AN1829
Rev. 1, 8/2005

Software Differences Between the DSP56002 and the DSP56303

By Barbara Johnson

The software differences between the DSP56002 and the DSP56303 must be considered when a system based on the DSP56002 is redesigned to use the DSP56303. This application note describes these software differences in detail. For details on the hardware differences between these two DSP products, refer to the application note AN1830, *Hardware Differences Between the DSP56002 and the DSP56303*, which covers memory, interrupt and mode control, core, and peripherals.

This application note does not describe an application *per se*. Rather it summarizes the differences between two DSP products as a convenient reference for designers/programmers who are migrating to the DSP56303 from the DSP56002. It saves you the time that would be required to determine what these differences are.

CONTENTS

1 Introduction

Figure 1 and **Figure 2** show the block diagrams of the DSP56002 and the DSP56303, respectively, in order to provide a context for the next sections, which compare various features and modules of these two DSP products.

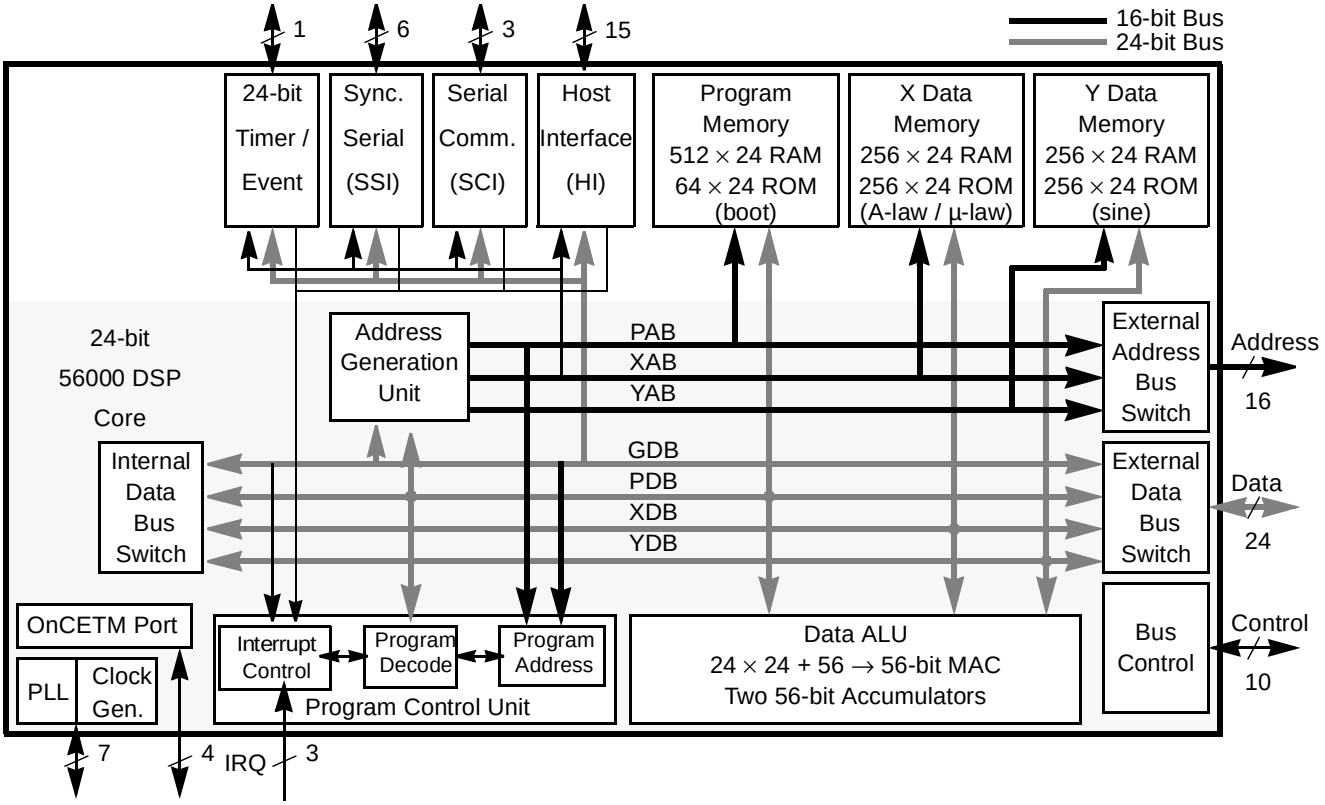

Figure 1. DSP56002 Block Diagram

2 Instruction Pipeline

In the DSP56002, the program control unit implements a three-stage instruction execution pipeline:

1. prefetch
2. decode
3. execute

Each instruction requires a minimum of three instruction cycles to move through this pipeline. There is a delay of three instruction cycles on power-up to fill the pipeline. A new instruction can begin executing immediately after the previous instruction executes. Two-word instructions require a minimum of four instruction cycles to execute—that is, three cycles for the first instruction word to move through the pipeline and execute and one more cycle for the second word to execute. A new instruction can start after two instruction cycles.

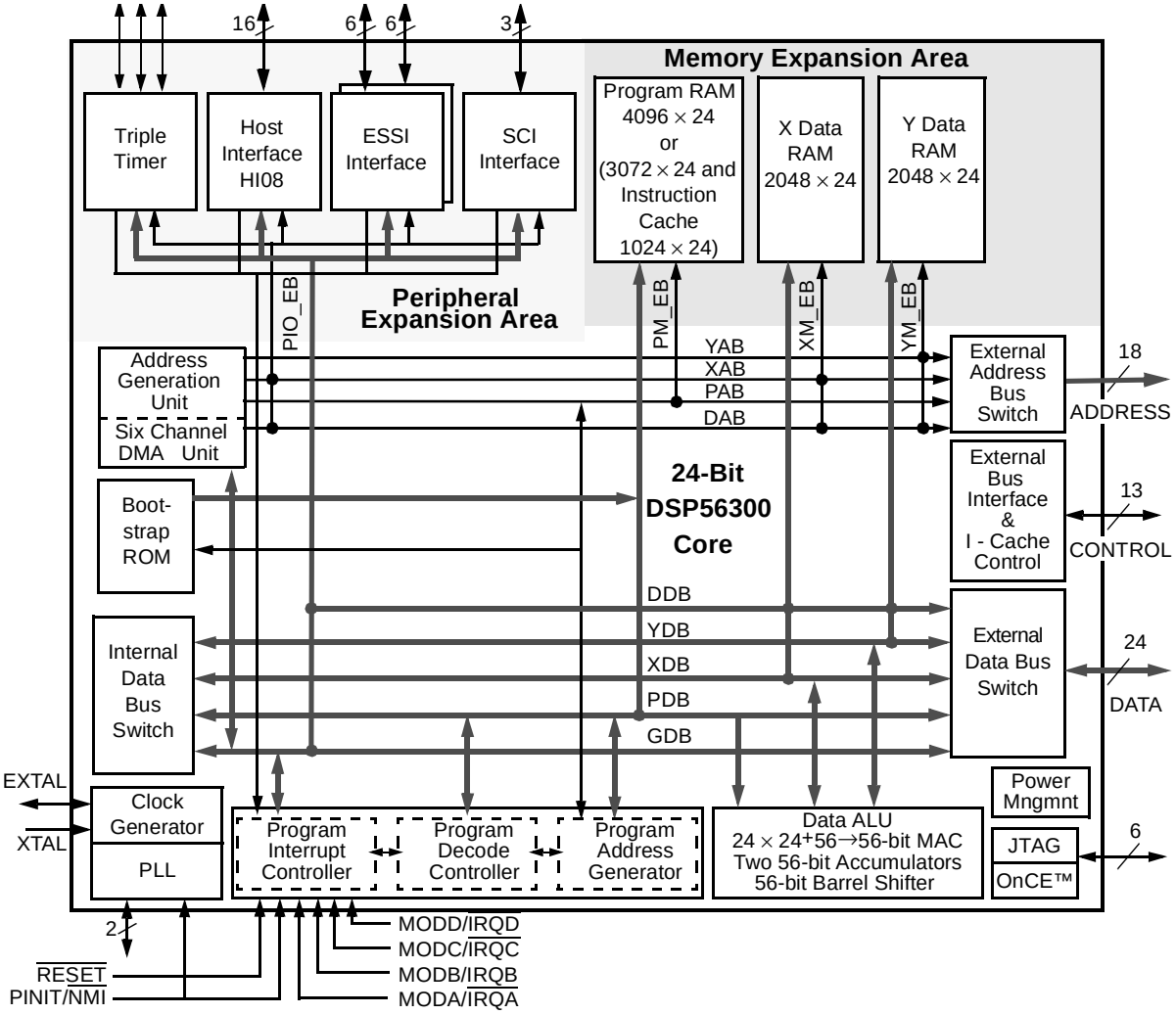

Figure 2. DSP56303 Block Diagram

In the DSP56303, the program control unit implements a seven-stage pipeline:

1. prefetch-I
2. prefetch-II
3. decode
4. address generation-I
5. address generation-II
6. execute-I
7. execute-II

Each instruction requires a minimum of seven instruction cycles to move through this pipeline. There is a delay of seven instruction cycles on power-up to fill the pipeline. A new instruction can begin executing immediately after the previous instruction executes. Two-word instructions require a minimum of eight instruction cycles to execute—that is, seven cycles for the first instruction word to move through the pipeline and execute and one more cycle for the second word to execute. A new instruction can start after two instruction cycles.

3 Instruction Cache Controller

The instruction cache controller is not available on the DSP56000 core. On the DSP56300 core, the instruction cache stores frequently-used program instructions. When instruction words required by a program are available in the on-chip cache, an increase in throughput may result, and the time required to access them on the external bus is eliminated.

3.1 Software Control

On the DSP56300 core, switching between PRAM mode and cache mode is controlled via the Cache Enable CE bit in the Expanded Mode Register (EMR). If EMR[CE] is set, the cache is enabled, instructions are cached into the internal PRAM and fetched from there. If EMR[CE] is cleared, the cache is disabled, and the core fetches instructions from external or internal program memory.

3.2 Instruction Cache Structure

The instruction cache memory array contains 1024 x 24-bit words logically divided into eight 128-word cache sectors. The 24-bit address is divided into the following fields:

- vbit field, 7 least significant bits for the word displacement in the sector
- tag field , 17 most significant bits for the sector base address

The cache controller compares its tag field to the tag values currently stored in the tag register file. These tag values are the tag fields of the base addresses of the memory sectors currently mapped into the cache. Each word in each cache sector is associated with a cache word valid bit that specifies whether the data in that word has been fetched from external memory and is therefore valid. There is a total 2048 valid bits arranged as 8 banks of 128 or 256 valid bits each, one bank for every sector. Of these valid bits, 1024 are not used if the instruction cache size is 1024 bytes.

3.3 Burst Mode

When the Burst Mode Enable (BE) bit in the Extended Chip Operating Mode Register (EOM) is set, up to four program words are fetched from external memory upon detection of an instruction cache miss. When EOM[BE] is cleared, the burst mode is disabled, and only one program word is fetched from external memory upon detection of an instruction cache miss.

3.4 Cache Visibility Via OnCE

When the DSP is in Debug mode, you can observe the memory sectors that are currently mapped into cache, the locked cache sectors, the Least Recently Used cache sector, and the occurrence of a hit. You can also read the valid bits of specific cache locations.

4 Arithmetic Logic Unit

Table 1 summarizes the differences in the arithmetic logic unit between the DSP56002 and the DSP56303.

Table 1. Arithmetic Logic Unit, Comparison Between DSP56002 and DSP56303

DSP56002	DSP56303
MAC Unit	
MAC operation is a non-pipelined single-cycle operation.	MAC operation is fully pipelined and requires two clock cycles to complete. In the first clock cycle, the multiply is performed and the product is stored in the pipeline register. In the second clock cycle, the accumulator is added or subtracted.
Bit Field Unit (BFU)	
Not Available.	The BFU contains a 56-bit parallel bidirectional shifter with a 56-bit input and a 56-bit output mask generation unit and logic unit. It is used in the following operations: • Multiple left/right shift (arithmetic or logical) for ASL/ASR, LSL/LSR • Bit field merge, insert and extract for MERGE, INSERT, EXTRACT, EXTRACTU,Count leading bits for CLB • Fast normalization for NORMF The addition of the BFU on the DSP56300 core has produced new shift and program control-relative instructions (refer to **Section 7**, "Instruction Set," on page 10).
Sixteen-Bit Arithmetic Mode	
Not Available.	Sixteen-Bit Arithmetic mode is enabled by setting the SA (bit 17) in the Status Register (SR). The 16-bit data is right-aligned in 24-bit memory locations and non-Data ALU registers as shown here. **Table 4-1.** **23–16** **15–0** The bit positions in the memory long word in Sixteen-Bit Arithmetic mode are as follows: **Table 4-2.** **23:16** **15:0** **23:16** **15:0** The 16-bit data is left-aligned in Data ALU input registers as follows: **Table 4-3.** **47:32** **31:24** **23:8** **7:0** X1/Y1 Undefined X0/Y0 Undefined The 16-bit data is left-aligned in Data ALU accumulator registers as follows: **Table 4-4.** **55:48** **47:32** **31:24** **23:8** **7:0** A2/B2 A1/B1 Undefined A0/B0 Undefined

Table 1. Arithmetic Logic Unit, Comparison Between DSP56002 and DSP56303 (Continued)

DSP56002	DSP56303
Rounding Modes	
Convergent rounding mode is implemented.	Convergent and twos complement rounding are implemented. To select Rounding mode, set the RM bit (bit 21) in the SR. If RM is set, twos complement rounding is selected; if RM is cleared, convergent rounding is selected. Convergent rounding rounds down if the number is even (LSB=0) and rounds up if the number is odd (LSB=1). Twos complement rounding rounds down all values below one-half and rounds up all values of one-half or greater.
Saturation Mode	
Not Available	The arithmetic unit's result is limited to 48 bits by setting the SM (bit 20) in the SR. This bit provides a saturation mode for algorithms that do not recognize or cannot take advantage of the extension accumulator.

5 Address Generation Unit

Table 2 summarizes the differences in the Address Generation Unit (AGU) between the DSP56002 and the DSP56303.

Table 2. Address Generation Unit, Comparison of DSP56002 and DSP56303

DSP56002	DSP56303
AGU Registers	
The 24 Address Generation Unit (AGU) registers (address registers R0–R7, offset registers N0–N7, and modifier registers M0–M7) are 16 bits wide. When used as a source operand, these registers occupy the low-order portion of the 24-bit word; the high-order portion is read as zeros. When used as a destination operand, these registers receive the low-order portion of the word; the high-order portion is not used.	The 24 AGU registers are 24-bits wide.
Sixteen-Bit Compatibility Mode	
Not Available	When the Sixteen-Bit Compatibility Mode bit in the MR is set, move operations to and from the LA, LC, SP, SSL, SSH, EP, SZ, VBA, and SC registers clear the eight most significant bits of the destination. This guarantees compatibility for object code written for the DSP56000 family of DSPs.

6 Programming Model

This section compares key registers of the DSP56002 and the DSP56303. The registers covered are as follows:

- Status Register (SR), **Table 3** on page 7
- Operating Mode Register (OMR), **Table 4** on page 8
- System Stack (SS), **Table 5** on page 8

- Program Counter (PC), **Table 6** on page 9
- Vector Base Address Register (VBA), **Table 7** on page 9
- Loop Counter Register (LC), **Table 8** on page 9
- Loop Address Register (LA), **Table 9** on page 9
- Stack Extension Pointer (EP), **Table 10** on page 9
- Stack Size Register (SZ), **Table 11** on page 9
- Stack Counter Register (SC), **Table 12** on page 10
- Stack Pointer Register (SC), **Table 13** on page 10

Table 3. Status Register, Comparison of the DSP56002 and the DSP56303

DSP56002				DSP56303			
The Status Register (SR) is a 16-bit register that consists of a Mode Register MR in the high-order eight bits and a Condition Code Register CCR in the low-order eight bits.				The SR is 24-bit register that consists of an Extended Mode Register EMR in the high-order eight bits, an MR in the middle-order eight bits, and a CCR in the low-order eight bits.			
				EMR	23–22	CP1–CP0	Core Priority
					21	RM	Rounding Mode
					20	SM	Arithmetic Saturation Mode
					19	CE	Instruction Cache Enable
					18	—	Reserved
					17	SA	Sixteen-Bit Arithmetic Mode
					16	FV	DO-Forever Flag
					15	LF	DO-Loop Flag
MR	15	LF	Loop Flag	MR	15	LF	Loop Flag
	14	DM	Double Precision Multiply Mode		14	DM	Double-Precision Multiply Mode
	13	T	Trace Mode		13	SC	Sixteen-Bit Compatibility Mode
	12	—	Reserved		12	—	Reserved
	11-10	S1 – S0	Scaling Mode		11-10	S1 – S0	Scaling Mode
	9–8	I1–I0	Interrupt Mask		9–8	I1 – I0	Interrupt Mask
CCR	7	S	Scaling	CCR	7	S	Scaling
	6	L	Limit		6	L	Limit
	5	E	Extension		5	E	Extension
	4	U	Unnormalized		4	U	Unnormalized
	3	N	Negative		3	N	Negative
	2	Z	Zero		2	Z	Zero
	1	V	Overflow		1	V	Overflow
	0	C	Carry		0	C	Carry

Table 4. Operating Mode Register, Comparison of the DSP56002 and the DSP56303

DSP56002				DSP56303			
The Operating Mode Register (OMR) is a 24-bit register with only six bits defined.				The OMR is a 24-bit register divided into three 8-bit sections. The upper byte consists of the System Stack Control Status Register (SCS), which controls and monitors the stack extension. The two lower bytes consist of the Extended Chip Operating Mode Register (EOM) and the Chip Operating Mode Register (COM), which control the DSP operating mode.			
				SCS	23–21	—	Reserved
					20	SEN	Stack Extension Enable
					19	WRP	Extended Stack Wrap Flag
					18	EOV	Extended Stack Overflow Flag
					17	EUN	Extended Stack Underflow Flag
					16	XYS	Stack Extension Space Select
				EOM	15	ATE	Address Tracing Enable
					14–13	—	Reserved
					12	BRT	Bus Release Timing
					11	TAS	TA Synchronize Select
					10	BE	Burst Mode Enable
					9–8	CDP1 – CDP0	Interrupt Mask
				COM	7	MS	Memory Switch Mode
	6	SD	Stop Delay		6	SD	Stop Delay
	4	MC	Operating Mode C		5	—	Reserved
	3	YD	Internal Y Memory Disable		4	EBD	External Bus Disable
	2	DE	Data ROM Enable		3–0	MD – MA	Operating Mode D–A
	1	MB	Operating Mode B				
	0	MA	Operating Mode A				

Table 5. System Stack, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
The System Stack (SS) is a separate 15-level by 32-bit internal memory divided into two 16-bit wide banks: the System Stack High (SSH) and the System Stack Low (SSL). The SS accommodates up to 15 long interrupts, 7 DO loops, 15 JSRs, or combinations of these.	The SS is a separate 16-level by 48-bit internal memory divided into two 24-bit wide banks: the System Stack High (SSH) and the System Stack Low (SSL). If the Stack Extension Enable (SEN) bit in the OMR is cleared, the SS accommodates up to 15 long interrupts, 7 DO loops, 15 JSRs, or combinations of these. If the SEN bit is set, the stack extension is enabled and the limits on the level of nesting of subroutines or DO loops can be set to any desired value.

Table 6. Program Counter, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
The Program Counter (PC) is a 16-bit register that contains the address of the next location to be fetched from the program memory space.	The PC is a 24-bit register.

Table 7. Vector Base Address Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
Not Available	The Vector Base Address Register (VBA) is a 24-bit register that is used as a base address of the interrupt vector and interrupt vector+1. Bits 7–0 are read-only and are always cleared.

Table 8. Loop Counter Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
The Loop Counter Register (LC) is a 16-bit counter that specifies the number of times a hardware program loop is to be repeated.	The LC is a 24-bit counter.

Table 9. Loop Address Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
The Loop Address Register (LA) is a 16-bit register that indicates the location of the last instruction word in a hardware loop.	The LA is a 24-bit register.

Table 10. Stack Extension Pointer Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
Not Available	The Stack Extension Pointer Register (EP) is a 24-bit register that points to the stack extension in data memory when the stack extension is enabled and move operations to/from the on-chip hardware stack are needed.

Table 11. Stack Size Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
Not Available	The Stack Size Register (SZ) is a 24-bit register that determines the number of data words allocated in memory for the stack in the extended mode.

Table 12. Stack Counter Register, Comparison of the DSP56002 and the DSP56303

DSP56002	DSP56303
Not Available	The Stack Counter Register (SC) is a 5-bit register that monitors how many entries of the hardware stack are in use.

Table 13. Stack Pointer Register, Comparison of the DSP56002 and the DSP56303

DSP56002

The Stack Pointer Register (SP) is a 6-bit register that indicates the location of the top of the SS and the status of the SS.

Bit No.	Description	
5	UF	Underflow Flag
4	SE	Stack Error Flag
3–0	P3–P0	Stack Pointer

DSP56303

The SP is a 24-bit register. When the extended mode is disabled, the status of the SS is also indicated in the SP.

Bit No.	Description		
		Extended Mode	Non-Extended Mode
23–6	P23–P6	Bits 23–6 of the Stack Pointer	
5	UF/P5	Bit 5 of the Stack Pointer	Underflow Flag
4	SE/P4	Bit 4 of the Stack Pointer	Stack Error Flag
3–0	P3–P0	Bits 3–0 of the Stack Pointer	

7 Instruction Set

Table 14 shows the DSP56300 instructions that are either available on the DSP56303 but not on the DSP56002 or that are enhanced on the DSP56303.

Table 14. DSP56300 Instructions

Type	Instruction	Description	Status
Arithmetic	ADD	Add	Enhanced on DSP56300
	ASL	Arithmetic Shift Accumulator Left	Enhanced on DSP56300
	ASR	Arithmetic Shift Accumulator Right	Enhanced on DSP56300
	CMP	Compare	Enhanced on DSP56300
	CMPU	Compare Unsigned	DSP56300 only
	DMAC	Double (Multi) Precision Multiply Accumulate with Right Shift	DSP56300 only
	MACI	Signed Multiply-Accumulate with Immediate Operand	DSP56300 only
	MACsu	Mixed Multiply Accumulate (S1 signed, S2 unsigned)	DSP56300 only
	MACuu	Mixed Multiply Accumulate (S1 and S2 unsigned)	DSP56300 only
	MACRI	Signed Multiply-Accumulate and Round with Immediate Operand	DSP56300 only

Table 14. DSP56300 Instructions (Continued)

Type	Instruction	Description	Status
Arithmetic cont.	MAX	Transfer by Signed Value	DSP56300 only
	MAXM	Transfer by Magnitude	DSP56300 only
	MERGE	Merge Two Half Words	DSP56300 only
	MPYsu	Mixed Multiply (S1 signed, S2 unsigned)	DSP56300 only
	MPYuu	Mixed Multiply (S1 and S2 unsigned)	DSP56300 only
	MPYI	Signed Multiply with Immediate Operand	DSP56300 only
	MPYRI	Signed Multiply and Round with Immediate Operand	DSP56300 only
	NORMF	Fast Accumulator Normalization	DSP56300 only
	SUB	Subtract	Enhanced on DSP56300
Logical	AND	Logical AND	Enhanced on DSP56300
	CLB	Count Leading Bits	DSP56300 only
	EOR	Logical Exclusive OR	Enhanced on DSP56300
	EXTRACT	Extract Bit Field	DSP56300 only
	EXTRACTU	Extract Unsigned Bit Field	DSP56300 only
	INSERT	Insert Bit Field	DSP56300 only
	LSL	Logical Shift Left	Enhanced on DSP56300
	LSR	Logical Shift Right	Enhanced on DSP56300
	OR	Logical OR	Enhanced on DSP56300
Loop	DOR	Start PC Relative Hardware Loop	DSP56300 only
	BRKcc	Exit Current Do Loop Conditionally	DSP56300 only
Move	LRA	Load PC Relative Address	DSP56300 only
Program Control	IFcc	Execute Conditionally	DSP56300 only
	IFcc.U	Execute Conditionally and Update Condition Code Register	DSP56300 only
	BRCLR	Branch if Bit Clear	DSP56300 only
	BRSET	Branch if Bit Set	DSP56300 only
	BSCLR	Branch to Subroutine if Bit Clear	DSP56300 only
	BSSET	Branch to Subroutine if Bit Set	DSP56300 only
	PLOCK	Lock Instruction Cache Sector	DSP56300 only
	PUNLOCK	Unlock Instruction Cache Sector	DSP56300 only
	PLOCKR	Lock Instruction Cache Relative Sector	DSP56300 only
	PUNLOCKR	Unlock Instruction Cache Relative Sector	DSP56300 only
	PFREE	Program Cache Global Unlock	DSP56300 only
	PFLUSH	Program Cache Flush	DSP56300 only
	PFLUSHUN	Program Cache Flush Unlocked Sectors	DSP56300 only
	TRAP	Software Interrupt	DSP56300 only
	TRAPcc	Conditional Software Interrupt	DSP56300 only

7.1 Arithmetic Instructions

Table 15 lists the arithmetic instructions that are either available on the DSP56303 but not on the DSP56002 or enhanced on the DSP56303.

Table 15. Arithmetic Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
ADD	Enhanced to support the addition of a 6-bit short immediate or a 24-bit long immediate data to the destination accumulator.	**ADD S,D** ;S is source register X0, X1, Y0, Y1, X, Y, A, B ;D is destination accumulator A, B **ADD #ii,D** ;#ii is 6-bit immediate short data **ADD #iiiiii,D** ;#iiiiii is 24-bit immediate long data extension ;word
ASL/ASR	Enhanced to support the multi-bit shifting.	**ASL D** ;D is destination accumulator A,B **ASL #ii,S2,D** ;#ii is 6-bit unsigned integer [0-55] ;denoting the shift amount ;S2 is source accumulator A, B **ASL S1,S2,D** ;S1 is control register X0, X1, Y0, Y1, A1, B1
CMP	Enhanced to support the comparison of a 6-bit short immediate data or a 24-bit long immediate data to a source accumulator.	**CMP S1,S2** ;S1 is source one register X0, X1, Y0, Y1, A, B ;S2 is source two accumulator A, B **CMP #ii,S2** ;#ii is 6-bit immediate short data **CMP #iiiiii,S2** ;#iiiiii is 24-bit immediate long data ;extension word
CMPU	DSP56300 only. Subtracts the source operand from the source accumulator and updates the condition code register. The result of the operation is not stored.	**CMPU S1,S2** ;S1 is source one register X0, X1, Y0, Y1, A, B ;S2 is source two accumulator A, B
DMAC	DSP56300 only. Multiplies the two signed 24-bit source operands and adds/subtracts the product to/from the specified 56-bit destination accumulator which has been previously shifted 24 bits to the right.	**DMAC (+/-)S1,S2,D** ;S1, S2 are source registers X0, Y0, X1, Y1 ;D is destination accumulator A, B
MACI	DSP56300 only. Multiplies the two signed 24-bit source operands and adds/subtracts the product to/from the specified 56-bit destination accumulator.	**MACI (+/-)#iiiiii,S,D** ;#iiiiii is 24-bit immediate long data ;extension word ;S is source register ;X0, Y0, X1, Y1 ;D is destination accumulator A, B

Table 15. Arithmetic Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002 (Continued)

Instruction	Description	Valid Assembler Syntax
MACsu/ MACuu	DSP56300 only. MACsu multiplies the two 24-bit source operands, in which one source operand is unsigned and the other is signed, and adds/subtracts the product to/from the specified 56-bit destination accumulator. MACuu multiplies the two 24-bit source operands, in which both source operands are unsigned, and adds/subtracts the product to/from the specified 56-bit destination accumulator.	**MACsu (+/-)S1,S2,D** ;S1, S2 are source registers X0, X1, ;Y0, Y1 ;D is destination accumulator A, B
MACRI	DSP56300 only. Multiplies the two signed 24-bit source operands and adds/subtracts the product to/from the specified 56-bit destination accumulator.	**MACRI (+/-)#iiiiii,S,D** ;#iiiiii is 24-bit immediate long data ;extension word ;S is source register X0, Y0, X1, Y1 ;D is destination accumulator A, B
MAX	DSP56300 only. Subtracts the signed value of the source accumulator from the signed value of the destination accumulator. If the difference is negative or zero, the source accumulator is transferred to the destination accumulator. Otherwise, the destination accumulator remains unchanged.	**MAX A,B** ;A, B are source accumulators
MAXM	DSP56300 only. Subtracts the magnitude of the source accumulator from the magnitude of the destination accumulator. If the difference is negative or zero, the source accumulator is transferred to the destination accumulator. Otherwise, the destination accumulator remains unchanged.	**MAXM A,B** ;A, B are source accumulators
MERGE	DSP56300 only. Concatenates the contents of bits 11–0 of the source register to the contents of bits 35–24 of the destination accumulator. In 16-bit arithmetic mode, the contents of bits 15–8 of the source register are concatenated to the contents of bits 39–32 of the destination accumulator. The result is placed in bits 47–32 of the destination accumulator.	**MERGE S,D** ;S is source register X0, X1, Y0, Y1, A1, B1 ;D is destination accumulator A, B
MPYsu/ MPYuu	DSP56300 only. MPYsu multiplies the two 24-bit source operands, in which one source operand is unsigned and the other is signed, and stores the product in the specified 56-bit destination accumulator. MPYuu multiplies the two 24-bit source operands, in which both source operands are unsigned, and stores the product in the specified 56-bit destination accumulator.	**MPYsu (+/-)S1,S2,D** ;S1, S2 are source registers X0, X1, Y0, ;Y1 ;D is destination accumulator A, B

Table 15. Arithmetic Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002 (Continued)

Instruction	Description	Valid Assembler Syntax
MPYI	DSP56300 only. Multiplies the immediate 24-bit source operand with the 24-bit source operand and stores the resulting product in the specified 56-bit destination accumulator.	**MPYI (+/-)#iiiiii,S,D** ;#iiiiii is 24-bit immediate long data ;extension word ;S is source register X0, Y0, X1, Y1 ;D is destination accumulator A, B
MPYRI	DSP56300 only. Multiplies the two signed 24-bit source operands, rounds the result using either convergent or twos complement rounding, and stores the result in the specified 56-bit destination accumulator.	**MPYRI (+/-)#iiiiii,S,D** ;#iiiiii is 24-bit immediate long data ;extension word ;S is source register X0, Y0, X1, Y1 ;D is destination accumulator A, B
NORMF	DSP56300 only. Arithmetically shifts the destination accumulator either left or right as specified by the source operand sign and value. If the source operand is negative, then the accumulator is left shifted, and if the source operand is positive, then it is right shifted. It can be used to bring the destination accumulator's leading one or zero to bit location 46.	**NORMF S,D** ;S is source register X0, Y0, X1, Y1, A1, B1 ;D is destination accumulator A, B
SUB	Enhanced to support the subtraction of a 6-bit short immediate or a 24-bit long immediate data from the destination accumulator.	**SUB S,D** ;S is source register X0, X1, Y0, Y1, X, Y, A, B ;D is destination accumulator A, B **SUB #ii,D** ;#ii is 6-bit immediate short data **SUB #iiiiii,D** ;#iiiiii is 24-bit immediate long data ;extension word

7.2 Logical Instructions

Table 16 lists the logical instructions that are either available on the DSP56303 but not on the DSP56002 or that are enhanced on the DSP56303.

Table 16. Logical Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
AND	Enhanced to support the logical AND operation between a 6-bit short immediate data or a 24-bit long immediate data and a source accumulator.	**AND S,D** ;S is source input register X0, X1, Y0, Y1 ;D is destination accumulator A, B **AND #ii,D** ;#ii is 6-bit immediate short data **AND #iiiiii,D** ;#iiiiii is 24-bit immediate long ;data extension word
CLB	DSP56300 only. Counts the leading zeros or ones according to bit 55 of the source accumulator. If bit 55 is zero, then bits 47–24 of the destination accumulator are loaded with 9 minus the number of consecutive leading zeros in bits 55–0 of the source accumulator. If bit 55 is one, then bits 47–24 of the destination accumulator is loaded with 9 minus the number of consecutive leading ones in bits 55–0 of the source accumulator.	**CLB S,D** ;S is source accumulator A, B ;D is destination accumulator A, B
EOR	Enhanced to support the logical Exclusive OR operation between a 6-bit short immediate data or a 24-bit long immediate data and a source accumulator.	**EOR S,D** ;S is source input register X0, X1, Y0, Y1 ;D is destination accumulator A, B **EOR #ii,D** ;#ii is 6-bit immediate short data **EOR #iiiiii,D** ;#iiiiii is 24-bit immediate long data ;extension word
EXTRACT/ EXTRACTU	DSP56300 only. Extracts a bit-field from source accumulator S2, while the instruction EXTRACTU extracts an unsigned bit-field from source accumulator S2. The bit-field width is specified by bits 17–12 in S1 register or in immediate control word #CO. The offset from the least significant bit is specified by bits 5–0 in S1 or in #CO. In 16-bit arithmetic mode, the offset field is located in bits 13–8 of the control register and the width field is located in bits 21–16 of the control register.	**EXTRACT S1,S2,D** ; S1 is control register X0, X1, ;Y0, Y1, A1, B1 ;S2 is source accumulator A, B ;D is destination accumulator A, B **EXTRACTU #CO,S2,D** ;#CO is control word extension

Table 16. Logical Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002 (Continued)

Instruction	Description	Valid Assembler Syntax
INSERT	DSP56300 only. Inserts a bit-field into the destination accumulator. The bit-field width is specified by bits 17–12 in S1 register and begins at the least significant bit of the S2 register. The bit-field is inserted into the destination accumulator with an offset according to bits 5–0 in S1 register. S1 can be an immediate control word #CO. In 16-bit arithmetic mode, the offset field is located in bits 13–8 of the control register and the width field is located in bits 21–16 of the control register.	**INSERT S1,S2,D** ;S1 is control register X0, X1, Y0, Y1, A1, B1 ;S2 is source accumulator A, B ;D is destination accumulator A, B **INSERT #CO,S2,D** ;#CO is control word extension
LSL/LSR	Enhanced to support the multi-bit shifting.	**LSL D** ;D is destination accumulator A, B **LSL #ii,D** ;#ii is 5-bit unsigned integer [0-23] denoting the ;shift amount **LSL S,D** ;S is control register X0, X1, Y0, Y1, A1, B1
OR	Enhanced to support the logical inclusive OR operation between a 6-bit short immediate data or a 24-bit long immediate data and a source accumulator.	**OR S,D** ;S is source input register X0, X1, Y0, Y1 ;D is destination accumulator A, B **OR #ii,D** ;#ii is 6-bit immediate short data **OR #iiiiii,D** ;#iiiiii is 24-bit immediate long data ;extension word

7.3 New Loop Instructions

Table 17 lists the loop instructions that are either available on the DSP56303 but not on the DSP56002 or enhanced on the DSP56303.

Table 17. Loop Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

<table>
<tr><th>Instruction</th><th>Description</th><th>Valid Assembler Syntax</th></tr>
<tr><td>DOR</td><td>DSP56300 only. Initiates the beginning of a PC-relative hardware program loop.</td><td>DOR [X or Y]:ea,label
;ea is effective address
;label is 24-bit address displacement in 24-bit
;extension word

DOR [X or Y]:aa,label
;aa is absolute address

DOR #iii,label
;#iii is immediate short data

DOR S,label ;S is source register</td></tr>
<tr><td>DOR FOREVER</td><td>DSP56300 only. Begins a hardware DO loop that is to be repeated forever and whose range of execution is terminated by the destination operand.</td><td></td></tr>
<tr><td>BRKcc</td><td>DSP56300 only.Conditionally exits the current hardware DO loop before the current loop counter equals one. It also terminates the DO FOREVER or DOR FOREVER loop.</td><td>The conditions that the term “cc” can specify are listed as follows:

<table>
<tr><th></th><th>cc Mnemonic</th><th>Condition</th></tr>
<tr><td>CC(HS)</td><td>carry clear (higher or same)</td><td>C = 0</td></tr>
<tr><td>CS(LO)</td><td>carry set (lower)</td><td>C = 1</td></tr>
<tr><td>EC</td><td>extension clear</td><td>E = 0</td></tr>
<tr><td>EQ</td><td>equal</td><td>Z = 1</td></tr>
<tr><td>ES</td><td>extension set</td><td>E = 1</td></tr>
<tr><td>GE</td><td>greater than or equal</td><td>N eor V = 0</td></tr>
<tr><td>GT</td><td>greater than</td><td>Z or (N eor V) = 0</td></tr>
<tr><td>LC</td><td>limit clear</td><td>L = 0</td></tr>
<tr><td>LE</td><td>less than or equal</td><td>Z or (N eor V) = 1</td></tr>
<tr><td>LS</td><td>limit set</td><td>L = 1</td></tr>
<tr><td>LT</td><td>less than</td><td>N eor V = 1</td></tr>
<tr><td>MI</td><td>minus</td><td>N = 1</td></tr>
<tr><td>NE</td><td>not equal</td><td>Z = 0</td></tr>
<tr><td>NR</td><td>normalized</td><td>Z or (U and E) = 1</td></tr>
<tr><td>PL</td><td>plus</td><td>N=0</td></tr>
<tr><td>NN</td><td>not normalized</td><td>Z or (U and E) = 0</td></tr>
</table>
</td></tr>
</table>

7.4 Move Instruction

Table 18 lists the move instruction that is available on the DSP56303 but not on the DSP56002.

Table 18. Move Instruction Added on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
LRA	DSP56300 only. Adds the PC to the specified displacement	**LRA Rn,D** ;Rn is address register R0–R7 ;D is destination address register X0, X1, Y0, ;Y1, A0, B0, ;A2, B2, A1, B1, A, B, R0:R7, N0–N7 **LRA xxxx,D** ;xxxx is 24-bit PC long displacement

7.5 Program Control Instructions

Table 19 lists the program control instructions that are available on the DSP56303 but not on the DSP56002.

Table 19. Program Control Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
IFcc	DSP56300 only. Executes and stores the result of the specified Data ALU operation if the specified condition is true. If the specified condition is false, no destination is altered. The instructions that can conditionally be executed by using this instruction are shown in **Table 17.**	**IFFcc**
IFcc.U	DSP56300 only. Executes and stores the result of the specified Data ALU operation if the specified condition is true and updates the condition code register with the status information generated by the Data ALU operation. If the specified condition is false, no destination is altered and the condition code register is not affected.	**IFFcc.U**

Table 19. Program Control Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

<table>
<tr><th>Instruction</th><th>Description</th><th>Valid Assembler Syntax</th></tr>
<tr><td>BRCLR</td><td>DSP56300 only. Tests the nth bit in the source operand. If the tested bit is cleared, program execution continues at location PC+displacement. If the tested bit is set, the PC is incremented and program execution continues sequentially.</td><td>BRCLR #n,[X or Y]:ea,xxxx
;#n is bit number [0-23]
;ea is effective address (see the following table)
;xxxx is 24-bit PC relative displacement
BRCLR #n,[X or Y]:aa,xxxx;aa is absolute address [0-63]

BRCLR #n,[X or Y]:pp,xxxx
;pp is I/O short address [64
;addresses: $FFFFC0-$FFFFFF]

BRCLR #n,[X or Y]:qq,xxxx
;qq is I/O short address [64
;addresses: $FFFF80-$FFFFBF]

BRCLR #n,S,xxxx
;S is all in-chip registers

<table><tr><th>Effective Addressing Mode</th></tr><tr><td>(Rn)-Nn</td></tr><tr><td>(Rn)+Nn</td></tr><tr><td>(Rn)</td></tr><tr><td>(Rn)+</td></tr><tr><td>(Rn)</td></tr><tr><td>(Rn+Nn)</td></tr><tr><td>-(Rn)</td></tr></table></td></tr>
<tr><td>BRSET</td><td>DSP56300 only. Tests the nth bit in the source operand. If the tested bit is set, program execution continues at location PC+displacement. If the tested bit is cleared, the PC is incremented and program execution continues sequentially.</td><td>BRSET #n,[X or Y]:ea,xxxx
;#n is bit number [0-23]
;ea is effective address (see Table 17).
;xxxx is 24-bit PC relative displacement

BRSET #n,[X or Y]:aa,xxxx;aa is absolute address [0-63]

BRSET #n,[X or Y]:pp,xxxx
;pp is I/O short address [64
;addresses: $FFFFC0-$FFFFFF]

BRSET #n,[X or Y]:qq,xxxx
;qq is I/O short address
;[64 addresses: $FFFF80-$FFFFBF]

BRSET #n,S,xxxx
;S is all in-chip registers</td></tr>
</table>

Table 19. Program Control Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
BSCLR	DSP56300 only. Tests the nth bit in the source operand. If the tested bit is cleared, the address of the instruction immediately following the BSCLR instruction and the status register are pushed onto the stack. Program execution then continues at location PC+displacement. If the tested bit is set, the PC is incremented and program execution continues sequentially.	**BSCLR #n,[X or Y]:ea,xxxx** ;#n is bit number [0-23] ;ea is effective address (see **Table 17**) ;xxxx is 24-bit PC relative displacement **BSCLR #n,[X or Y]:aa,xxxx** **;**aa is absolute address [0-63] **BSCLR #n,[X or Y]:pp,xxxx** **;**pp is I/O short address [64 ;addresses: $FFFFC0-$FFFFFF] **BSCLR #n,[X or Y]:qq,xxxx** ;qq is I/O short address ;[64 addresses: $FFFF80-$FFFFBF] **BSCLR #n,S,xxxx** ;S is all in-chip registers
BSSET	DSP56300 only. Tests the nth bit in the source operand. If the tested bit is set, the address of the instruction immediately following the BSSET instruction and the status register are pushed onto the stack. Program execution then continues at location PC+displacement. If the tested bit is cleared, the PC is incremented and program execution continues sequentially.	**BSSET #n,[X or Y]:ea,xxxx** ;#n is bit number [0-23] ;ea is effective address (see **Table 17**) ;xxxx is 24-bit PC relative displacement **BSSET #n,[X or Y]:aa,xxxx** ;aa is absolute address [0-63] **BSSET #n,[X or Y]:pp,xxxx** ;pp is I/O short address ;[64 addresses: $FFFFC0-$FFFFFF] **BSSET #n,[X or Y]:qq,xxxx** ;qq is I/O short address ;[64 addresses: $FFFF80-$FFFFBF] **BSSET #n,S,xxxx** ;S is all in-chip registers
PLOCK/ PUNLOCK	DSP56300 only. PLOCK locks the cache sector to which the specified effective address belongs. If the sum does not belong to any cache sector, it loads the 17 most significant bits of the sum into the least recently used cache sector tag, and then locks that cache sector. PUNLOCK unlocks the cache sector to which the specified effective address belongs. If the sum does not belong to any cache sector, and is therefore definitely unlocked, PUNLOCK loads the least recently used cache sector tag with the 17 most significant bits of the sum.	**PLOCK ea** ;ea is effective address

Table 19. Program Control Instructions Added/Enhanced on the DSP56303 But Not on the DSP56002

Instruction	Description	Valid Assembler Syntax
PLOCKR/ PUNLOCKR	DSP56300 only. PLOCKR locks the cache sector to which the sum PC + specified displacement belongs. If the sum does not belong to any cache sector, then it loads the 17 most significant bits of the sum into the least recently used cache sector tag, and then locks that cache sector. PUNLOCKR unlocks the cache sector to which the sum PC + specified displacement belongs. If the sum does not belong to any cache sector, and is therefore definitely unlocked, PUNLOCKR loads the least recently used cache sector tag with the 17 most significant bits of the sum.	**PLOCKR xxxx** ;xxxx is twos complement 24-bit integer ;displacement
PFREE	DSP56300 only. Unlocks all the locked cache sectors in the instruction cache.	
PFLUSH/ PFLUSHUN	DSP56300 only. PFLUSH flushes the whole instruction cache, unlocks the cache sectors, and sets the LRU stack and tag registers to their default values. PFLUSHUN flushes the instruction cache which is unlocked, sets the LRU stack to its default value and sets the unlocked tag registers to their default values.	
TRAP/ TRAPcc	TRAP suspends normal instruction execution and begins TRAP exception processing. TRAPcc suspends normal instruction execution and software exception processing is initiated if the specified condition is true. If the specified condition is false, instruction execution continues with the next instruction.	

NOTES:

NOTES:

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Order No.: AN1829
Rev. 1
8/2005