1-1

Motorola DSP Products

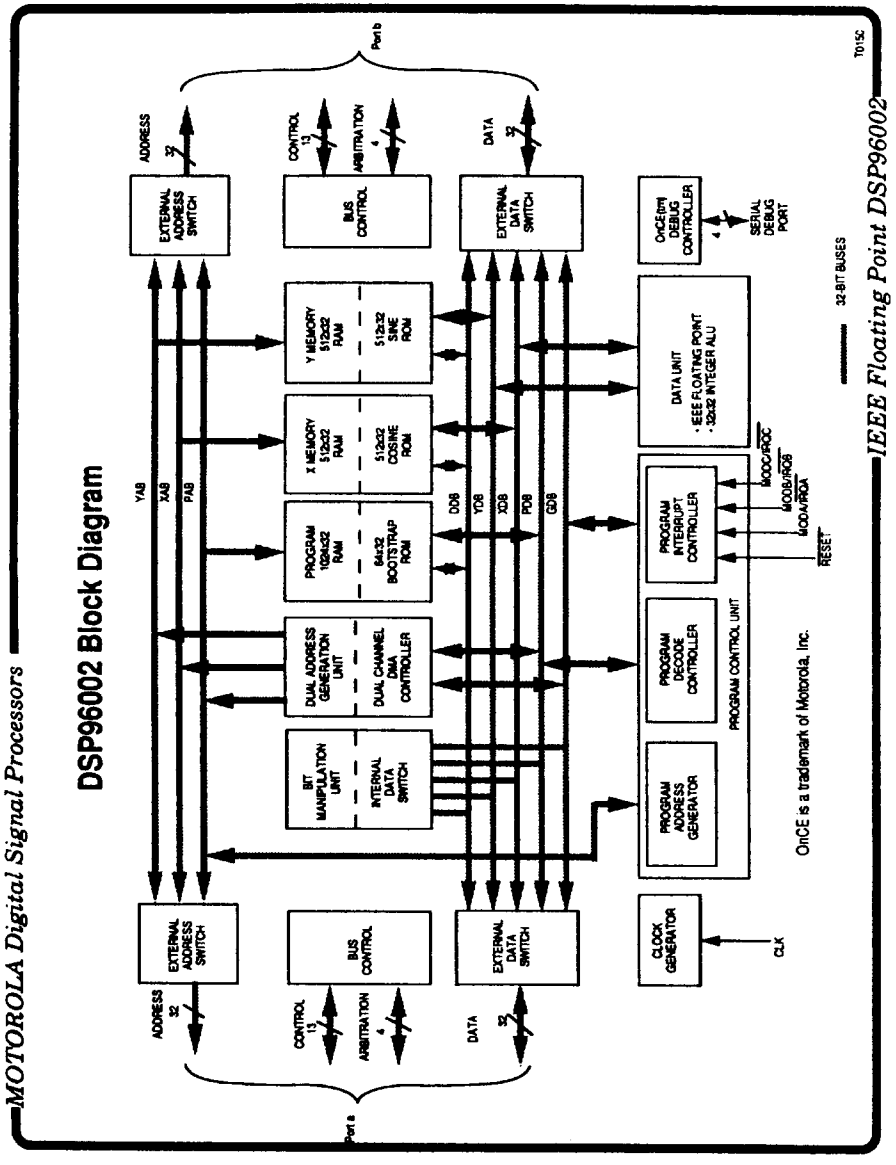
DSP96002

32/96 Bit IEEE Floating Point General Purpose Digital Signal Processor

TERRY SCHULTZ
MOTOROLA
P.C. WORKSTATIONS SEGMENT
ANNAPOLIS CALIF

DSP96002 APPLICATIONS

- ❑ Engineering Workstations
- ❑ Personal Computers
- ❑ Graphics Processing
- ❑ Image Processing
- ❑ Numeric / Array Processing
- ❑ Laser Printers/PostScript Processing
- ❑ Radar / Sonar Processing
- ❑ Spectral Analysis
- ❑ Medical Equipment
- ❑ Digital Audio
- ❑ High Speed Control
- ❑ Speech Processing
- ❑ Instrumentation

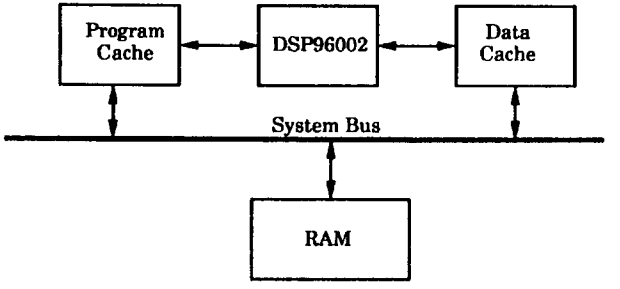


DSP96002 Port A/B Applications

- **Graphics Processor**
  - * Dual buses and use of DRAM page mode achieve a 5X performance improvement over a single bus system without DRAM support.

- **FFT Array Processor**
  - * Data bandwidth doubles with FFT code in internal RAM.

- **Simulation Engine**
  - * Large caches reduce system bus loading.



T0170

DSP96002 Multiprocessor Support

The DSP96002 bus represents a significant advancement in:

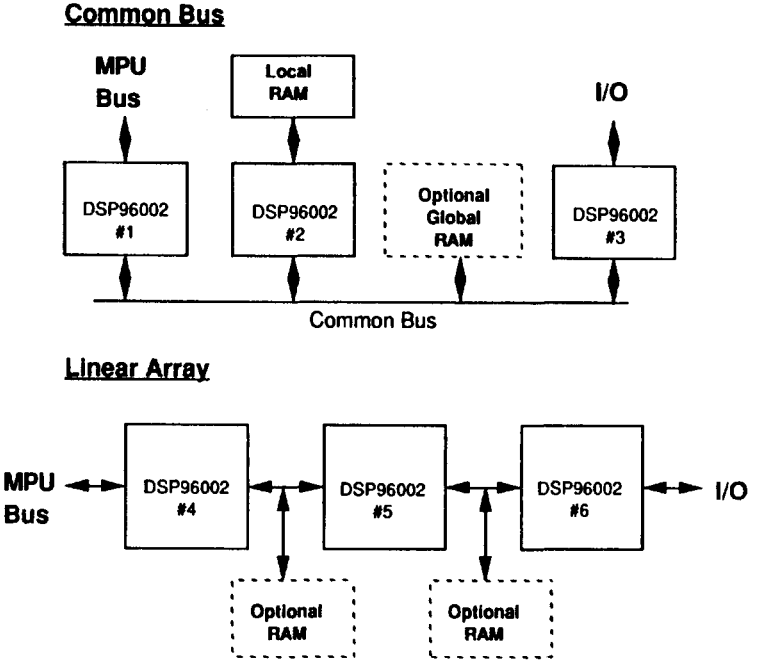
- ***Multiprocessor systems and communications***
- ***DRAM/VRAM support***
- ***Interfacing to 32 bit microprocessors***

DSP96002 user benefits include:

- ***Higher performance, more flexibility***
- ***Fewer support chips, smaller footprint***
- ***Lowest system cost***

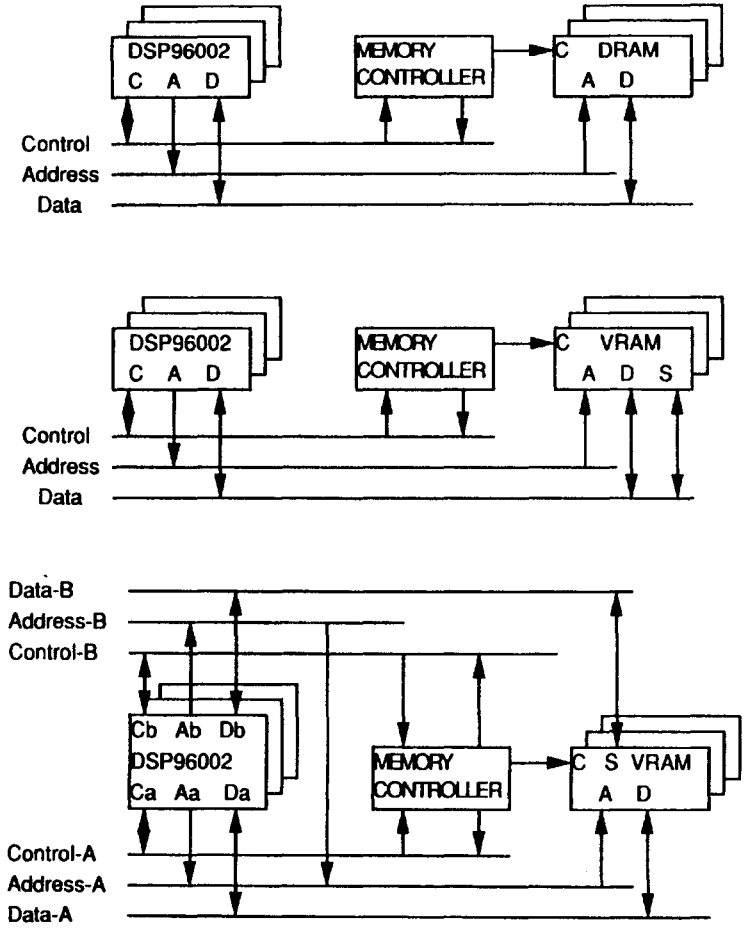
T0180

DSP96002 Multiprocessor Systems



T0190

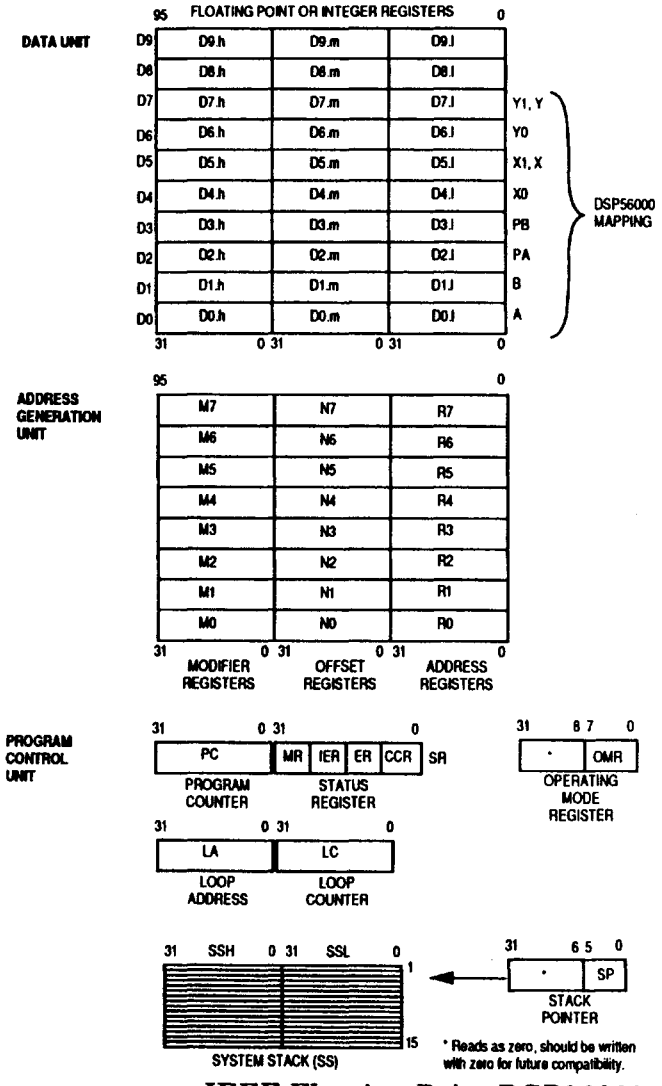
DSP96002 DRAM/VRAM Systems



DSP96002 Dual Channel Direct Memory Access (DMA) Controller

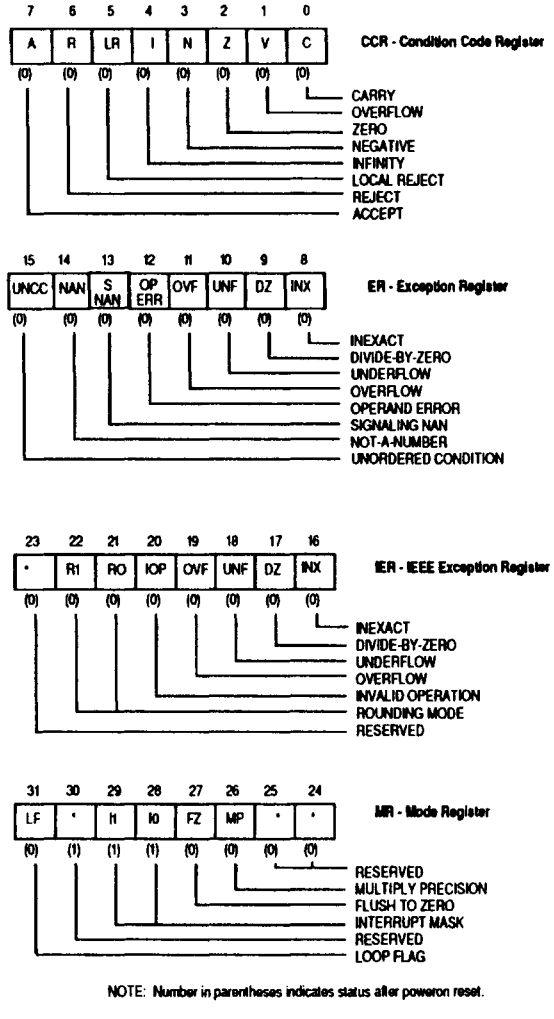
- **Two Independent DMA Channels**
- **Generic Data Transfers**
  * Memory to Memory, Including Memory Mapped Peripherals
  * Handshake Modes for Data Throttling and Peripherals
  * Internal and/or External
  * Port A and/or B sources and destinations.
- **Linear, Modulo or Bit Reversed DMA Addressing**
  * Shares Addressing Units with CPU
  * Separate R, N and M registers for Source and Destination (Each Channel)
- **High Performance in Parallel with the CPU**
  * Transfer Rates up to 54 Mbytes per Second
  * Port to Port Transfers Transparent to the CPU
  * DMA Wait States Do Not Slow Down the CPU
- **Dual Access Internal Memories**
  * P, X and Y RAM and ROM - DMA Access Transparent to the CPU
  * One CPU Time Slot per Instruction Cycle
  * One DMA Time Slot per Instruction Cycle
- **Programmable Bus Access Priority**
  * If Both CPU and DMA want the Same External Bus
  * If Both CPU and DMA want the Global Data Bus

DSP96002 User Programming Model



T0210

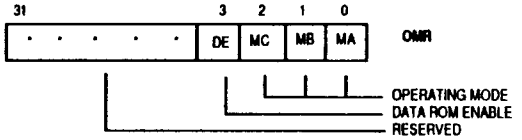
DSP96002 Status Register Format



NOTE: Number in parentheses indicates status after poweron reset.

T0220

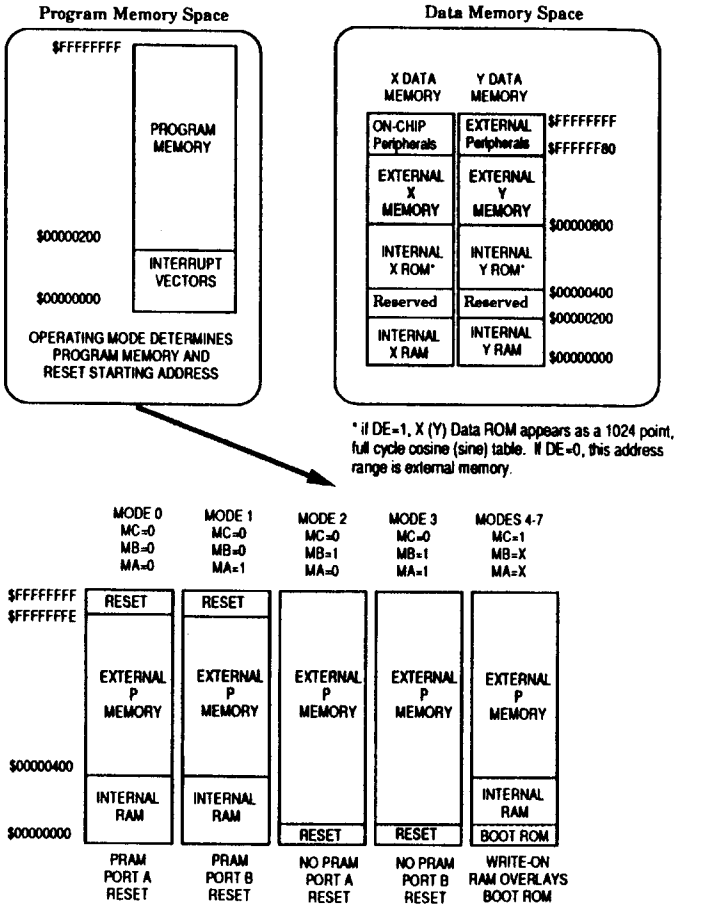
DSP96002 Operating Mode Register



DSP96002 Bootstrap Modes

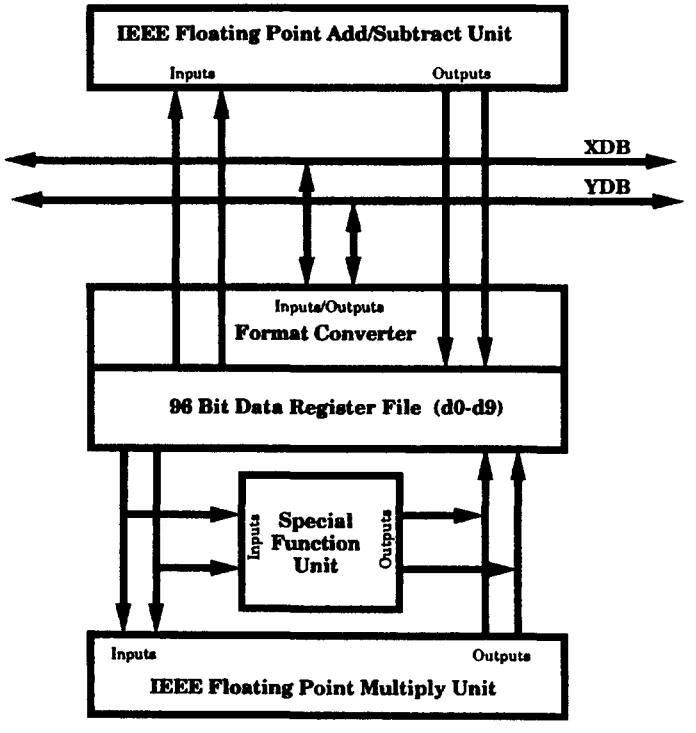
- MA, MB and MC mode bits are latched from MODA, MODB and MODC pins when RESET is negated.
- If MC=1, an internal self-bootstrap program begins executing from the bootstrap ROM.
- MA and MB are read by the self-bootstrap program to select the bootstrap source (for example, an external byte-wide EPROM).
- User code is moved from the bootstrap source to the Program RAM.
- Self-bootstrap ends by jumping to the user code in Program RAM.

DSP96002 Memory Maps



DSP96002 Data ALU Block Diagram

- IEEE Floating Point Add/Subtract Unit
- IEEE Floating Point Multiply Unit
- Special Function Unit
  - * Logical
  - * Divide
  - * Square Root



T0250

IEEE Floating Point Numbers

Numbers of the form: $(-1)^{s}\ 2^{E}\ (b_0.b_1b_2b_3\ldots b_{p-1})$

Where:
- s = 0 or 1
- E = any integer between Emin and Emax, inclusive
- b_i = 0 or 1 (b_0 is a hidden integer bit)

Encodings for:
- +Zero and -Zero
- +Infinity and -Infinity
- Denormalized Numbers (E=Emin, unnormalized mantissa)
- At least one signaling Not-a-Number (NaN)
- At least one quiet NaN

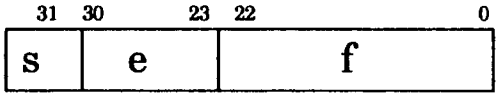
IEEE Floating Point Precisions

Parameter	Single (SP)	Single Extended (SEP)	Double (DP)	Double Extended (DEP)
Mantissa width P	24	>=32	53	>=64
Exponent maximum Emax	+127	>=+1023	+1023	>=+16383
Exponent minimum Emin	-126	<=-1022	-1022	<=-16382
Exponent bias Ebias	+127	Unspecified	+1023	Unspecified
Exponent width	8	>=11	11	>=15
Format width	32	>=43	64	>=79
Implementation	Required	Optional	Optional	Optional
DSP96002 Data ALU Formats	Yes	Yes	No	No
DSP96002 Move Formats	Yes	(Yes)	Yes	(Yes)

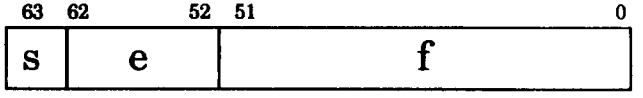
T0251

IEEE Standard Storage Formats
Bit Level Conformance Required

- SP Format (e=E+127)



- DP Format (e=E+1023)



- SEP and DEP (extended) storage formats are not specfied by IEEE and are typically not stored in memory.

Why Use Extended Precision?

- Greater precision and dynamic range for improved accuracy in intermediate calculations
  * Expression evaluation
  * Inner products

- SEP allows efficient computation of IEEE SP math library functions
  * Transcendentals
  * Trigonometric argument reduction
  * Power function: Y^X

T0252

Advantages of
IEEE Floating Point

Superior Compatibility

- Standard storage format for data exchange with other processors.
- Numerically identical results across many implementations (coprocessors, data path chips and integrated floating point chips).
- Ease of porting high-level language software and test files using IEEE floating point.
- Interface to IEEE 754 format data bases (graphics) is easy.

Superior Mathematical Accuracy

- Guaranteed error bounds for add, subtract, compare, multiply, divide, remainder, square root, and conversions between fixed and floating point formats.
- Support of denormalized numbers for gradual underflow.
- Support of four rounding modes:
  * Round to nearest (even)
  * Round to zero
  * Round to plus infinity
  * Round to minus infinity

Superior Error Handling

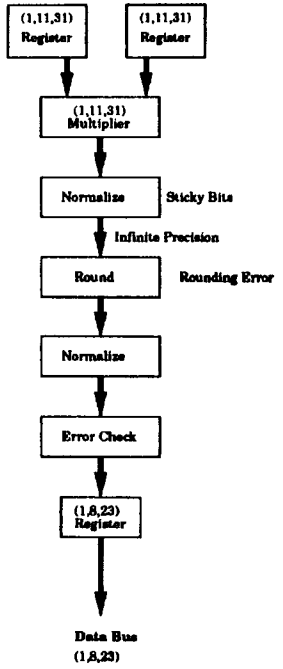
- Standard floating poing exception processing.
- Standard comparison predicates and relations for conditional branches.
- Support of quiet and signalling Not-a-Numbers (NaN).

T0253

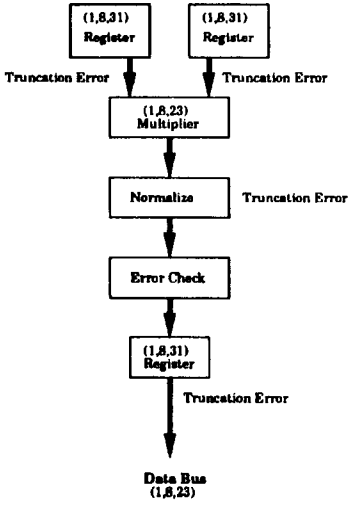
Floating Point Multiplication
Single Precision Accuracy Comparison

(Sign, Exponent, Mantissa) in bits

Motorola (IEEE Standard)



Competition (Proprietary)



IEEE: 1 Rounding Error

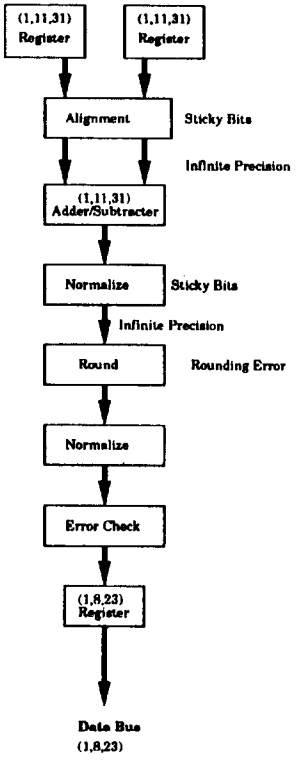
Proprietary: 4 Truncation Errors

T0254-1

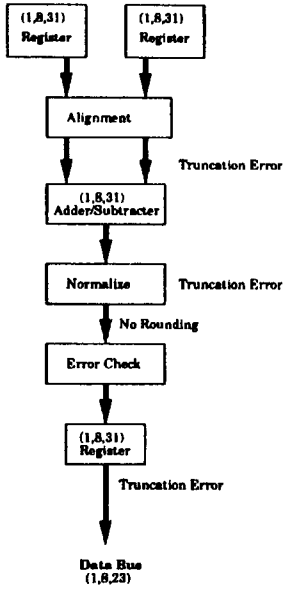
Floating Point Addition/Subtraction
Single Precision Accuracy Comparison

(Sign, Exponent, Mantissa) in bits

Motorola (IEEE Standard)



Competition (Proprietary)



IEEE: 1 Rounding Error

Proprietary: 3 Truncation Errors

T0254-2

DSP96002 Underflow Modes

Floating Point underflow occurs when the exponent of a normalized result is less than the format minimum. Thus, the underflowed result can not be stored as a normalized number.

- **IEEE Gradual Underflow Mode**
  - * The mantissa is denormalized until the exponent is equal to the format minimum.

s	e=0	f=.000110101

  - * If the exponent is zero, the hidden integer bit is also 0.
  - * If the denormalized mantissa is all zeros, the result is floating point zero.
  - * Special DSP96002 hardware detects denormalized numbers (including floating point zero) and inserts one additional instruction cycle to process each denormalized number. The additional cycles are data-dependent.
- **Non-IEEE Flush to Zero Mode**
  - * All floating point underflows are forced to floating point zero.

s	e=0	f=0

  - * No additional cycles (data-independent).

T0255

DSP96002 Exception Processing

- **The IEEE floating point standard defines five types of exceptions:**
  - * Invalid Operation (0xInfinity, 0/0, Infinity/Infinity)
  - * Divide by Zero
  - * Overflow
  - * Underflow
  - * Inexact
- **The IEEE standard also defines two responses for each floating point exception:**
  - * Trap-disabled exception processing (default)
    - Defines result to be delivered
    - Defines how status flags record the error
  - * Trap-enabled exception processing (optional)
    - Defines result to be delivered
    - Defines interface to trap handler
- **The DSP96002 implements IEEE trap-disabled exception processing in hardware:**
  - * "Sticky" status flags remember the error
  - * Real-time systems must deliver a result
  - * No additional cycles

T0254

DSP96002 Rounding Control

To meet IEEE error bounds, floating-point arithmetic is done to infinite precision with one final rounding error to the result precision.

- **IEEE Rounding Modes**
  - * Round to Nearest (Even) (default)
  - * Round to Zero
  - * Round to Plus Infinity
  - * Round to Minus Infinity

- **Three Instruction Classes**
  - * Single Precision Rounding

    fadd.s d1,d0
  - * Single Extended Precision Rounding

    fadd.x d1,d0
  - * No Rounding

    fcmp d1,d0

T0270

DSP96002 Opcode Summary

- **Floating-Point Operations**

fabs	Absolute Value
fadd	Add
faddsub	Add and Subtract
fclr	Clear
fcmp	Compare
fcmpg	Compare for Graphics
fcmpm	Compare Magnitude
fcopys	Copy Sign
fgetman	Get Mantissa
fint	Round to Integer
float	Convert Integer to Float
floatu	Convert Unsigned Integer to Float
floor	Round to Smallest Integer
fmpy	Multiply
fmpy // fadd	Multiply and Add
fmpy // faddsub	Multiply, Add and Subtract
fmpy // fsub	Multiply and Subtract
fneg	Negate
fscale	Scale (Add Exponent)
fseedd	Reciprocal Seed
fseedr	Reciprocal Square Root Seed
fsub	Subtract
ftfr	Transfer Register
ftst	Test

- **Integer Operations**

abs	Absolute Value
addc	Add with Carry
add	Add
asl	Arithmetic Shift Left
asr	Arithmetic Shift Right
bfind	Bit Find First One
clr	Clear
cmp	Compare
cmpg	Compare for Graphics
dec	Decrement
ext	Sign Extend Half-Word
extb	Sign Extend Byte
getexp	Get Exponent
inc	Increment
int	Convert Float to Integer
intrz	Convert Float to Integer, Round to Zero
intu	Convert Float to Unsigned Integer
inturz	Convert Float to Unsigned Integer, Round to Zero
mpy	Signed Multiply

DSP96002 Opcode Summary

- **Integer Operations (continued)**

mpyu	Unsigned Multiply
neg	Negate
negc	Negate with Carry
setr	Set Register
sub	Subtract
subc	Subtract with Carry
tfr	Transfer Register
tst	Test

- **Logical Operations**

and	AND
andc	AND with Complement
andi	AND with Control Register
btst	Bit Test
eor	Exclusive OR
join	Join Half-Words
joinb	Join Bytes
lsl	Logical Shift Left
lsr	Logical Shift Right
not	Complement
or	OR
orc	OR with Complement
ori	OR with Control Register
rol	Rotate Left
ror	Rotate Right
split	Split Half-Words
splitb	Split Bytes

- **Bit Manipulation Operations (Read-Modify-Write)**

bchg	Bit Test and Change
bclr	Bit Test and Clear
bset	Bit Test and Set

- **Move Operations**

move	Move
movec	Move Control Register
movep	Move Peripheral
moveta	Move and Test Address Register
lea	Load Effective Address
lra	Load PC Relative Address

- **Loop Control Operations**

do	Start Hardware Loop
dor	Start PC Relative Hardware Loop
enddo	End Current Loop
rep	Repeat Next Instruction

DSP96002 Opcode Summary

- **Program Control Operations**

bcc	Branch on Condition cc
brclr	Branch if Bit Clear
brset	Branch if Bit Set
bscc	Branch to Subroutine on Condition cc
bsclr	Branch to Subroutine if Bit Clear
bsset	Branch to Subroutine if Bit Set
fbcc	Float Branch on Condition cc
fbscc	Float Branch to Subroutine on Condition cc
ftrapcc	Float Trap on Condition cc
illegal	Illegal Instruction Trap
jcc	Jump on Condition cc
jclr	Jump if Bit Clear
jscc	Jump to Subroutine on Condition cc
jsclr	Jump to Subroutine if Bit Clear
jset	Jump if Bit Set
jsset	Jump to Subroutine if Bit Set
nop	No Operation
reset	Reset On-Chip Peripherals
rti	Return From Interrupt
rtr	Restore From Subroutine
rts	Return From Subroutine
stop	Stop Processing (Low Power Standby)
trapcc	Trap on Condition cc
wait	Wait for Interrupt (Low Power Standby)

DSP96002 Parallel Operations

Instruction Syntax

OPCODE (OPERANDS) (OPCODE OPERANDS) (PARALLEL MOVE) (PARALLEL MOVE)

Single Precision Parallel Moves	Example
None	fmpy d4,d5,d0 fadd.s d6,d1
Register	fmpy d4,d5,d0 fadd.s d6,d1 d1.s,d4.s
Update	fmpy d4,d5,d0 fadd.s d6,d1 (r0)+
X memory	fmpy d4,d5,d0 fadd.s d6,d1 x:(r7)-,d6.s
X memory and register	fmpy d4,d5,d0 fadd.s d6,d1 d3.s,x:(r2)+ d7.s,d3.s
Y memory	fmpy d4,d5,d0 fadd.s d6,d1 d0.l,y:(r0+$43)
Y memory and register	fmpy d4,d5,d0 fadd.s d6,d1 d2.s,d7.s y:(r6)+n6,d3.s
X and Y memory	fmpy d4,d5,d0 fadd.s d6,d1 x:(r3)+,d4.s d1.s,y:(r4)-
	fmpy d4,d5,d0 fadd.s d6,d1 x:(r5),d4.s y:,d5.s
Double Precision Parallel Moves	
Register	fmpy d4,d5,d0 fadd.s d6,d1 d2.d,d3.d
L(X:Y) memory	fmpy d4,d5,d0 fadd.s d6,d1 d2.d,l:(r0)-n0
Conditional Operations	
Integer cc, no CCR update	fmpy d4,d5,d0 fadd.s d6,d1 ifcs
Float cc, no CCR update	fmpy d4,d5,d0 fadd.s d6,d1 fflt
Integer cc, CCR update	lsl d1 ifcs.u
Float cc, CCR Update	fftr.s d0,d1 ffgt.u

T0300

MOTOROLA Digital Signal Processors

DSP96002 Parallel Operations

Instruction Syntax

OPCODE (OPERANDS) (OPCODE OPERANDS) (PARALLEL MOVE) (PARALLEL MOVE)

Single Precision Parallel Moves Example

None fmpy d4,d5,d0 fadd.s d6,d1
Register fmpy d4,d5,d0 fadd.s d6,d1 d1.s,d4.s
Update fmpy d4,d5,d0 fadd.s d6,d1 (r0)+
X memory fmpy d4,d5,d0 fadd.s d6,d1 x:(r7)-,d6.s
X memory and register fmpy d4,d5,d0 fadd.s d6,d1 d3.s,x:(r2)+ d7.s,d3.s
Y memory fmpy d4,d5,d0 fadd.s d6,d1 d0.l,y:(r0+$43)
Y memory and register fmpy d4,d5,d0 fadd.s d6,d1 d2.s,d7.s y:(r6)+n6,d3.s
X and Y memory fmpy d4,d5,d0 fadd.s d6,d1 x:(r3)+,d4.s d1.s,y:(r4)-
fmpy d4,d5,d0 fadd.s d6,d1 x:(r5),d4.s y:,d5.s

Double Precision Parallel Moves
Register fmpy d4,d5,d0 fadd.s d6,d1 d2.d,d3.d
L(X:Y) memory fmpy d4,d5,d0 fadd.s d6,d1 d2.d,l:(r0)-n0

Conditional Operations
Integer cc, no CCR update fmpy d4,d5,d0 fadd.s d6,d1 ifcs
Float cc, no CCR update fmpy d4,d5,d0 fadd.s d6,d1 fflt
Integer cc, CCR update lsl d1 ifcs.u
Float cc, CCR Update fftr.s d0,d1 ffgt.u

T0300

IEEE Floating Point DSP96002

DSP96002 Parallel Operations

Instruction Syntax

OPCODE (OPERANDS) (OPCODE OPERANDS) (PARALLEL MOVE) (PARALLEL MOVE)

Single Precision Parallel Moves	Example
None	fmpy d4,d5,d0 fadd.s d6,d1
Register	fmpy d4,d5,d0 fadd.s d6,d1 d1.s,d4.s
Update	fmpy d4,d5,d0 fadd.s d6,d1 (r0)+
X memory	fmpy d4,d5,d0 fadd.s d6,d1 x:(r7)-,d6.s
X memory and register	fmpy d4,d5,d0 fadd.s d6,d1 d3.s,x:(r2)+ d7.s,d3.s
Y memory	fmpy d4,d5,d0 fadd.s d6,d1 d0.l,y:(r0+$43)
Y memory and register	fmpy d4,d5,d0 fadd.s d6,d1 d2.s,d7.s y:(r6)+n6,d3.s
X and Y memory	fmpy d4,d5,d0 fadd.s d6,d1 x:(r3)+,d4.s d1.s,y:(r4)-
	fmpy d4,d5,d0 fadd.s d6,d1 x:(r5),d4.s y:,d5.s
Double Precision Parallel Moves	
Register	fmpy d4,d5,d0 fadd.s d6,d1 d2.d,d3.d
L(X:Y) memory	fmpy d4,d5,d0 fadd.s d6,d1 d2.d,l:(r0)-n0
Conditional Operations	
Integer cc, no CCR update	fmpy d4,d5,d0 fadd.s d6,d1 ifcs
Float cc, no CCR update	fmpy d4,d5,d0 fadd.s d6,d1 fflt
Integer cc, CCR update	lsl d1 ifcs.u
Float cc, CCR Update	fttr.s d0,d1 ffgt.u

T0300

DSP96002 Addressing Mode Summary

Addressing Mode	Modifiers	Assembler Syntax
Register Direct		
Low data register	no	Dn.l
Middle data register	no	Dn.m
High data register	no	Dn.h
Single precision integer	no	Dn.l
Double precision integer	no	Dn.ml
Single precision float	no	Dn.s
Double precision float	no	Dn.d
Address or Control	no	any register name
Address Register Indirect		
No update	yes	(Rn)
Post-increment by 1	yes	(Rn)+
Post-decrement by 1	yes	(Rn)-
Post-increment by offset Nn	yes	(Rn)+Nn
Post-decrement by offset Nn	yes	(Rn)-Nn
Indexed by offset Nn	yes	(Rn+Nn)
Indexed by displacement	yes	(Rn+expr)
Pre-decrement by 1	yes	-(Rn)
PC Relative Indirect		
15 bit short displacement	no	expr
32 bit long displacement	no	expr
Address register displacement	no	Rn
Special		
16 bit immediate short data	no	#expr
32 bit immediate long data	no	#expr
8 bit absolute short address	no	expr
12 bit short jump address	no	expr
32 bit absolute long address	no	expr
Implicit	no	

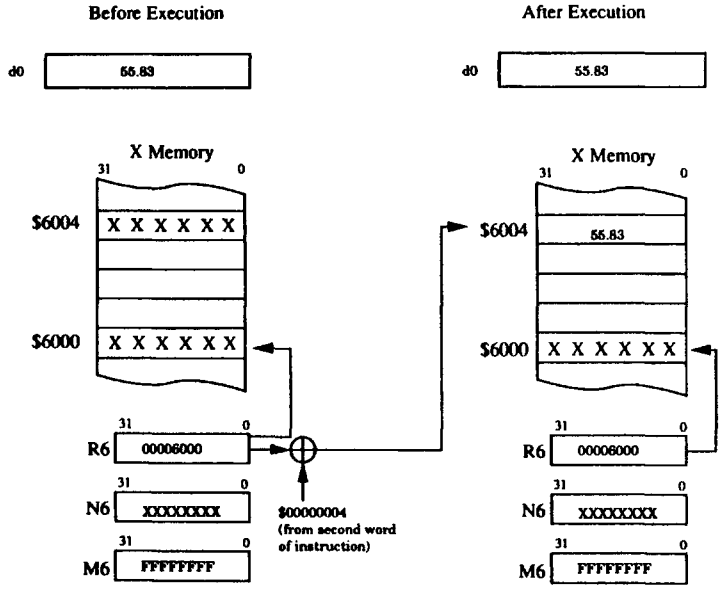
Where
- Dn is data register n = 0-9
- Rn is address register n = 0-7
- Nn is offset register n = 0-7
- expr is any valid assembler expression

T0315

Displacement Addressing Mode

- Assembler Syntax: (Rn + $AAAAAAAA)
- Operands referenced: P, X, Y, L memories.
- Additional instruction execution time (clocks):
  4
- Additional effective address words: 1

Example: MOVE d0.s,X:(R6+$4)

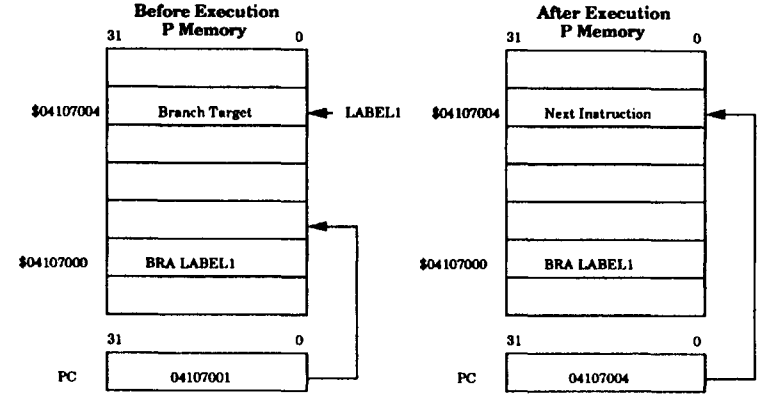


T0320

PC Relative Addressing Mode - Branches

- Assembler Syntax:
  BRA offset
- Operands referenced: P Memory
- Additional instruction execution time (clocks): 4 (15 bit signed offset)
  6 (32 bit offset)
- Additional effective address words: 0 (15 bit signed offset)
  1 (32 bit offset)

Example: BRA LABEL1



T0330

PC Relative Addressing Mode - Load Relative Address

- Assembler Syntax:
  LRA (PC + $AAAAAAAA),reg ;static offset
  LRA (PC + Rn),reg ; dynamic offset
- Additional instruction execution time (clocks): 4 Static, 2 Dynamic
- Additional effective address words: 1 static, 0 dynamic

Example:

$00000100	LRA (PC+$150),d4.l	**d4.l = $102 + $150 = $252**
$00000101	$00000150	
$00000102	xxxxxxxx	

Example:

$00000100	LRA (PC+r1),r4	r1 = $150
$00000101	xxxxxxxx	**r4 = $101 + $150 = $251**
$00000102	xxxxxxxx	

MOTOROLA Digital Signal Processors

PC Relative Addressing Mode - Load Relative Address

- Assembler Syntax:
  LRA (PC + $AAAAAAAA),reg ;static offset
  LRA (PC + Rn),reg ; dynamic offset
- Additional instruction execution time (clocks): 4 Static, 2 Dynamic
- Additional effective address words: 1 static, 0 dynamic

Example:

$00000100	LRA (PC+$150),d4.l	d4.l = $102 + $150 = $252
$00000101	$00000150	
$00000102	xxxxxxxx	

Example:

$00000100	LRA (PC+r1),r4	r1 = $150
$00000101	xxxxxxxx	r4 = $101 + $150 = $251
$00000102	xxxxxxxx	

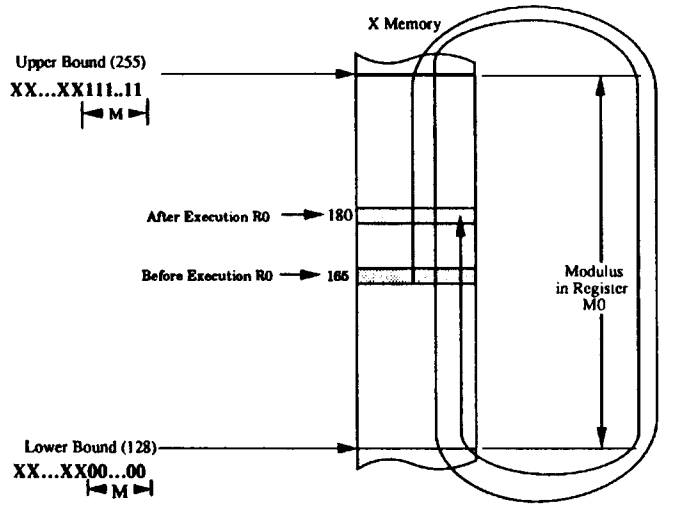
T0340
IEEE Floating Point DSP96002

Multiple Wraparound Modulo 2^M Addressing

1. Modifier Register Mn = $FFmm mmmm, where mm mmmm = bit mask.
2. Lower Bound = 31 XX...XX00...00 0 (M) where M = number of lsb's set in Mn mask.
3. Upper Bound = 31 XX...XX11..11 (M)
4. Lower Bound ≤ Address Register Rn ≤ Upper Bound
5. Offset Register Nn = any value

Example: MOVE d0.s,X:(R0)+N0

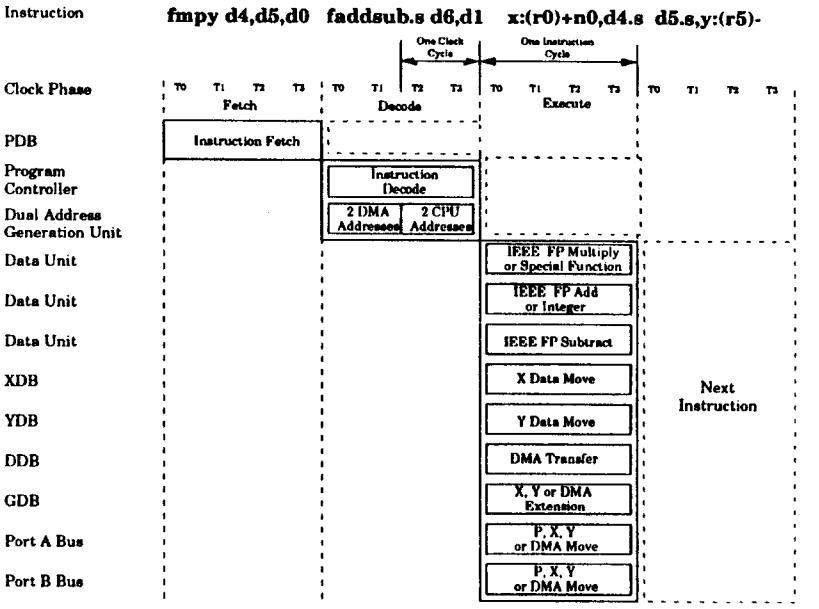
Starting Address		Increment		Modulo	
R0	165	N0	271	M0	$FF00 003F



Applications: Waveform Generation, Modulation

T0355

Pipeline Timing For One DSP96002 Instruction



Number of Instruction Words:	**1**
Execution Time:	**50 ns**
Peak Floating Point:	**60 MFLOPS**
Millions of Addresses per Second	**80**
Mbytes per Second DMA Data Transfer:	**80**

T0357

DSP96002 Benchmark Summary

Benchmark	Instruction Cycles	Execution Time (40 MHz, 1990)	MIPS	SP/SEP MFLOPS
• Real				
v = v + s	2N	-	20.0	10.0
v = v + v	2N	-	20.0	10.0
v = v * v	2N	-	20.0	10.0
v = v * s + v	2N	-	20.0	20.0
v = v * v + s	2N	-	20.0	20.0
v = v * v + v	2N	-	20.0	20.0
Polynomial Expansion	2N	-	20.0	29.9
Convolution or Correlation	1N	-	20.0	40.0
FIR Filter With Data Shift	1N	50 ns/tap	20.0	40.0
Biquad Filter With Data Shift	4N	200 ns/stage	20.0	40.0
Lattice Filter With Data Shift	3N	150 ns/stage	20.0	26.7
[10x10][10x10] = [10x10]	1583	79 us	18.5	29.0
1024 Point FFT With Bit Reversal	12880	644 us	19.1	35.0
• Complex				
v = v * v	4N	-	20.0	29.9
v = v * v + v	4N	-	20.0	40.0
Convolution or Correlation	4N	-	20.0	40.0
FIR Filter With Data Shift	4N	200 ns/tap	20.0	40.0
1024 Point FFT With Bit Reversal	20931	944 us	19.8	43.1
• Graphics/Image Processing				
Divide				
32 Bit Accuracy	7	0.30 us	20.0	23.3
16 Bit Accuracy	5	0.20 us	20.0	20.0
8 Bit Accuracy	2	0.10 us	20.0	20.0
Square Root				
32 Bit Accuracy	12	0.50 us	20.0	22.0
16 Bit Accuracy	8	0.30 us	20.0	20.0
8 Bit Accuracy	2	0.10 us	20.0	20.0
Transformations				
[1x3][3x3] = [1x3]	12	0.60 us	20.0	25.1
[3x3][3x3] = [3x3]	30	1.50 us	20.0	29.9
[1x4][4x4] = [1x4]	19	0.85 us	20.0	32.9
[4x4][4x4] = [4x4]	67	3.35 us	20.0	33.5
3D Volume Compare For Trivial Accept/Reject				
1 Point	8	0.40 us	20.0	15.0
2 Point Polyline	14	0.70 us	20.0	17.2
N Point Polygon	6N	-	20.0	20.0
Bezier Cubic Evaluation For Font Compilation	13	0.65 us	20.0	21.5
BITBLT (Bit Block Transfer), N 32 Bit Words, Replacement	3N	213 Mbits/sec	20.0	-
2D [3x3] Image Convolution	9N	-	20.0	37.8

s = scalar, v = vector, n=number of data points

T04166