ANALOG DEVICES

32-Bit IEEE Floating-Point Chipset

ADSP-3201/ADSP-3202

FEATURES

Complete Chipset Implementing Floating-Point Arithmetic
Fully Compatible with IEEE Standard 754
Arithmetic Operations on Three Data Formats:
32-Bit Single-Precision Floating Point
32-Bit Twos-Complement Fixed-Point
32-Bit Unsigned-Magnitude Fixed-Point
Pin-Compatible Single-Precision Versions of the ADSP-3211 Multiplier and ADSP-3221 ALU
Only One Internal Pipeline Stage
Single-Precision and Fixed-Point Multiplier and ALU Pipelined Throughput Rates to 10 MFLOPS
Low Latency for Scalar Operations
240ns for 32-Bit Multiplier and ALU Operations
IEEE Divide and Square Root
Either One or Two Input-Port Configuration Modes
750mW Maximum Power Dissipation per Chip with 1.5μm CMOS Technology
144-Lead Pin Grid Array
Available Specified to MIL-STD-883, Class B

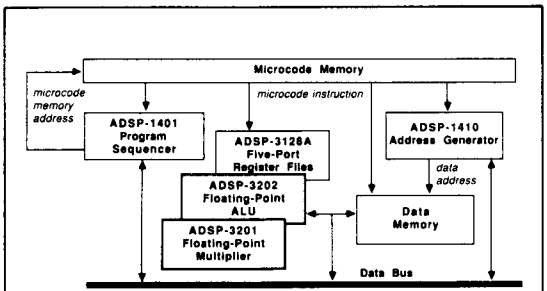
APPLICATIONS

High-Performance Digital Signal Processing
Floating-Point Accelerators
Array Processors
Graphics Numerics Processors

GENERAL DESCRIPTION

The ADSP-3201 Floating-Point Multiplier and the ADSP-3202 Floating-Point ALU are high-speed, low-power, 32-bit arithmetic processors conforming to IEEE Standard 754. This low-cost chipset comprises the basic computational elements for implementing a high-speed, single-precision numeric processor. Operations are supported on three data formats: 32-bit IEEE single-precision floating-point, 32-bit twos-complement fixed-point, and 32-bit unsigned-magnitude fixed-point.

The high throughput of these CMOS chips is achieved with only a single level of internal pipelining, greatly simplifying program development. Theoretical MFLOPS rates are much easier to approach in actual systems with this chip architecture than with alternative, more heavily pipelined chipsets. Also, the minimal internal pipelining in the ADSP-3201/3202 results in very low latency, important in scalar processing and in algorithms with data dependencies. To further reduce latency, input registers can be read into the chips' internal computational circuits at the rising edge that loads them from the input port (formerly called "direct operand feed").



Word-Slice® Microcoded System with ADSP-3201/3202

4

In conforming to IEEE Standard 754, these chips assure complete software portability for computational algorithms adhering to the Standard. All four rounding modes are supported for all floating-point data formats and conversions. Five IEEE exception conditions – overflow, underflow, invalid operation, inexact result, and division by zero – are available externally on status pins. The IEEE gradual underflow provisions are also supported, with special instructions for handling denormals. Alternatively, each chip offers a FAST mode which sets results less than the smallest IEEE normalized values to zero, thereby eliminating underflow exception handling when full conformance to the Standard is not essential.

The instruction sets of the ADSP-3201/3202 are oriented to system-level implementations of function calculations. Specific instructions are included to facilitate such operations as floating-point divide and square root, table lookup, quadrant normalization for trig functions, extended-precision integer operations, logical operations, and conversions between all data formats.

The ADSP-3201 Floating-Point Multiplier is a pin-compatible, 32-bit version of the 144-lead ADSP-3211 Floating-Point Multiplier. Like the ADSP-3211, it has two input ports and eight input registers. It executes all ADSP-3210 and ADSP-3211 32-bit operations. The ADSP-3201 supports twos-complement, unsigned-magnitude, and mixed-mode 32-bit fixed-point multiplications.

The ADSP-3202 Floating-Point ALU is a pin-compatible, 32-bit version of the 144-lead ADSP-3221 Floating-Point ALU. Like the ADSP-3211, it has two input ports and eight input registers. It executes all ADSP-3220 and ADSP-3221 32-bit operations, including IEEE division and square root.

The ADSP-3201/3202 chipset is fabricated in double-metal 1.5μm CMOS. Each chip consumes 750mW maximum, significantly less than comparable bipolar solutions. The differential between the chipset's junction temperature and the ambient temperature stays small because of this low-power dissipation. Thus the ADSP-3201/3202 can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-3201/3202 are available for both commercial and extended temperature ranges. Extended temperature range parts are available processed fully to MIL-STD-883, Class B. The ADSP-3201 and ADSP-3202 are packaged in ceramic 144-lead pin grid arrays.

TABLE OF CONTENTS **PAGE**

GENERAL DESCRIPTION 4–5
FUNCTIONAL DESCRIPTION OVERVIEW 4–6
PIN DEFINITIONS AND FUNCTIONAL BLOCK DIAGRAMS 4–8
METHOD OF OPERATION 4–10
DATA FORMATS 4–10
Single-Precision Floating-Point Data Format 4–10
Supported Floating-Point Data Types 4–11
32-Bit Fixed-Point Data Formats 4–11
CONTROLS 4–12
FAST/IEEE CONTROL 4–13
RESET CONTROL 4–13
PORT CONFIGURATION – IPORT CONTROLS . . 4–13
INPUT REGISTER LOADING AND OPERAND STORAGE – SELA/B CONTROLS 4–14
DATA FORMAT SELECTION – SP CONTROL . . 4–14
INPUT DATA REGISTER READ SELECTION – RDA/B CONTROLS 4–14
ABSOLUTE VALUE CONTROLS – ABSA/B 4–15
WRAPPED INPUT CONTROLS – WRAPA/B (and INEXIN and RNDCARI on the ADSP-3202) . 4–15
TWOS-COMPLEMENT INPUT CONTROLS – TCA/B (ADSP-3201) 4–15
ROUNDING – RND CONTROLS 4–15
STATUS FLAGS 4–16
Denormal Input 4–17
Invalid Operation and NAN Results 4–17
Division-by-Zero 4–17
Overflow 4–17
Underflow 4–17
Inexact 4–18
Less Than, Equal, Greater Than, Unordered 4–18
Special Flags for Unwrapping 4–18
INSTRUCTIONS AND OPERATIONS 4–19
Fixed-Point Arithmetic ALU Operations 4–20
Logical ALU Operations 4–21
Floating-Point ALU Operations 4–21
OUTPUT CONTROL – SHLP, OEN, MSWSEL, and HOLD 4–23
TIMING 4–23
GRADUAL UNDERFLOW 4–24
SPECIFICATIONS 4–34
ORDERING INFORMATION 4–35
PINOUTS 4–36

FUNCTIONAL DESCRIPTION OVERVIEW

The ADSP-3201/3202 share a common architecture (Figure 1) in which all input data is loaded to a set of input registers with both rising and falling clock edges. These registers can be read to the chip's computational circuitry as they are loaded on a rising edge. At the end of first processing clock cycle, partial results and most controls are clocked into a set of internal pipeline registers. In most cases, only a second clock cycle is required to conclude processing. (The exceptions are division and square root.) At the end of this second processing cycle, results are clocked into an output register. The contents of the output register can then be driven off-chip. An output multiplexer allows driving both halves of a 64-bit fixed-point multiplication result off-chip through the 32-bit output port in one output cycle.

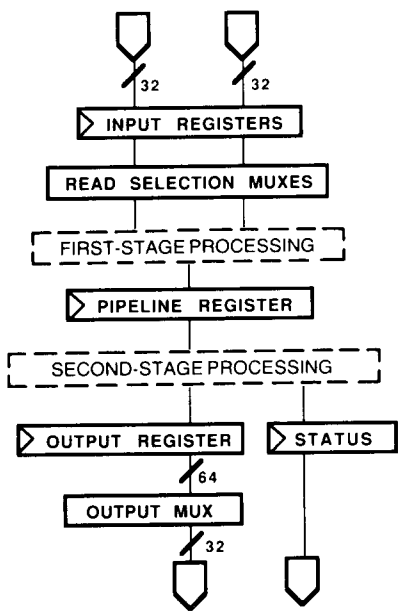


Figure 1. ADSP-3201/3202 Generic Architecture

Because all input and output data is internally registered and because of the single level of internal pipeline registers, operations can be overlapped for high levels of pipelined throughput. Figure 2 illustrates a typical sequence of pipelined operations. Note cycle #4 of Figure 2 after the data transfer and internal pipelines are full. While the final A results of the first operation are being driven off-chip, B processing can be concluding at the second

stage, C processing beginning at the first stage, and D data loading to the input registers.

All three-port members of this chipset can be configured for two-port operations, thereby reducing system busing requirements. However configured, the ADSP-3201/3202 can load data on rising edges of the clock and on falling edges of the clock, subject to constraints described in "Method of Operation." The port configuration chosen determines which registers load data on which edges. All input registers have their own independent load selection controls, allowing the same data to be loaded to multiple registers simultaneously.

A set of read selection multiplexers feeds input data from the input registers to the computational circuitry. These muxes can select data that was just loaded at the clock's rising edge ("direct operand feed"), if desired, with no throughput or cycle-time penalty.

All control signals need only be supplied to the chips at their cycle rate. This approach avoids requiring that the sequencing control cycle time be faster than the chipset's major processing cycle rate. Less expensive microcode memory can therefore be used. For this reason, load selection controls for registers to be loaded on the clock's falling edge need only be valid at the previous rising edge. (The designer may choose to supply the asynchronous output multiplexer and tristate controls at a higher rate, however.)

The ADSP-3201/3202 fully supports the gradual underflow provisions of IEEE Standard 754 for floating-point arithmetic. The Floating-Point ALU can operate directly on both normals and denormals, except in division and square root. The Floating-Point Multiplier operates on normals but cannot operate on denormals directly. Denormals must first be "wrapped" by an ALU to a format readable by a Multiplier. Several flags are available for detecting and handling exceptions caused by loading a denormal to a Floating-Point Multiplier. Information about rounding and inexact results generated by the Multiplier is needed by the ALU to produce results in conformance to Standard 754. All ADSP-3201/3202 chips include a "FAST" control that flushes all denormalized results to zero, avoiding the system delays of IEEE exception processing for gradual underflow.

All status output flags except denormal detection are registered at the output in parallel with their associated results. The asynchronous denormal flag allows an early detection of a denormalized number loaded to a Floating-Point Multiplier, speeding exception processing.

time (cycles)	Load Input Data	First-Stage Processing	Second-Stage Processing	Output Result
1	Data Set A			
2	Data Set B	Data Set A		
3	Data Set C	Data Set B	Data Set A	
4	Data Set D	Data Set C	Data Set B	Data Set A
5	Data Set E	Data Set D	Data Set C	Data Set B

Figure 2. Typical Pipelining with the ADSP-3201/3202

PIN DEFINITIONS AND FUNCTIONAL BLOCK DIAGRAMS

All control pins are active HI (positive true logic naming convention), except $\overline{\text{RESET}}$ and $\overline{\text{HOLD}}$. Some controls are registered at the clock's rising edge (REG); other controls are latched in clock HI and transparent in clock LO (LAT); and others are asynchronous (ASYN).

ADSP-3201 Floating-Point Multiplier Pin List

PIN NAME	DESCRIPTION	TYPE
Data Pins		
AIN_{31-0}	32-Bit Data Input	
BIN_{31-0}	32-Bit Data Input	
$DOUT_{31-0}$	32-Bit Data Output	
Control Pins		
$\overline{\text{RESET}}$	Reset	ASYN
$\overline{\text{HOLD}}$	Hold Control	ASYN
IPORT0	Input Port Configuration Control 0	ASYN
IPORT1	Input Port Configuration Control 1	ASYN
SELA0	Load Selection for A0	LAT
SELA1	Load Selection for A1	LAT
SELA2	Load Selection for A2	LAT
SELA3	Load Selection for A3	LAT
SELB0	Load Selection for B0	LAT
SELB1	Load Selection for B1	LAT
SELB2	Load Selection for B2	LAT
SELB3	Load Selection for B3	LAT
RDA0	Register Ax Read Selection Control 0	REG
RDA1	Register Ax Read Selection Control 1	REG
RDB0	Register Bx Read Selection Control 0	REG
RDB1	Register Bx Read Selection Control 1	REG
WRAPA	Wrapped Contents in Register Ax	REG
WRAPB	Wrapped Contents in Register Bx	REG
TCA	Twos-Complement Integer in Register Ax	REG
TCB	Twos-Complement Integer in Register Bx	REG
ABSA	Read Absolute Value of Ax	REG
ABSB	Read Absolute Value of Bx	REG
SP	Single-Precision Floating-Point Mode	REG
DP	Double-Precision Mode	REG
RND0	Rounding Mode Control 0	REG
RND1	Rounding Mode Control 1	REG
FAST	Fast Mode	REG
SHLP	Shift Left Fixed-Point Product	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN
Status Out		
INEXO	Inexact Result	
OVRFLO	Overflowed Result	
UNDFLO	Underflowed Result	
INVALOP	Invalid Operation	
DENORM	Denormal Output	
RNDCARO	Round Carry Propagation Out	
Miscellaneous		
CLK	Clock Input	
V_{DD}	+5V Power Supply (Four Lines)	
GND	Ground Supply (Eight Lines)	

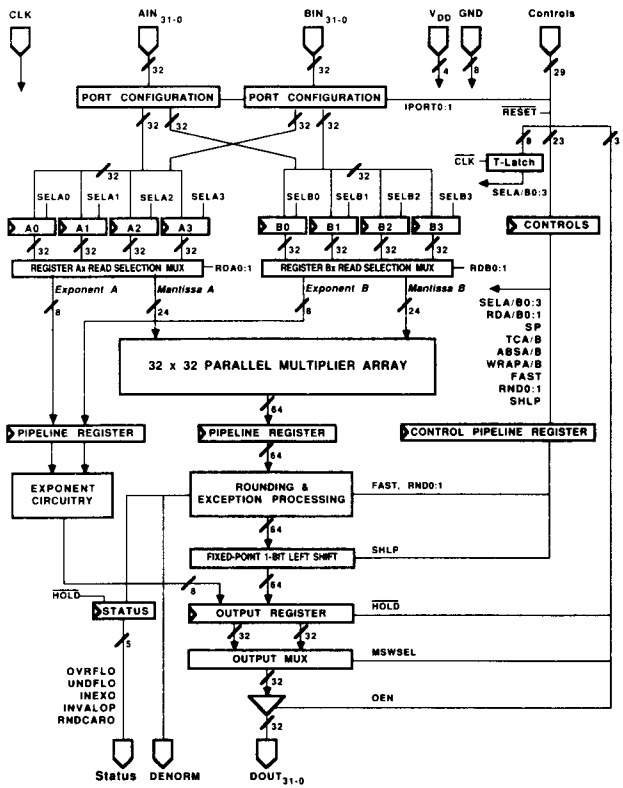


Figure 3. ADSP-3201 Functional Block Diagram

ADSP-3202 Floating-Point Multiplier Pin List

PIN NAME	DESCRIPTION	TYPE
Data Pins		
AIN_{31-0}	32-Bit Data Input	
BIN_{31-0}	32-Bit Data Input	
$DOUT_{31-0}$	32-Bit Data Output	
Control Pins		
$\overline{RESET}$	Reset	ASYN
IPORT0	Input Port Configuration Control 0	ASYN
IPORT1	Input Port Configuration Control 1	ASYN
SELA0	Load Selection for A0	LAT
SELA1	Load Selection for A1	LAT
SELA2	Load Selection for A2	LAT
SELA3	Load Selection for A3	LAT
SELB0	Load Selection for B0	LAT
SELB1	Load Selection for B1	LAT
SELB2	Load Selection for B2	LAT
SELB3	Load Selection for B3	LAT
RDA0	Register Ax Read Selection Control 0	REG
RDA1	Register Ax Read Selection Control 1	REG
RDB0	Register Bx Read Selection Control 0	REG
RDB1	Register Bx Read Selection Control 1	REG
I_{8-0}	ALU Instruction	REG
RND0	Rounding Mode Control 0	REG
RND1	Rounding Mode Control 1	REG
FAST	Fast Mode	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN
Status In		
INEXIN	Inexact Data In	REG
RNDCAR1	Round Carry Propagation In	REG
Status Out		
INEXO	Inexact Result	
OVRFLO	Overflowed Result	
UNDFLO	Underflowed Result	
INVALOP	Invalid Operation	
Miscellaneous		
CLK	Clock Input	
V_{DD}	+5V Power Supply (Four Lines)	
GND	Ground Supply (Four Lines)	

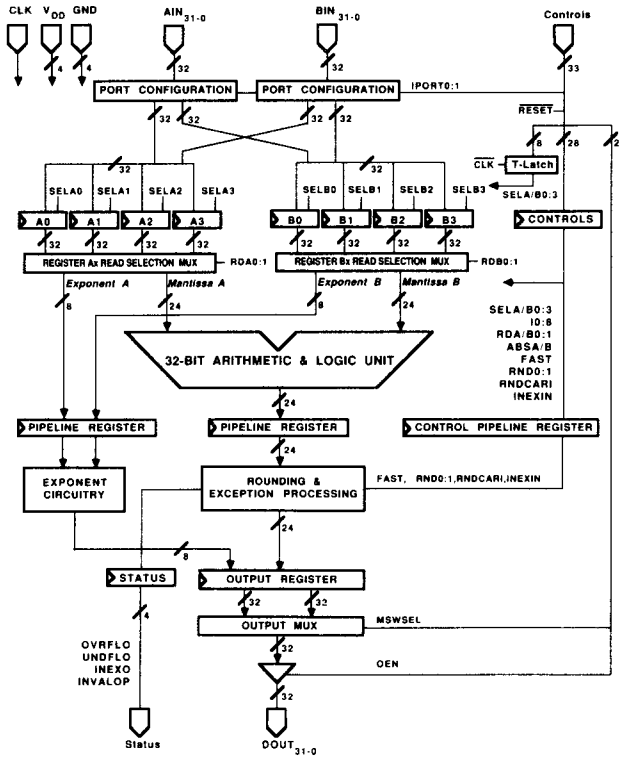


Figure 4. ADSP-3202 Functional Block Diagram

METHOD OF OPERATION

DATA FORMATS

The ADSP-3201/3202 chipset supports single-precision floating-point data formats and operations as defined in IEEE Standard 754-1985. 32-bit twos-complement fixed-point data formats and operations are also supported by all four chips. 32-bit unsigned-magnitude data formats and operations are supported by the ADSP-3201 Multiplier and ADSP-3202 ALU. This chipset operates directly on 32-bit fixed-point data. (No time-consuming conversions to and from floating-point formats are required.)

Single-Precision Floating-Point Data Format

IEEE Standard 754 specifies a 32-bit single-precision floating-point format,

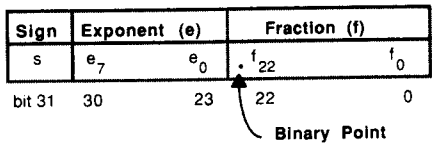


Figure 5. Single-Precision Floating-Point Format

which consists of a sign bit s, a 24-bit significand, and an 8-bit unsigned-magnitude exponent e. For normalized numbers, this significand consists of a 23-bit fraction f and a "hidden" bit of 1 that is implicitly presumed to precede f_{22} in the significand. The binary point is presumed to lie between this hidden bit and f_{22}. The least significant bit of the fraction is f_0; the LSB of the exponent is e_0. The hidden bit effectively increases the precision of the floating-point significand to 24 bits from the 23 bits actually stored in the data format. It also insures that the significand of any number in the IEEE normalized-number format is always greater than or equal to 1 and less than 2.

The unsigned exponent e for normals can range between $1 \le e \le 254$ in the single-precision format. This exponent is *biased* by +127 in the single-precision format. This means that to calculate the "true" *unbiased* exponent, 127 must be subtracted from e.

The IEEE Standard also provides for several special data types. In the single-precision floating-point format, an exponent value of 255 (all ones) with a nonzero fraction is a not-a-number (NAN). NANs are usually used as flags for data flow control, for the values of uninitialized variables, and for the results of invalid operations such as $0 \cdot \infty$. Infinity is represented as an exponent of 255 and a zero fraction. Note that because the fraction is signed, both positive and negative INF can be represented.

The IEEE Standard requires the support of denormalized data formats and operations. A denormalized number, or "denormal," is a number with a magnitude less than the minimum normalized ("normal") number in the IEEE format. Denormals have a zero exponent and a nonzero fraction. Denormals have no hidden "one" bit. (Equivalently, the hidden bit of a denormal is zero.)

The unbiased (true) value of a denormal's exponent is −126 in the single-precision format, i.e., one minus the exponent bias. Note that because denormals are not required to have a significant leading one bit, the precision of a denormal's significand can be as little as one bit for the minimum representable denormal.

ZERO is represented by a zero exponent and a zero fraction. As with INF, both positive ZERO and negative ZERO can be represented.

The IEEE single-precision floating-point data types and their interpretations are summarized in Table I.

The ADSP-3201/3202 chipset also supports two data types not included in the IEEE Standard, "wrapped" and "unnormal." These data types are necessitated by the fact that the ADSP-3201 Multiplier and the ADSP-3202 ALU during division and square root do not operate directly on denormals. (To do so, they would need shifting hardware that would slow them significantly.) Denormal operands must first be translated by the ADSP-3202 ALU to wrapped numbers to be readable by the Multiplier. Wrapped and unnormal Multiplier products must also be unwrapped by an ALU before an ALU can operate on these results in general. (See "Gradual Underflow and IEEE Exceptions.")

The interpretation of wrapped numbers differs from normals only in that the exponent is treated as a twos-complement number. Single-precision wrapped numbers have a hidden bit of one and an exponent bias of +127. All single-precision denormals can be mapped onto wrapped numbers where the exponent e ranges between $-22 \le e \le 0$. WRAPA and WRAPB controls on the ADSP-3201 tell the Multiplier to interpret a data value as a wrapped number.

The ranges of the various single-precision floating-point data formats supported by the ADSP-3201/3202 are summarized in Table II.

The multiplication of two wrapped numbers can produce a number smaller than can be represented as a wrapped number. Such numbers are called "unnormals." Unnormals are interpreted exactly as are wrapped numbers. They differ only in the range of their exponents, which fall between $-171 \le e \le -23$ for single-precision unnormals. The smallest unnormal is the result of multiplying WRAP.MIN by itself. Unnormals, because they are smaller than DRNM.MIN, generally unwrap to ZERO. (UNRM.MAX can unwrap to DRNM.MIN, depending on rounding mode.)

The underflow flag should be thought of as an implicit most significant ninth bit, the sign bit. For unnormals for which $-171 \le e < -128$, the most significant bit in the eight-bit exponent field (e_7, bit 30) will be zero, but the underflow flag understood as weighted by −256 allows their representation without ambiguity. This sign bit is implicitly assumed by the ALU to be present when unwrapping unnormals, making this convention for very small unnormals transparent to the user.

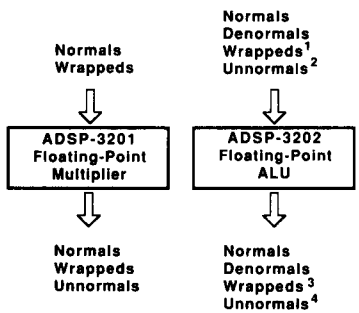
Mnemonic	Exponent	Fraction	Value	Name	IEEE Format?
NAN	255	non-zero	undefined	not-a-number	yes
INF	255	zero	$(-1)^s$(infinity)	infinity	yes
NORM	1 thru 254	any	$(-1)^s(1.f)2^{e-127}$	normal	yes
DNRM	0	non-zero	$(-1)^s(0.f)2^{-126}$	denormal	yes
ZERO	0	zero	$(-1)^s 0.0$	zero	yes
WRAP	−22 thru 0	any	$(-1)^s(1.f)2^{e-127}$	wrapped	no
UNRM	−171 thru −23	any	$(-1)^s(1.f)2^{e-127}$	unnormal	no

Table I. Single-Precision Floating-Point Data Types and Interpretations

Data name (positive)	Exponent	Exp. data type	Exponent bias	Hidden bit	Fraction (binary)	Unbiased absolute value
NORM.MAX	254	unsigned	+127	1	111.....11	$2^{+127} \cdot (2-2^{-23})$
NORM.MIN	1	unsigned	+127	1	000.....00	2^{-126}
DNRM.MAX	0	unsigned	+126	0	111.....11	$2^{-126} \cdot (1-2^{-23})$
DNRM.MIN	0	unsigned	+126	0	000.....01	$2^{-126} \cdot 2^{-23}$
WRAP.MAX	0	2scmplmt	+127	1	111.....11	$2^{-127} \cdot (2-2^{-23})$
WRAP.MIN	−22	2scmplmt	+127	1	000.....00	2^{-149}
UNRM.MAX	−23	2scmplmt	+127	1	111.....11	$2^{-150} \cdot (2-2^{-23})$
UNRM.MIN	−171	2scmplmt	+127	1	000.....00	2^{-298}

Table II. Single-Precision Floating-Point Range Limits

Supported Floating-Point Data Types
The direct floating-point data types support provided by the members of this chipset can be summarized:



1. for unwrapping, division, and square root
2. for unwrapping only
3. from wrapping and division
4. from division

Figure 6. Data Types Directly Supported by the ADSP-3201/3202

Not every member of the ADSP-3201/3202 chipset supports all the data types described above directly. See the section below, "Gradual Underflow and IEEE Exceptions," for a full description of how the chips work together to implement the IEEE Standard. For systems not requiring full conformance to Standard 754, the section below, "FAST/IEEE Control," describes a simplified operation for this chipset that avoids denormals, wrappeds, and unnormals altogether.

32-Bit Fixed-Point Data Formats
The ADSP-3201/3202 chipset supports two 32-bit fixed-point formats: twos-complement and unsigned-magnitude. With the ALU, the output data format is identical with the input data format, i.e., 32 bits wide. In contrast, the Multiplier produces a 64-bit product from two 32-bit inputs.

The 32-bit twos-complement data format for Multiplier inputs and ALU inputs and outputs is:

	Sign				
WEIGHT	-2^{k+31}	2^{k+30}	2^{k+29}	...	2^{k}
VALUE	i_{31}	i_{30}	i_{29}	...	i_0
POSITION	31	30	29	...	0

Figure 7. 32-Bit Twos-Complement Fixed-Point Data Format

The MSB is i_{31}, which is also the sign bit; the LSB is i_0. Note that the sign bit is negatively weighted in twos-complement format. The position of the binary point for fixed-point data is represented here in full generality by the integer *k*. Integers (binary point right of bit position 0) are represented when k = 0; signed fractional numbers (binary point between bit positions 31 and 30) are represented when k = 31. The value of k is for user interpretation only and in general does not affect the operation of the chips. The only exceptions are the ALU conversion operations between floating-point and fixed-point. For these operations, the fixed-point format is presumed to be twos-complement integers, i.e., k = 0.

The ADSP-3201 Multiplier produces a 64-bit product at its Output Register. The ADSP-3201 will produce results in the format of Figure 8 at the DOUT port if the Shift Left Fixed-Point Product (SHLP) control (described below in "Output Control") is LO:

	Sign							
WEIGHT	-2^{r+63}	2^{r+62}	...	2^{r+32}	2^{r+31}	...	2^{r+1}	2^{r}
VALUE	i_{63}	i_{62}	...	i_{32}	i_{31}	...	i_1	i_0
POSITION	63	62	...	32	31	...	1	0

Most Significant Product — Least Significant Product

Figure 8. 64-Bit Twos-Complement Fixed-Point Data Format at Multiplier Output Register with SHLP LO

The weighting of the product bits is given by the integer *r*. When k_A represents the weighting of operand A and k_B the weighting of operand B, then $r = k_A + k_B$.

When HI, the SHLP control shifts all bits left one position as they are loaded to the Output Register. The results will then be in the format:

WEIGHT	Sign -2^{r+62}	2^{r+61}	...	2^{r+31}	2^{r+30}	...	2^{r}	2^{r-1}
VALUE	i_{62}	i_{61}	...	i_{31}	i_{30}	...	i_0	0
POSITION	63	62	...	32	31	...	1	0

Most Significant Product Least Significant Product

Figure 9. 64-Bit Twos-Complement Fixed-Point Data Format at Multiplier Output Register with SHLP HI

The LSB becomes zero and i_{62} moves into the sign bit position. Normally i_{63} and i_{62} will be identical in twos-complement products. (The only exception is full-scale negative multiplied by itself.) Hence, a one-bit left-shift normally removes a redundant sign bit, thereby increasing the precision of the Most Significant Product. Also, if the fixed-point data format is fractional ($k = -31$ in Figure 7), then a single-bit left-shift will renormalize the MSP to a fractional format (because $r = 2 \cdot k = 2 \cdot (31) = -62$).

For unsigned-magnitude data formats, inputs to the ADSP-3201 Multiplier and inputs and outputs from the ADSP-3202 ALU will be 32 bits wide. The 32-bit unsigned-magnitude data format is:

WEIGHT	2^{k+31}	2^{k+30}	2^{k+29}	...	2^{k}
VALUE	i_{31}	i_{30}	i_{29}	...	i_0
POSITION	31	30	29	...	0

Figure 10. 32-Bit Unsigned-Magnitude Fixed-Point Data Format

Again, the position of the binary point for fixed-point data is represented here in full generality by the integer k. Integers (binary point right of bit position 0) are represented when $k = 0$; unsigned fractional numbers (binary point left of bit position 31) are represented when $k = -32$. The value of k is for user interpretation only and, except for conversions to fixed-point, does not affect the operation of the chips.

The ADSP-3201 Multiplier discriminates twos-complement from unsigned-magnitude inputs with TCA and TCB controls. (See "Controls.") When TCA and TCB are both LO, the ADSP-3201 produces a 64-bit unsigned-magnitude product at its Output Register. The ADSP-3201 will produce results in this format if SHLP is LO:

WEIGHT	2^{r+63}	2^{r+62}	...	2^{r+32}	2^{r+31}	...	2^{r+1}	2^{r}
VALUE	i_{63}	i_{62}	...	i_{32}	i_{31}	...	i_1	i_0
POSITION	63	62	...	32	31	...	1	0

Most Significant Product Least Significant Product

Figure 11. 64-Bit Unsigned-Magnitude Fixed-Point Data Format at Multiplier Output Register with SHLP LO

Again, the weighting of the product bits is given by the integer r. When k_A represents the weighting of operand A, and k_B the weighting of operand B, then $r = k_A + k_B$.

If SHLP is HI, the data at the Output Register will have been shifted left one position and zero-filled in the format:

WEIGHT	2^{r+62}	2^{r+61}	...	2^{r+31}	2^{r+30}	...	2^{r}	2^{r-1}
VALUE	i_{62}	i_{61}	...	i_{31}	i_{30}	...	i_0	0
POSITION	63	62	...	32	31	...	1	0

Most Significant Product Least Significant Product

Figure 12. 64-Bit Unsigned-Magnitude Fixed-Point Data Format at Multiplier Output Register with SHLP HI

The ADSP-3201 also supports mixed-mode multiplications, i.e., twos-complement by unsigned-magnitude. These are valuable in extended-precision fixed-point multiplications, e.g., 64×64 and 128×128. The result of a mixed-mode multiplication will be in a twos-complement format. Unlike twos-complement multiplications, however, mixed-mode results do not in general have a redundant sign bit in i_{62}. Hence, mixed-mode results should be read out with SHLP LO as in Figure 8.

CONTROLS

The controls for the ADSP-3201/3202 (see Pin Lists above) are all active HI, with the exceptions of $\overline{\text{RESET}}$ and $\overline{\text{HOLD}}$. The controls are either *registered* into the Input Control Register at the clock's rising edge, *latched* into the Input Control Register with clock HI and transparent in clock LO, or *asynchronous*. The controls are discussed below in the order in which they affect data flowing through the chipset.

Registered controls, in general, are pipelined to match the flow of data. All data and control pipelines advance with the rising edge of each clock cycle. For example, to perform n optional fixed-point one-bit left-shift on output with the product of X and Y, you would assert the registered, pipelined control SHLP on the rising edge that causes X and Y inputs to be read into the multiplier array. Just before the result was ready to be loaded to the Output Register, the pipelined SHLP control would perform the proper shift. After the initiation of a multicycle operation, registered control inputs are ignored until the end of the operation time. (See "Timing" below for a precise definition of "operation time.")

Because this chipset uses CMOS static logic throughout and controls are pipelined, the clock can be stopped as long as desired for generating wait-states, diagnostic analysis, or whatever. These chips can also be easily adapted to "state-push" implementations. The machine's state can be pushed forward one stage by simply providing a rising edge to the clock input when desired.

The only controls that are latched (as opposed to registered) are the Load Selection Controls. They are transparent in clock LO and latched with clock HI. Load Selection Controls are setup to the chips exactly as if they were registered, with the same setup time. The fact that they are transparent in clock LO allows them to select input registers in parallel with the setup of data to be loaded on the rising edge. Because they are latched with clock HI, microcode need only be presented at the clock rate, though data is loaded on both clock rising and falling edges.

A few controls are asynchronous. These controls take effect immediately and are thus neither registered nor pipelined. Each has an independently specified setup time.

FAST/IEEE CONTROL (REG)

FAST is a pipelined, registered control. It affects the interpretation of data read into processing circuitry immediately after having been loaded to the input control register. FAST affects the format of results in the rounding & exception processing pipeline stage. FAST also affects the definition of some exception flags. (See "Exception Flags.")

IEEE Standard 754 requires a system to perform operations on denormal operands (which are smaller in magnitude than the minimum representable normalized number). This capability to accommodate these numbers is known as "gradual underflow". For floating-point systems not requiring strict adherence to the IEEE Standard, the ADSP-3201/3202 provides a FAST mode (FAST control pin HI) which consistently flushes post-rounded results less than NORM.MIN to ZERO. This approach greatly simplifies exception processing and avoids generating the denormal, wrapped, and unnormal data types described above. When in FAST mode, the Multiplier will treat denormal inputs as ZERO and produces a ZERO result. The ALU will treat denormal inputs exactly as it does in IEEE mode but still flush post-rounded results less than NORM.MIN to ZERO.

Systems implementing gradual underflow with the ADSP-3201/3202 must treat the multiplication of operands that include a denormal as an exception to normal process flow. FAST should be LO on all chips. See the section below, "Gradual Underflow and IEEE Exceptions," for a fuller discussion of the details of implementing an IEEE system with this chipset.

RESET CONTROL (ASYN)

The asynchronous, active LO $\overline{\text{RESET}}$ control clears all control functions in the ADSP-3201/3202. $\overline{\text{RESET}}$ should be asserted on power up to insure proper initialization. $\overline{\text{RESET}}$ will abort any multicycle operation in progress. Status flags are cleared by $\overline{\text{RESET}}$. No input register contents are affected by $\overline{\text{RESET}}$; however, the output register can be invalidated if $\overline{\text{RESET}}$ is asserted LO during a multicycle operation. All load selection controls (SELA/B) must be LO at $\overline{\text{RESET}}$.

PORT CONFIGURATION – IPORT CONTROLS (ASYN)

This chipset offers several options on its input port configuration. The options are controlled by the two asynchronous lines, IPORT0:1. They are intended to be hardwired to the desired port configuration. If the user wants to change the port configuration under microcode control, the timing requirements of Figure 14 must be met.

The first and last configurations in Figure 13 are called "two-port" configurations; the middle pair, "one-port" configurations. Whether an input register loads its data on a rising or falling clock edge will depend in general on whether the chip is wired in a one-port or two-port configuration.

In one-port configurations, the unused port effectively becomes a no-connect, reducing the number of external buses required to operate these chips. The full pipelined throughput can be maintained for the Multiplier and the ALU in the one-port configuration for all 32-bit operations.

IPORT1	IPORT0	PORT CONFIGURATION
0	0	AIN BIN *two port* → A registers, B registers
0	1	AIN BIN *one port* → A registers, B registers
1	0	AIN BIN *one port* → A registers, B registers
1	1	AIN BIN *two port* → A registers, B registers

Figure 13. ADSP-3201/3202 Input Port Configurations

The port configuration of the ADSP-3201/3202 can be changed under microcode control. However, as described in the section below, "Input Register Loading", the selected port configuration affects whether a given register loads on rising or falling clock edges. The transition between port configurations can cause inadvertent data loads, destroying data held in input registers. Therefore, all input registers must be deselected for data loading (all SELA/B controls must be *held* LO throughout the period when IPORT0:1 are changing; see "Input Register Loading") during both the cycle in which IPORT bits are changed and the cycle following:

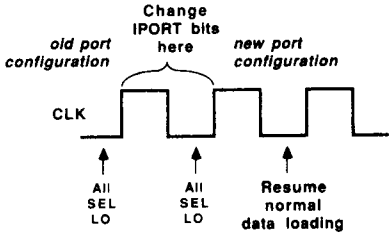


Figure 14. Timing Requirements for Changing the ADSP-3201/3202 Input Port Configurations

Thus, data loading will be interrupted for two cycles whenever changing the ADSP-3201/3202's port configuration. All other processing is unaffected.

INPUT REGISTER LOADING AND OPERAND STORAGE – SELA/B CONTROLS (LAT)

The chipset's 32-bit input registers are selected for data loading with the latched Load Selection Controls, SELA/B0:3. Since each input register has its own control, the Load Selection Controls are independent of one another. Multiple registers can be selected for parallel loads of the same input data, if desired. The Load Selection Controls' effects on data loading are summarized:

SEL control	register loaded
SELA0	A0
SELA1	A1
SELA2	A2
SELA3	A3
SELB0	B0
SELB1	B1
SELB2	B2
SELB3	B3

Figure 15. ADSP 3201/3202 Load Selection Controls

Restrictions on Register Loading

Input port configuration affects whether input registers load data on rising or falling edges. Devices in one-port configurations load A registers on rising edges and B registers on falling edges. Devices in two-port configurations load even-numbered registers on rising edges and odd-numbered registers on falling edges (which is typically simpler to implement). Devices in the two-port configuration load data:

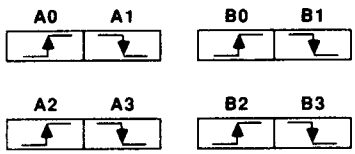


Figure 16. ADSP-3201/3202 Clock Edge for Data Loading – Two Port Configuration

Eight-register devices (ADSP-3201/3202) in the one-port configuration load data to A registers on the rising edge and B registers on the falling edge:

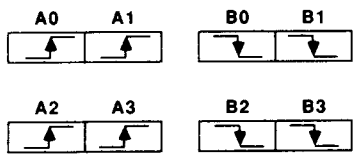


Figure 17. ADSP-3201/3202 Clock Edge for Data Loading – One Port Configuration

Restrictions on Register Storage

For single-precision and fixed-point data, any convenient register can be used. The only restriction is that the register being loaded is not currently in use by the chip's processing elements. For all single-precision Multiplier and most ALU operations, input registers are only read into the computational circuits for one cycle. Do not load a register for 32-bit operations on the clock's falling edge when that register has been selected to feed the chips processing circuits in that same cycle (with the RDA/B controls described in "Input Data Read Selection"). Pick a register not in use.

The ADSP-3202 ALU is capable of two multicycle operations: IEEE floating-point division and square root. For single-precision floating-point division, the dividend can be stored in any A register and the divisor can be stored in any B register. Single-precision operands for IEEE square root can be stored in any B register. The registers selected to the computational circuits for these operations must be stable until the end of the operation time. (See "Timing" and the timing diagrams below for a precision definition of "operation time.")

DATA FORMAT SELECTION – SP CONTROL (REG)

The two data formats processed by the ADSP-3201/3202 chipset are *single-precision* floating-point and *fixed*. With the ADSP-3201 Multiplier, the data format is indicated explicitly by the states of the SP registered control:

SP	Data Format Selection
0	fixed
1	single-precision

Figure 18. ADSP-3201 Multiplier Data Format Selection

The state of the SP control at the rising edge when data is read into the Multiplier Array determines whether the data is interpreted as single-precision floating-point or fixed-point. Once initiated, the state of SP doesn't matter until the next data is read to the processing circuitry.

For the ADSP-3202 ALU, data format selection is implicit in the ALU instruction, I_{8-0}. (See "ALU Operation" section below.)

INPUT DATA REGISTER READ SELECTION – RDA/B CONTROLS (REG)

The Register Read Selection Controls, RDA/B0:1, are registered controls and select the input registers that are read into the chipset's processing circuitry. Any pair of input registers can be read into the processing circuitry. (For single-operand operations, the state of the Selection controls for the unused register bank doesn't matter.) Data loaded to an input register on a rising edge can be read into the processing circuitry on that same edge ("direct operand feed").

For the ADSP-3201/3202, register read selection is defined:

RDA1	RDA0	SP & Fixed: A register selected
0	0	A2
0	1	A3
1	0	A0
1	1	A1

RDB1	RDB0	SP & Fixed: B register selected
0	0	B2
0	1	B3
1	0	B0
1	1	B1

Figure 19. ADSP-3201/3202 Input Register Read Selection

After the initiation of multicycle operations, the RDA/B controls are ignored. The chips themselves take over the sequencing of register read selection until the multicycle operation is completed.

ABSOLUTE VALUE CONTROLS – ABSA/B (REG)

The registered Absolute Value Controls convert an operand selected by the Read Selection Controls to its absolute value before processing. Asserting ABSA (HI) causes the A operand to be converted to its absolute value; asserting ABSB (HI) causes the B operand to be converted to its absolute value. The contents of the input registers remain unaffected.

With the ADSP-3202 ALU, the ABSA/B controls are effective with most fixed-point and all single-precision operations. If the ABSA/B controls are asserted in logical operations, the results will be undefined.

For the ADSP-3201 Multiplier, the absolute value operation is available on single-precision floating point operands only. If the ABSA/B controls are asserted with a Multiplier for a fixed-point operation, the results will be undefined.

WRAPPED INPUT CONTROLS – WRAPA/B (REG) (and INEXIN and RNDCARI on the ADSP-3202)

The ADSP-3201 cannot operate directly on denormals; denormals to be multiplied must first be converted by an ALU to the "wrapped" format. (See "Gradual Underflow and IEEE Exceptions" below). The Multiplier must be told that an input is in the wrapped format so that its exponent can be interpreted properly as a twos-complement number.

The registered WRAPA/B controls inform a Multiplier that a wrapped number has been selected as an operand (RDA/B controls) to the multiplier array. WRAPA indicates (HI) that the selected A register contains a wrapped number; WRAP B, that the selected B register contains a wrapped number.

The ALU in general operates directly on denormals and hence don't need a similar set of controls. However, for ADSP-3202 IEEE division and square root operations, the ALU cannot operate directly on denormals. Like the Multiplier, it needs denormals to be converted to wraps before processing. To indicate that the dividend in the A register is a wrapped, INEXIN should be asserted (HI) exactly as WRAPA would be asserted on the Multiplier. To indicate that either the divisor in a B register or a square root operand in a B register is a wrapped, RNDCARI should be asserted (HI). Except for unwrap, division, and square root operations, both INEXIN and RNDCARI should be held LO.

TWOS-COMPLEMENT INPUT CONTROLS – TCA/B (REG)

The registered ADSP-3201's Twos-Complement Input Controls inform the Multiplier to interpret the selected fixed-point inputs in the twos-complement data format. (See "32-Bit Fixed-Point Data Formats" above.) TCA HI indicates that the selected A register is twos-complement; TCB HI indicates a twos-complement B register. A LO value on either control for fixed-point multiplication indicates that the selected input is in unsigned-magnitude format. Mixed-mode (twos-complement times unsigned-magnitude) multiplications are permitted. The TCA/B controls are operative in fixed-point mode only; in floating-point mode, they are ignored.

ROUNDING – RND CONTROLS (REG)

For floating-point operations, the ADSP-3201/3202 chipset supports all four rounding modes of IEEE Standard 754. These are: Round-to-Nearest, Round-toward-Zero, Round-toward-Plus-Infinity, and Round-toward-Minus-Infinity. For fixed-point operations, two rounding modes are available: Round-to-Nearest, and Unrounded.

Rounding is involved in all operations in which the precision of the destination format is less than the precision of the intermediate results from the operation. Multiplications internally generate twice as many bits in the intermediate result significand as can be stored in the destination format. Data conversions to a destination format of lesser precision than the source also always force rounding unless the source value fits exactly.

Rounding with the ADSP-3201/3202 chipset is controlled by a pair of pipelined, registered round controls, RND0:1. They should be setup with the input data whose result is to be rounded. Rounding is performed in the last stage of processing; the Output Register always contains rounded results. The effects of the Round Controls are defined in Figure 20.

The four floating-point modes of the IEEE Standard can be summarized as follows. In all cases, if the result before rounding can be expressed exactly in the destination format without loss of accuracy, then that will be the destination format result, regardless of specified rounding mode.

Round-toward-Plus-Infinity (RP): "When rounding toward $+\infty$, the result shall be the format's value (possibly $+\infty$) closest to and no less than the infinitely precise result." (Std 754-1985, Sec. 4.2) If the result before rounding (the "infinitely precise result") is not exactly representable in the destination format, then the result will be that number which is nearer to positive infinity. Round-toward-Plus-Infinity is available in floating-point operations only. If the result before rounding is greater than NORM.MAX but not equal to Plus Infinity, the result will be Plus Infinity. If the result before rounding is less than $-$NORM.MAX but not equal to Minus Infinity, the result will be $-$NORM.MAX. For fixed-point destination formats, the results of RP are undefined.

Mnemonic	RND1	RND0	Floating-Point	Fixed-Point
RN	0	0	Round-to-Nearest	Round-to-Nearest
RZ	0	1	Round-toward-Zero	Unrounded
RP	1	0	Round-toward-Plus-Infinity	illegal state
RM	1	1	Round-toward-Minus-Infinity	illegal state

Figure 20. Round Controls

Round-toward-Minus-Infinity (RM): When rounding toward $-\infty$, the result shall be the format's value (possibly $-\infty$) closest to and no greater than the infinitely precise result." (Std 754-1985, Sec. 4.2) If the result before rounding is not exactly representable in the destination format, the result will be that number which is nearer to Minus Infinity. Round-toward-Minus-Infinity is available in floating-point operations only. If the result before rounding is greater than NORM.MAX but not equal to Plus Infinity, the result will be NORM.MAX. If the result before rounding is less than $-$NORM.MAX but not equal to Minus Infinity, the result will be Minus Infinity. For fixed-point destination formats, the results of RM are undefined.

Round-toward-Zero and Unrounded (RZ): "When rounding toward 0, the result shall be the format's value closest to and no greater in magnitude than the infinitely precise result." (Std 754-1985, Sec. 4.2) If the result before rounding is not exactly representable in the destination format, the result will be that number which is nearer to zero. The Round-toward-Zero operation is available in floating-point operations only. It is equivalent to truncation of the (unsigned-magnitude) significand. If the result before rounding has a magnitude greater than NORM.MAX but not equal to Infinity, the result will be NORM.MAX of the same sign.

For fixed-point destination formats, the RZ mode is *Unrounded*. For fixed-point operations, RZ has no effect on the result at the Output Register and should be specified whenever unmodified fixed-point results are desired. (Treating the unrounded Most Significant Product as the final result and throwing away the LSP is logically equivalent to Round-toward-Minus-Infinity for twos-complement numbers and equivalent to Round-toward-Zero [truncation] for unsigned-magnitude numbers.)

Round-to-Nearest (RN): When rounding to nearest, "the representable value nearest to the infinitely precise result shall be delivered; if the two nearest representable values are equally near, the one with its least significant bit zero shall be delivered." (Std 754-1985, Sec. 4.1) If the result before rounding is not exactly representable in the destination format, the result will be that number which is nearer to the result before rounding. In the case that the result before rounding is exactly half way between two numbers in the destination format differing by an LSB, the result will be that number which has an LSB equal to zero.If the result before rounding overflows, i.e., has a magnitude greater than or equal to NORM.MAX + 1/2LSB in the destination format, the result will be the Infinity of the same sign.

Round-to-Nearest is available in both floating-point and fixed-point operations. In fixed-point, Round-to-Nearest treats the Most Significant Product *after having been shifted in accordance with SHLP* (see Figures 8, 9, 11, and 12) as the destination format.

The four rounding modes are illustrated by number lines in Figure 21. The direction of rounding is indicated by an arrow. Numbers exactly representable in the destination format are indicated by "•"s. In subdividing the number lines, square brackets are inclusive of the points on the line they intersect. Note that brackets intersect points representable in the destination format except for Round-to-Nearest, where they intersect the line midway between representable points. Slashes are used to indicate a break in the number line of arbitrary size.

Note that Round-to-Nearest is unique among the rounding modes in that it is *unbiased*. The large-sample statistical mean from a set of numbers rounded in the other modes will be displaced from the true mean. The other three modes will exhibit a large-sample statistical *bias* in the direction of the rounding operation performed.

STATUS FLAGS

The ADSP-3201/3202 chipset generates on dedicated pins the following exception flags specified in the IEEE Standard: Overflow (OVRFLO), Underflow (UNDFLO), Inexact Result (INEXO), and Invalid Operation (INVALOP). The IEEE exception condition Division-by-Zero is flagged by the simultaneous assertion of both OVRFLO and INVALOP pins. The five IEEE exceptions are defined in accordance to the default assumption of Std 754 of nontrapping exceptions.

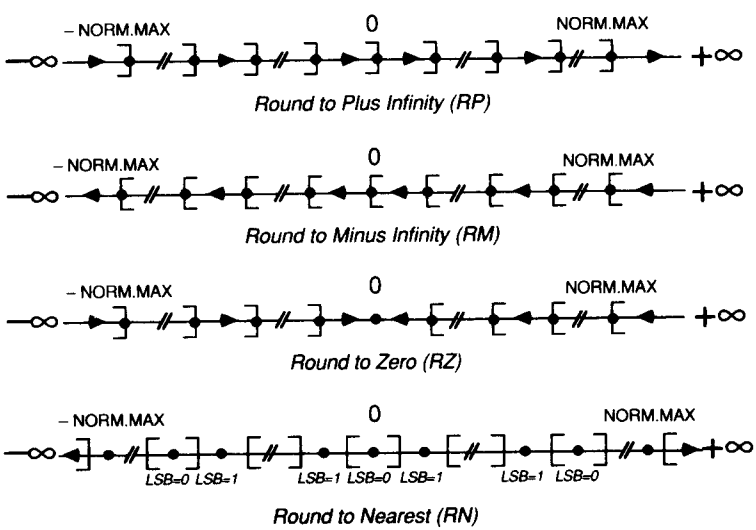


Figure 21. IEEE Rounding Modes

These four flag results are registered in the Status Output Register when the results they reflect are clocked to the Output Register. They are held valid until the next rising clock edge. The IEEE Standard specifies that exception flags when set remain set until reset by the user. For full conformance to the standard, the status outputs from this chipset should be individually latched externally.

Denormal Input

In addition to the IEEE status flags, the ADSP-3201 Multiplier has a DENORM output flag that signals the presence of a denormalized number at one of the input registers being read into the multiplier array. This denormal must be wrapped by the ALU before the Multiplier can read it. To minimize the system response time to a denormal input exception, the DENORM flag comes out earlier than the associated IEEE status flags. DENORM is normally in an indeterminate state. For single-precision multiplications, DENORM goes HI during the cycle after a denormal was read into the array (with the RDA/B controls). (See Figure T4.) The Multiplier produces ZERO results under these conditions. The DENORM flag is asserted in both IEEE and FAST modes.

Some multiplications with denormal operands do not require wrapping and therefore do not cause the assertion of the DENORM flag. These are DNRM•ZERO, DNRM•INF, and DNRM•NAN. Multiplication of a finite number by zero always yields zero – the result the Multiplier will produce anyway – so there is no need to signal an exception. Any finite number multiplied by INF should yield INF, and the ADSP-3201 Multiplier will produce this result with a DNRM operand, hence no wrapping is required. And multiplication of any number by a NAN produces a NAN (and the INVALOP flag); no wrapping is necessary for the Multiplier to produce this correct IEEE result.

Note that the ALU in general operate directly on denormals and therefore do not flag any exception. The ADSP-3202 ALU, however, cannot operate directly on denormals in its division and square root operations. For these operations, denormal inputs will cause the simultaneous assertion of UNDFLO and INVALOP in IEEE mode. For divisions, INEXO HI indicates that the dividend is a DNRM; INEXO LO indicates that the divisor or both operands are DNRMs. In FAST mode, only INVALOP will be asserted. This denormal exception information becomes available with the status outputs, i.e., at the end of an attempted multicycle division or square root. In both modes for both division and square root, a properly signed all-ones NAN will be produced.

Invalid Operation and NAN Results

INVALOP is generated whenever attempting to execute an invalid operation, as defined in Std 754 Section 7.1. The INVALOP output is also used in conjunction with other pins to indicate the Division-by-Zero exception and denormal divisor or dividend. The default nontrapping result is required to be a quiet NAN. Except when passing a NAN with PASS or copying a sign bit to a NAN, the ADSP-3201/3202 chipset will always produce a NAN with an exponent and fraction of all ones as a result of an invalid operation.

Conditions that cause the assertion of INVALOP are:

- NAN input read to computational circuitry (except for logical PASS)
- Multiplication of either ±INF by either ±ZERO
- In FAST mode, multiplication of either ±INF by either ±DNRM
- Subtraction of liked-signed INFs or addition of opposite-signed INFs
- Conversion of a NAN or INF to fixed-point
- Wrapping an operand that is neither a denormal nor ZERO
- Division of either ±ZERO by either ±ZERO or of either ±INF by either ±INF
- Attempting the square root of a negative number
- In conjunction with OVRFLO, the Division-by-Zero exception
- In FAST mode, a denormal divisor or dividend. In IEEE mode, in conjunction with UNDFLO, a denormal divisor or dividend
- In conjunction with UNDFLO, a denormal input operand to square root.

Division-by-Zero

The Division-by-Zero exception is generated whenever attempting to divide a finite nonzero dividend by a divisor of zero (Std 754 Section 7.2). The Division-by-Zero exception is indicated on the ADSP-3202 ALU by the simultaneous assertion of both OVRFLO and INVALOP. The ALU result is always a correctly signed INF.

4

Overflow

OVRFLO is generated whenever the unbounded (i.e., supposing hypothetically no bounds on the exponent range of the result), post-rounded result exceeds in magnitude NORM.MAX in the destination format, as defined in Std 754 Section 7.3. Note that the overflow condition can occur both during computations and during data format conversions. The result will be either ±INF or ±NORM.MAX, depending on the sign of the result and the operative rounding mode. (See "Rounding – RND Controls" above.) The OVRFLO pin is also used to signal additional exception conditions.

Conditions that cause the assertion of OVRFLO are:

- Unbounded, post-rounded result exceeds destination format in computation or conversion
- In conjunction with INVALOP, the Division-by-Zero exception on the ADSP-3202 ALU
- Comparison when operand A is greater than operand B
- Exponent subtraction when the resultant exponent is more positive than can be represented in the destination format
- Twos-complement fixed-point additions and subtractions that overflow.

Note that OVRFLO is always LO when the ADSP-3201 Multiplier is in fixed-point mode.

Underflow

Underflow is defined in four ways in Std 754 Section 7.4. The IEEE Standard allows the implementer to choose which definition of underflow to use and provides no guidance. The first option is whether to flag underflow based on results before or after rounding. Consistent with the definition of overflow, underflow is always flagged with this chipset based on results *after* rounding (except for the operations of conversion from floating-point to fixed-point and logical downshifts). Thus, a result whose infinitely precise value is less than NORM.MIN yet which rounds to NORM.MIN will *not* be considered to have underflowed.

The second option is how to interpret what the Standard calls an "extraordinary loss of accuracy." The first way is in terms of the creation of nonzero, post-rounded numbers smaller in magnitude than NORM.MIN. The second way is in terms of loss of

accuracy when representing numbers as denormals. With the ADSP-3201/3202 chipset, the conditions under which UNDFLO is asserted depend on whether the chip in question can generate denormals in its current operating mode. If the chip cannot generate denormals, the definition in terms of numbers smaller in magnitude than NORM.MIN will apply; if it can generate denormals, the definition in terms of inexact denormals will apply. Thus, which definition applies will depend on whether chipset is operating in IEEE or FAST mode, whether the result is generated by a Multiplier or an ALU, and whether the operation is division or not.

With the ADSP-3201 Multiplier, UNDFLO is generated whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format, both in FAST and IEEE modes. In FAST mode, the data result will be ZERO; in IEEE mode, the data result will be in the wrapped format. An exact ZERO result will never cause the assertion of UNDFLO.

With the ADSP-3202 ALU in the FAST mode, UNDFLO is also generated whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format for standard ALU operations as well as for division and square root. For FAST mode underflows, the ALU result will always be ZERO. The only exception to this rule is for sums of and differences between DNRMs; if the unbounded, post-rounded, non-zero result of (DNRM ± DNRM) is of lesser magnitude than NORM.MIN in FAST, then UNDFLO will *not* be set. The ALU result will still be ZERO.

With the ADSP-3202 ALU in IEEE mode, UNDFLO is generated (except for divisions) whenever the unbounded, infinitely precise (i.e., supposing hypothetically no bounds on the precision of the result), post-rounded result is a denormal and does not fit into the denormal destination format *without a loss of accuracy*. In other words, UNDFLO will be generated whenever an inexact denormal result is produced. (See "Inexact" below.) If the result is a denormal and does fit exactly, neither UNDFLO nor INEXO will be asserted. Note that additions, subtractions, and comparisons cannot generate this underflow condition (since no operand contains significant bits of lesser magnitude than DNRM.MIN). IEEE-mode ALU underflow exceptions occur only during conversions and divisions.

The division operation is treated like a multiplication operation in IEEE mode rather than an ALU operation in the definition of underflow. A quotient from division smaller in magnitude than NORM.MIN will always be flagged as underflowed with the ADSP-3202 ALU. The data result will be in the wrapped format. Note that $\sqrt{(\overline{DNRM.MIN})} \geq NORM.MIN$. Therefore, square root will never underflow with operands greater than or equal to DNRM.MIN.

Conditions that cause the assertion of UNDFLO are:
- With the ADSP-3201 Multiplier, whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format
- With the ADSP-3202 ALU in the FAST mode, whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format
- With the ADSP-3202 ALU in IEEE mode, whenever an inexact denormal is produced or whenever the unbounded, post-rounded, nonzero quotient from division is of lesser magnitude than NORM.MIN in the destination format
- Conversions to integer if the magnitude of the floating-point source *before* rounding is less than one
- Comparison when operand A is less than operand B
- Attempting to wrap a ZERO
- Unwrapping if there is a loss of accuracy
- Exponent subtraction when the resultant exponent is more negative than can be represented in the destination format
- Logical downshift that *before* rounding would have shifted all bits out of the destination format
- In conjunction with INVALOP, a denormal divisor or dividend
- A quotient from division less than NORM.MIN
- In IEEE mode, in conjunction with INVALOP, a denormal input operand for square root.

Inexact

The inexact exception is defined in Std 754 Section 7.5 as the loss of accuracy of the unbounded, infinitely precise result when fitted to the destination format. It is signalled on the ADSP-3201/3202 chipset by INEXO.

For fixed-point operations, the ADSP-3201 Multiplier will assert INEXO HI if and only if any of the least-significant 32-bits of the pre-rounded 64-bit product are ones. It never asserts INEXO for logical operations. The ADSP-3202 ALU never asserts INEXO for fixed-point or logical operations.

In an ADSP-3202 division operation, either a denormal divisor or a denormal dividend will cause the simultaneous assertion of UNDFLO and INVALOP. INEXO will, in that context, signal which of the two was the denormal: INEXO LO indicates that the divisor is a denormal; INEXO HI indicates that the dividend is a denormal.

Conditions that cause the assertion of INEXO are:
- Loss of accuracy when fitting result to destination format
- For fixed-point operations, the prerounded multiplier 64-bit product contains ones in the least-significant 32-bits
- In IEEE mode, in conjunction with both UNDFLO and INVALOP, dividend is a denormal (HI) or divisor is a denormal or both are denormals (LO).

Less Than, Equal, Greater Than, and Unordered

For comparison operations in the ALU, the OVRFLO, UNDFLO, and INVALOP status outputs are used to indicate the four comparison conditions of IEEE Std 754, Section 5.7. They are defined as follows:
- "Less than" is signalled by the assertion of UNDFLO (while OVRFLO is LO)
- "Equal" is signalled by *not* asserting either OVRFLO or UNDFLO (i.e., both LO)
- "Greater than" is signalled by the assertion of OVRFLO (while UNDFLO is LO)
- "Unordered" is signalled by the assertion of INVALOP, caused by attempting a comparison with at least one NAN operand.

The data result from a comparison operation is identical to subtracting operand B from operand A. See Tables VIII and IX.

In IEEE comparisons, the data types are always ordered in ascending sequence: −INF, −NORM, −DRNM, ZERO, DNRM, NORM and INF. Comparisons between like signed INFs will generate the "Equal" status condition. Comparisons between signed ZEROs will also generate the "Equal" status. Any comparison to a NAN will also cause INVALOP and produce an all-ones NAN. Even in FAST mode, DNRMs will be compared based on their true value (rather than all being treated as ZEROs).

Special Flags for Unwrapping

The ADSP-3201 generates a Round Carry Propagation Out flag,

RNDCARO, that indicates whether or not a carry bit propagated into the destination formats fraction during the Multipliers floating-point rounding operation. The rounding that the Multiplier does in creating the wrapped or unnormal result may cause a carry bit into the LSB in the destinations formats fraction. This rounding position will not in general be correct for a properly rounded denormal. Thus, when the underflowed Multiplier result is unwrapped to a denormal, the ALU has to undo the Multipliers rounding and re-round to achieve the properly rounded denormal.

To do this, the ALU has to know if any carry bits in the Multiplier's rounding operation propagated into the fraction of the result. This information is provided in the Multiplier's RNDCARO flag. The ALU also needs to know if the Multiplier's rounded result caused a loss of accuracy when expressed in its destination wrapped format, indicated by the Multiplier's Inexact Result (INEXO) flag.

The ADSP-3202 ALU has a corresponding pair of flag status input pins: Round Carry Propagation In (RNDCARI) and Inexact Data In (INEXIN). In an unwrap operation, these flags are used by the ALU when converting from a WNRM to a DNRM to obtain the properly rounded result. RNDCARI and INEXIN should be setup to the ALU with the instruction for the unwrap operation. Both Multiplier and ALU must be using the same rounding mode.

The ADSP-3202 ALU itself generates WNRMs in underflowed division operations. These WNRMs must be fed back to the ALU to be unwrapped to DNRMs. The ADSP-3202, unlike the Multiplier, does not have a RNDCARO pin to signal whether or not a carry bit propagated into the destination format on rounding. For this reason, WNRMs produced by the ADSP-3202 ALU in division are rounded differently than they are on the Multiplier; underflowed (only) quotients are always truncated (Round-toward-Zero) to the destination wrapped format. Hence there *is* no carry bit propagation. When unwrapping a WNRM produced in division, RNDCARI should always be held LO. INEXIN should reflect the status of INEXO when the ALU produced the underflowed wrapped quotient.

The ADSP-3202 ALU also uses the RNDCARI and INEXIN pins to indicated wrapped A and B operands, respectively, to division and square root operations. Both RNDCARI and INEXIN should be held LO except for unwrap, division, and square root operations.

INSTRUCTIONS AND OPERATIONS

The ADSP-3201 Multiplier executes the same instruction every cycle: multiply. It need not be specified explicitly in microcode. The data format of results and status flags from multiplication are shown in Tables VI and VII.

Denormal input operands will generally cause the DENORM exception (see "Status Flags" above) and correctly signed ZERO results. FAST mode suppresses the DENORM exception. In either FAST or IEEE, DNRM•ZERO will be ZERO without exception. DNRM•INF will be a correctly signed INF without exception in IEEE mode and a NAN and INVALOP in FAST mode. DNRM•NAN will be a correctly signed NAN with INVALOP asserted. The sign bit of the NAN generated from any invalid operation will depend on the operands. (The IEEE Standard does not specify conditions for the sign bit of a NAN.) On the ADSP-3201 Multiplier, the sign of a NAN result will be the exclusive OR of the signs of the input operands.

The product of INF with anything except ZERO or NAN is a correctly signed INF. INF•ZERO will cause INVALOP and yield a NAN. NAN times anything will also cause INVALOP and yield a NAN.

The ADSP-3202 ALU, in contrast to the Multiplier, is instruction-driven with the operation specified by I_{8-0}. The ALU instructions fall into three categories: Fixed-Point, Logical, and Single-Precision Floating-Point. Instructions are summarized in Tables III through V and described below. The data format of results and status flags from the various ALU operations are shown in Tables VIII and IX. Division is shown in Tables X and XI; square root in Table XII. Conversions from single-precision floating-point to two-complement integer are illustrated in Table XIII.

The ADSP-3202 Fixed-Point Arithmetic Operations are:

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
IADD	001	000	011	Fixed-point A + B
ISUBB	001	001	011	Fixed-point A − B
ISUBA	001	000	111	Fixed-point B − A
IADDWC	001	010	011	Fixed-point A + B with carry
ISUBWBB	001	011	011	Fixed-point A − B with borrow
ISUBWBA	001	010	111	Fixed-point B − A with borrow
INEGA	001	000	101	Fixed-point − A. ABSA/B must be LO.
INEGB	001	001	010	Fixed-point − B. ABSA/B must be LO.
IADDAS	001	100	011	Fixed-point |A + B|
ISUBBAS	001	101	011	Fixed-point |A − B| ABSA/B must be LO.
ISUBAAS	001	100	111	Fixed-point |B − A| ABSA/B must be LO.

Table III. ADSP-3202 Fixed-Point ALU Operations

The ADSP-3202 Logical Operations are:

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
COMPLA	000	000	101	Ones-complement A
COMPLB	000	001	010	Ones-complement B
PASSA	000	000	001	Pass A unmodified. Set no flags.
PASSB	000	000	010	Pass B unmodified. Set no flags.
AANDB	000	010	010	Bitwise logical AND
AORB	000	100	010	Bitwise logical OR
AXORB	000	110	010	Bitwise logical XOR
NOP	000	000	000	No operation. Preserve status flags and Output contents.
CLR	100	000	000	Clear all status flags. Data register contents are unaffected.

Table IV. ADSP-3202 ALU Logical Operations

The ADSP-3202 Single-Precision Floating-Point Operations are:

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
SADD	111	000	011	SP FltgPt (A + B)
SSUBB	111	000	111	SP FltgPt (A − B)
SSUBA	111	001	011	SP FltgPt (B − A)
SCOMP	111	001	111	SP FltgPt comparison of A to B. Result is (A − B) Greater Than=OVRFLO HI Equal=(OVRFLO LO & UNDFLO LO) Less Than=UNDFLO HI Unordered=INVALOP HI
SADDAS	011	000	011	SP FltgPt |A + B|
SSUBBAS	011	000	111	SP FltgPt |A − B|
SSUBAAS	011	001	011	SP FltgPt |B − A|
SFIXA	011	001	101	Convert SP FltgPt A to twos-complement Integer
SFIXB	011	001	110	Convert SP FltgPt B to twos-complement Integer
SFLOATA,	011	100	101	Convert twos-complement integer A to SP FltgPt
SFLOATB	011	100	110	Convert twos-complement integer B to SP FltgPt
SPASSA	011	110	001	Pass SP FltgPt A. NANs cause INVALOP.
SPASSB	011	110	010	Pass SP FltgPt B. NANs cause INVALOP.
SWRAPA	011	100	001	Wrap SP DNRM A to SP WNRM
SWRAPB	011	100	010	Wrap SP DNRM B to SP WNRM
SUNWRAPA	011	010	001	Unwrap SP WNRM A to SP DNRM
SUNWRAPB	011	010	010	Unwrap SP WNRM B to SP DNRM
SSIGN	011	111	101	Copy sign from SP FltgPt B to SP FltgPt A. Result is [sign B, exponent A, fraction A].
SXSUB	011	111	001	Subtract B exponent from A exponent. Result is [sign A, (expt A − expt B), fraction A] for all data types. If the unbiased exponent ≥ + 128, INF results. If the unbiased exponent is ≤ − 127, ZERO results.
SITRN	011	010	101	Downshift SP FltgPt A mantissa (with hidden bit) logically by the unbiased SP FltgPt B exponent to a 32-bit unsigned-magnitude integer. Use RZ only.
Use RZ only:				
SDIV	011	110	111	SP FltgPt (A ÷ B)
SSQR	111	110	110	SP FltgPt $\sqrt{B}$

Table V. ADSP-3202 ALU Single-Precision Floating-Point Operations

Fixed-Point Arithmetic ALU Operations

The negation operation is a twos-complementing of the input operand.

The OVRFLO flags can be set by fixed-point ALU operations. The twos-complement data format is presumed in the definition of fixed-point overflow.

Absolute Value Controls

Absolute value controls (ABSA/B) cannot be used with all operands input to all fixed-point ALU operations. ABSA/B must be LO for negation (INEGA/B) and absolute difference (ISUBBAS/ ISUBAAS) operations, or results will be undefined. Absolute value controls can be used with all other fixed-point operations.

Extended-Precision Fixed-Point Arithmetic
The ADSP-3202's integer ALU operations include three operations for extended fixed-point precision: addition with carry and two subtractions with borrow. The carry bit generated by an addition or subtraction is latched internally for one cycle only.

To illustrate, these instructions can be used to add two 64-bit fixed-point numbers. The two least-significant 32-bit halves can be added with IADD. Any carry bit generated would be latched internally in the ADSP-3202. On the next cycle, the most-significant 32-bit halves can be added with IADDWC, which would also add in the carry bit from the previous operation, if any. The two fixed-point results will be latched in the Output Register in consecutive cycles. As with all fixed-point results, they will appear in consecutive cycles in the most-significant 32-bits of the Output Register (bit positions 63 through 32).

Extended-precision fixed-point subtraction is exactly analogous. The least-significant 32-bit halves can be subtracted with either ISUBA or ISUBB. On the next cycle, the most-significant 32-bit halves can be subtracted with either ISUBWBA or ISUBWBB.

Fixed-Point Zero and Equality Tests
The ADSP-3202 do not directly support fixed-point zero-test or comparison operations. However, both can be accomplished using other ALU operations. A zero-test will result from executing a single-precision floating-point wrap instruction (SWRAPA/B) on the fixed-point data in question. UNDFLO will be asserted if and only if the operand is ZERO, which is bitwise equivalent to an operand of all zero bits.

A fixed-point test for equality will result from a bitwise XOR of A and B operands (AXORB) followed by the zero-test using SWRAPA/B described in the previous paragraph. In this context, UNDFLO will flag fixed-point equality.

Logical ALU Operations
The ones-complement instructions (COMPLA/B) change every one bit in the operand to a zero bit and every zero bit in the operand to a one bit. Ones-complementing is equivalent to a bitwise logical NOT operation on the 32-bit operand. The pass instructions (PASSA/B) pass all operands unmodified, including NANs, without signaling an INVALOP exception. PASSA/B set no flags.

The logical AND, OR, and XOR (AANDB, AORB, AXORB) operate bitwise on all 32-bits in their pair of operand fields to produce a 32-bit result.

NOP will advance the ALU pipeline one cycle. Status flags and Output Register contents will be preserved. CLR simply resets all status flags. Note that CLR is pipelined and takes effect one cycle after it is presented. All data register contents, including the Output Register, remain unaffected.

Do not assert the absolute value controls (ABSA/B) with logical operations. The results will be undefined.

Floating-Point ALU Operations
The data types and flags resulting from single-precision floating-point additions, subtractions, comparisons, absolute sums, and absolute differences are shown in Tables VIII and IX. The INEXO flag is not shown explicitly in these tables (or any other) since it may or may not be set, depending on whether the result is inexact.

Absolute Value Controls
Absolute value controls (ABSA/B) can be used with all operands input to all floating-point ALU operations.

Sign of NAN Results
On the ADSP-3222, the sign of a NAN resulting from any operation (except division) involving at least one NAN operand will be the sign which would be produced if the magnitude portion (sign plus fraction) of the NAN operand(s) were treated as normal numbers.

Some ALU operations with two INF inputs can cause INVALOP and generate NANs. The assignment of sign to the NAN is analogous to additions with signed zeros:

$(\pm\text{INF})+(\pm\text{INF})=(\pm\text{INF})-(\mp\text{INF})\rightarrow\pm\text{INF}$
$(\pm\text{INF})+(\mp\text{INF})=(\pm\text{INF})-(\pm\text{INF})\rightarrow+\text{NAN}$
(RN, RZ, RP rounding modes)
$(\pm\text{INF})+(\mp\text{INF})=(\pm\text{INF})-(\pm\text{INF})\rightarrow-\text{NAN}$
(RM rounding mode)

In this notation, the first line refers to either $+\text{INF}+\text{INF}$ or $-\text{INF}-\text{INF}$. The second and third lines refer to $+\text{INF}-\text{INF}$ or $-\text{INF}+\text{INF}$.

Comparisons
Comparison generates the data result, (operand A minus operand B). The flags, however, are defined to indicate the comparison conditions rather than the flag conditions for subtraction. Signed INFs will be compared as expected. A NAN input to the comparison operation will cause the unordered flag result (INVALOP) and the production of an all-ones NAN. Even in FAST mode, the ALU will accept denormals as inputs to the comparison operation. See "Less Than, Equal, Greater Than, and Unordered" in the "Status Flag" section above for a complete discussion of these flags in comparison operations.

Conversions: Floating to Fixed
Conversions from floating-point to twos-complement integer (SFIXA/B) are considered "floating-point" operations, and all four rounding modes are available. If the operand *after rounding* overflows the destination format, OVRFLO will be set, and the results will be undefined. Thus, OVRFLO for fixed-point operations is treated exactly as it is for floating-point operations.

If the nonzero operand *before rounding* is of magnitude less than one, UNDFLO will be set in a conversion to integer. The magnitude of the result may be either one or zero, depending on the rounding mode. Conversion to integer is the only operation where UNDFLO depends on the *pre*-rounded result. The reason for this is that the infinitely precise result could be almost one integer unit away from the post-rounded result, potentially a large difference. We have chosen to flag underflow whenever the magnitude of the source operand is less than one, thereby alerting the user to a potentially significant loss of accuracy.

INEXO will be asserted if the conversion is inexact. NANs and INFs will convert to a same-signed single-precision floating-point all-ones NAN. INVALOP will be asserted. The twos-complement integer interpretation of $+\text{NAN}$ is full-scale positive and of $-\text{NAN}$, minus one. See Table XIII for illustrations of fixing single-precision floating-point numbers.

Conversions: Fixed to Floating
All four rounding modes are also available for conversions from twos-complement integer to floating-point. For conversion to single-precision floating-point (SFLOATA/B), the numerical result will always be IEEE normals. The only flag ever set is INEXO. INEXO will be set if and only if the source integer contains more than 24 bits of significance. "Significance" is defined as follows: For positive twos-complement integers, the number of significant bits is ([32 minus the number of leading zeros] minus the number of trailing zeros). "Leading zeros" are the contiguous string of zeros starting from the most significant bit. "Trailing zeros" are the contiguous string of zeros starting from the least significant bit. For negative twos-complement integers, the number of significant bits is ([33 minus the number of leading ones] minus the number of trailing zeros).

Pass
Pass instructions (SPASSA/B) pass all operands unmodified. Unlike the PASSA/B instructions, the floating-point pass instructions will cause INVALOP if a NAN is passed. The NAN will pass unmodified. INFs are passed without setting any flags. The absolute value controls can be used with the floating-point pass instructions to reset the unmodified NAN's sign bit to zero.

Wrap
Wrap instructions (SWRAPA/B) convert a denormal to a wrapped number readable by a Multiplier or the ADSP-3202 ALU in division and square root operations. Since the wrapped format has an additional bit of precision (the hidden bit), all wrapping is exact. If the operand is ZERO, then UNDFLO will be set. If the operand is neither a DNRM nor ZERO, INVALOP will be set.

Unwrap
Unwrapping instructions (SUNWRAP/B) convert a wrapped number to the IEEE denormal format. After rounding, the result may turn out to be NORM.MIN or ZERO. WRAP.MAX, whose infinitely precise value is between NORM.MIN and DNRM.MAX, will round to NORM.MIN or DNRM.MAX, depending on rounding mode:

+WRAP.MAX→NORM.MIN (RN, RP modes)
+WRAP.MAX→DNRM.MAX (RZ, RM modes)
−WRAP.MAX→NORM.MIN (RN, RM modes)
−WRAP.MAX→DNRM.MAX (RZ, RP modes).

INEXO will always be set when unwrapping WRAP.MAX. If the unwrapping operation, after rounding, shifts all ones out of the DNRM destination format, ZERO will result. Whenever this happens, UNDFLO and INEXO will always both be set.

The UNDFLO condition for unwrapping is based on the IEEE definition in terms of loss of accuracy when representing a denormal (see "Underflow" in "Status Flags" above.) That is, UNDFLO will only be set when the unbounded, post-rounded result cannot be expressed exactly in the destination denormal format. UNDFLO will always be set in conjunction with INEXO when unwrapping.

Inexactness can be caused by a loss of accuracy when unwrapping the operand supplied to the ALU. The ADSP-3202 also considers whether the multiplication, division, or square root that generated the wrapped number caused a loss of accuracy. It determines this information by reading the INEXIN flag input to the ALU.

The INEXIN is essential to the unwrapping operation in the ALU. The state of INEXIN input when wrapping should reflect the state of INEXO when the wrapped number was generated during multiplication, division, or square root. The ADSP-3202 uses this information to determine if the operation creating the wrapped number was inexact. When the ADSP-3202 unwraps a wrapped number, its INEXO will be asserted if *either* the originating operation or the unwrapping operation caused a loss of accuracy.

Copy Sign
The SSIGN operation copies the sign of the B operand to the A operand. The result is (sign B, exponent A, fraction A). Rounding modes have no effect on this operation since the precision of the result is exactly that of the source, i.e., all "roundings" are exact. The only condition that generates a flag is a NAN as the A operand; INVALOP will be set. This instruction is useful for quadrant normalization of trigonometric functions. Trigonometric identities allow mapping an angle of interest to a quadrant for which lookup tables exist. SSIGN simplifies this mapping. For example, $\sin(-37°) = -\sin(37°)$. By looking up sin (37°) and transferring the sign of the angle (−37°, the B operand) to the value from the lookup table (0.60182, the A operand), the correct result is obtained (−0.60182).

Exponent Subtraction
Exponent subtraction (SXSUB) subtracts the exponent of the B operand from the A operand. The A operand is the destination format: (sign A, [expt A − expt B], fraction A). INFs and NANs are valid inputs to the SXSUB operation; INVALOP is never asserted. If the unbounded result is greater than that of NORM.MAX, INF will be produced and OVRFLO will be set. If the unbounded result is less than that of NORM.MIN, ZERO will be produced and UNDFLO will be set.

Exponent subtraction is useful as the first step in the Newton-Raphson division by recursion algorithm. This operation allows an improved implementation of this algorithm. For the details, see the Application Note, "Floating-Point Division using Analog Devices ADSP-3210 and ADSP-3220", available from Analog Devices' DSP Applications Engineering.

Logical Downshift
The mantissa of a floating-point A operand (with hidden bit restored) can be downshifted logically to an unsigned-magnitude integer destination format using the SITRN operation (see Figure 22). The source mantissa is treated as a right-justified unsigned integer. The unbiased (i.e., the "true" exponent after the bias has been subtracted) exponent of the B operand determines the amount of the downshift. The unbiased B exponent is interpreted as an unsigned number which indicates how many bit positions the mantissa should be downshifted. (A negative unbiased exponent will cause a very large downshift. The mantissa will be completely shifted out of range, and the result will be zero.) The result will a be left-zero-filled unsigned-magnitude integer. Like all fixed-point results, it will appear in the most significant bit positions of the Output Register.

Logical downshift is only defined for NORMs. Results from operands that are not normals are undefined. A NAN A-operand input to SITRN will cause INVALOP and produce all-ones NANs of the same sign. Round-toward-Zero (RZ) must be specified for SITRN. Otherwise, the result is undefined. If the shifted result *before rounding* is all zeros, UNDFLO will be set. (Actually, with RZ, the shifted result before rounding is the same as the shifted result after rounding.) If any bits are shifted out of the range of the destination format, INEXO will be set.

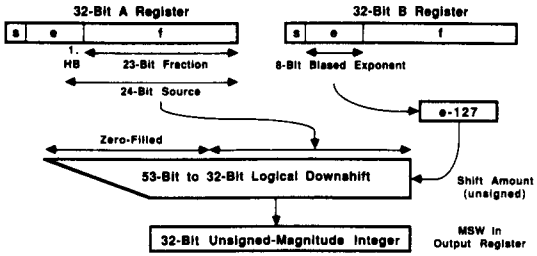


Figure 22. ADSP-3202 SITRN Instruction

The logical downshift operations can be useful to generate table lookup addresses. In this application, the most-significant mantissa bits would be used as table addresses. Because different B exponents can be applied to the same A mantissa, the same datum can be used to address multiple tables with differently sized address fields.

Division and Square Root

The ADSP-3202 ALU support multicycle division (SDIV) and square root (SSQR) operations. Tables X and XI illustrate the resultant data types and status conditions for division. Table XII serves a similar role for square root. Neither operation can accept denormal inputs directly; they must be wrapped to the wrapped data format first. Denormal inputs to division and square root operations will cause the simultaneous assertion of UNDFLO and INVALOP in IEEE mode. For divisions, INEXO HI indicates that the dividend is a DNRM; INEXO LO indicates that the divisor or both operands are DNRMs. In FAST mode, only INVALOP will be asserted. In both modes for both division and square root, a properly signed all-ones NAN will be produced.

The square root of any non-negative normal or wrapped number will be an IEEE normal number. The square root of a negative number is an all-ones −NAN. The square root of +INF is +INF without exception. The square root of a NAN is a same-signed all-ones NAN.

Division can produce wrappeds and unnormals; these must be passed back to the ALU for unwrapping. INF dividends cause correctly signed INFs without flags except when the divisor is also an INF. Either ±INF divided by either ±INF or any NAN input will generate INVALOP and an all-ones NAN. For ADSP-3202 division operations, the sign of the NAN will be the exclusive OR of the signs of the dividend and the divisor.

OUTPUT CONTROL – SHLP (REG), OEN (ASYN), MSWSEL (ASYN), and HOLD (ASYN)

Both members of the ADSP-3201/3202 chipset have a 64-bit Output Register. The Output Registers are clocked every cycle, except for multi-cycle operations (division and square root), when HOLD is LO on the ADSP-3201, and when the ADSP-3202 is executing NOP. Output Registers are clocked at the conclusion of multicycle operations and not before.

Results appear in the Multiplier's Output Register as follows:

Bit 63 ... 32	31 ... 0
SP FltgPt Product	not meaningful
FxdPt Most Significant Product	FxdPt Least Significant Product

Figure 23. ADSP-3201 Multiplier Output Register

When the destination format from multiplication is single-precision floating-point, the fraction bits that are less than the least-significant bit in the destination format are stored in the least-significant half of the Output Register.

The Multiplier has a pipelined, registered fixed-point shift-left control, SHLP. When HI, SHLP will cause a one-bit left shift in the 64-bit product that appears in the Multiplier's Output Register. The least-significant bit in the Output Register will be zero. See "32-Bit Fixed-Point Data Formats" above for more details of the effects of SHLP. SHLP has no effect on floating-point multiplications. Note that SHLP should be setup at the clock edge when the multiplication operands are read into the multiplier array.

Results appear in the ALU's Output Registers as follows:

Bit 63 ... 32	31 ... 0
SP FltgPt Product	not meaningful
FxdPt Result	not meaningful

Figure 24. ADSP-3202 ALU Output Register

All members of this chipset have an asynchronous output enable control, OEN. When HI, outputs are enabled; when LO, output drivers at $DOUT_{31-0}$ are put into a high-impedance state. Note that status flags are always driven off-chip, regardless of the state of OEN. See Figure T1 for the timing of OEN.

All members of this chipset also have an asynchronous MSW select control, MSWSEL. When outputs are enabled and MSWSEL is HI, the most-significant half (bits 63 through 32) of the Output Register will be driven to the output port, $DOUT_{31-0}$. When outputs are enabled and MSWSEL is LO, the least-significant half (bits 31 through 0) of the Output Register will be driven to the output port, $DOUT_{31-0}$. The operation of MSWSEL is illustrated in all timing diagrams where 64-bit outputs are produced.

The ADSP-3201 Multiplier has an asynchronous, active LO control, $\overline{HOLD}$, that prevents the Output Register from being updated. $\overline{HOLD}$ must be set up prior to the clock edge when the Output Register would have otherwise been updated. See Figure T3. For normal operations where the Output Register is updated, $\overline{HOLD}$ must be held HI.

TIMING

Timing diagrams are numbered Figures T1 through T7. Three-state timing for DOUT is shown in Figure T1. Output disable time, t_{DIS}, is measured from the time OEN reaches 1.5V to the time when all outputs have ceased driving. This is calculated by measuring the time, $t_{measured}$, from the same starting point to when the output voltages have changed by 0.5V toward +1.5V. From the tester capacitive loading, C_L, and the measured current, i_L, the decay time, t_{DECAY}, can be approximated to first order by:

$$t_{DECAY} = \frac{C_L \cdot 0.5V}{i_L}$$

from which

$$t_{DIS} = t_{measured} - t_{DECAY}$$

is calculated. Disable times are longest at the highest specified temperature.

The minimum output enable time, minimum t_{ENA}, is the earliest that outputs begin to drive. It is measured from the control

signal OEN reaching 1.5V to the point at which the fastest outputs have changed by 0.1V from $V_{tristate}$ toward their final output voltages. Minimum enable times are shortest at the lowest specified temperature.

The maximum output enable time, maximum t_{ENA}, is also measured from OEN at 1.5V to the time when all outputs have reached TTL input levels (V_{OH} or V_{OL}). This could also be considered as "data valid." Maximum enable times are longest at the highest specified temperature.

Reset timing is shown in T2. RESET must be LO for at least t_{RS}. In addition, RESET must return HI at least t_{SU} before the first rising clock edge of operation. Hold timing is shown in T3. $\overline{HOLD}$ must go LO t_{HS} before the rising edge at which the Output Register is *not* updated. $\overline{HOLD}$ must also be held t_{HH} after the clock edge.

All data, registered and latched controls, and instructions shown in T4 through T7 must be set up t_{DS} before the rising edge and held t_{DH}. Both input-port configurations are shown in most of these diagrams. Data is shown loaded for minimum latency. Other sequencing options are possible and may be more convenient, depending on the system. These other options, however, require that data be loaded to the input registers earlier than as shown in these diagrams and not overwritten. See "Input Register Loading and Operand Storage" above for constraints on register loading and operand storage that must be observed.

The operation time, t_{OPD}, is the time required to advance the internal pipelines one stage. It reflects the pipelined throughput of the device for that operation. The latency, t_{LAD}, is the time it takes for the chip to produce a valid result at DOUT from valid data at its input ports. (Latency is the true measure of the internal speed of the chip.) Latency is referenced from data valid of the earliest required input to data valid of the first 32-bit output.

The asynchronous MSWSEL control's delay is t_{ENO}. The maximum specification for t_{ENO} is the delay which guarantees valid data. The minimum specification for t_{ENO} is the earliest time after the MSWSEL control is changed that data can change.

Status flags have a maximum output delay of t_{SO} referenced from the clock rising edge. All status flags except the Multiplier's DENORM are available in parallel with their associated output results. DENORM is available earlier to speed up recovery from a denormal input exception. Note that DENORM is indeterminate (not necessarily LO) except in the cycles indicated in T4. DENORM should therefore not be used by itself to externally trigger a denormal input exception processing routine.

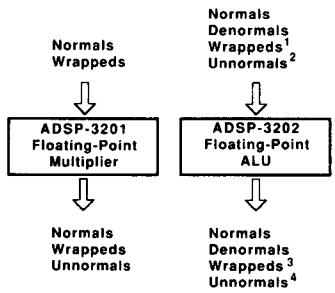
Note that for all operations (Figures T4 through T7) a new operation can begin the cycle before output results and status flags (other than DENORM) results from the previous operation are driven off chip. This feature leads to improved pipeline throughput.

GRADUAL UNDERFLOW AND IEEE EXCEPTIONS

The data types that each chip operates on directly is shown in Figure 25.

Denormals are detected by the Multiplier when read into their processing circuitry. The ADSP-3201 will produce a flag output, DENORM, when one or both of the operands read into the array are denormals. The occurrence of DENORM should trigger exception processing. (See Status Flags above for a discussion of DENORM and its timing.) Controlling hardware must recover the denormal(s) that was input to a Multiplier and present it to an ALU for wrapping.

The ADSP-3202 ALU will also detect denormals when read into internal circuitry for division or square root operations. The



1. for unwrapping, division, and square root
2. for unwrapping only
3. from wrapping and division
4. from division

Figure 25. Data Types Directly Supported by the ADSP-3201/3202

UNDFLO and INVALOP flags will both be asserted on the ADSP-3202 to signal the presence of a denormal input to these operations. INEXO will indicate whether the denormal input is the A operand or B operand. (See "Status Flags" above for a fuller discussion of denormal detection in the ADSP-3202.)

The ALU wraps denormals with its SWRAP instruction. Note from Table II that any denormal can be represented as a wrapped without loss of precision (hence triggers no exception flags in the ALU).

The wrapped equivalent from the ALU must now be passed to the Multiplier for multiplication or the ADSP-3202ALU for division or square root. The controlling system must tell the Multiplier to interpret the wrapped input as wrapped by asserting WRAPA/B when it is read into the Multiplier's processing circuitry. For division and square root, the controlling system must tell the ALU to interpret the wrapped operand A as wrapped by asserting INEXIN when it is read into the ALUs processing circuitry and to interpret the wrapped operand B as wrapped by asserting RNDCARI. The result of the multiplication or division can be a normal, a wrapped, or an unnormal (see Tables VI, VII, X, and XI). Square root on IEEE numbers only produces normals (see Tables VIII and IX). An underflowed result (wrapped or unnormal) from either Multiplier or ALU will be indicated by the UNDFLO flag and must be passed to the ALU for unwrapping.

For full conformance to the IEEE Standard, all wrapped and unnormal results must be unwrapped in an ALU (with the SUNWRAP instruction) to an IEEE sanctioned destination format before any further operations on the data. If the result from unwrapping is a DNRM, then that data will have to be wrapped before it can be used in multiplication, division, or square root operations.

The reason why WNRMs and UNRMs should always be unwrapped upon their production is that the wrapped and unnormal data formats often contain "spurious" accuracy, i.e., more precision than can be represented in the normal and denormal data formats. If WNRMs or UNRMs produced by the system were used directly as inputs to multiplication, division, or square root operations, the results could be more accurate than, and hence incompatible with, the IEEE Standard.

When unwrapping, additional information about underflowed results must accompany their input to the ALU. See "Special Flags for Unwrapping" in "Status Flags" above for details of how INEXO and RNDCARO status flag outputs must be used with INEXIN and RNDCARI inputs.

A final point about conformance with IEEE Std 754 pertains to NANs. The Standard distinguishes between signalling NANs and quiet NANs, based on differing values of the fraction field. Signalling NANs can represent uninitialized variables or specialized data values particular to an implementation. Quiet NANs provide diagnostic information resulting from invalid data or results. The ADSP-3201/3202 generally produce all-ones outputs from invalid operations resulting from NAN inputs. So a system that implements operations on quiet and signalling NANs will have to modify the NAN output from these chips externally. See Section 6.2 of Std 754-1985 for the details of these operations.

A operand \ B operand	ZERO		DNRM		WRAP		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	ZERO		ZERO		ZERO		ZERO		NAN	INVALOP	NAN	INVALOP
DNRM	ZERO		ZERO	DENORM	ZERO	DENORM	ZERO	DENORM	INF		NAN	INVALOP
WRAP	ZERO		ZERO	DENORM	UNRM	UNDFLO	NORM WRAP UNRM	 UNDFLO UNDFLO	INF		NAN	INVALOP
NORM	ZERO		ZERO	DENORM	NORM WRAP UNRM	 UNDFLO UNDFLO	INF,NORM.MAX[1] NORM WRAP	OVRFLO UNDFLO	INF		NAN	INVALOP
INF	NAN	INVALOP	INF		INF		INF		INF		NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."

Table VI. ADSP-3201 Floating-Point Multiplication (IEEE Mode)

A operand \ B operand	ZERO		DNRM		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	ZERO		ZERO		ZERO		NAN	INVALOP	NAN	INVALOP
DNRM	ZERO		ZERO	DENORM	ZERO	DENORM	NAN	INVALOP	NAN	INVALOP
NORM	ZERO		ZERO	DENORM	INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	INF		NAN	INVALOP
INF	NAN	INVALOP	INF	INVALOP	INF		INF		NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."
2. In FAST mode, WRAP inputs are illegal.

Table VII. ADSP-3201 Floating-Point Multiplication (FAST Mode)

A operand \ B operand	ZERO		DNRM		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	ZERO[2]		DNRM		NORM		INF		NAN	INVALOP
DNRM	DNRM		NORM DNRM ZERO		INF,NORM.MAX[1] NORM DNRM	OVRFLO	INF		NAN	INVALOP
NORM	NORM		INF,NORM.MAX[1] NORM DNRM	OVRFLO	INF,NORM.MAX[1] NORM DNRM ZERO	OVRFLO	INF		NAN	INVALOP
INF	INF		INF		INF		INF[3] NAN[3]	INVALOP	NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."
2. (± ZERO)+ (± ZERO) = (±ZERO)–(∓ZERO) ⇒ ± ZERO
   (± ZERO)+(∓ ZERO) = (±ZERO)–(±ZERO)⇒ + ZERO (RN, RZ, RP rounding modes)
   (± ZERO)+(∓ ZERO) = (±ZERO)–(±ZERO)⇒ – ZERO (RM rounding mode)
3. (± INF)+(± INF) = (±INF)–(∓INF) ⇒ ± INF
   (± INF)+(∓ INF) = (±INF)–(±INF) ⇒ +NAN (RN, RZ, RP rounding modes)
   (± INF)+(∓ INF) = (±INF)–(±INF) ⇒ –NAN (RM rounding mode)
4. If DNRM result is inexact, UNDFLO will be set.

Table VIII. ADSP-3202 Floating-Point Addition/Subtraction (IEEE Mode)

A operand \ B operand	ZERO		DNRM		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	ZERO[2]		ZERO		NORM		INF		NAN	INVALOP
DNRM	ZERO		NORM ZERO		INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	INF		NAN	INVALOP
NORM	ZERO		INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	INF,NORM.MAX[1] NORM ZERO[4] ZERO[4]	OVRFLO UNDFLO	INF		NAN	INVALOP
INF	INF		INF		INF		INF[3] NAN[3]	INVALOP	NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."
2. ± ZERO ± ZERO ⇒ ± ZERO
   ± ZERO ∓ ZERO ⇒ + ZERO (RN, RZ, RP rounding modes)
   ± ZERO ∓ ZERO ⇒ – ZERO (RM rounding mode)
3. ± INF ± INF ⇒ ± INF
   ± INF ∓ INF⇒ +NAN (RN, RZ, RP rounding modes)
   ± INF ∓ INF⇒ –NAN (RM rounding mode)
4. Exact result.

Table IX. ADSP-3202 Floating-Point Addition/Subtraction (FAST Mode)

A operand \ B operand	ZERO		DNRM		WRAP		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	NAN	INVALOP	ZERO		ZERO		ZERO		ZERO		NAN	INVALOP
DNRM	INF[1]	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	NAN	UNDFLO INVALOP	NAN	UNDFLO INVALOP	ZERO		NAN	INVALOP
WRAP	INF[1]	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	NORM		NORM WRAP UNRM	UNDFLO UNDFLO	ZERO		NAN	INVALOP
NORM	INF[1]	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	INF,NORM.MAX[1] NORM	OVRFLO	INF,NORM.MAX[1] NORM WRAP UNRM	OVRFLO UNDFLO UNDFLO	ZERO		NAN	INVALOP
INF	INF		INF		INF		INF		NAN	INVALOP	NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."

Table X. ADSP-3202 Floating-Point Division (A÷B) (IEEE Mode)

4

A operand \ B operand	ZERO		DNRM		NORM		INF		NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
ZERO	NAN	INVALOP	NAN	INVALOP	ZERO		ZERO		NAN	INVALOP
DNRM	NAN	INVALOP	NAN	INVALOP	ZERO		ZERO		NAN	INVALOP
NORM	INF[1]	OVRFLO& INVALOP	INF[1]	OVRFLO& INVALOP	INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	ZERO		NAN	INVALOP
INF	INF		INF		INF		NAN	INVALOP	NAN	INVALOP
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP

1. Either INF or NORM.MAX, depending on rounding mode. See "Round Controls."

Table XI. ADSP-3202 Floating-Point Division (A÷B) (FAST Mode)

Mode \ B operand	B<ZERO		±ZERO		+DNRM		+WRAP		+NORM		+INF		±NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
IEEE	–NAN	INVALOP	±ZERO		+NAN	UNDFLO& INVALOP	NORM		NORM		+INF		±NAN	INVALOP
FAST	–NAN	INVALOP	±ZERO		+ZERO		NORM		NORM		+INF		±NAN	INVALOP

Table XII. ADSP-3202 Floating-Point Division Square Root $\sqrt{B}$)

Sign	HB	f22 . . . f1	f0		Unbiased Expnt	Source Name	Sign	i30	i29	i28	i27	i26	i25	i24	i23	i22 . . .	i7	i6	i5	i4	i3	i2	i1	i0	Rounding Modes	Status Flags
0	1	X . . . X	X	2**	128	+ NAN	0	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP
0	1	0 . . . 0	0	2**	128	+ INF	0	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP
0	1	0 . . . 0	0	2**	31		U*	U	U	U	U	U	U	U	U	U . . .	U	U	U	U	U	U	U	U	all	OVRFLO
0	1	1 . . . 1	1	2**	30		0	1	1	1	1	1	1	1	1	1 . . .	1	0	0	0	0	0	0	0	all	
0	1	1 . . . 1	1	2**	23		0	0	0	0	0	0	0	0	1	1 . . .	1	1	1	1	1	1	1	1	all	
0	1	0 . . . 0	0	2**	23		0	0	0	0	0	0	0	0	1	0 . . .	0	0	0	0	0	0	0	0	all	
0	1	1 . . . 1	1	2**	22		0	0	0	0	0	0	0	0	1	0 . . .	0	0	0	0	0	0	0	0	RN,RP	INEXO
0	1	1 . . . 1	1	2**	22		0	0	0	0	0	0	0	0	0	1 . . .	1	1	1	1	1	1	1	1	RZ,RM	INEXO
0	1	0 . . . 0	0	2**	0	one	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	all	
0	1	1 . . . 1	1	2**	− 1	one − 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	RN, RP	UNDFLO,INEXO
0	1	1 . . . 1	1	2**	− 1	one − 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RZ,RM	UNDFLO,INEXO
0	1	0 . . . 0	1	2**	− 1	1/2 + 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	RN,RP	UNDFLO,INEXO
0	1	0 . . . 0	1	2**	− 1	1/2 + 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RZ,RM	UNDFLO,INEXO
0	1	0 . . . 0	0	2**	− 1	1/2	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	RP	UNDFLO,INEXO
0	1	0 . . . 0	0	2**	− 1	1/2	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RM,RN,RZ	UNDFLO,INEXO
0	1	0 . . . 0	0	2**	− 126	+ NORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	RP	UNDFLO,INEXO
0	1	0 . . . 0	0	2**	− 126	+ NORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RM,RN,RZ	UNDFLO,INEXO
0	0	0 . . . 0	1	2**	− 126	+ DENORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	RP	UNDFLO,INEXO
0	0	0 . . . 0	1	2**	− 126	+ DENORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RM,RN,RZ	UNDFLO,INEXO
0	0	0 . . . 0	0		0	+ ZERO	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	all	
1	0	0 . . . 0	1	2**	− 126	− DENORM.MIN	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	RM	UNDFLO,INEXO
1	0	0 . . . 0	1	2**	− 126	− DENORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	− 126	− NORM.MIN	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	RM	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	− 126	− NORM.MIN	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	− 1	− 1/2	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	RM	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	− 1	− 1/2	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	1	2**	− 1	− 1/2 − 1LSB	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	RM,RN	UNDFLO,INEXO
1	1	0 . . . 0	1	2**	− 1	− 1/2 − 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RP,RZ	UNDFLO,INEXO
1	1	1 . . . 1	1	2**	− 1	− one + 1LSB	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	RM,RN	UNDFLO,INEXO
1	1	1 . . . 1	1	2**	− 1	− one + 1LSB	0	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	RP,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	0	− one	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	
1	1	1 . . . 1	1	2**	22		1	1	1	1	1	1	1	1	1	0 . . .	0	0	0	0	0	0	0	0	RM,RN	INEXO
1	1	1 . . . 1	1	2**	22		1	1	1	1	1	1	1	1	1	0 . . .	0	0	0	0	0	0	0	1	RP,RZ	INEXO
1	1	0 . . . 0	0	2**	23		1	1	1	1	1	1	1	1	1	0 . . .	0	0	0	0	0	0	0	0	all	
1	1	1 . . . 1	1	2**	23		1	1	1	1	1	1	1	1	0	0 . . .	0	0	0	0	0	0	0	1	all	
1	1	1 . . . 1	1	2**	30		1	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	all	
1	1	0 . . . 0	0	2**	31		1	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	all	
1	1	0 . . . 0	1	2**	31		U	U	U	U	U	U	U	U	U	U . . .	U	U	U	U	U	U	U	U	all	
1	1	0 . . . 0	0	2**	128	− INF	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP
1	1	X . . . X	X	2**	128	− NAN	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP

*"U" denotes an undefined result.

Table XIII. Conversion of 32-Bit Single-Precision Floating- Point to 32-Bit Twos-Complement Integer

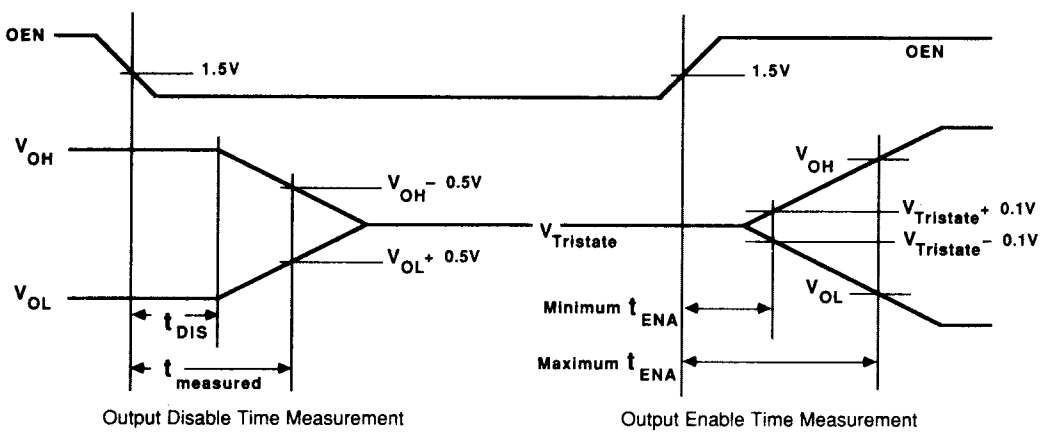


Figure T1. ADSP-3201/3202 Three-State Disable and Enable Timing

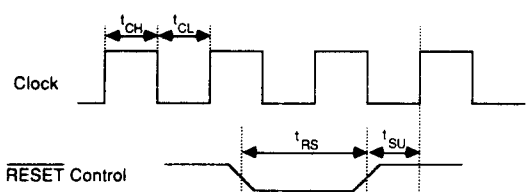


Figure T2. ADSP-3201/3202 Reset Timing

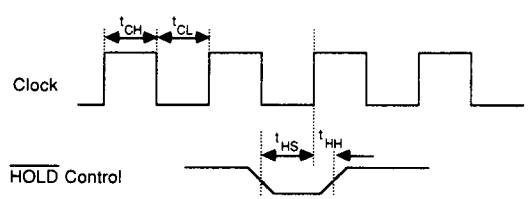
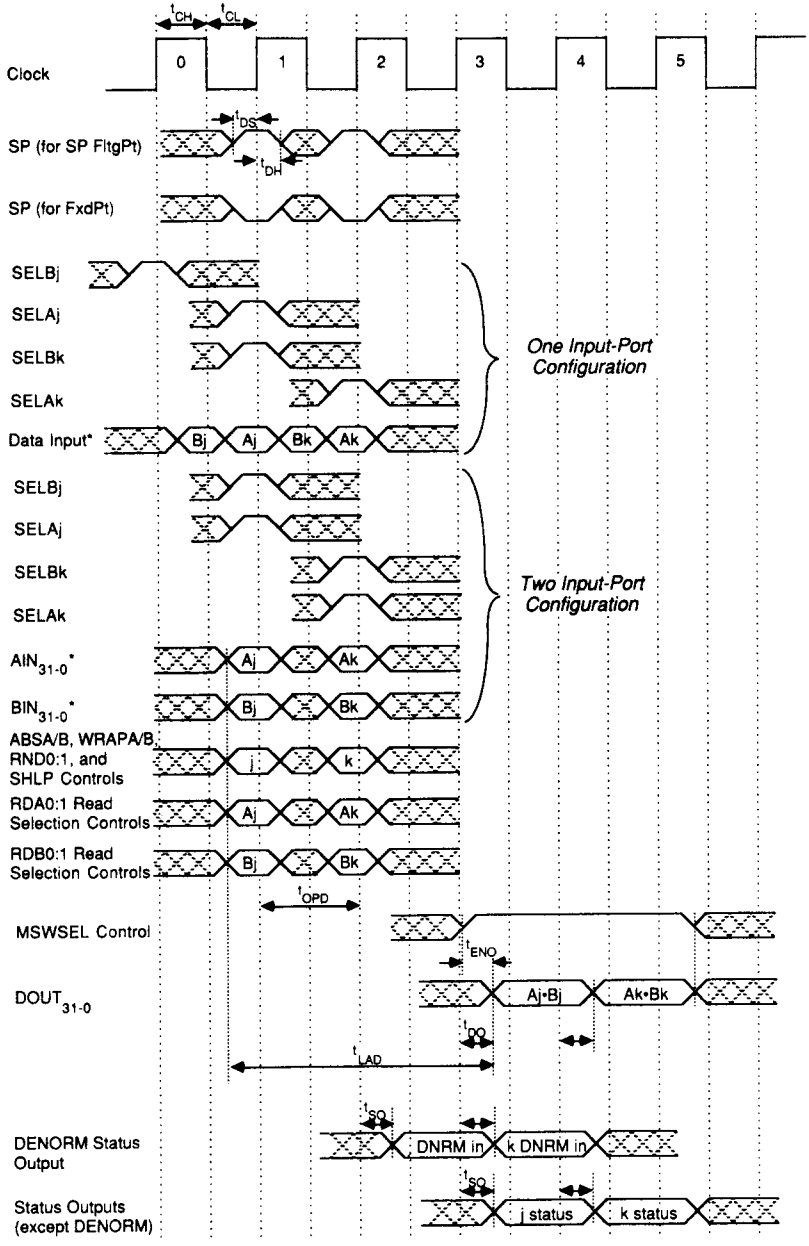
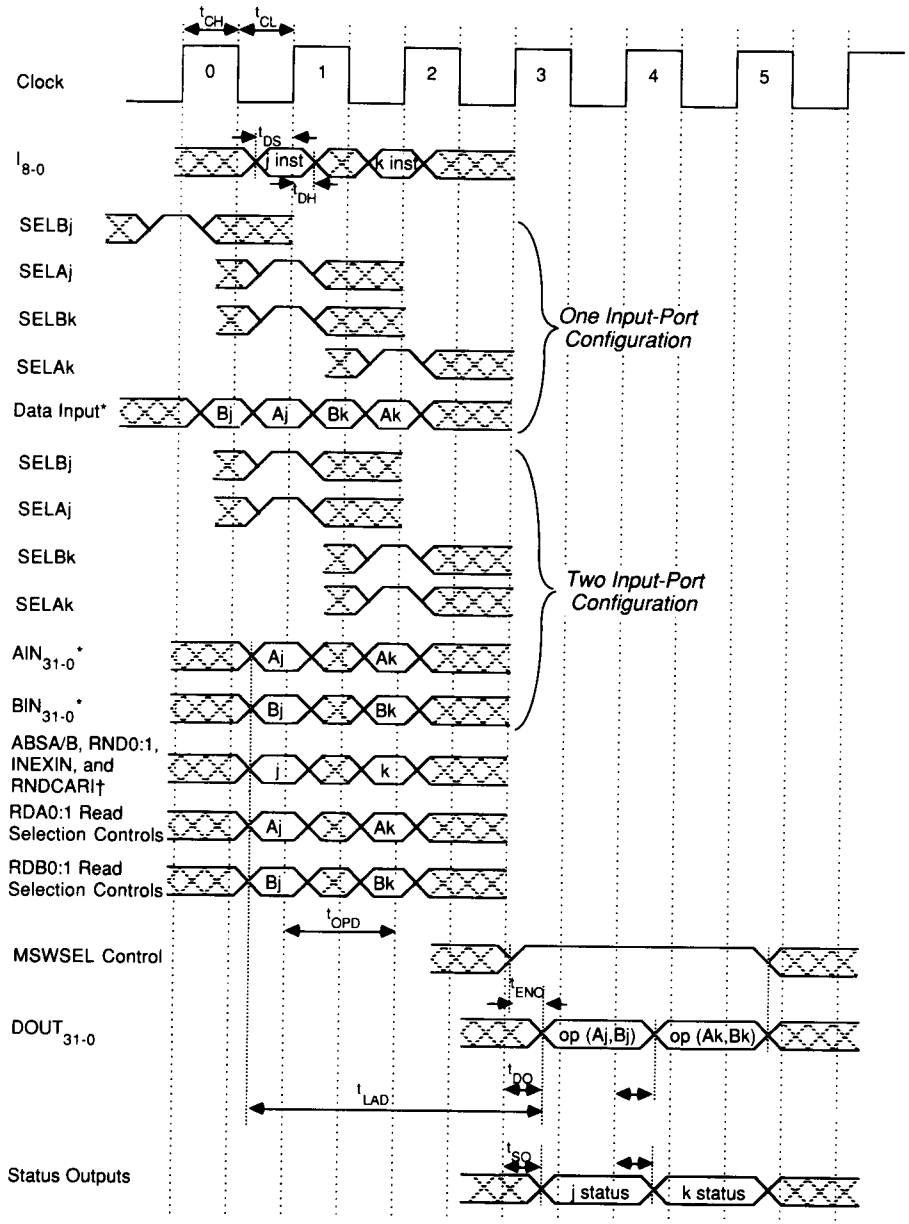


Figure T3. ADSP-3201 Multiplier Output Register Hold Timing



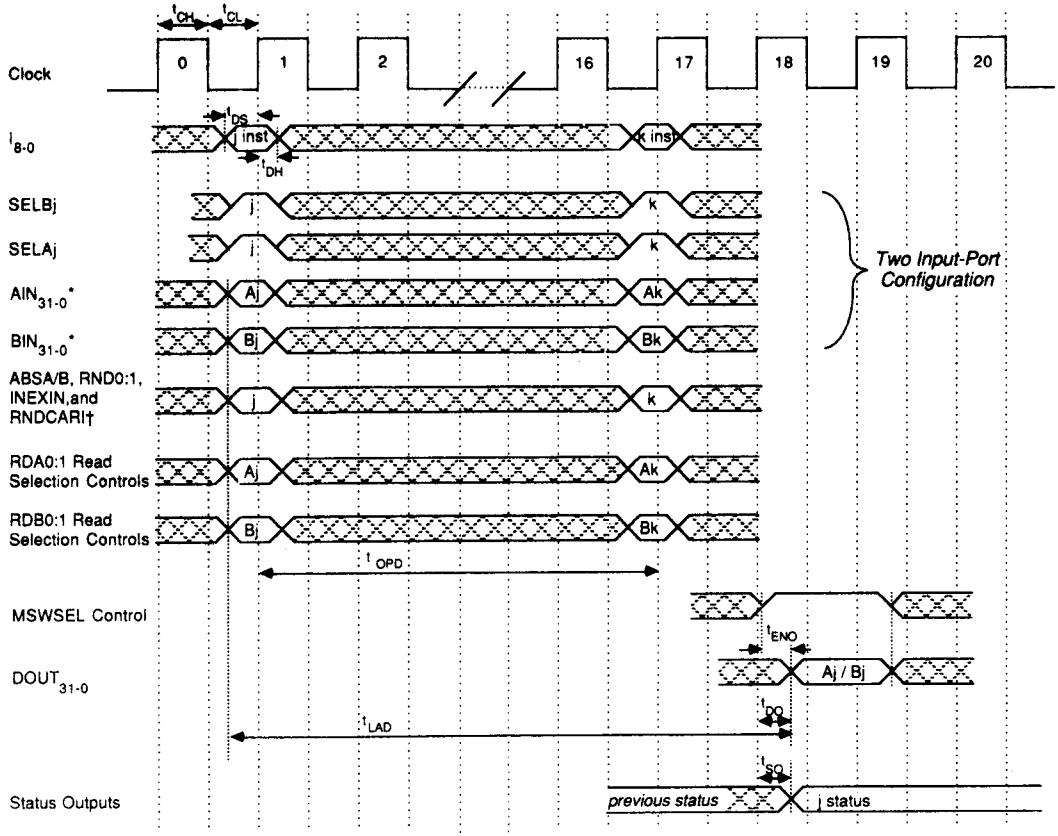
* See "Timing" section for additional sequencing options.

Figure T4. ADSP-3201 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplications



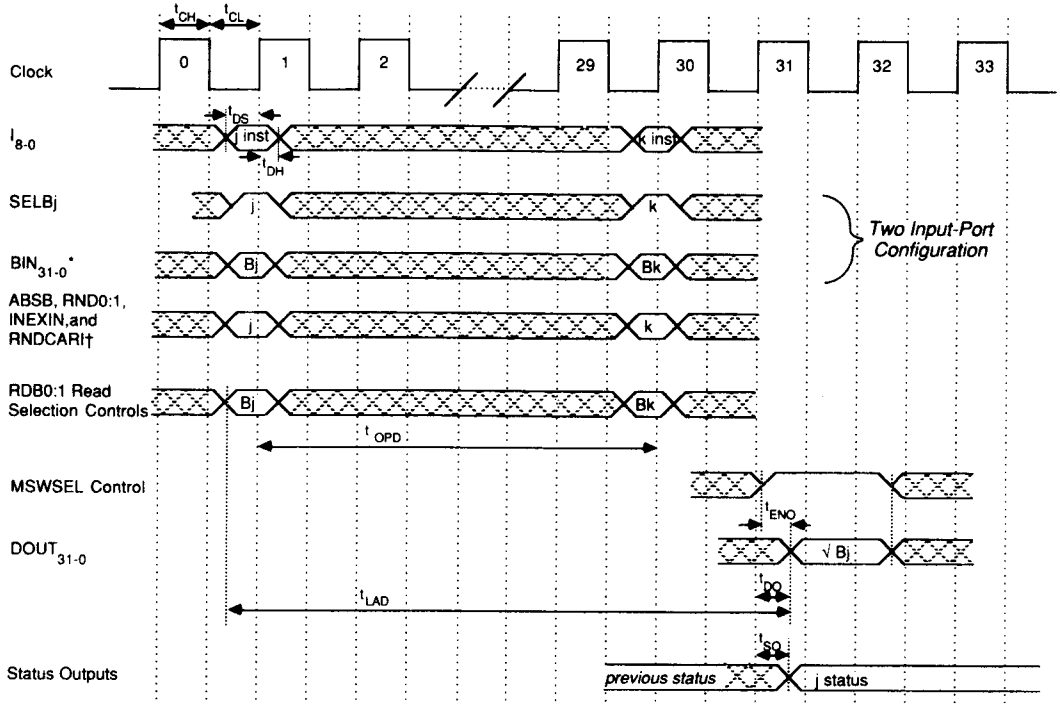
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T5. ADSP-3202 32-Bit Single-Precision Floating-Point Logical, and Fixed-Point ALU Operations



* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T6. ADSP-3202 32-Bit Single-Precision Floating-Point Division – Two Input-Port Configuration



* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T7. ADSP-3202 32-Bit Single-Precision Floating-Point Square Root – Two Input-Port Configuration

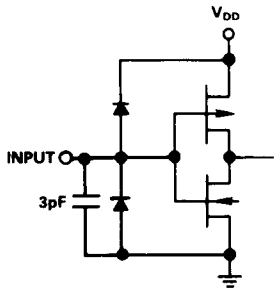


Figure 26. Equivalent Input Circuits

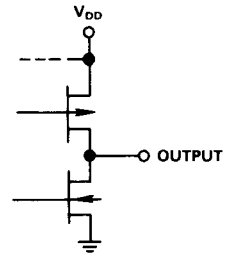


Figure 27. Equivalent Output Circuits

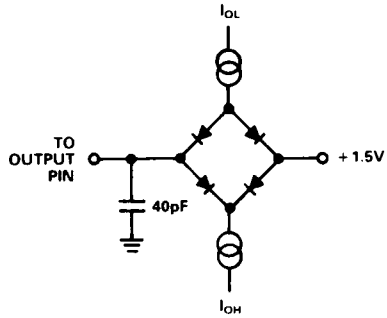


Figure 28. Normal Load for ac Measurements

4

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3201/3202 J and K Grades Min	J and K Grades Max	S and T Grades[2] Min	S and T Grades[2] Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_{AMB}	Operating Temperature (Ambient)	0	+70	−55	+125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	ADSP-3201/3202 J and K Grades Min	J and K Grades Max	S and T Grades[2] Min	S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		V
V_{IHA}	High-Level Input Voltage, CLK and Asynchronous Controls	@ V_{DD} = max	2.6		3.0		V
V_{IL}	Low-Level Input Voltage	@ V_{DD} = min		0.8		0.8	V
V_{OH}	High-Level Output Voltage	@ V_{DD} = min & I_{OH} = −1.0mA	2.4		2.4		V
V_{OL}	Low-Level Output Voltage	@ V_{DD} = min & I_{OL} = 4.0mA		0.5		0.6	V
I_{IH}	High-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 5.0V		10		10	μA
I_{IL}	Low-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 0V		10		10	μA
I_{OZ}	Three-State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = 0V or max		50		50	μA
I_{DD}	Supply Current	@ max clock rate; TTL inputs		150		200	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		50		60	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-3201/3202 J Grade 0 to +70°C Min	J Grade 0 to +70°C Max	K Grade 0 to +70°C Min	K Grade 0 to +70°C Max	S Grade[2] −55°C to +125°C Min	S Grade[2] −55°C to +125°C Max	T Grade[2] −55°C to +125°C Min	T Grade[2] −55°C to +125°C Max	Unit
t_{CY}	Clock Cycle		125		100		150		125	ns
t_{CL}	Clock LO	20		20		30		30		ns
t_{CH}	Clock HI	20		20		30		30		ns
t_{DS}	Data & Control Setup	20		15		25		20		ns
t_{DH}	Data & Control Hold	3		3		3		3		ns
t_{DO}	Data Output Delay		30		25		35		30	ns
t_{SO}	Status Output Delay		30		25		35		30	ns
t_{ENO}	MSWSEL-to-Data Delay		25		20		30		25	ns
t_{DIS}	Three-State Disable Delay		18		15		25		20	ns
t_{ENA}	Three-State Enable Delay	3	25	3	20	2	30	2	25	ns
t_{SU}	RESET Setup	20		15		25		20		ns
t_{RS}	RESET Pulse Duration	50		50		50		50		ns
t_{HS}	HOLD Setup	20		15		22		18		ns
t_{HH}	HOLD Hold	3		3		3		3		ns
t_{OPD}	Operation Time									
	32-Bit Multiplication		125		100		150		125	ns
	32-Bit ALU Operations		125		100		150		125	ns
	32-Bit Division (3202)		2.0		1.6		2.4		2.0	μs
	32-Bit Square Root (3202)		3.625		2.9		4.35		3.625	μs

Parameter		ADSP-3201/3202 J Grade 0 to +70°C Min	J Grade 0 to +70°C Max	K Grade 0 to +70°C Min	K Grade 0 to +70°C Max	S Grade[2] −55°C to +125°C Min	S Grade[2] −55°C to +125°C Max	T Grade[2] −55°C to +125°C Min	T Grade[2] −55°C to +125°C Max	Unit
t_{LAD}	Total Latency									
	32-Bit Multiplication		300		240		360		300	ns
	32-Bit ALU Operation		300		240		360		300	ns
	32-Bit Division		2.175		1.74		2.61		2.175	µs
	32-Bit Square Root (3202)		3.8		3.04		4.56		3.8	µs

NOTES
[1]All min and max specifications are over power-supply and temperature range indicated.
[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883B. The processing and test methods used for S/883B and T/883B versions of the ADSP-3201/3202 can be found in Analog Devices' Military Databook.
[3]Input levels are GND and +3.0V. Rise times are 5ns. Input timing reference levels and output reference levels are 1.5V, except for 1) t_{ENA} and t_{DIS} which are as indicated in Figure T1 and 2) t_{DS} and t_{DH} which are measured from clock V_{THA} to data input V_{TH} or V_{TL} crossing points.

Specifications subject to change without notice.

4

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	−0.3V to +7V	Operating Temperature Range (Ambient)	−55°C to +125°C
Input Voltage	−0.3V to V_{DD}	Storage Temperature Range	−65°C to +150°C
Output Voltage Swing	−0.3V to V_{DD}	Lead Temperature (10 Sec)	+300°C

ESD SENSITIVITY

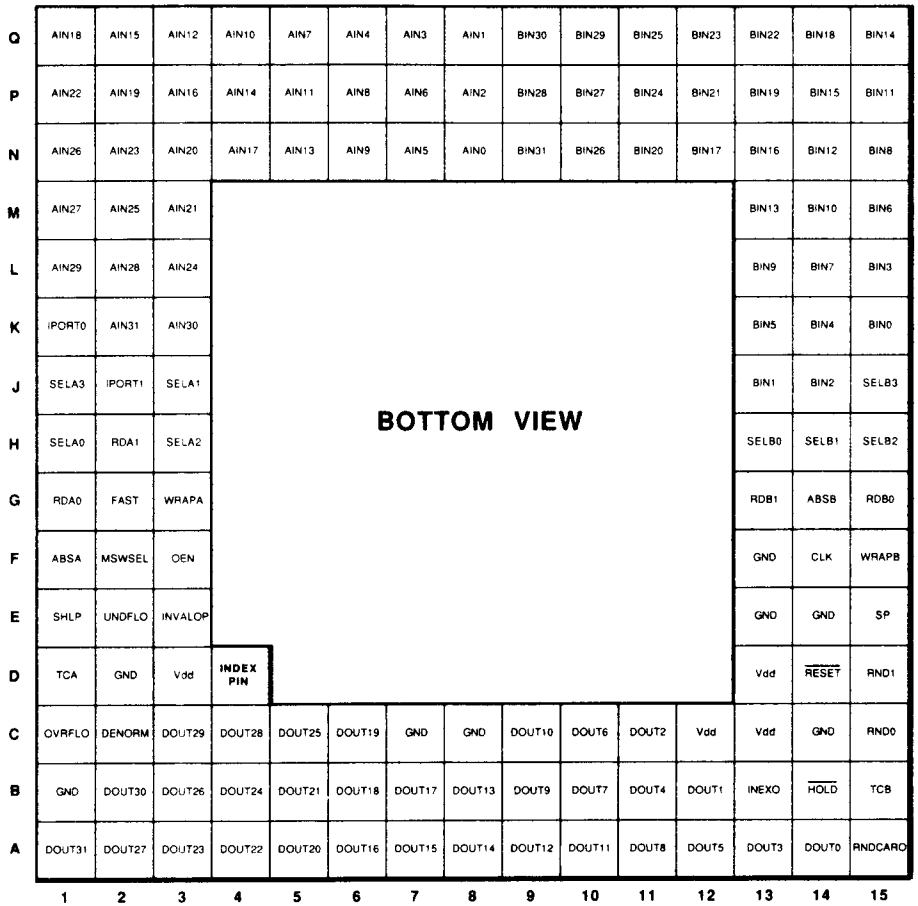
The ADSP-3201/3202 feature proprietary input protection to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-3201/3202 have been classified as Class 1 devices.

Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should be discharged to the destination socket before devices are removed. For further information on ESD precautions, refer to Analog Devices' *ESD Prevention Manual*.

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-3201JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3201KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3201SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3201TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3201SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3201TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3202JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3202KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3202SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3202TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3202SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3202TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A

Contact DSP Marketing in Norwood concerning the availability of other package types.



ADSP-3201 Multiplier Pinouts

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q	AIN18	AIN15	AIN12	AIN10	AIN7	AIN4	AIN3	AIN1	BIN30	BIN29	BIN25	BIN23	BIN22	BIN18	BIN14
P	AIN22	AIN19	AIN16	AIN14	AIN11	AIN8	AIN6	AIN2	BIN28	BIN27	BIN24	BIN21	BIN19	BIN15	BIN11
N	AIN26	AIN23	AIN20	AIN17	AIN13	AIN9	AIN5	AIN0	BIN31	BIN26	BIN20	BIN17	BIN16	BIN12	BIN8
M	AIN27	AIN25	AIN21										BIN13	BIN10	BIN6
L	AIN29	AIN28	AIN24										BIN9	BIN7	BIN3
K	RND1	AIN31	AIN30										BIN5	BIN4	BIN0
J	RNDCARI	RND0	CLK										BIN1	BIN2	IPORT1
H	ABSB	ABSA	$\overline{\text{RESET}}$				**BOTTOM VIEW**						RDA0	IPORT0	RDA1
G	I0	I3	I2										SELA0	SELA3	SELA1
F	I1	I5	I6										RDB0	RDB1	SELA2
E	I4	I8	FAST										N.C	SELB1	SELB0
D	I7	GND	Vdd	INDEX PIN									Vdd	N.C	SELB2
C	INEXIN	OVRFLO	INEXO	DOUT31	DOUT28	DOUT22	GND	GND	DOUT13	DOUT9	DOUT5	Vdd	Vdd	MSWSEL	SELB3
B	GND	UNDFLO	DOUT29	DOUT27	DOUT24	DOUT21	DOUT20	DOUT16	DOUT12	DOUT10	DOUT7	DOUT4	DOUT2	DOUT0	OEN
A	INVALOP	DOUT30	DOUT26	DOUT25	DOUT23	DOUT19	DOUT18	DOUT17	DOUT15	DOUT14	DOUT11	DOUT8	DOUT6	DOUT3	DOUT1

ADSP-3202 ALU Pinouts