be at IA.
_LOOP4
        bra <FINISH            ; and go get another 24-bit word.
                                ; finish bootstrap

;=====
; This is the routine that loads from the Host Interface in PCI mode.
; MD:MC:MB:MA=x100 - Host PCI

PCIHSTLD
        bset #20,X:M_DCTR      ; Configure HI32 as PCI
UB3_CONT
        jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
        movep X:M_DRXR,a0      ; Store number of words
        jclr #2,X:M_DSR,*      ; Wait for SRRQ to go high (i.e. data ready)
        movep X:M_DRXR,r0      ; Store starting address
        move r0,r1             ; save r0

```

```

do a0,_LOOP5          ; Load instruction words
_LBLK
    jset #2,X:M_DSR,_LBLK ; Wait for SRRQ to go high (i.e. data ready)
    jclr #3,X:M_DSR,_LBLK ; If HF0=1, stop loading data. Else check SRRQ.
    bra <TERMINATE        ; Terminate loop (enddo) and finish
_LBLK
    movep X:M_DRXR,P:(R0)+ ; Store 24-bit data in P mem.
    nop                  ; movem cannot be at LA.
_LOOP5                  ; and go get another 24-bit word.
                        ; finish bootstrap
    bra <FINISH          ;

;=====
EPRSCILD
    jclr #1,omr,EPROMLD ; If MD:MC:MB:MA=x001, go load from EPROM
    jclr #0,omr,SCILD   ; If MD:MC:MB:MA=x010, go load from SCI
                        ; If MD:MC:MB:MA=x011, DSP56301-to-DSP56301 boot

;=====
; This is the routine for DSP56301-to-DSP56301 boot.
; MD:MC:MB:MA=x011 - HI32 in UB mode, double strobe, HTA pin active low

UB3HOSTLD
    movep #$268000,x:M_DCTR ; HM=$2 (UB)
                        ; HIRD=0 (HIRQ pin - drive high disabled, open drain)
                        ; HIRH=1 (HIRQ pin - handshake enabled)
                        ; HRSP=1 (HRST pin - active low)
                        ; HDRP=0 (HDRQ pin - active high)
                        ; HTAP=1 (HTA pin - active low)
                        ; HDSM=0 (Double-strobe pin mode enabled)

    bra <UB3_CONT        ; continue

;=====
; This is the routine that loads from the SCI.
; MD:MC:MB:MA=x010 - external SCI clock

SCILD
    movep #$0302,X:M_SCR ; Configure SCI Control Reg
    movep #$C000,X:M_SCCR ; Configure SCI Clock Control Reg
    movep #7,X:M_PCRE    ; Configure SCLK, TXD and RXD

    do #6,_LOOP6         ; get 3 bytes for number of
                        ; program words and 3 bytes
                        ; for the starting address

    jclr #2,X:M_SSR,*    ; Wait for RDRF to go high
    movep X:M_SRXL,A2    ; Put 8 bits in A2
    jclr #1,X:M_SSR,*    ; Wait for TDRE to go high
    movep A2,X:M_STXL    ; echo the received byte
    asr #8,a,a
_LOOP6
    move a1,r0           ; starting address for load

```

Bootstrap Program

```

        move a1,r1                ; save starting address

        do a0,_LOOP7              ; Receive program words
        do #3,_LOOP8
        jclr #2,X:M_SSR,*         ; Wait for RDRF to go high
        movep X:M_SRXL,A2         ; Put 8 bits in A2
        jclr #1,X:M_SSR,*         ; Wait for TDRE to go high
        movep a2,X:M_STXL         ; echo the received byte
        asr #8,a,a
_LOOP8
        movem a1,p:(r0)+          ; Store 24-bit result in P mem.
        nop                      ; movem cannot be at LA.
_LOOP7
        bra <FINISH               ; Boot from SCI done

;=====
; This is the routine that loads from external EPROM.
; MD:MC:MB:MA=x001

EPROMLD
        move #BOOT,r2             ; r2 = address of external EPROM
        movep #AARV,X:M_AAR1     ; aar1 configured for SRAM types of access

        do #6,_LOOP9              ; read number of words and starting address
        movem p:(r2)+,a2          ; Get the 8 LSB from ext. P mem.
        asr #8,a,a               ; Shift 8 bit data into A1
_LOOP9
        ;
        move a1,r0                ; starting address for load
        move a1,r1                ; save it in r1
        ; a0 holds the number of words

        do a0,_LOOP10             ; read program words
        do #3,_LOOP11             ; Each instruction has 3 bytes
        movem p:(r2)+,a2          ; Get the 8 LSB from ext. P mem.
        asr #8,a,a               ; Shift 8 bit data into A1
_LOOP11
        ; Go get another byte.
        movem a1,p:(r0)+          ; Store 24-bit result in P mem.
        nop                      ; movem cannot be at LA.
_LOOP10
        ; and go get another 24-bit word.
        bra <FINISH               ; Boot from EPROM done

;=====
TERMINATE
        enddo                     ; End the loop before exit.
FINISH

; This is the exit handler that returns execution to normal
; expanded mode and jumps to the RESET vector.

        andi #$0,CCR              ; Clear CCR as if RESET to 0.
        jmp (r1)                  ; Then go to starting Prog addr.
; End of bootstrap code. Number of program words: 191.

```

APPENDIX E

DSP56301 CHIP ERRATA FIXED IN REVISION B

Note: Refer to the Motorola World-Wide Web site at <http://www.motorola-dsp.com> for the chip errata lists for Rev. A silicon.

Table 10 DSP56301 Rev. A Errata Fixed in Rev. B

Errata No.	Brief Description (See 0F92R or 1F92R Errata List for detailed descriptions)
1	Jcc/Bcc to LA does not work properly in interrupts are enabled.
2	Various JTAG related errors.
3	Cannot use low frequency (< 500 kHz) crystal for clock source.
4	Second DMA channel does not transfer data if first DMA channel stalls or locks.
5	Two sequential 1-cycle writes to the same peripheral do not work properly.
6	If OMR[4] is set, DMA and core contention may prevent proper DMA operation.
7	Certain conditions may cause damage to stack extension operation.
8	STOP instruction does not work properly.
9	The interrupt, HCLK, and $\overline{\text{RESET}}$ pins do not have proper 5 V protection.
10	Certain conditions may cause false assertion of the $\overline{\text{HIRQ}}$ signal.
11	Jcc/Bcc to subroutine causes stack extension mechanism not to work properly.
12	Triggered-by-request DMA transfers in Wait state may cause a false DMA transfer to occur (i.e., a request may cause two transfers instead of one).
15	The CILP register location is incorrectly defined (at address \$FC instead of \$3C).
17	Certain conditions may prevent the proper occurrence of a DMA interrupt.
18	Enabling stack extension may cause improper operation for a MOVE to/from SSH if followed by a Type0 Address Generation Interlock.
20	Enabling stack extension may cause improper operation for change of flow instructions at LA-1, LA-2, or LA-3.

Table 10 DSP56301 Rev. A Errata Fixed in Rev. B


Errata No.	Brief Description (See 0F92R or 1F92R Errata List for detailed descriptions)
21	Reading the HCVR in PCI mode with DMA transfers to DTXS enabled may cause false DMA transfers.
22	If the PCI master inserts more than one wait state when reading the HCVR with data in HRXS, it reads HRXS instead of HCVR.
23	The HC bit may remain set even after the HCP status bit in DSR is cleared.
24	If a DMA zero-wait state transfer occurs with the DMA interrupt enabled, two interrupts may occur (one real and one false).
25	The PCAP pin has a latchup sensitivity.
26	Under certain conditions, writing a zero to RREQ and TREQ in HCTR does not clear the interrupt request.
28	Trace mode does not work properly during REP instruction execution.
29	A target disconnect when the HI32 is a PCI master may cause the Remaining Data Count to be erroneous.
30	Voltage is clamped to $V_{cc} + 0.4\text{ V}$ for certain 5 V tolerant pins used in open drain mode.
31	The JTAG port incorrectly reports “Debug” mode instead of “User” mode if the chip is in Debug mode and $\overline{\text{RESET}}$ is asserted without asserting $\overline{\text{TRST}}$.
32	The JTAG port incorrectly reports “User” mode instead of “Debug” mode if the chip is in Debug mode and $\overline{\text{TRST}}$ is asserted.
33	The last bit of the transmitted byte may be truncated to half the serial cycle in Synchronous mode.
44	The HI32 DRXR FIFO pointers may be corrupted by certain multiple PCI master situations.



OnCE and Mfax are trademarks of Motorola, Inc.

© 1997 Motorola, Inc.



Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/Europe/Locations Not Listed:

Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
303-675-2140
1 (800) 441-2447

Mfax™:

RMFAX0@email.sps.mot.com
TOUCHTONE (602) 244-6609
US & Canada ONLY (800) 774-1848

Asia/Pacific:

Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
852-26629298

Technical Resource Center:

1 (800) 521-6274

DSP Helpline

dsphelp@dsp.sps.mot.com

Japan:

Nippon Motorola Ltd
SPD, Strategic Planning Office
4-32-1, Nishi-Gotanda
Shinagawa-ku, Tokyo 141, Japan
81-3-5487-8488

Internet:

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



MOTOROLA