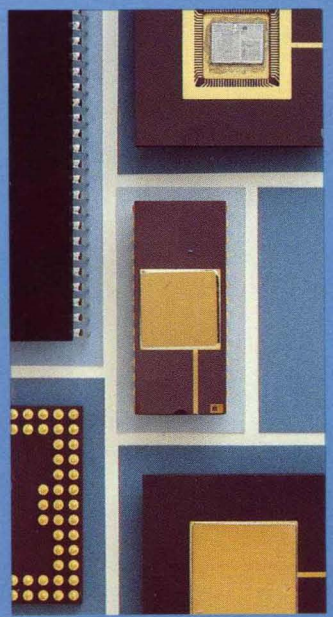
ANALOG DEVICES

1989



DSP PRODUCTS DATABOOK

DSP PROCESSORS
MICROCODED SUPPORT COMPONENTS
FLOATING-POINT COMPONENTS
FIXED-POINT COMPONENTS

DSP PRODUCTS DATABOOK 1989
ANALOG DEVICES

How to Find Product Data in This Databook

THIS VOLUME

Contains Data Sheets, Selection Guides, Application Notes, and a wealth of background information on components for number crunching and digital signal processing (DSP).

It is one member of a three-volume, 2,000-page set of Databooks describing and specifying Linear, Conversion, and DSP products from Analog Devices, Inc., in IC, hybrid, and assembled form for measurement, control, and real-world signal processing.

IF YOU KNOW THE MODEL NUMBER

Turn to the product index on inside back cover at the back of the book and look up the model number. You will find the Section-Page location of data sheets bound into this volume.

If you're looking for a form-and-function-compatible version of a product originally brought to market by some other manufacturer (second source), add our "ADSP" prefix and look it up in the index.

IF YOU DON'T KNOW THE MODEL NUMBER

Find your function in the list on the opposite page or in the Table of Contents on pages 1–3 and 1–4. Turn directly to the appropriate Section. You will find a functional Selection Guide at the beginning of the Section. The Selection Guides will help you find the products that are the closest to satisfying your need. Use them to compare all products in the category by salient criteria.

IF YOU CAN'T FIND IT HERE . . . ASK!

If it's not a DSP product, it's probably in one of the two sister volumes, the *Linear Products Databook* or the *Data Conversion Products Databook*. If you don't already own these volumes, you can have them FREE by getting in touch with Analog Devices or the nearest sales office, or phoning (617)-329-4700, Extension 3392.

See Worldwide Service Directory on pages 8–6 and 8–7 at the back of this volume for our sales-office phone numbers.

Contents of Other Databooks

DATA CONVERSION PRODUCTS DATABOOK

D/A Converters
A/D Converters
V/F & F/V Converters
Synchro & Resolver Converters
Sample/Track-Hold Amplifiers
CMOS Switches & Multiplexers
Voltage References
Data Acquisition Subsystems
Application Specific ICs
Power Supplies
Component Test Systems

LINEAR PRODUCTS DATABOOK

Operational Amplifiers
Comparators
Instrumentation Amplifiers
Isolation Amplifiers
Analog Multipliers/Dividers
Log/Antilog Amplifiers
RMS-to-DC Converters
Special Function Components
Temperature Transducers
Signal Conditioning Components & Subsystems
Digital Panel Instruments
Application Specific ICs
Power Supplies
Component Test Systems

1989 DSP PRODUCTS DATABOOK

ANALOG DEVICES

DSP PRODUCTS DATABOOK
April 1989

Products in this book may be covered by one or more of the following patents. Additional patents are pending.

U.S.:
RE29,619, RE29,992, RE30,586, RE31,850, DES. 233,909, 3,007,114, 3,278,736, 3,355,670, 3,441,913, 3,467,908, 3,500,218, 3,530,390, 3,533,002, 3,685,045, 3,729,660, 3,793,563, 3,803,590, 3,842,412, 3,868,583, 3,890,611, 3,906,486, 3,909,908, 3,932,863, 3,940,760, 3,942,173, 3,946,324, 3,950,603, 3,961,326, 3,978,473, 3,979,688, 4,016,559, 4,020,486, 4,029,974, 4,034,366, 4,054,829, 4,092,698, 4,123,698, 4,136,349, 4,141,004, 4,213,806, 4,250,445, 4,268,759, 4,270,118, 4,286,225, 4,309,693, 4,313,083, 4,323,795, 4,338,591, 4,349,811, 4,363,024, 4,374,314, 4,383,222, 4,395,647, 4,399,345, 4,400,689, 4,400,690, 4,427,973, 4,439,724, 4,460,891, 4,475,103, 4,475,169, 4,476,538, 4,481,708, 4,484,149, 4,485,372, 4,491,825, 4,511,413, 4,521,764, 4,543,560, 4,543,561, 4,547,766, 4,547,961, 4,556,870, 4,558,242, 4,562,400, 4,565,000, 4,586,019, 4,586,155, 4,590,456, 4,596,976, 4,601,760, 4,604,532, 4,608,541, 4,622,512, 4,626,769, 4,639,683, 4,644,253, 4,646,056, 4,646,238, 4,678,936, 4,684,922, 4,685,200, 4,694,276, 4,697,151, 4,703,283, 4,707,682, 4,709,167, 4,717,883, 4,722,910, 4,742,331, 4,751,455, 4,752,900 4,761,636, 4,769,564, 4,771,011, 4,774,685, 4,791,551

France:
111.833, 70.10561, 75.27557, 76 08238, 77 20799, 78 10462, 79 24041, 80 00960, 80 11312, 81 02661, 81 14845, 82 09758, 83 03140

Japan:
1,092,928, 1,242,936, 1,242,965, 1,306,235, 1,337,318, 1,401,661, 1,412,991

West Germany:
2,014 034, 25 40 451.7, 26 11 858.1

U.K.:
1,310,591, 1,310,592, 1,537,542, 1,590,136, 1,590,137, 1,599,538, 2,008,876, 2,032,659, 2,040,087, 2,050,740, 2,054,992, 2,075,295, 2,081,040, 2,100,081, 2,103,884, 2,104,288, 2,107,951, 2,115,932, 2,118,386, 2,119,139, 2,119,547, 2,126,445, 2,126,814, 2,135,545, 2,137,787

Canada:
984,015, 1,006,236, 1,025,558, 1,035,464, 1,054,248, 1,141,034, 1,141,820, 1,142,445, 1,143,306, 1,150,414, 1,153,607, 1,157,571, 1,159,956, 1,177,127, 1,177,966, 1,184,662, 1,184,663, 1,191,715, 1,192,310, 1,192,311, 1,192,312, 1,203,628, 1,205,920, 1,212,730, 1,214,282, 1,219,679, 1,219,966, 1,223,086

Sweden:
7603320-8

General Information
Contents

Introduction

DSP AT ANALOG DEVICES

Analog Devices is the industry's leading supplier of high performance signal processing integrated circuits. As the leader, Analog Devices was quick to recognize the important opportunities made possible by the growth of digital signal processing. In 1983, the DSP Division introduced the industry's first CMOS fixed-point multipliers and multiplier-accumulators. These matched the speed of bipolar alternatives while cutting power requirements by a factor of twenty. This breakthrough shifted the focus of the industry from bipolar to CMOS for high speed VLSI circuits.

PRODUCT GROWTH & INNOVATION

From a base in industry-standard components, we brought out a complete line of building block VLSI processors for high end DSP and numeric processing systems. These include several 64-bit IEEE floating-point chipsets and a single-precision (32-bit) version of one of the same chipsets. Other products in the family are an address generator, two program sequencers and a register file.

In 1986 we introduced the first full off-chip Harvard architecture DSP microprocessor, the ADSP-2100, complemented by a superior set of interactive development tools.

TECHNOLOGY GROWTH

From our original CMOS wafer fabrication in 5 micron geometries we moved, in 1985, to 1.5 micron double-layer metal CMOS. We are currently in production with both the 1.5 micron and our newer 1.0 micron CMOS processes. Our 12.5MHz ADSP-2100A is the 1.0 micron version of our original 1.5 micron ADSP-2100, for example. Analog Devices continues to develop advanced processes such as specialized bipolar and gallium arsenide both internally and through our strategic investments.

Our manufacturing facilities include factories in Wilmington, Massachusetts and assembly in the Philippines. Our digital VLSI test capability is located in Norwood, Massachusetts, the Division's headquarters.

APPLICATIONS & SUPPORT GROWTH

Analog Devices supports its products with a technically strong direct sales force and readily available applications assistance. Our Applications Engineering staff in Norwood, Massachusetts; Santa Ana, California; Tokyo, Japan; Newbury, UK and other locations worldwide understands the specialized requirements of designing and supporting DSP systems. Our quarterly DSP applications newsletter, *DSPatch*, brings you up-to-date applications information and is available free by request.

DSP PRODUCTS DATABOOK

This book provides complete technical data on DSP products from Analog Devices. Included are:

- Comprehensive Data Sheets on some 20 significant product families
- Selection Guides for rapid product finding
- DSP Application Notes
- List of available Technical Publications on real-world analog and digital signal processing
- Worldwide Service Directory
- Index.

Besides this Databook, the present series includes a Linear Products Databook and a Data Conversion Products Databook; like this book, the latest versions of both are available free upon request.

TECHNICAL SUPPORT

Our extensive technical literature discusses the technology and applications of products for precision measurement and control. Besides tutorial material and comprehensive data sheets, including a large amount in our Databooks, we offer Application Notes, Application Guides, Technical Handbooks (at reasonable prices), and several serial publications; for example, *Analog Productlog* provides brief information on new products being introduced, and *Analog Dialogue*, our technical magazine, provides in-depth discussions of new developments in analog and digital circuit technology as applied to data acquisition, signal processing, control, and test. We maintain a mailing list of engineers, scientists, and technicians with a serious interest in our products. In addition to Databook catalogs, we also publish several short-form catalogs on specific product families. You will find typical publications described on pages 8-2 and 8-3 at the back of the book.

SALES OFFICES

Backing up our design and manufacturing capabilities and our extensive array of publications is a network of sales offices and representatives throughout the United States and most of the world. They are staffed by experienced sales and applications engineers, and many of them maintain a local stock of Analog Devices products. Our Worldwide Service Directory, as of the publication date, appears on pages 8-6 and 8-7 at the back of the book.

RELIABILITY

The manufacture of reliable products is a key objective at Analog Devices. We maintain facilities that have been qualified under such standards as MIL-M-38510 for ICs in the U.S. and Ireland and MIL-STD-1772 for hybrids. A growing number of our products have qualified for JAN part numbers; others are in the process. Most of our ICs are available in versions that comply with MIL-STD-883C Class B.

We publish a *Military Products Databook* for designers who specify ICs and hybrids for military contracts (the 1987 issue contains data on nearly 150 available product families). A newsletter, *Analog Briefings*, provides current information about the status of reliability at ADI.

PRICES

Accurate, up-to-date prices are an important consideration in making a choice among the many available product families. Since prices are subject to change, current prices lists and/or quotations are available upon request from our sales offices.

Table of Contents

DSP Processors
Contents

2

Introduction

The ADSP-2100 family of digital signal processors provides a core architecture optimized for digital signal processing and other high speed numeric processing applications. The family consists of the ADSP-2100 and ADSP-2100A microprocessors and the ADSP-2101 and ADSP-2102 microcomputers. All devices share the core set of features:

1. Easy-to-Attain High Performance
   The ADSP-2100 core integrates an arithmetic/logic unit (ALU), multiplier-accumulator (MAC), barrel shifter, data address generators and a program sequencer in a single device. It incorporates modified Harvard architecture (that is, data can also be stored in program memory) for efficient access to program and data memories. The result combines the functions and performance of a bit-slice or building block system with the ease-of-design and development of a general-purpose microprocessor.
2. Easy-to-Understand Instruction Set
   The ADSP-2100 family instruction set uses an algebraic syntax, similar to high level languages, making it easier to write and understand source code. This results in easier and faster code development and maintenance.
3. Easy-to-Use Development Tools & Support
   The complete set of development tools available for the family (including a C Compiler, Simulator and In-circuit Emulator each described later in this section) minimizes both design time and effort. Your application is up and running faster with this powerful development. In addition, our Applications Engineering Group supports only DSP with application notes, applications handbooks, a customer newsletter, a bulletin board service and excellent telephone support.
4. Easy-to-Design System Interface
   The advanced design of the ADSP-2100 family allows simple interconnections of memories and I/O devices, minimizes the external logic required to handle interrupts and supports straightforward host interface and multiprocessing designs.

ADSP-2100/ADSP-2100A Microprocessor

The microprocessor members of the family include the ADSP-2100 and the 1.0 micron ADSP-2100A which are pin and code-compatible. In addition to the core, these devices offer the following:

- Modified off-chip Harvard architecture. The processor can access up to 16K words of 16-bit data memory and up to 32K 24-bit words of program memory containing both instructions and data.
- A 16-instruction on-chip cache memory with allows the processor to fetch two operands in parallel when executing out of the cache. Since the instruction set supports a high degree of parallelism, the loops of many algorithms can be efficiently coded in 16 instructions or less.

A sample set of benchmarks for the 12.5MHz ADSP-2100A is shown in the table below.

ADSP-2101/ADSP-2102 Microcomputer

The microcomputer members of the family include the RAM-based ADSP-2101 and the mask programmable ROM-based ADSP-2102. Both are upwardly code compatible with the ADSP-2100 and ADSP-2100A. In addition to the core, these devices offer the following:

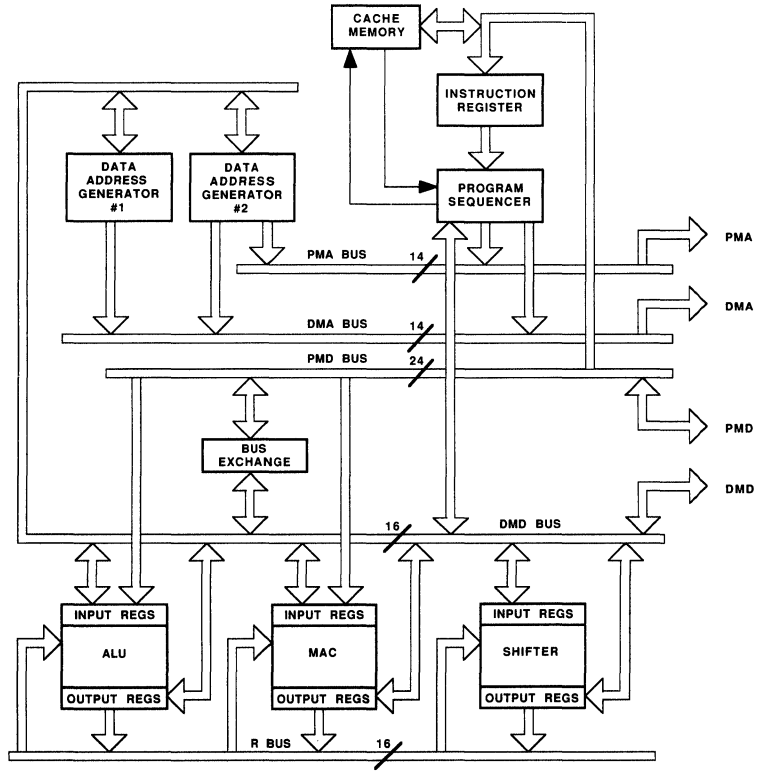
- Modified on-board Harvard architecture. The processor has 2K words of (24-bit) program memory RAM and 1K of 16-bit data memory RAM on-chip. Off-chip memories share one address and one data bus which can be used to fetch instructions, data and to boot the processor from external memory.
- A 16-bit programmable timer with an 8-bit prescaling factor that generates its own interrupt.
- Two serial ports offering a wide set of possible framing and timing options for interfacing easily to any serial device. Companding is supported in hardware.

The diagram on the following page graphically shows the microprocessor and microcomputer devices.

ADSP-2100 Family Benchmarks

Algorithm	Performance @ 12.5MHz
FIR Filter	80ns per Tap (1 cycle per Tap)
Complex FIR Filter	320ns per Tap (4 cycles per Tap)
Biquad Filter Section	560ns per Section (7 cycles per Section)
Lattice Filter Section	400ns per Section (5 cycles per Section)
1024-point Complex FFT (Radix-2)	2.9ms
4096-point Complex FFT (Radix-2)	19.8ms

ADSP-2100/ADSP-2100A



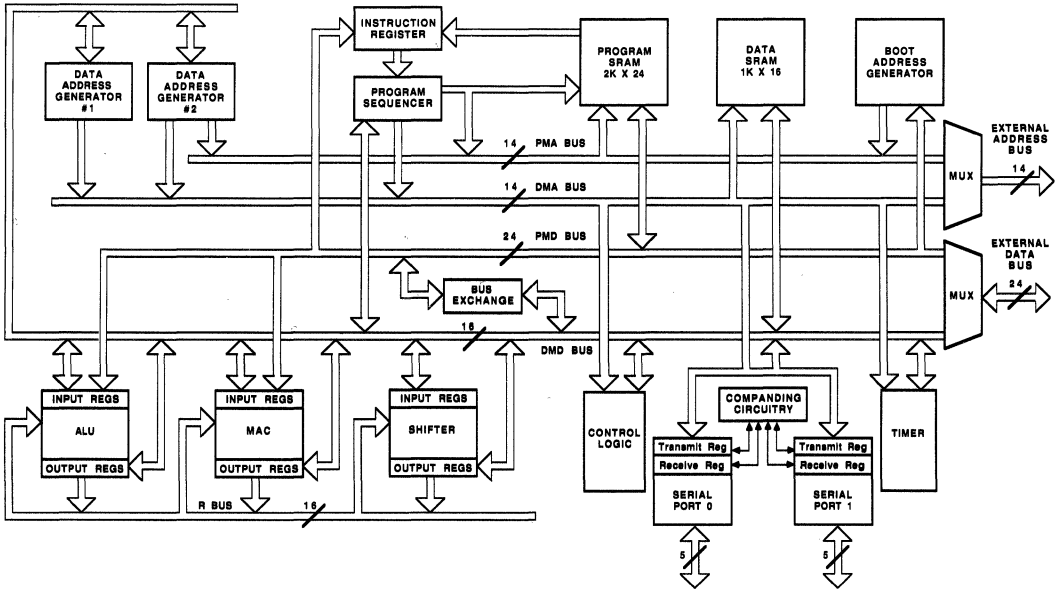
Core

- ALU, MAC and Barrel Shifter
- Two Data Address Generators, one with bit-reversing capability
- Separate Program Memory and Data Memory address and data buses
- Powerful Sequencer for Zero Overhead looping and single-cycle branches
- Bus Grant and Bus Request Signals for host interfacing
- Highly Readable Source Code for ease of development and maintenance

ADSP-2100/ADSP-2100A Specific Features

- All Program and Data Memory buses extended off-chip
- Single-cycle access to external memory
- Up to 16K of 16-bit word data memory
- Up to 32K of 24-bit word program memory (may also hold data)
- Data Memory Acknowledge Signal (DMACK) for interfacing to slow, memory-mapped peripherals
- On-chip instruction cache for three bus performance
- Four interrupt request lines
- 100-pin PGA and 100-lead PQFP packages

ADSP-2101/ADSP-2102



Core

- ALU, MAC and Barrel Shifter
- Two Data Address Generators, one with bit-reversing capability
- Separate Program Memory and Data Memory address and data buses
- Powerful Sequencer for Zero Overhead looping and single-cycle branches
- Bus Grant and Bus Request Signals for host interfacing
- Highly Readable Source Code for ease of development and maintenance

ADSP-2101/ADSP-2102 Specific Features

- 2K of 24-bit on-chip program memory RAM
- 1K of 16-bit on-chip data memory RAM
- Up to 16K of 16-bit word data memory using external memory
- Up to 16K of 24-bit word program memory using external memory
- Up to three memory accesses (one may be off-chip) in a single cycle
- Timer interrupt with programmable period and prescaler
- Two complete serial ports with companding in hardware
- 68-pin PGA and 68-lead PLCC packages

ADDS-21XX

FEATURES

Release 1.5 Supports the ADSP-2100 and ADSP-2100A DSP Microprocessors

C COMPILER

Programming in C Eases Development of Applications Software
Supports In-Line Assembly Code
Provides FRACT Data Type (1.15 Format) for DSP Algorithms
Complete Calling Interface to Assembly Language Routines
Produces ROMable Code
Floating Point Emulation Support
Conforms to ANSI Draft Standard (X3J11)

SYSTEM BUILDER

Architecture Description File Specifies Target Hardware

ASSEMBLER

Supports High Level Constructs
Supports Flexible Macro Processing
Encourages Modular Code Development
Provides a Full Range of Diagnostics

LINKER

Library Support
Maps Assembler Output to Target Hardware

PROM SPLITTER

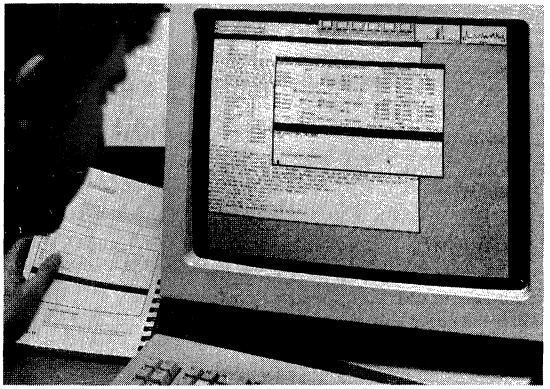
Formats ROM Memory Image for Uploading to PROM Programmers

SIMULATOR

Interactive User-Friendly Interface
Full Symbolic Disassembly
Simulates Hardware Configuration
Simulates Port I/O Handling
Flags Illegal Operations

GENERAL DESCRIPTION

The ADSP-2100 Cross-Software Development tools allow the programmer to develop applications software for implementation on ADSP-2100 and ADSP-2100A DSP microprocessors. The software tools include the C compiler, System Builder, Assembler, Linker, PROM Splitter and Simulator.



C COMPILER

The C Compiler supports the development of application programs in the C programming language. Consisting of a Preprocessor and Compiler which conform to the ANSI draft standard (X3J11), the C Compiler produces ADSP-2100 assembly language source code. Applications written in C are then compiled, assembled and linked to produce code that can be simulated using the Simulator or executed on the Emulator or Evaluation Board.

The Preprocessor supports the complete ANSI draft standard set of options, and reads directives such as *#include*. The *#pragma* directive supports in-line assembly code in a C program. This allows the user to execute efficient assembly language routines within the C environment.

From the code produced by the Preprocessor, the compiler creates a stack-oriented run-time environment using the Data Address Generators to implement the stack. The stack may be located in program or data memory RAM. It is used for parameter passing and local and temporary storage. Because the ADSP-2100 cannot write an immediate value to program memory, locating the stack in data memory is usually more efficient.

The example in Figure 1 illustrates how a simple function implemented in ADSP-2100 source code is interfaced to a C function call.

```
int i,j,k;
main()
{
  k = add(i,j);
}
add(x,y)
{
#pragma ADSP2100
{  Function add (x,y)                }
{    int x,y;                        }
{                                    }
{  Returns: z = x+y;                 }
  dm(i4,m7) = ay0;          { save registers }
  dm(i4,m7) = ar;
  i6 = 1;                   { get first parameter }
  modify(i6,m4);
  ax0 = dm(i6,m5);          { m5 = 1, i6 points to 2nd parameter }
  ay0 = dm(i6,m5);          { get second parameter }
  ar = ax0+ay0;             { perform addition }
  ax0 = ar;                 { return 16-bit values in ax0 }
  i6 = -1;                  { restore registers }
  modify(i6, m4);
  ay0 = dm(i6,m7);
  ar = dm(i6,m7);
#pragma ADSP2100
}
```

Figure 1. Assembly Language to C Language Interface

The stack is managed by a frame pointer and stack pointer. The following diagram illustrates the implementation of the stack during a call. The previous frame pointer and local variables are popped unto the stack.

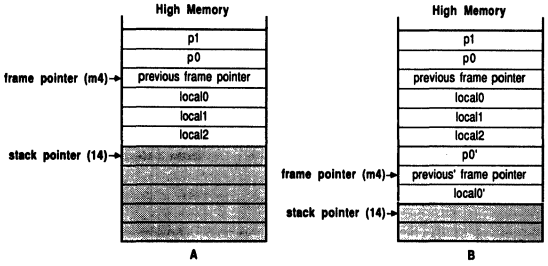


Figure 2. Stack Implementation in ADSP-2100 Memory Space

Though the ADSP-2100 is a 16-bit processor, the C Compiler supports certain 32-bit operations. The following arithmetic data types are supported directly:

int	16-bit twos-complement value
long int	32-bit twos-complement value
unsigned int	16-bit unsigned value
unsigned long int	32-bit unsigned values
fract	16-bit fractional value (1.15 format)
float	32-bit real.

Type *fract* is not a standard C data type but is an extension created to support the 1.15 data format used in digital signal processing applications. The compiler also supports all standard storage classes, types and modifiers.

Classes	*auto, extern, register, static, typedef*
Types	All including *void*
Modifiers	*const, volatile* plus *pm, dm, ram, rom*

Register values, though accepted by the compiler, are not implemented as actual processor registers. The modifiers *pm, dm, rom* and *ram* are extensions that are supported. In addition, the *fastswitch* statement, an extension to the language, has been added to support the DO UNTIL capability of the processors. It is syntactically identical to the standard *switch* statement but produces faster ADSP-2100 assembly code.

SYSTEM BUILDER

The System Builder translates a user-defined description of the target hardware system into a form which can be utilized by other Cross-Software Modules. The Cross-Software Modules require knowledge of the target hardware system for the Linker to place relocatable segments, the Simulator to simulate external memory configurations, and for the PROM Splitter to generate separate program and data files. The user specifies the target program memory, data memory and I/O port configurations by writing a System Specification Source File. The System Builder translates this into an Architecture Description File which is read by the other Cross-Software Modules. For example, the Linker resolves the references in the source code and the actual addresses by reading the Architecture File.

The Architecture File is comprised of the following directives that define the ADSP-2100 system:

.SYSTEM	first statement in .ACH file, specifies the name of the system
.ENDSYS	last statement in .ACH file, specifies the end of the file
.CONST	defines constants
.PORT	declares memory-mapped I/O ports
.SEG	specifies the type of memory in the system (program or data, RAM or ROM).

The following example of an architecture (.ACH) file shows the use of the directives:

```
.SYSTEM fir_system;                          {system name for fir_system}
.SEG/ROM/ABS = 0/PM/CODE program_mem[4096]   {declare code space}
.SEG/RAM/ABS = 4096/PM/DATA coeff_storage[15] {declare coeff table}
.SEG/RAM/ABS = 0/DM/DATA delay_line[15]      {declare data memory}
.PORT/ABS = 16382 ad_sample                  {declare i/o ports}
.ENDSYS                                      {indicates end of file}.
```

The .SYSTEM directive defines the name of the ADSP-2100 system. This name is used by the other software modules. The .SEG directive declares memory segments specifying the physical address, segment length, memory area (PM, DM), memory type (RAM, ROM) and memory attributes (CODE, DATA or both). In the above example, *program_mem* is a 4K-word buffer located in program memory ROM beginning at address 0 consisting of program code. The buffer, *coeff_storage*, is fifteen words of data located in program memory RAM beginning at address 4096. Finally, *delay_line* is a 15-word buffer located in data memory RAM starting at address 0. The .PORT directive declares memory-mapped I/O ports by specifying a name for the port and the absolute physical address. In the example, an analog-to-digital converter named *ad_sample* occupies location 16382 in data memory space.

ASSEMBLER

The Assembler translates source code modules into relocatable object code modules. The user creates an assembler source code module using the ADSP-2100 Assembly Language and defining variable data buffers and symbolic constants using the Assembler Directives. An assembly module becomes a unit of the complete system source code. Separately assembled object code modules are linked together to form the final running system using the Linker.

Assembler directives support a variety of data and program structures. Invocation switches modify the assembly process.

.MODULE	defines the beginning of an assembly module
.ENDMOD	the last statement in a source code file
.VAR	declare variables and data buffers, the /CIRC qualifier defines circular buffers
.CONST	declare constants
.PORT	declares a memory-mapped I/O port in data memory
.INIT	use to initialize declared variables and data buffers
.INCLUDE	use to read another source file
.MACRO	defines the beginning of a macro
.ENDMACRO	terminates a macro
.LOCAL	use only within a macro, directs the Assembler to create a unique label with local scope
.EXTERNAL	assigns external attribute to identifiers declared in other modules
.GLOBAL	assigns the global attribute to ports, variables and buffers
.ENTRY	assigns entry attribute to label names

Macros can be created using the .MACRO directive. For example, the macro shown below is a general purpose memory transfer routine which can transfer data buffers from one memory area (program or data memory) to the other.

```
{MACRO declaration}

.MACRO memory_transf (%0, %1, %2, %3, %4);   {pass five arguments}
.LOCAL transf;
        I4 = %0;                    {set I4 to source start address}
        I5 = %1;                    {set I5 to destination start address}
        M4 = 1;                     {set pointer update to single increment}
        CNTR = %2;                  {set length of buffer}
        DO transf UNTIL CE;         {transfer data}
        SI = %3(I4,M4);             {transfer from type %3 memory}
transf: %4(I5,M4) = SI;             {transfer from type %4 memory}
.ENDMACRO
```

To call the macro within an assembly language program, execute:

memory_transf (^coeff_table, ^buffer, buff_length, PM, DM);

LINKER

The Linker generates the Program Memory/Data Memory Image File, a complete executable program, by linking together object-code modules which were assembled separately. The hardware environment defined by the Architecture File is used by the Linker to place program and data in the defined memory area and location. This output file is used by the Simulator, PROM Splitter, Emulator, and Evaluation Board. Another Linker output, the Debug Symbol Table File, contains a list of all symbols encountered by the Linker and enables the Simulator to utilize user-defined source code level symbols in its interface with user.

To aid the user in interpreting the Linker result, a Map Listing file can be generated.

This file includes:

1. A cross-reference listing of all symbols encountered, arranged by module. Information on each symbol such at memory type, absolute address, length and symbol type is given.
2. A map of the memory sections and the attributes of each section.
3. A map of the allocated segments in program memory, listed sequentially from low order address to high order address.
4. A map of the allocated segments in data memory, listed sequentially from low order address to high order address.
5. Linker error messages.
6. A list of libraries searched and used.

PROM SPLITTER

The PROM Splitter extracts the address information and the contents of the ROM portion of the PM/DM Image File and formats the extracted images for uploading to PROM burners. Commercially available PROM burners expect input data to be eight bits wide. The PROM Splitter separates the memory image into a byte-wide format. It creates three one-byte wide PROM image files for the 24-bit program memory, and two one-byte wide PROM image files for the 16-bit data memory. Both program and data memory can be optionally output as a single stream of one-byte wide file. The PROM image file is generated in either Motorola S Record, Intel Hex Record or Daisy VLA format. For one-byte wide files, the Motorola S2 format is supported.

SIMULATOR

The Simulator simulates the operation of the ADSP-2100 and allows the user to observe the contents of the registers, buses, stacks and program and data memories as a program is being executed. The Simulator is user friendly, interactive and screen-oriented. Figure 4 shows the basic Register Display.

```
ADSP-2100 Simulator V1.5 Analog Devices Inc.     FIR_SYSTEM
          ALU                                          Address Generator #1
AX0 uuuu                        AC 0  AQ 0        I0 0009   M0 0001   L0 000F
AX1 uuuu                AR uuuu AN 0              I1 uuuu   M1 uuuu   L1 0000
AY0 uuuu                AF uuuu AV 0              I2 uuuu   M2 uuuu   L2 0000
AY1 uuuu                        AZ 0  AS 0        I3 uuuu   M3 uuuu   L3 0000
     Multiplier-Accumulator                                Addr
MX0 2000                                          Address Generator #2
MX1 uuuu    MR2 00      MR1 004A      MR0 0644    I4 1000   M4 0001   L4 000F
MY0 FE8C                MF  0000                  I5 uuuu   M5 uuuu   L5 0000
MY1 uuuu                        MV 0              I6 uuuu   M6 uuuu   L6 0000
          Shifter                                 I7 uuuu   M7 uuuu   L7 0000
SI  2000    SE uu       SR1 uuuu      SR0 uuuu              Addr
            SB uu                     SS 0        Data DM 004A   PM 3C00E5   PX 00
CNTR 0000  Astat 00  Mstat 0  Sstat 44   Imask 0  Icntl 00  PC 0015  Time 125200
                halt   : Module: FIR ROUTINE                : emulat  register hex
dspp_sim()-CMD_INP_ACKL:
>
>
>
>
>
> register
>
```

Figure 3. Register Display

By reading the Architecture Description File output of the System Builder, the Simulator configures itself to match the target system hardware. This enables the Simulator to flag operations such as attempting to write to ROM or nonexistent memory locations.

The Simulator supports full symbolic disassembly via the Debug Symbol Table File output of the Linker.

The Simulator supports three execution modes: Emulator, Extend and Single-Step. In Emulator mode, the Simulator runs at its fastest speed. The display is updated every 256 cycles. In Extend mode, the display is updated every cycle. In Single-Step mode the Simulator executes a single instruction per run command and updates the screen.

The basic format of the Simulator display includes a status line containing information about the status words, program counter and accumulated cycle time. It also provides a command window for interactive typing of commands and display of error messages and warnings. The Simulator's major informational displays include the following:

Register	The register display shows the basic processor data registers (primary or secondary bank), arithmetic status and the state of the buses and data address generators.
Program Memory	This window displays instructions in fully symbolic form. The user can change opcodes and instructions as needed.
Data Memory	This window displays the numeric contents of data memory.
Data Memory Plot	This window plots the contents of a selected range of data memory on hardware configurations that support graphics.
Status	This informational display shows breakpoints, watchpoints, port status and interrupt status.
Stack	The four columns of the Stack window each represent one of the four stacks of the processor. The user can modify the values through push and pop operations.
Trace Buffer	This displays the history of up to 4K states of the four external buses of the processor.
Cache Memory	This displays (symbolically) the contents of cache memory and whether or not an instruction in the cache is deemed valid.
Cross Reference	Displays the location of all symbol names.
Modules	Lists all available modules by name.
Help	Displays a list of Simulator commands and provides further information on them as requested.

In addition, the Simulator allows the user to modify the contents of most registers, memories and status words. Breakpoints can be set in Program Memory and watchpoints in Data Memory. Command files can be created to execute the same set of commands. This is useful for repetitive commands to bring the simulation to a specific starting condition.

User-defined addresses or values can be displayed symbolically. The state of the Simulator can be saved and restored for future simulation sessions. Contents of program and data memory can be dumped to files for use with the hardware development tools. The Simulator supports decimal and hexadecimal numeric formats. I/O to and from ports reads and writes data files which can later be analyzed.

Simulator Commands

Simulator commands allow the user to change the state of the processor. A quick summary of Simulator commands is shown below. Only the letters shown in caps must be entered to invoke the command.

Display Control Commands

ALternate	displays secondary data registers
BACkup	forces PM/DM/Trace displays to scroll back
BEep	enables beeps on user's terminal
CAche	invokes cache display mode
DECimal	forces all numbers to be displayed in decimal format
DM	invokes data memory display mode
FOrward	forces PM/DM/Trace displays to scroll forward
HELp	displays command list for access to help information
HEXadecimal	forces all numbers to be displayed in hexadecimal (the default)
Modules	displays all source modules
NOBeep	suppresses beeps at the user's terminal
NOSymbolic	forces the simulator to be non-symbolic
PLotdm	plots the contents of a selected section of DM
PM	invokes program memory display mode
PRimary	displays primary data registers
REGister	invokes register display mode
STACk	invokes stack display mode
STATus	displays Interrupt, Break and Port status
SYmbolic	forces the simulator to be symbolic (default)
TOggle	toggles display of primary and secondary register banks
TRace	invokes trace display mode
Wipe	rewrites current display
Xreference	displays cross-reference list

Operation Control Commands

EMulator	invokes emulator mode
EXTend	invokes extend mode
SInglestep	invokes single-step mode

Break Control Commands

CLEARBreak	clears a PM break address
CLEARStoptime	clears any stop times currently defined
CLEARWatch	clears a DM access watch address
COunt	sets iteration count and delay on break points
SETBreak	sets a PM break address
SETStoptime	sets a time in ns for the processor to halt
SETWatch	sets a DM access watch address

Context Control Commands

SETModule	sets the module the Simulator uses for symbolic context

File Control Commands

COMmfile	executes simulator commands found in a batch file
DUMPDm	forces a DM image dump to a file
DUMPPm	forces a PM image dump to a file
Load	reads .EXE and .SYM files and sets default module context
READImage	reads a memory image file
READSymbol	reads a symbol table file

Modify/Inspect Control Commands

CLEARTime	clears the time display
CYCLetime	sets the cycle period in ns
FINDDm	finds the occurrence of a value in DM
FINDPm	finds the occurrence of a value in PM
RESEtstack	clears stacks and reset pointers
SETDm	sets a segment of DM
SETPC	sets the PC
SETPM	sets a segment of PM
SETRegister	sets a register value

Assembly Commands

ADdsymbol	adds a user-defined symbol name
DELete	deletes one line of assembly code from PM
EXEcute	executes an assembly instruction
PAtch	patches one line of assembly code into PM
REMovesymbol	deletes a user-defined symbol name

Configuration Control Commands

BATch	turns off screen update in Emulator mode
CHipreset	simulates the hardware chip RESET
CLOse	closes a DM memory mapped I/O port
HArdware	simulates hardware powerup and sets ROM to undefined
Interrupts	activates the interrupt source
Open	opens a DM memory mapped I/O port
POwerup	simulates the hardware powerup condition

Execution Control Commands

RUn	starts processor running in Extend and Emulator modes
<cr>	starts processor running in Single-step mode

Exit Command

EXIt	exits from the Simulator and returns to the host

ADDITIONAL INFORMATION

The ADSP-2100 Software Development System is available for the PC-DOS*, MS-DOS, VAX/VMS* and UNIX* BSD 4.2 on the Sun-3. The *ADSP-2100 Cross-Software Manual* provides complete information on these tools.

Analog Devices offers a hands-on multiday workshop on programming the ADSP-2100 family of processors. The workshop is taught by our DSP Applications Engineering group and is presented several times a year at the factory in Norwood, Massachusetts. The fee includes all manuals and workbooks and lab time. The workshop can also be conducted at your site; consult us for site pricing.

ORDERING INFORMATION

Part Number	Description
ADDS-2110	Cross-Software for VAX/VMS
ADDS-2121†	System Builder, Assembler, Linker, PROM Splitter for IBM-PC*
ADDS-2122†	Simulator for IBM-PC
ADDS-2123-C	Cross-Software for Sun-3 (UNIX BSD 4.2)
ADDS-2130	C Compiler and Cross-Software for VAX/VMS*
ADDS-2131	C Compiler and Cross-Software for IBM-PC
ADDS-2133-C	C Compiler and Cross-Software for Sun-3 (UNIX BSD 4.2)
ADDS-2190	ADSP-2100 Family Workshop

*PC-DOS and IBM PC are trademarks of International Business Machines Corp. VAX/VMS is a trademark of Digital Equipment Corp. UNIX is a trademark of AT&T.

†Note that ADDS-2121 and ADDS-2122 must both be ordered to make up a complete IBM-PC Cross-Software system without the C Compiler.

ANALOG DEVICES

DSP Hardware Development Tools

ADDS-21XX

FEATURES

ADSP-2100A EVALUATION BOARD

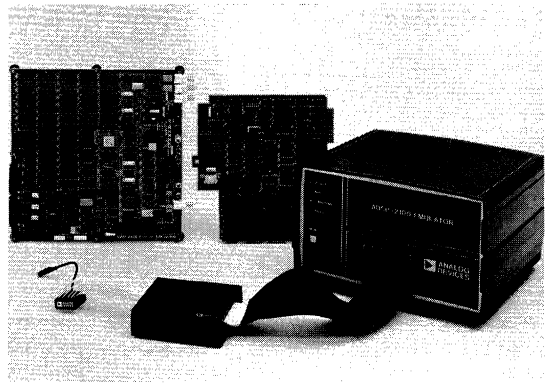
- Can Be Used to Benchmark Real-Time Performance
- Interfaces to an IBM-PC or VAX Host via RS-232 Connectors
- Operates at 8MHz
- Same Interactive, Symbolic User Interface as the Emulator and Simulator
- Three Execution Modes: Single-Step, Extend, Emulator
- Displays Contents of ADSP-2100A Registers, Program Memory, Data Memory and Stack
- Multiple Program Memory Breakpoints Supported
- 4K Program and 2K Data Memory Installed with Sockets for Expanding to Full 32K Program and 16K Data Memory
- Fully Documented Prototyping Expansion Connector to Customize Evaluation Board to Your Application
- Bidirectional Codec Channel to Process Real-World Signals
- 12-Bit Linear DAC Provides an Oscilloscope Interface
- Input Preamp with Microphone Jack and Output Amplifier with Speaker Jack Directly Supports Audio and Speech Applications

ADSP-2100A IN-CIRCUIT EMULATOR

- Performs In-Circuit Emulation
- Interfaces to an IBM-PC or VAX Host via Two RS-232 Connectors
- Operates at 8MHz
- Same Interactive, Symbolic User Interface as the ADSP-2100 Simulator and Evaluation Board
- Three Execution Modes: Single-Step, Extend, Emulator
- Displays Contents of ADSP-2100A Registers, Program Memory, Data Memory and Stack
- Supports Multiple Program Memory Breakpoints
- User-Selectable Program Memory Source: Emulator or Target System
- User-Selectable System Clock Source: Emulator, Target System or External

OPTIONAL TRACE BOARD FOR IN-CIRCUIT EMULATOR

- Buffers Up To 8K of Bus Activity for Display and Analysis
- Break Triggering on an Extensive Set of Possible Bus Conditions
- Buffer Can Be Uploaded to Host for Further Analysis
- Installs Inside Emulator Case



GENERAL DESCRIPTION

The ADSP-2100A Hardware Development Tools support the prototyping, development and debugging of applications in hardware.

The Evaluation Board allows the user to benchmark real-time performance by executing Analog Devices-supplied or user-developed DSP routines.

The In-Circuit Emulator allows the user to debug code in the actual target system.

The Trace Board enhances the In-Circuit Emulator by capturing activity on the four external buses of the processor.

The Hardware Development Tools have the same interactive, symbolic user interface as the Simulator. Single-step, extend and emulator execution modes run the processor as required for your debugging activity. Four major display modes enable users to examine contents of ADSP-2100A registers, program memory, data memory and stack. Multiple program memory breakpoints are supported.

ADSP-2100A EVALUATION BOARD

The Evaluation Board is an easy-to-use development tool for evaluating the ADSP-2100A DSP Microprocessor in real-time applications. It has three roles in the design process. As a demonstration system, you can observe the ADSP-2100A's real-time performance in executing standard DSP benchmarks. As an evaluation system, it can be used prior to designing hardware for the real-time execution of your application routines. As a simulation accelerator, application code can be executed in real time for increased productivity of software developers.

The Evaluation Board is a stand alone system consisting of an ADSP-2100A DSP Microprocessor, 4K words of (24-bit) program memory, and 2K words of (16-bit) data memory. Additional program and data memory sockets are provided and can be populated as desired up to the full 32K program and 16K data memory address space.

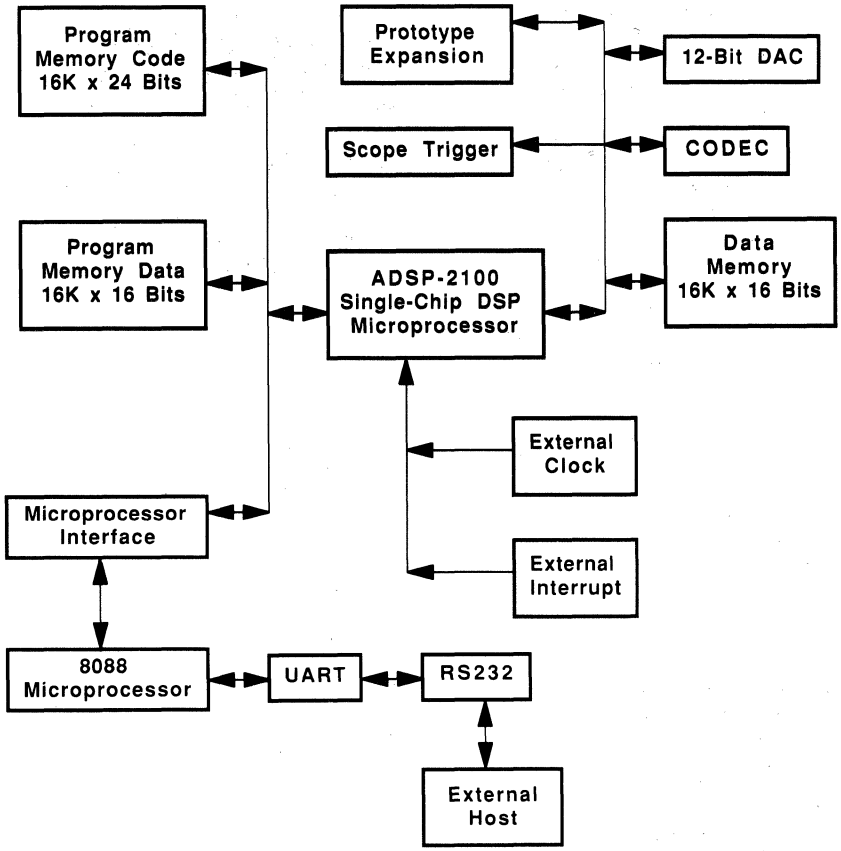


Figure 1. Evaluation Board Block Diagram

The Evaluation Board's ADSP-2100A runs under the control of an on-board host processor enabling the user to access a variety of powerful debugging tools. When interfaced to an external host computer system running the Cross-Software, the Evaluation Board serves as a real-time development tool.

The emulator mode runs the processor at full speed. Extend mode updates the screen every cycle during program execution. Single-step mode executes a single instruction per carriage return. In addition, multiple program memory breakpoints are supported.

The Evaluation Board has four major display modes: register, program memory, data memory and stack. Register mode displays the contents of the ADSP-2100A's primary and alternate registers. Program memory mode displays the contents of program memory. Data memory mode displays the contents of data memory. Stack display shows the contents of the ADSP-2100A's program counter stack and count stack.

The Evaluation Board connects to a terminal and host computer via two RS-232C serial connectors.

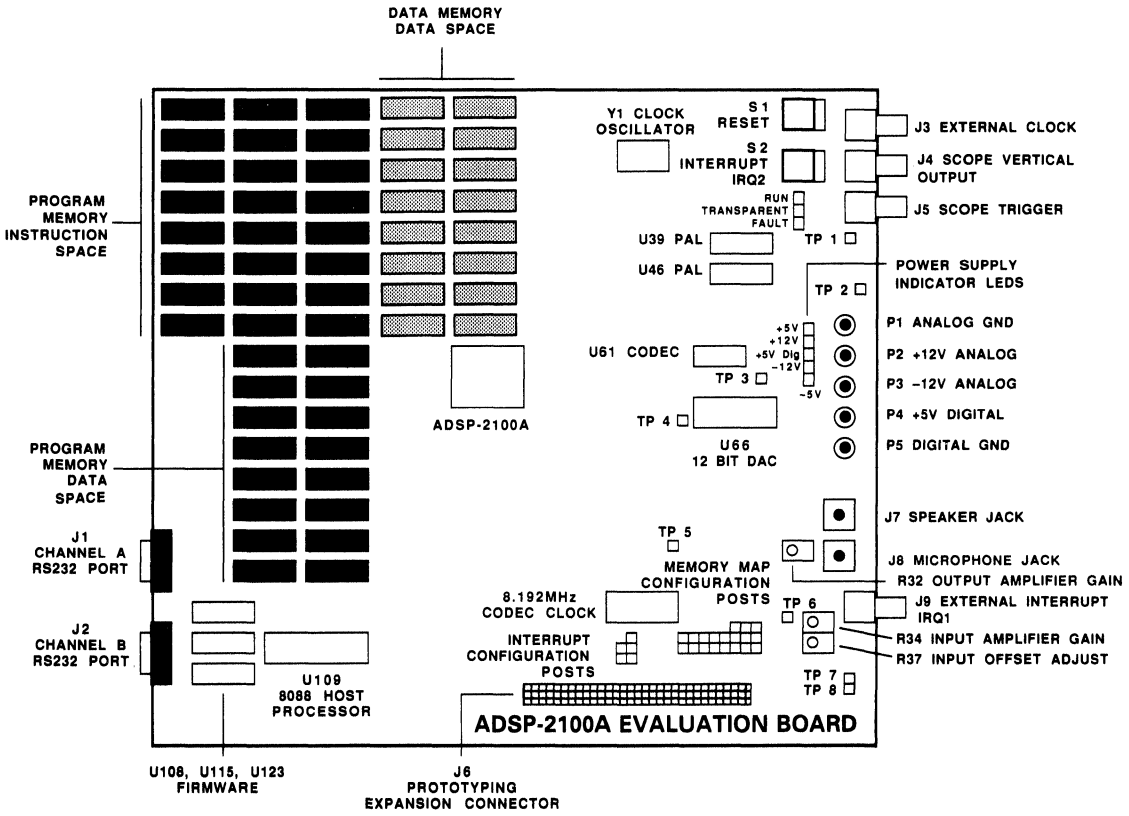


Figure 2. Evaluation Board

Built-in analog interfaces provide access to real signals for easy implementation of audio, speech and telecommunications applications. A bidirectional codec channel and an undedicated 12-bit linear D/A converter process real-world signals. The prototyping expansion bus allows you to construct custom hardware to reflect or test the eventual hardware environment. In addition, three BNC connectors interface to external instrumentation. An integral microphone jack and input pre-amplifier, along with a speaker jack and output amplifier, support speech and telecommunication applications.

With a microphone, speaker and oscilloscope you can easily implement audio and speech applications. The microphone and speaker are connected to the bidirectional codec channel via jacks on the input preamp and output amplifier. The codec is a National Semiconductor TP3051. It is a memory-mapped peripheral of the ADSP-2100A that can be written to or read from using the Data Memory Read and Data Memory Write commands. The codec represents the input/output sample in an 8-bit binary form. By using the standard μ-law nonlinear transformation, the codec's effective dynamic range can be extended to 13 bits. The codec samples data at a frequency of 8.192kHz using a dedicated clock generator. Communication between the codec and the ADSP-2100A is synchronized with the DMACK signal. The codec rejects signals that do not fall in the range of 200Hz to 3400Hz and should be used only in speech or audio applications in which telephone-quality signals are adequate. An input pre-amplifier (Analog Devices AD741 operational amplifier) and output audio amplifier (National Semiconductor LM338 Audio Power Amplifier) are connected to the input and output of the codec.

To display processed data, the oscilloscope is connected to the 12-bit linear DAC via a BNC connector. The DAC is an Analog Devices AD667 12-bit D/A converter. It is a memory-mapped peripheral of the ADSP-2100A that can be written using the Data Memory Write commands. The DAC is intended for use as an analog output for the display of processed data on an oscilloscope. It is not intended as a means of reconstructing sampled data processed by the ADSP-2100A; it lacks the deglitching circuitry and anti-imaging filtering required of such a system. The user can construct a linear analog interface consisting of an A/D converter, D/A converter, antialiasing filter and anti-imaging filter using the prototyping expansion connector.

The prototyping expansion connector provides the data, address and interface signals for customizing the Evaluation Board. For example, analog circuitry composed of linear A/D and D/A converters and antialiasing filters may be connected to the Evaluation Board for implementing filtering applications. The 96-contact prototyping expansion connector brings out the following signals:

Input Signals

EIRQ3	External Interrupt Request 3 (Highest Priority)
EIRQ2	External Interrupt Request 2
EIRQ1	External Interrupt Request 1
EIRQ0	External Interrupt Request 0
EBR	External Bus Request . Allows your target board to request control of the data memory interface.
EDMACK	Data Memory Acknowledge. Used for asynchronous transfers across the data memory interface.
THALT	Processor Halt by Target System. Assertion of THALT halts the ADSP-2100A.
RESETOUT	System Reset Output. The ADSP-2100A's RESET line is available at this contact as an output only.

Output Signals

+12V	+12V Analog
AGND	Analog Ground
−12V	−12V Analog
GND	Digital Ground
$\overline{BG}$	Bus Grant. Acknowledges an external bus request ($\overline{BR}$).
DMA13–0	Data Memory Address bits
$\overline{DMRD}$	Data Memory Read. Indicates a read operation on the data memory interface.
$\overline{DMWR}$	Data Memory Write. Indicates a write operation on the data memory interface.
$\overline{DMS}$	Data Memory Select. Signals a data memory access on the data memory interface.
TRAP	Indicates the execution of a TRAP instruction. The ADSP-2100A halts execution and the TRAP signal remains asserted until THALT is asserted.
ECE8–1	External Chip Enables 8 through 1. These outputs are memory-mapped locations.

Bidirectional Signals

DMD15–0	Data Memory Data Bus

The Evaluation Board must be interfaced to an IBM-PC (with VT100 emulation) or VAX/VMS system via the RS-232 connectors. This host computer must also run the ADSP-2100 Cross-Software. The board requires ±12V and +5V power supplies.

ADSP-2100A IN-CIRCUIT EMULATOR

The In-Circuit Emulator allows you to debug code (developed with the Software Development tools) in the actual target system. The Emulator uses an ADSP-2100A to emulate the processor. It plugs into the target system's ADSP-2100A socket and operates at the ADSP-2100A's cycle rate. The Emulator provides a software interface similar to the Cross-Software Simulator and to the Evaluation Board.

Once your program has been debugged in the software environment you can further prove and debug in the hardware area using the Emulator. It provides a variety of ways to download your program into the actual hardware, executing out of emulator program memory or target system program memory, for example, or using any of three sources for the system clock.

The Emulator supports three execution modes. In emulator mode the Emulator runs at the full processor speed and halts only when a break condition is encountered. Break conditions include breakpoints, traps, halt on keyboard interrupt and target system voltage below 4.5V. While the Emulator is running in emulator mode, only the program counter and elapsed time information is updated. When the Emulator halts, the full screen is updated.

Extend mode runs the processor in a continuous single-step manner, updating the display after each processor cycle. Instructions are disassembled on the screen as they are executed. In emulator and extend modes, emulation can be halted by setting a breakpoint at a specified location in program memory.

In single-step mode, the Emulator executes one instruction and halts. All display contents are updated and instructions are disassembled as they are executed. The next instruction is executed if you type a carriage return or enter the RUN command.

The Emulator has four major display modes (five with the Trace Board installed). Register display shows the contents of the ADSP-2100A's primary and secondary registers. The program memory and data memory displays show the contents of program and data memories. Stack display shows the contents of the ADSP-2100A's program counter stack and count stack.

Using the same interactive, symbolic user interface as the Simulator, the Emulator allows the user to modify the contents of registers, program memory, data memory and the program counter. Breakpoints can be set in the emulator-based program memory. User-defined addresses and values can be displayed symbolically. Numbers can be specified and displayed in either decimal or hexadecimal format.

The Emulator has other features. The baud rate and parity settings for communications between the Emulator and the host computer can be specified by the user's terminal. The Emulator Pod can be activated and deactivated under software control. The program memory source can be either the Emulator's internal program memory RAM or the target system's program memory. Also, files can be downloaded from the host system. The system clock can be selected from either the Emulator's internal clock, the target system's clock or an external clock generator.

Propagation Delays

Although the Emulator matches the ADSP-2100A closely in performance for a few signals, its timing is degraded somewhat from that of the processor. Propagation delays and, in some cases, software overhead account for the delays. The signals with degraded timing are:

- CLKIN
- $\overline{\text{IRQ}}$
- $\overline{\text{BR}}$
- $\overline{\text{RESET}}$
- $\overline{\text{HALT}}$
- TRAP
- $\overline{\text{PMWR}}$ and $\overline{\text{PMRD}}$

All other signals operate at essentially the same timing as the processor in a non-emulator system. Complete information and timing diagrams are given in Appendix B of the *ADSP-2100 Emulator Manual*.

TRACE BOARD FOR ADSP-2100A IN-CIRCUIT EMULATOR

The Emulator supports an optional, factory-installed Trace Board. The Trace Board keeps a running history of past external bus states PMA, DMA and DMD in an 8K buffer. The Trace Buffer Display shows the past external bus states of the ADSP-2100A.

The Trace Board allows you to trigger on bus conditions. Emulation can be halted after detecting a specified combination of bus states. The IGNORE option turns off the trace during certain PMA ranges in order to skip over sections of code. The Trace Board can trigger on the following eleven different bus combinations:

PMA AND DMA
PMA AND DMD
DMA AND DMD
PMA AND DMA AND DMD
PMA OR (DMA AND DMD)
DMA OR (PMA AND DMD)
DMD OR (PMA AND DMA)
PMA OR DMA OR DMD
(PMA AND DMA) OR (PMA AND DMD)
(PMA AND DMA) OR (DMA AND DMD)
(PMA AND DMA) OR (DMA AND DMD)

In addition, the trace buffer can be uploaded from trace board to host computer.

ADDITIONAL INFORMATION

Request the *ADSP-2100 Emulator Manual* or the *ADSP-2100 Evaluation Board Manual* from your Analog Devices Sales Engineer for further information.

ORDERING INFORMATION

Part Number	Description
ADDS-2150A*	8MHz ADSP-2100A In-Circuit Emulator (110V)
ADDS-2150AE*	8MHz ADSP-2100A In-Circuit Emulator (220V)
ADDS-2151A*	8MHz ADSP-2100A In-Circuit Emulator with Trace Board (110V)
ADDS-2151AE*	8MHz ADSP-2100A In-Circuit Emulator with Trace Board (220V)
ADDS-2160*	8MHz ADSP-2100A Evaluation Board
Upgrade Kits	
ADDS-2161	Trace Board Upgrade for ADDS-2150
ADDS-2162	Trace Board Upgrade for ADDS-2150A

*A 12.5MHz version of this product is planned. Please contact factory for further information.

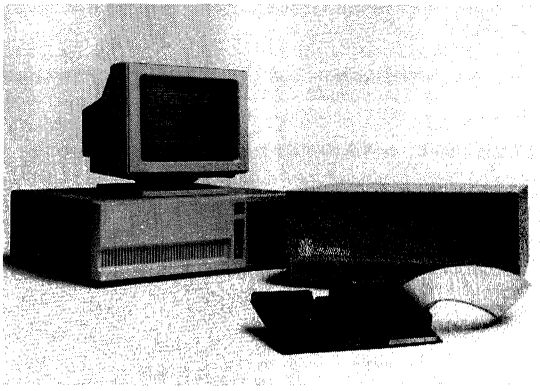
ANALOG DEVICES

In-Circuit Emulator

ADSP-2101

FEATURES

Supports the ADSP-2101 DSP Microcomputer
Performs In-Circuit Emulation
Operates at the Full Clock Rate of the ADSP-2101 (12.5MHz)
Same Interactive, Symbolic User Interface as the ADSP-2101 Simulator
Single-Step, Full Speed and Periodic Update Execution
Supports Breakpoints and Triggers
User Selectable Memory Source: Emulator or Target System
User Selectable Clock Source: Emulator or Target System
RS-232C Interface to Host System Supporting Up to 19.2 kb/s
8K Trace Buffer
On-Line Assembly/Disassembly
Performance Analysis
Histogram Profiling of Executing Code
Time-Tags on Trace Buffer Contents
Modular Hardware Based on VME Bus



GENERAL DESCRIPTION

The ADSP-2101 In-Circuit Emulator allows the user to debug code developed with the ADSP-210X Cross-Software Modules in an actual target system. The Emulator, which uses an ADSP-2101 to emulate the processor, plugs into the target system's ADSP-2101 socket and operates at the ADSP-2101's cycle rate. The Emulator supports three different types of memories: program memory, data memory and boot memory. All memories can be downloaded by the user. The boot memory interface is supported. The Emulator can operate from either emulator or target system based memory.

The Emulator can run at the full processor speed updating the display only when execution halts. The Emulator can also run in semi-real time updating the display at a predetermined rate up to every cycle. The user can also single-step through code from the keyboard.

The Emulator displays information about the state of the emulation in a variety of windows, similar to the ADSP-2101 Simulator. These include register, memory, execution profile and trace windows. The contents of the ADSP-2101's registers are displayed including serial port control registers, interval timer control registers and memory-mapped registers. The memory windows can show the contents of all memories on and off chip including boot memory. The trace display shows the contents of the 8K deep trace buffer. The execution profile shows the use of program modules to measure the efficiency and performance of code.

The Emulator allows the user to modify the contents of registers and memory. Breakpoints can be set and user-defined addresses and values can be displayed symbolically. For ease of use, the user can either specify decimal or hexadecimal format.

On-line assembly allows users to modify the code starting at a specified location and load instructions on a line-by-line basis. The disassembled contents of each address can be displayed before a new assembled instruction is stored.

The Emulator supports an 8192-frame deep trace buffer that stores data and address buses as well as control signals. Triggering on bus events, control lines and serial ports is supported. These events can be logically ANDed, ORed or negated to define a trigger event.

Consult the factory for current status.

ANALOG DEVICES

12.5 MIPS DSP Microprocessor

ADSP-2100/ADSP-2100A

FEATURES

Pin- and Code-Compatible DSP Microprocessors
ADSP-2100, 6.144MHz and 8.192MHz
ADSP-2100A, 10.24MHz and 12.5MHz
Separate Program and Data Buses, Extended Off-Chip
Single-Cycle Direct Access to 16K x 16 of Data Memory
Single-Cycle Direct Access to 32K x 24 of Program Memory
Dual Purpose Program Memory for Both Instruction and Data Storage
Three Independent Computational Units: ALU, Multiplier/Accumulator and Barrel Shifter
Two Independent Data Address Generators
Powerful Program Sequencer
Internal Instruction Cache
Provisions for Multiprecision Computation and Saturation Logic
Single-Cycle Instruction Execution
Multifunction Instructions
Four External Interrupts
80ns Cycle Time (ADSP-2100A)
790mW Maximum Power Dissipation (ADSP-2100A, J and K Grades)
100-Pin Grid Array, 100-Lead PQFP (JEDEC Style)

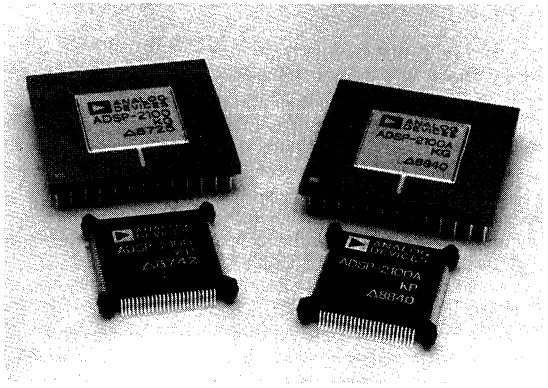
APPLICATIONS

Optimized for DSP Algorithms Including
Digital Filtering
Fast Fourier Transforms
Applications Include
Image Processing
Radar, Sonar
Speech Processing
Telecommunications

GENERAL DESCRIPTION

The ADSP-2100 and ADSP-2100A are pin- and code-compatible single-chip microprocessors optimized for digital signal processing (DSP) and other high-speed numeric processing applications. The ADSP-2100 and ADSP-2100A are both fabricated in a low-power double-layer metal CMOS process. Together, they offer a span of performance from 6MHz to 12.5MHz. All descriptions of the ADSP-2100 in the text of this data sheet refer to both the ADSP-2100A and the ADSP-2100 versions since they have identical architectures and instruction sets. Timing and electrical specifications differ as shown in those sections of the data sheet.

Both processors integrate computational units, data address generators and a program sequencer in a single device. The ADSP-2100 architecture makes efficient use of external memories for program and data storage, freeing silicon area for increased processor performance. The resulting processor combines the functions and performance of a bit-slice/building block system with the ease of design and development support of a general purpose microprocessor.



The ADSP-2100A (K grade) operates at 12.5MHz. Every instruction executes in a single 80ns cycle. The ADSP-2100A (J and K grades) dissipates less than 790mW while the ADSP-2100 dissipates less than 475mW.

The ADSP-2100's flexible architecture and comprehensive instruction set support a high degree of operational parallelism. Because all instructions execute in a single cycle, MHz = MIPS. In one cycle the ADSP-2100 can:

- generate the next program address
- fetch the next instruction
- perform one or two data moves
- update one or two data address pointers
- perform a computational operation.

DEVELOPMENT SYSTEM

The ADSP-2100 and ADSP-2100A are supported by a complete set of tools for software and hardware system development. The Cross-Software System provides a System Builder for defining the architecture of simulated systems under development, an Assembler, a Linker and a interactive Simulator. An ANSI (draft) Standard C Compiler supports program development in this widely used programming language, producing ADSP-2100 Assembly code which may be assembled, linked and simulated with the other development system tools. A PROM Splitter generates PROM burner compatible files. An In-Circuit Emulator is available for hardware debugging.

An Evaluation Board is available for quick assessment of actual processor performance in a prepackaged hardware environment.

ADDITIONAL INFORMATION

For additional information on the architecture and instruction set of the processor, refer to the *ADSP-2100 User's Manual*. For more information about programming and the Development System, refer to the *ADSP-2100 Cross-Software Manual* and the *ADSP-2100 Emulator Manual*. For examples of applications routines, refer to the *ADSP-2100 Applications Handbook, Volume 1* or *Volume 2*. Manuals are available only from your local Analog Devices sales office. There is also a quarterly newsletter, *DSPatch*™, supporting Analog Devices' digital signal processing customers.

ARCHITECTURE OVERVIEW

Figure 1 is an overall block diagram of the ADSP-2100. The processor contains three independent computational units: the ALU, the multiplier/accumulator (MAC) and the Shifter. The computational units process 16-bit data directly and have provisions to support multiprecision computations. The ALU performs a standard set of arithmetic and logic operations; division primitives are also supported. The MAC performs single-cycle multiply, multiply/add and multiply/subtract operations. The Shifter performs logical and arithmetic shifts, normalization, denormalization and derive exponent operations. The Shifter can be used to efficiently implement any degree of numeric format control, up to and including full floating point representations. The computational units are arranged side-by-side instead of serially for flexible operation sequencing. The internal result (R) bus directly connects the computational units so that the output of any unit may be the input of any unit on the next cycle.

A powerful program sequencer and two dedicated data address generators ensure efficient use of these computational units. The program sequencer generates the next instruction address. To minimize overhead cycles, the sequencer supports conditional jumps, subroutine calls and returns in a single cycle. With internal loop counters and loop stacks, the ADSP-2100 executes looped code with zero overhead; no explicit jump instructions are required to maintain the loop.

The data address generators (DAGs) handle address pointer updates. Each DAG keeps track of up to four address pointers. Whenever the pointer is used to access external data (indirect addressing), it is modified by a prespecified value. A length value may be associated with each pointer to implement automatic modulo addressing for circular buffers. With two independent DAGs, the processor can generate two addresses simultaneously for dual operand fetches.

Efficient data transfer is achieved with the use of five internal buses.

- Program Memory Address (PMA) bus
- Program Memory Data (PMD) bus
- Data Memory Address (DMA) bus
- Data Memory Data (DMD) bus
- Result (R) bus

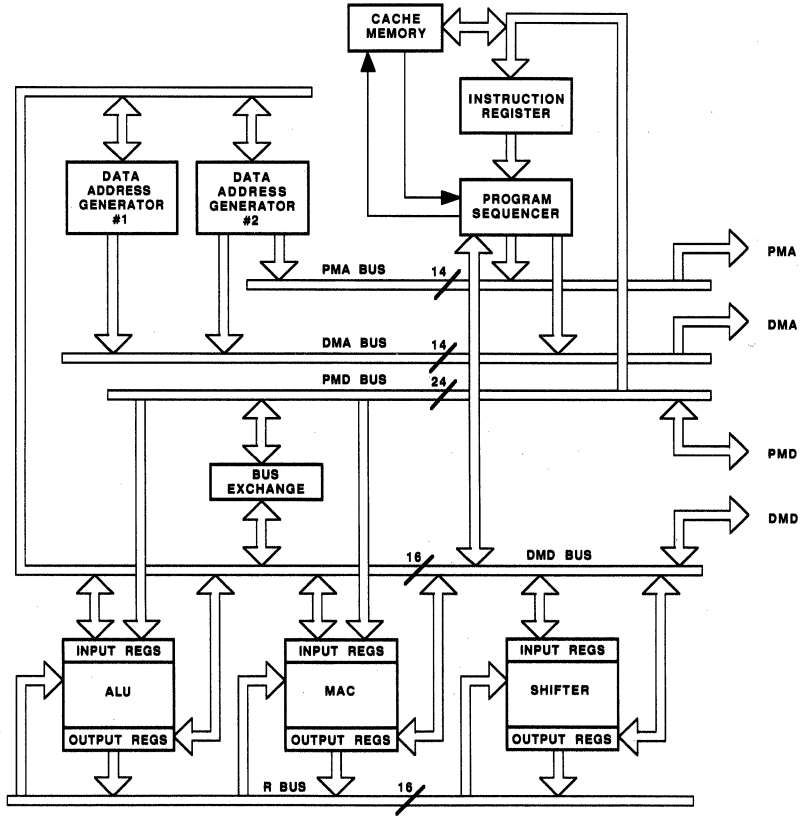


Figure 1. ADSP-2100 Block Diagram

DSPatch is a trademark of Analog Devices, Inc.

The program memory (PMD, PMA) buses and data memory (DMA, DMD) buses extend off-chip to provide direct connections to external memories. The DMD bus is the primary bus for routing data internally and to/from external data memory. The 14-bit DMA bus provides direct addressing of 16K × 16 of external memory. Although the primary function of the program memory is for storing instructions, it can also store data. In this case, the PMD bus provides a path for routing data to/from program memory, permitting dual operand fetches. The 14-bit PMA bus provides direct addressing of 16K × 24 of external memory, expandable to 32K × 24 by using the program memory data access (PMDA) signal as the 15th address line.

When a data fetch from program memory is required, an extra memory cycle is automatically appended to enable the next instruction fetch. To avoid this extra cycle, the ADSP-2100 has an internal instruction cache (16 instructions deep) which serves as an alternate source for the next instruction. The cache monitor circuit transparently determines when the cache contents are valid. When the next instruction is in the cache, no extra cycle is necessary.

The data memory interface supports slower memories and memory-mapped peripherals with wait states. The data memory acknowledge (DMACK) signal provides the necessary handshake. External devices can gain control of program or data buses independently with bus request/ grant signals ($\overline{BR}$, and $\overline{BG}$).

The ADSP-2100 can respond to four external interrupts, which are internally prioritized, maskable and independently programmable as either edge- or level-sensitive. Additional external controls are provided by the $\overline{RESET}$, $\overline{HALT}$ and TRAP signals. With both $\overline{BR}$ and $\overline{RESET}$ recognized, the ADSP-2100 idles, consuming the least possible current.

The ADSP-2100 instruction set provides flexible data moves and multifunction (data moves with a computation) instructions. Every instruction can be executed in a single processor cycle. The ADSP-2100 assembly language uses an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

A pin description and detailed discussion of each section of the ADSP-2100 follows.

2

Pin Description

This section summarizes the pin description of the processor by interface. In this data sheet, when groups of pins are identified with subscripts, as in PMD_{23-0}, the highest numbered pin (PMD_{23}) is the MSB.

Pin Name	Type	Function
Clocks:		
CLKIN	Input	Master input clock operating at four times the processor instruction rate. Nominally 50% duty cycle. The phases of CLKIN define the eight internal processor states making up one instruction cycle.
CLKOUT	Output	Output clock operating at the processor instruction rate with a 50% duty cycle. Synchronized to the internal processor states.
Interrupt Request Lines:		
$\overline{IRQ}_{3-0}$	Input	Interrupt Request lines that may be either edge triggered or level sensitive. Interrupts are prioritized and individually maskable.
Control Interface:		
$\overline{RESET}$	Input	Master Reset must be asserted long enough to assure proper reset. When $\overline{RESET}$ is released, execution begins at program memory location 0004.
$\overline{HALT}$	Input	Used to halt the processor. All control signals become inactive and the address and data buses are driven for observation.
TRAP	Output	Used to indicate the execution of a TRAP instruction. Remains asserted until $\overline{HALT}$ is asserted by an external device.
$\overline{BR}$	Input	Bus Request used by an external device to request control of the program and data memory interface. Upon receiving $\overline{BR}$ the processor halts execution at the completion of the current cycle and relinquishes the program and data memory interface by tristating PMA, PMD, $\overline{PMS}$, $\overline{PMWR}$, $\overline{PMRD}$, PMDA, DMA, DMD, $\overline{DMS}$, $\overline{DMRD}$ and $\overline{DMWR}$. The processor regains control when $\overline{BR}$ is released.
$\overline{BG}$	Output	Bus Grant. Acknowledges a bus request ($\overline{BR}$), indicating that the external device may take control. $\overline{BG}$ is held asserted until $\overline{BR}$ is released.
Program Memory Interface:		
PMA_{13-0}	Output	Program Memory Address Bus; tristated when $\overline{BG}$ is asserted.
PMD_{23-0}	Bidirectional	Program Memory Data Bus; tristated when $\overline{BG}$ is asserted.
$\overline{PMS}$	Output	Program Memory Select signals a program memory access on the PM interface. Usable as a chip select signal for external memories. Remains asserted on successive program memory accesses. HI only when the processor is halted or after execution of a TRAP instruction. Tristated when $\overline{BG}$ is asserted.

Program Memory Interface:

$\overline{PMRD}$	Output	Program Memory Read indicates a read operation on the PM interface. Also usable as a read strobe or output enable signal. Tristated when $\overline{BG}$ is asserted.
$\overline{PMWR}$	Output	Program Memory Write establishes the direction of data transfer on the PM interface. Also usable as a write strobe. Tristated when $\overline{BG}$ is asserted.
PMDA	Output	Program Memory Data Access used to distinguish instruction and data fetches from PM. Asserted high when data, as opposed to instruction, are accessed. Also usable as a fifteenth PM address bit. Tristated when $\overline{BG}$ is asserted.

Data Memory Interface:

DMA_{13-0}	Output	Data Memory Address Bus; tristated when $\overline{BG}$ is asserted.
DMD_{15-0}	Bidirectional	Data Memory Data Bus; tristated when $\overline{BG}$ is asserted.
$\overline{DMS}$	Output	Data Memory Select signals a Data Memory Access on the Data Memory interface. Usable as a chip select signal for external memories. Remains asserted on successive data memory accesses. HI only when the processor is halted or after execution of a TRAP instruction. Tristated when $\overline{BG}$ is asserted.
$\overline{DMRD}$	Output	Data Memory Read indicates a read operation on the Data Memory interface. Also usable as a read strobe or output enable signal. Tristated when $\overline{BG}$ is asserted.
$\overline{DMWR}$	Output	Data Memory Write indicates a write operation on the Data Memory interface. Also usable as a write strobe. Tristated when $\overline{BG}$ is asserted.
DMACK	Input	Data Memory Acknowledge signal used for asynchronous transfers across the DM interface. Indicates that data memory or memory-mapped peripherals are ready for data transfer. If DMACK is not asserted when checked by the processor, wait states are automatically generated until DMACK is asserted.

Supply Rails:

V_{DD}	Supply	Power supply rail nominally +5VDC. There are four V_{DD} pins.
GND	Ground	Power supply return. There are nine GND pins.

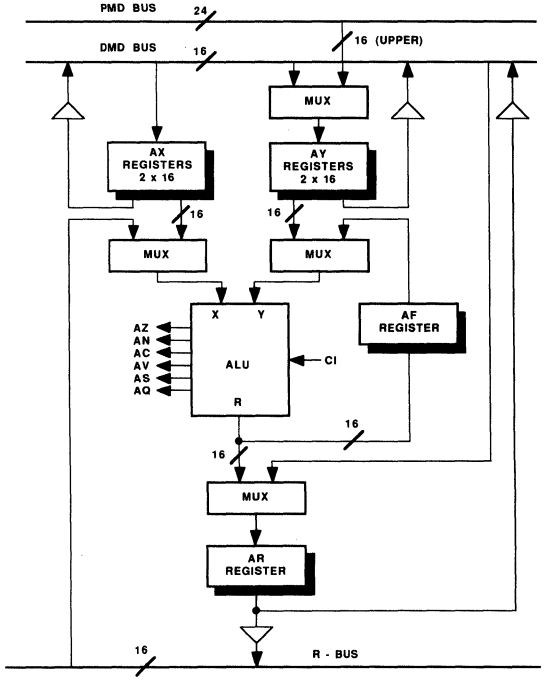


Figure 2. ALU Block Diagram

Arithmetic/Logic Unit

Figure 2 shows a block diagram of the Arithmetic/Logic Unit (ALU).

The ALU provides a standard set of general purpose arithmetic and logic functions: add, subtract, negate, increment, decrement, absolute value, AND, OR, Exclusive OR and NOT. Two divide primitives are also provided to facilitate division. The ALU takes two 16-bit inputs, X and Y, and generates one 16-bit output, R. It accepts the carry (AC) bit in the arithmetic status register (ASTAT) as the carry-in (CI) bit. The carry-in feature enables multiprecision computations. Six arithmetic status bits are generated: AZ (zero), AN (negative), AV (overflow), AC (carry), AS (sign) and AQ (quotient). These status bits are latched in ASTAT.

The X input port can be fed by either the AX register file or any result registers on the R-bus (AR, MR0, MR1, MR2, SR0, or SR1). The AX register file contains two registers, AX0 and AX1. The AX registers can be loaded from the DMD bus. The Y input port can be fed by either the AY register file or the ALU feedback (AF) register. The AY register file contains two registers, AY0 and AY1. The AY registers can be loaded from either the DMD bus or the PMD bus.

The register file outputs are dual ported so that one register can drive the ALU input while either one simultaneously drives the DMD bus. The ALU output can be latched in either the AR register or the AF register.

The AR register has a saturation capability; it can automatically output plus or minus the maximum value if an overflow or underflow occurs. The saturation mode is enabled by a bit in the mode status register (MSTAT). The AR register can drive both the R-bus and the DMD bus and can be loaded from the DMD bus.

The ALU contains a duplicate bank of registers shown in Figure 2 as a "shadow" behind the primary registers. The secondary set contains all the registers described above (AX0, AX1, AY0, AY1, AF, AR). Only one set is accessible at a time. The two sets of registers allow fast context switching for interrupt servicing. The active set is determined by a bit in MSTAT.

Multiplier/Accumulator

The multiplier/accumulator (MAC) implements high-speed multiply, multiply/add and multiply/subtract operations. Figure 3 shows a block diagram of the MAC section.

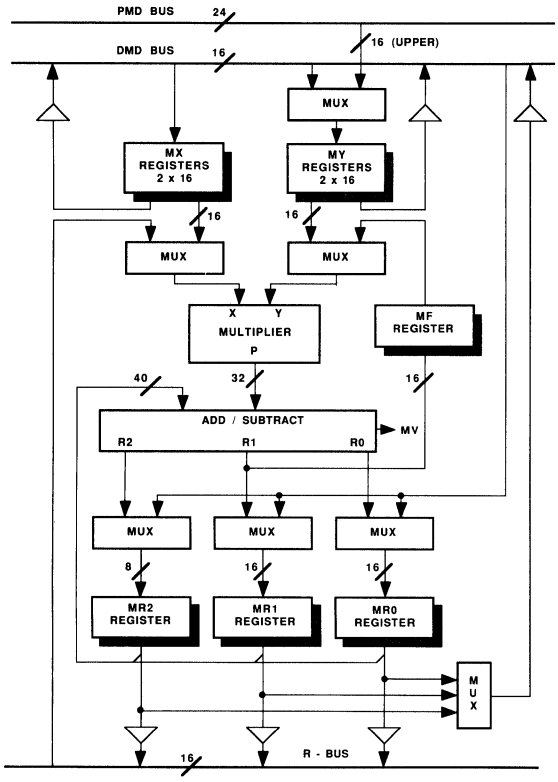


Figure 3. MAC Block Diagram

The multiplier takes two 16-bit inputs, X and Y, and generates one 32-bit output, P. The 32-bit output is routed to a 40-bit accumulator which can add or subtract the P output from the value in MR. MR is a 40-bit register which is divided into three sections: MR0 (bits 0-15), MR1 (bits 16-31), and MR2 (bits 32-39). The result of the accumulator is either loaded into the MR register or into the 16-bit MAC feedback (MF) register. The multiplier accepts the X and Y inputs in either signed or unsigned formats. The result is shifted one bit to the left automatically to remove the redundant sign bit for fractional justification. The accumulator generates one status bit, MV, which is set when the accumulator result overflows the 32-bit boundary. A saturate command is available to change the content of the MR register to the maximum or minimum 32-bit value when MV is set. The accumulator also has the capability for rounding the 40-bit result at the boundary between bit 15 and bit 16.

The MAC and ALU registers are similar. The X input port can be fed by either the MX register file (MX0, MX1) or any result registers on the R-bus (AR, MR0, MR1, MR2, SR0 or SR1).

The MX register file is readable and loadable from the DMD bus and has dual-ported outputs.

The Y input port can be fed by either the MY register file (MY0, MY1) or the MF register. The MY register file is readable from the DMD bus and readable and loadable from both the DMD and the PMD bus. Its outputs are dual ported.

The accumulator output can be latched in either the MR register or the MF register. The MR register is connected to both the R-bus and the DMD-bus. Like the ALU section, the MAC section contains two complete banks of registers (MX0, MX1, MY0, MY1, MF, MR0, MR1, MR2) to allow fast context switching.

Shifter

The Shifter gives the ADSP-2100 its unique capability to handle data formatting and numeric scaling. Figure 4 shows a block diagram of the Shifter.

The Shifter can be divided into the following components: the shifter array, the OR/PASS logic, the exponent detector and the exponent compare logic. These components give the Shifter its six basic functions: arithmetic shift, logical shift, normalization, denormalization, derive exponent and derive block exponent.

The shifter array is a 16×32-barrel shifter. It accepts a 16-bit input and can place it anywhere in the 32-bit output field, from off-scale right to off-scale left. The Shifter can perform arithmetic shifts (shifter output is sign-extended to the left) or logical shifts (shifter output is zero-filled to the left). The placement of the 16-bit input is determined by the control code (C) and the HI/LO reference signal. The control code can come from one of three sources: directly from the instruction (immediate arithmetic or logical shift), from the SE register (denormalization) or the negated value of the SE register (normalization). The shifter input can come from either the 16-bit SI register or any result register on the R-bus. The 32-bit output of the shifter array is fed to the OR/PASS circuit. The result can be either logically OR-ed with the current contents of the SR register or passed directly to the SR register. The SR register is divided into two 16-bit sections: SR0 (bits 0-15) and SR1 (bits 16-31).

The shifter input is also routed to the exponent detector circuitry. The exponent detector generates a value to indicate how many places the input must be up-shifted to eliminate all but one of the sign bits. This value is effectively the base 2 exponent of the number. The result of the exponent detector can be latched into the SE register (for a normalize operation) or can be sent to the exponent compare logic. The exponent compare logic compares the derived exponent with the value in the SB register and updates the SB register only when the derived exponent value is larger than the current value in the SB register. Therefore, the exponent compare logic can be used to find the largest exponent value in an array of shifter inputs.

The Shifter includes the following registers: the SI register, the SE register, the SB register and the SR register. All these registers are readable and loadable from the DMD-bus. The SR register can also drive the R-bus. Like the ALU and MAC, the Shifter contains two complete banks of registers for context switching. Each set contains all the registers described above, but only one set is accessible at a time. The active set is determined by a bit in MSTAT.

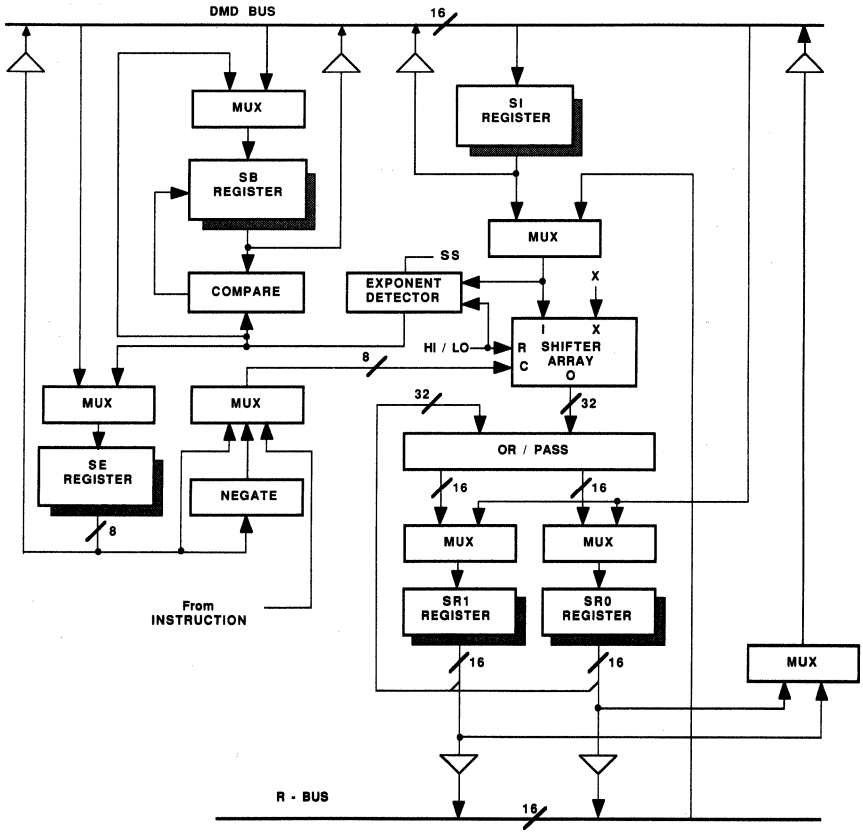


Figure 4. Shifter Block Diagram

Data Address Generators

Figure 5 shows a block diagram of a data address generator.

The data address generators (DAGs) provide indirect addressing for data stored in external memories. The processor contains two independent DAGs so that two data operands (one in program memory and one in data memory) can be addressed simultaneously. The two data address generators are identical except that DAG1 has a bit reversal option on the output and can only generate

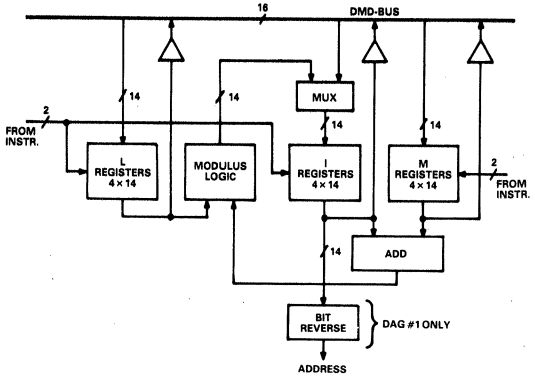


Figure 5. Data Address Generator

data memory addresses, while DAG2 can generate both program and data memory addresses but has no bit reversal capability.

There are three register files in each DAG: the modify (M) register file, the indirect (I) register file, and the length (L) register file. Each of these register files contain four 14-bit registers which are readable and loadable from the DMD-bus. The I registers hold the actual addresses used to access external memory. When using the indirect addressing mode, the selected I register content is driven onto either the PMA or DMA bus. This value is post-modified by adding the content of the selected M register. The modified address is passed through the modulus logic. Associated with each I register is an L register which may contain the length of the buffer addressed by the I register. The L register and the modulus logic together enable circular buffer addressing with automatic wrap around at the buffer boundary. The modulus logic is disabled by setting the length of the associated buffer to zero.

Program Sequencer

The program sequencer incorporates powerful and flexible mechanisms for program flow control such as zero-overhead looping, single-cycle branching (both conditional and unconditional), and automatic interrupt processing. Figure 6 shows a block diagram of the program sequencer.

The sequencing logic controls the flow of the program execution. It outputs a program memory address onto the PMA bus from

one of four sources: the PC incrementer, PC stack, instruction register or interrupt controller. The next address source selector controls which of these four sources are selected based on the current instruction word and the processor status. A fifth possible source for the next program memory address is provided by DAG2 when a register indirect jump is executed.

The program counter (PC) is a 14-bit register which contains the address of the currently executing instruction. The PC output goes to the incrementer. The incremented output is selected as the next program memory address if program flow is sequential. The PC value is pushed onto the 16 × 14 PC stack when a CALL instruction is executed or when an interrupt is processed. The PC stack is popped when a return from subroutine or interrupt is executed. The PC stack is also used in zero-overhead looping.

The program sequencer section contains five status registers. These are the Arithmetic Status register (ASTAT), the Stack Status register (SSTAT), the Mode Status register (MSTAT), the Interrupt Control register (ICNTL) and the Interrupt Mask register (IMASK). These registers are described in detail in the next section.

The interrupt controller allows the processor to respond to one of four external interrupts with a minimum of overhead. The interrupts are internally prioritized and are individually maskable. Each interrupt can be set to be either edge- or level-sensitive. Depending on a bit in the interrupt control register (ICNTL), interrupt routines can either be nested, with higher priority interrupts taking precedence, or processed sequentially, with only one interrupt service active at a time. When responding to an interrupt, the status registers ASTAT, MSTAT, IMASK are pushed onto the status stack and the PC counter is loaded with the appropriate vectored address. The status stack is four levels deep to allow four levels of interrupt nesting. The stack is automatically popped when return from interrupt is executed.

The vector addresses for each interrupt are fixed at the lowest four addresses in the program memory space. Single-word, single-cycle branch instructions may be placed at these locations to transfer control to the appropriate interrupt service routine.

The down counter and the count stack implement a powerful looping mechanism. The down counter is a 14-bit register with

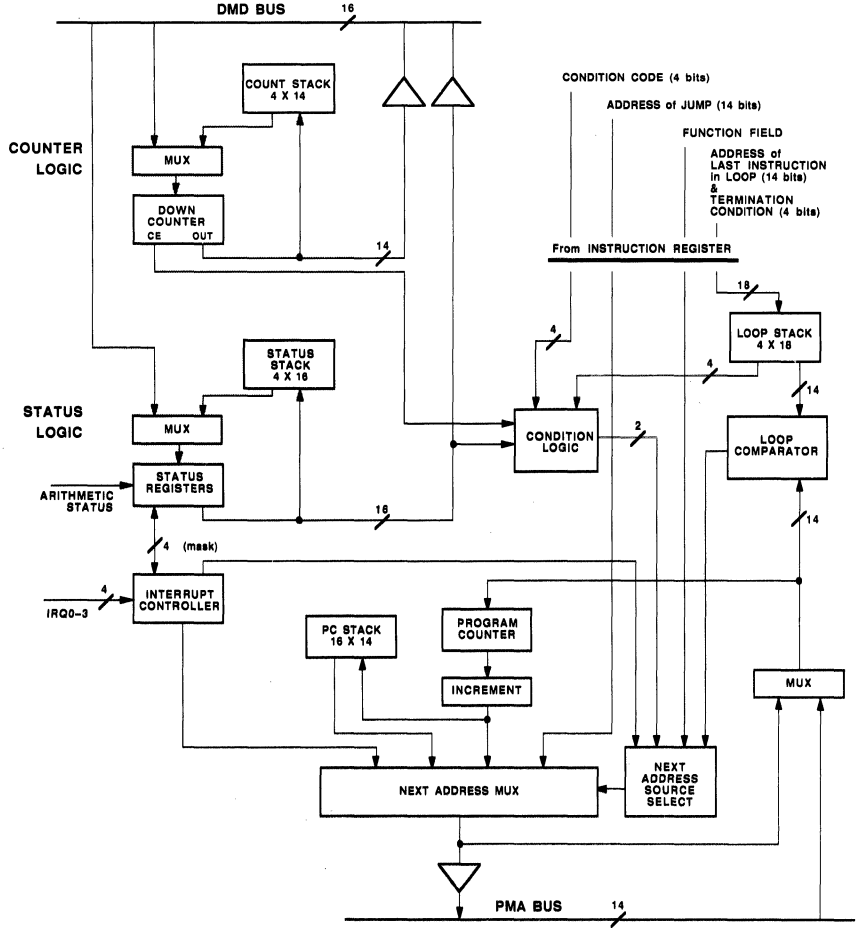


Figure 6. Program Sequencer

auto-decrement capability. It is loaded from the DMD bus with the loop count. The count is decremented every time the counter value is checked; when the count expires, the counter expired (CE) flag is set. The count stack allows the nesting of loops by storing temporarily dormant loop counts. When a new value is loaded into the counter from the DMD bus, the current counter value is automatically pushed onto the count stack as program flow enters a loop. The count stack is automatically popped whenever the CE flag is tested and is true, thereby resuming execution of the code outside the loop.

The DO UNTIL instruction executes a zero-overhead loop using the loop stack and the loop comparator. For a DO UNTIL instruction, a 14-bit termination address and a 4-bit termination condition are pushed onto the 18-bit loop stack. The address of the next instruction (which identifies the top of the loop) is pushed onto the PC stack. The loop comparator continuously compares the current PC value against the termination address on the top of the loop stack. When the termination address is detected, the processor checks if the termination condition is met. If the termination condition is not met, then the top of the PC stack is used as the next PC address, returning program flow to the beginning of the loop. If the termination condition is met, then the PC stack is popped, the current PC is incremented by one, and program flow falls out of the loop. The loop stack is four levels deep, permitting four levels of zero-overhead loop nesting.

Instruction Cache Memory

The instruction cache memory is 16 levels deep and one instruction (24 bits) wide. The cache memory maintains a short history of previously executed instructions so they can be fetched internally if they are needed again.

Every time an instruction is fetched from external memory, it is also written into the cache memory. When the program enters a loop which fits within the cache, all the instructions in the loop are stored in cache during the first pass. On subsequent passes, the instructions can be fetched from the instruction cache when a program memory data access is required. This allows the program memory to be used for data access without penalty. The ADSP-2100 then becomes, in effect, a three-bus system with two data buses and one program bus. For the multiply/accumulate operations typical of digital signal processing algorithms, this gives significant speed advantages.

Instructions are fetched from cache memory *only* when a program memory data fetch is required. The cache monitor circuit automatically keeps track of when the next instruction is contained in the cache. No maintenance or overhead is needed to store externally fetched instructions in the cache or to read previously fetched instructions from cache.

PMD-DMD Bus Exchange

The PMD-DMD bus exchange circuit couples the PMD and DMD buses. The PMD bus is 24 bits wide and the DMD bus is 16 bits wide. The upper 16 bits of PMD are connected to the DMD bus. An 8-bit register (PX) allows transfer of the full width of the PMD bus. When data is read from the PMD bus, the lower 8 bits of the PMD bus are loaded into PX. When writing to the PMD bus, the contents of PX are appended to the upper 16 bits, forming a 24-bit value. The PX register is readable and loadable from the DMD bus.

STATUS REGISTERS

The ADSP-2100 maintains five status registers, each of which can be read over the DMD bus and four of which can be written. These registers are:

ASTAT	Arithmetic Status register
SSTAT	Stack Status register (read-only)
MSTAT	Mode Status register
ICNTL	Interrupt Control register
IMASK	Interrupt Mask register

ASTAT

ASTAT is 8 bits wide and holds the status information generated by the computational sections of the processor. The bits in ASTAT are defined as follows:

0	AZ	(ALU result zero)
1	AN	(ALU result negative)
2	AV	(ALU overflow)
3	AC	(ALU carry)
4	AS	(ALU X input sign)
5	AQ	(ALU quotient flag)
6	MV	(MAC overflow)
7	SS	(Shifter input sign)

The bits which express a particular condition (AZ, AN, AV, AC, MV) are all positive sense (1 = true, 0 = false). Each of the bits are automatically updated whenever a new status is generated by an arithmetic operation. As such, each bit is affected only by a certain subset of arithmetic operations, as defined by the following table:

Status Bit	Updated on:
AZ, AN, AV, AC	Any ALU operation except division
AS	ALU absolute value operation
AQ	ALU divide operations
MV	Any MAC operation except saturate MR
SS	Shifter exponent detect operation

SSTAT

SSTAT is 8 bits wide and holds the status of the four internal stacks. The bits in SSTAT are:

0	PC Stack Empty
1	PC Stack Overflow
2	Count Stack Empty
3	Count Stack Overflow
4	Status Stack Empty
5	Status Stack Overflow
6	Loop Stack Empty
7	Loop Stack Overflow

All of the bits are positive sense (1 = true, 0 = false). The *empty* status bits indicate that the number of pop operations for the stack is greater than or equal to the number of push operations (if no stack overflow has occurred) since the last reset. The *overflow* status bits indicate that the number of push operations for the stack has exceeded the number of pop operations by an amount that is greater than the depth of the stack. When this occurs, the item(s) most recently pushed will be missing from the stack (old data is considered more important than new). The stack overflow status bits "stick" once they are set, so that subsequent pop operations have no effect on them. A processor reset must be executed to clear the stack overflow status.

MSTAT
MSTAT is a 4-bit register that defines various operating modes of the processor. The Mode Control instruction enables or disables the four operating modes. The bits in MSTAT are:

0 Data Register Bank Select
1 Bit Reverse Mode (DAG1 only)
2 ALU Overflow Latch Mode
3 AR Saturation Mode

The data register bank select bit determines which set of data registers is currently active (0 = primary, 1 = secondary). The data registers include all of the result and input registers to the ALU, MAC, and Shifter (AX0, AX1, AY0, AY1, AF, AR, MX0, MX1, MY0, MY1, MF, MR0, MR1, MR2, SB, SE, SI, SR0 and SR1). At initialization, the data register bank select bit is cleared.

The bit reverse mode, when enabled, bit-wise reverses all addresses generated by DAG1. This is most useful for reordering the input or output data in a radix-2 FFT algorithm.

The ALU overflow latch mode causes the AV (ALU overflow) status bit to "stick" once it is set. In this mode, when an ALU overflow occurs, AV will be set and remain set, even if subsequent ALU operations do not generate overflows. AV can then only be cleared by writing a zero into it from the DMD bus.

The AR saturation mode, when set, causes ALU results to be saturated to the maximum positive (H#7FFF) or negative (H#8000) values when an ALU overflow occurs.

IMASK
IMASK is four bits wide and allows the four interrupt inputs to be individually enabled or disabled. The bits in IMASK are:

0 $\overline{\text{IRQ0}}$ Enable
1 $\overline{\text{IRQ1}}$ Enable
2 $\overline{\text{IRQ2}}$ Enable
3 $\overline{\text{IRQ3}}$ Enable

The bits are all positive sense (0 = disabled, 1 = enabled). IMASK is set to zero upon a processor reset so that all interrupts are disabled initially.

ICNTL
ICNTL is a 5-bit register configuring the interrupt modes of the processor. The bits in ICNTL are:

0 $\overline{\text{IRQ0}}$ Sensitivity
1 $\overline{\text{IRQ1}}$ Sensitivity
2 $\overline{\text{IRQ2}}$ Sensitivity
3 $\overline{\text{IRQ3}}$ Sensitivity
4 Interrupt Nesting Mode

The IRQ sensitivity bits determine whether a given interrupt input is edge- or level-sensitive (0 = level-sensitive, 1 = edge-sensitive). These bits are all undefined after a processor reset.

The interrupt nesting mode determines whether nesting of interrupt service routines is allowed. When set to zero, all interrupt levels will be masked automatically when an interrupt service routine is entered. When set to one, IMASK will be set so that only equal and lower priority interrupts will be masked, permitting higher priority interrupts to interrupt the current interrupt service routine. This bit is undefined after a processor reset.

CONDITION CODES
The condition codes are used to determine whether a conditional instruction, such as a jump, trap, call, return, MAC saturation or arithmetic operation, is performed. The sixteen composite status conditions and their derivations are given in Table I. Since arithmetic status is latched into ASTAT at the end of a processor cycle, the condition logic outputs represent conditions generated on a previous cycle.

Code	Status Condition	True If:
EQ	ALU Equal Zero	AZ = 1
NE	ALU Not Equal Zero	AZ = 0
LT	ALU Less Than Zero	AN .XOR. AV = 1
GE	ALU Greater Than or Equal Zero	AN .XOR. AV = 0
LE	ALU Less Than or Equal Zero	(AN .XOR. AV) .OR. AZ = 1
GT	ALU Greater Than Zero	(AN .XOR. AV) .OR. AZ = 0
AC	ALU Carry	AC = 1
NOT AC	Not ALU Carry	AC = 0
AV	ALU Overflow	AV = 1
NOT AV	Not ALU Overflow	AV = 0
MV	MAC Overflow	MV = 1
NOT MV	Not MAC Overflow	MV = 0
NEG	ALU X Input Sign Negative	AS = 1
POS	ALU X Input Sign Positive	AS = 0
NOT CE	Not Counter Expired	CE ≠ 0
TRUE	True	Always True

Table I. Condition Codes

SYSTEM INTERFACE

Figure 7 shows a basic system configuration with the ADSP-2100.

Clock Signals

The ADSP-2100 takes a TTL-compatible clock signal, CLKIN, running at four times the basic processor cycle time as an input. Using this clock input, the processor divides the internal processor cycle into eight states, defined by the edges of the input clock. The active processor cycle consists of states 1 through 7. State 8 is a dead zone to provide a neutral stopping point for halting the processor.

A clock output (CLKOUT) signal is generated by the processor to synchronize external devices to the processor's internal cycles. CLKOUT is high during states 8, 1, 2 and 3, and low during states 4, 5, 6 and 7. Its frequency is one-fourth of that of CLKIN. Except during $\overline{\text{RESET}}$, the CLKOUT signal runs continuously.

Bus Interface

The ADSP-2100 can relinquish control of the memory buses to an external device. When the external device requires access to memory, it asserts the Bus Request ($\overline{\text{BR}}$) signal. After completing the current instruction, the processor halts program execution, tristates the PMA, PMD, $\overline{\text{PMS}}$, $\overline{\text{PMRD}}$, $\overline{\text{PMWR}}$ and PMDA output drivers and the DMA, DMD, $\overline{\text{DMS}}$, $\overline{\text{DMRD}}$ and $\overline{\text{DMWR}}$ output drivers, and asserts the Bus Grant ($\overline{\text{BG}}$) signal. When the $\overline{\text{BR}}$ signal is released, the processor re-enables the output drivers, releases the $\overline{\text{BG}}$ signal, and continues program execution from the point where it stopped.

Program Memory Interface

The Program Memory Interface supports two buses: the program memory address bus (PMA) and the program memory data bus (PMD). The 14-bit PMA bus directly addresses up to 16K words. The PMD bus is bidirectional and 24 bits wide.

Since program memory can be used for both instruction code and data storage, the Program Memory Data Access (PMDA) signal is asserted whenever data, as opposed to an instruction code, is fetched. There is no placement restriction for instruction code and data in program memory area if less than 16K words are used. Since the timing of PMDA is compatible with that of the PMA lines, it may be used as a 15th address line if desired. This effectively doubles the program memory area to 32K, which must be split into 16K dedicated to instruction codes and 16K to data.

The program memory data lines are bidirectional. The Program Memory Select ($\overline{\text{PMS}}$) signal indicates access to the Program Memory and can be used as a chip select signal. The Program Memory Write ($\overline{\text{PMWR}}$) signal indicates a write operation and can be used as a write strobe. The Program Memory Read ($\overline{\text{PMRD}}$) signal indicates a read operation and can be used as a read strobe or output enable signal.

Although the processor internal data bus is only 16 bits, the ADSP-2100 can write to the full 24-bit program memory using the PX register.

Data Memory Interface

The Data Memory Interface supports two buses: the Data Memory Address bus (DMA) and the Data Memory Data bus (DMD). The 14-bit DMA bus directly addresses up to 16K words of data. The DMD bus is bidirectional and 16 bits wide. The Data Memory Select ($\overline{\text{DMS}}$) signal indicates access to the Data Memory and can be used as a chip select signal. The Data Memory Write ($\overline{\text{DMWR}}$) signal indicates a write operation and can be used as a write strobe. The Data Memory Read ($\overline{\text{DMRD}}$) signal indicates a read operation and can be used as a read strobe or output enable signal.

The ADSP-2100 supports memory-mapped I/O, with the peripherals memory mapped into the data memory address space and accessed by the processor in the same manner as data memory.

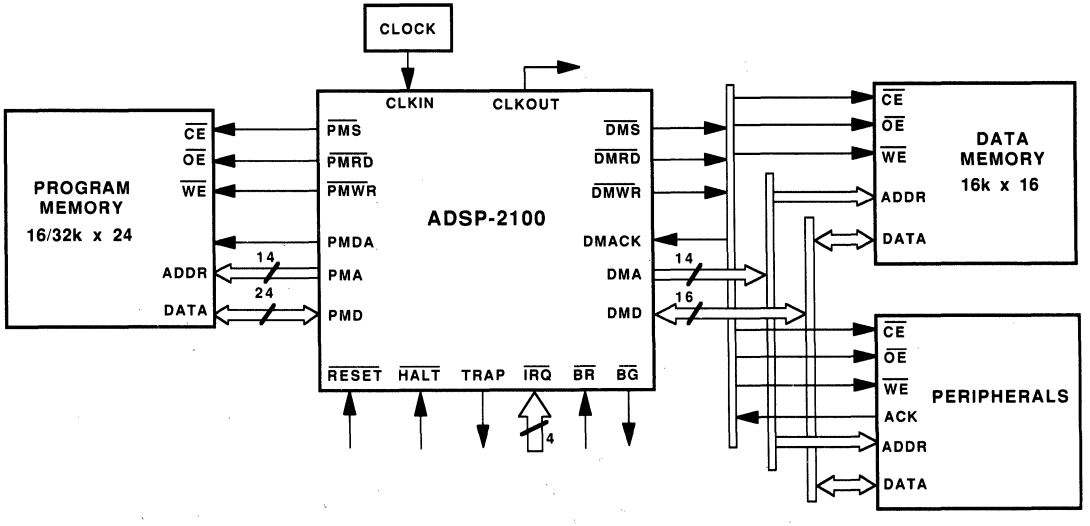


Figure 7. Basic System Configuration

To allow interfacing to slower peripherals, the data memory acknowledge (DMACK) signal is provided. The ADSP-2100 checks the status of the DMACK signal at the end of each processor cycle. If the DMACK signal is not asserted, the processor extends the current cycle by another full cycle. This extension occurs as many times as necessary until the DMACK signal is asserted and the access is completed.

Interrupt Handling

The ADSP-2100 provides four direct interrupt input pins, $\overline{IRQ}_0$ to $\overline{IRQ}_3$. Each interrupt pin corresponds to a particular interrupt priority level from 3 (highest) to 0 (lowest). The four interrupt levels are internally prioritized and individually maskable. These input pins can be programmed to be either level- or edge-sensitive.

The ADSP-2100 supports a vectored interrupt scheme: when an external interrupt is acknowledged, the processor switches program control to the interrupt vector address corresponding to the interrupt level (program memory locations 0000 to 0003). Interrupts can optionally be nested so that a higher priority interrupt can preempt the currently executing interrupt service routine.

Processor Control Interface

The processor control interface provides external control over the activity of the processor. The control signals are $\overline{RESET}$, $\overline{HALT}$ and TRAP.

The $\overline{RESET}$ signal initiates a master reset of the ADSP-2100. The $\overline{RESET}$ signal must be asserted after the chip is powered up to assure proper initialization. The master reset performs the following:

1 Initialize internal clock circuitry
2 Reset all internal stack pointers
3 Clear the cache memory monitor
4 If there is no pending bus request, PMA is driven with 0004
5 Mask all interrupts
6 Clear MSTAT register.

The $\overline{HALT}$ signal is used to suspend program execution temporarily. When $\overline{HALT}$ is asserted, the processor stops at the end of the current instruction. To ensure that the processor always halts after completion of an instruction fetch, an external fetch of the next instruction is forced even if the instruction is available from internal cache memory. Since the processor always stops after an external instruction fetch cycle, the controlling device is able to observe the instruction address where the program was stopped. The halt condition can be sustained for any length of time, during which all signals generated by the processor will remain static (maintaining the output at state 8). The processor will continue normal execution when the $\overline{HALT}$ line is released.

The TRAP signal is generated by the processor whenever a TRAP instruction is executed. Assertion of the TRAP signal indicates that the processor has stopped instruction execution just after the end of the cycle which executed the TRAP instruction. The TRAP state is identical to the $\overline{HALT}$ state, with the processor output frozen in state 8. In this case, the processor PMA bus contains the address of the instruction following the TRAP instruction. The TRAP signal remains asserted until the $\overline{HALT}$ signal is asserted externally. When the $\overline{HALT}$ signal assertion is sensed, the processor releases the TRAP signal. However, the processor remains in the halt condition until the $\overline{HALT}$ line is released.

Multiprocessor Synchronization

Even when multiple ADSP-2100s are driven from the same CLKIN signal, there is a phase ambiguity between the various processors. This ambiguity can be prevented by using a single master $\overline{RESET}$ signal synchronized to CLKIN. When the master $\overline{RESET}$ is released, all the processors begin state 5 on the same edge of CLKIN. Once initialized in this manner, the cycle states of the processors remain synchronized with each other.

INSTRUCTION SET DESCRIPTION

The ADSP-2100 assembly language uses an algebraic syntax for ease of coding and readability. The sources and destinations of computations and data movements are written explicitly in each assembly statement, eliminating cryptic assembler mnemonics. Nevertheless, every instruction assembles into a single 24-bit word and executes in a single cycle. The instructions encompass a wide variety of instruction types along with a high degree of operational parallelism. There are five basic categories of instructions: data move instructions, computational instructions, multifunction instructions, program flow control instructions and miscellaneous instructions. Each of these instruction types is described briefly. The complete instruction set is summarized in Table IV at the end of this section.

Data Move Instructions

Table II gives a list of all registers that are accessible using the data move instructions. (Only the program counter (PC), the instruction register, the arithmetic feedback register (AF) and the multiplier feedback register (MF) are not on this list.) This set of registers is denoted as *reg* in the instruction set summary given in Table IV. A subset of the *reg* group associated with the computational units, which generally hold data as opposed to address or status information, is denoted as *dreg*.

The data move instructions include transfers between internal registers, between data memories and internal registers, between program memories and internal registers, and immediate value loading of registers and data memories. The content of every *reg*

AX0, AX1 AY0, AY1 AR MX0, MX1 MY0, MY1 MR0, MR1, MR2 SI SE SR0, SR1	Data Registers (dreg)
SB PX I0, I1, I2, I3, I4, I5, I6, I7 M0, M1, M2, M3, M4, M5, M6, M7 L0, L1, L2, L3, L4, L5, L6, L7 CNTR ASTAT MSTAT SSTAT IMASK ICNTL	Accessible Registers (reg)

Table II. Register Classification

can also be loaded to any other *reg*. Every *reg* can be loaded with an immediate value which is the full width of the particular register being loaded.

Two addressing modes are supported for data memory transfers: direct addressing and indirect addressing. In direct addressing, the memory address is supplied from the instruction word. In indirect addressing, one of the data address generators provides the address. Using direct addressing, the content of a data memory location can be written and read by any *reg*. Using indirect addressing, the content of a data memory location can only be written and read by a *dreg*. Immediate data load to data memory is permitted with indirect addressing. Only the indirect addressing mode is supported for program memory data transfers, and contents of a program memory location can be read and written to any *dreg*.

Computational Instructions
There are three types of operations associated with the computational units: ALU operations, MAC operations and shifter operations. With few exceptions, all these computational instructions can be made conditional. (The permissible conditions are specified in Table I.) Each computational unit has a set of input registers and output registers. A list of permissible input operands and result registers for each of the units is given in Table III.

Multifunction Instructions
Multifunction instructions execute one computational operation with one or two data moves. All of the multifunction instructions utilize various combinations of the computational and data move operations described above. Since the instruction word is only 24 bits wide, only certain combinations are valid. In general, the following rules are followed.

1 Only one unconditional computational operation can be specified
2 Any memory transfer must use the indirect addressing mode
3 Data move operations can only involve data registers (*dregs*)
4 Only an ALU or a MAC operation can be specified with two operand fetches, one from program memory and one from data memory.

Program Flow Control Instructions
Program flow control instructions include JUMP, CALL, return from subroutine, return from interrupt, DO UNTIL and TRAP. All of these instructions can be made conditional. The JUMP and CALL instructions support both direct addressing, with the destination address specified by the instruction word, and indirect addressing, with the destination address specified by one of the I registers in DAG2.

Miscellaneous Instructions
Miscellaneous instructions include indirect register modify, stack control, mode control and NOP operations.

ALU

Source for X input port (xop)	Source for Y input port (yop)	Destination for output port R
AX0, AX1 AR MR0, MR1, MR2 SR0, SR1	AY0, AY1 AF	AR AF

MAC

Source for X input port (xop)	Source for Y input port (yop)	Destination for output port R
MX0, MX1 AR MR0, MR1, MR2 SR0, SR1	MY0, MY1 MF	MR (MR2, MR1, MR0) MF

Shifter

Source for Shifter input (xop)		Destination for Shifter output
SI AR MR0, MR1, MR2 SR0, SR1		SR (SR1, SR0)

Table III. Computational Input/Output Registers

These conventions are used in Table IV:

1. All keywords are shown in capital letters.
2. Brackets enclose optional parts of the syntax.
3. Vertical lines indicate that one parameter must be chosen from those enclosed.
4. Table I defines the conditions for *condition*.
5. Table II defines the set of registers for *dreg* and *reg*.
6. Table III defines the set of registers for *xop* and *yop*.
7. <data> represents an immediate value.
8. <address> may be an immediate value or label.
9. <comp>, in a multifunction instruction, represents all legal ALU, MAC or Shifter operations with these restrictions:
   - All operations are performed unconditionally
   - Shift Immediate operations are not allowed
   - ALU division (DIVS, DIVQ) is not allowed

DATA MOVE INSTRUCTIONS

Register Move

reg = reg;

Load Register Immediate

reg = <data>;

Data Memory Read (direct address)

reg = DM(<address>);

Data Memory Read (indirect address)

dreg = DM(I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 , M0 | M1 | M2 | M3 | M4 | M5 | M6 | M7);

Program Memory Read (indirect address)

dreg = PM(I4 | I5 | I6 | I7 , M4 | M5 | M6 | M7);

Data Memory Write (direct address)

DM(<address>) = reg;

Data Memory Write (indirect address)

DM(I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 , M0 | M1 | M2 | M3 | M4 | M5 | M6 | M7) = dreg | <data>;

Program Memory Write (indirect address)

PM(I4 | I5 | I6 | I7 , M4 | M5 | M6 | M7) = dreg;

COMPUTATIONAL INSTRUCTIONS: ALU

Add/Add with Carry

[IF condition] AR | AF = xop +yop | +C | +yop+C ;

Subtract X-Y/Subtract X-Y with Borrow

[IF condition] AR | AF = xop −yop | −yop+C−1 ;

Subtract Y-X/Subtract Y-X with Borrow

[IF condition] AR | AF = yop −xop | −xop+C−1 ;

AND, OR, Exclusive OR

[IF condition] AR | AF = xop AND | OR | XOR yop ;

Pass/Clear

[IF condition] AR | AF = PASS xop | yop ;

Negate

[IF condition] AR | AF = − xop | yop ;

NOT

[IF condition] AR | AF = NOT xop | yop ;

Absolute Value

[IF condition] AR | AF = ABS xop | yop ;

Increment

[IF condition] AR | AF = yop +1 ;

Decrement

[IF condition] AR | AF = yop −1 ;

Divide

DIVS yop, xop ;
DIVQ xop ;

COMPUTATIONAL INSTRUCTIONS: SHIFTER

Arithmetic Shift

[IF condition] SR = [SR OR] ASHIFT xop (HI) | (LO) ;

Logical Shift

[IF condition] SR = [SR OR] LSHIFT xop (HI) | (LO) ;

Normalize

[IF condition] SR = [SR OR] NORM xop (HI) | (LO) ;

Derive Exponent

[IF condition] SE = EXP xop (HI) | (LO) | (HIX) ;

Block Exponent Adjust

[IF condition] SB = EXPADJ xop ;

Arithmetic Shift Immediate
SR = [SR OR] ASHIFT xop BY <data> [(HI) | (LO)] ;

Logical Shift Immediate
SR = [SR OR] LSHIFT xop BY <data> [(HI) | (LO)] ;

COMPUTATIONAL INSTRUCTIONS: MAC

Multiply
[IF condition] [MR | MF] = xop*yop ([SS | SU | US | UU | RND]) ;

Multiply Accumulate
[IF condition] [MR | MF] = MR + xop*yop ([SS | SU | US | UU | RND]) ;

Multiply Subtract
[IF condition] [MR | MF] = MR − xop*yop ([SS | SU | US | UU | RND]) ;

Clear
[IF condition] [MR | MF] = 0 ;

Transfer MR
[IF condition] [MR | MF] = MR [(RND)] ;

Conditional MR Saturation
IF MV SAT MR ;

PROGRAM FLOW CONTROL INSTRUCTIONS

Jump
[IF condition] JUMP ([I4 | I5 | I6 | I7 | <address>]) ;

Call
[IF condition] CALL ([I4 | I5 | I6 | I7 | <address>]) ;

Return from Subroutine
[IF condition] RTS ;

Return from Interrupt
[IF condition] RTI ;

Do Until
DO <address> [UNTIL condition] ;

Trap
[IF condition] TRAP ;

MULTIFUNCTION INSTRUCTIONS

Computation with Memory Read
<comp> , dreg = [DM([I0 | I1 | I2 | I3] , [M0 | M1 | M2 | M3]) | PM ([I4 | I5 | I6 | I7] , [M4 | M5 | M6 | M7])] ;

Computation with Data Register Move
<comp> , dreg = dreg ;

Computation with Memory Write
[DM([I0 | I1 | I2 | I3] , [M0 | M1 | M2 | M3]) | PM ([I4 | I5 | I6 | I7] , [M4 | M5 | M6 | M7])] = dreg, <comp> ;

Data & Program Memory Read

AX0 AX1 MX0 MX1	= DM (	I0 I1 I2 I3	,	M0 M1 M2 M3	) ,	AY0 AY1 MY0 MY1	= PM (	I4 I5 I6 I7	,	M4 M5 M6 M7	) ;			

ALU/MAC Operation with Data & Program Memory Read*

<ALU> <MAC>	,	AX0 AX1 MX0 MX1	= DM (	I0 I1 I2 I3	,	M0 M1 M2 M3	) ,	AY0 AY1 MY0 MY1	= PM (	I4 I5 I6 I7	,	M4 M5 M6 M7	) ;	

*ALU Division operations not allowed.

MISCELLANEOUS INSTRUCTIONS

Stack Control

[PUSH] STS] [,POP CNTR] [,POP PC] [,POP LOOP] ;
[POP]

Mode Control

[ENA] BIT_REV][, ENA] AV_LATCH][, ENA] AR_SAT][, ENA] SEC_REG];
[DIS] [DIS] [DIS] [DIS]

Modify Address Register

MODIFY (I0 , M0) ;
I1 M1
I2 M2
I3 M3
I4 M4
I5 M5
I6 M6
I7 M7

No Operation

NOP ;

Table IV. Instruction Set Summary

SPECIFICATIONS

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-2100/ADSP-2100A J, K, AJ, AK Grades Min	J, K, AJ, AK Grades Max	S, AS, AT Grades Min	S, AS, AT Grades Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.50	5.50	V
T_{AMB}	Ambient Operating Temperature	0	+70	−55	+125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	ADSP-2100 J & K Grades Min	J & K Grades Max	S Grade Min	S Grade Max	Unit
V_{IH}	Hi-Level Input Voltage[1]	@V_{DD} = max	2.0		2.2		
V_{IL}	Lo-Level Input Voltage[1]	@V_{DD} = min		0.8		0.8	V
V_{OH}	Hi-Level Output Voltage[2]	@V_{DD} = min, I_{OH} = −1mA	2.4		2.4		V
V_{OL}	Lo-Level Output Voltage[2]	@V_{DD} = min, I_{OL} = 4mA		0.4		0.6	V
I_{IH}	Hi-Level Input Current[3]	@V_{DD} = max, V_{IN} = max		10		10	μA
I_{IL}	Lo-Level Input Current[3]	@V_{DD} = max, V_{IN} = 0V		10		10	μA
I_{OZH}	Tristate Leakage Current[4]	@V_{DD} = max, V_{IN} = max[7]		10		10	μA
I_{OZL}	Tristate Leakage Current[5]	@V_{DD} = max, V_{IN} = 0V[7]		10		10	μA
I_{OZL}	Tristate Pullup Current[6]	@V_{DD} = max, V_{IN} = 0V[7]		150		150	μA
I_{DD}	Supply Current (Power-Down)[9]	@V_{DD} = max, V_{IN} = 0V[6,7]		10		15	mA
I_{DD}	Supply Current (Dynamic)	@V_{DD} = max, max clock rate[8]		90		100	mA

Parameter		Test Conditions	ADSP-2100A AJ&AK Grades Min	AJ&AK Grades Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Unit
V_{IH}	Hi-Level Input Voltage[1]	@V_{DD} = max	2.0		2.2		2.2		V
V_{IH}	Hi-Level Input Voltage at CLKIN	@V_{DD} = max	2.2		2.4		2.4		V
V_{IL}	Lo-Level Input Voltage[1]	@V_{DD} = min		0.8		0.8		0.8	V
V_{IL}	Lo-Level Input Voltage at CLKIN	@V_{DD} = min		0.8		0.8		0.8	V
V_{OH}	Hi-Level Output Voltage[2]	@V_{DD} = min, I_{OH} = −1mA	2.4		2.4		2.4		V
V_{OL}	Lo-Level Output Voltage[2]	@V_{DD} = min, I_{OL} = 4mA		0.4		0.6		0.6	V
I_{IH}	Hi-Level Input Current[3]	@V_{DD} = max, V_{IN} = max		10		10		10	μA
I_{IL}	Lo-Level Input Current[3]	@V_{DD} = max, V_{IN} = 0V		10		10		10	μA
I_{OZH}	Tristate Leakage Current[4]	@V_{DD} = max, V_{IN} = max[7]		10		10		10	μA
I_{OZL}	Tristate Leakage Current[5]	@V_{DD} = max, V_{IN} = 0V[7]		10		10		10	μA
I_{OZL}	Tristate Pullup Current[6]	@V_{DD} = max, V_{IN} = 0V[7]		180		180		180	μA
I_{DD}	Supply Current (Power-Down)[9]	@V_{DD} = max, V_{IN} = 0V[6,7]		10		15		15	mA
I_{DD}	Supply Current (Dynamic)	@V_{DD} = max, max clock rate[8]		150		130		180	mA

NOTES

[1]Applies to pins: PMD_{0-23}, DMD_{0-15}, $\overline{BR}$, $\overline{IRQ}_{0-3}$, DMACK, $\overline{RESET}$, $\overline{HALT}$, (48 input pins for ADSP-2100A). Includes CLKIN for ADSP-2100 (49 input pins).

[2]Applies to pins: PMA_{0-13}, $\overline{PMS}$, PMD_{0-23}, $\overline{PMRD}$. $\overline{PMWR}$, PMDA, $\overline{BG}$, DMA_{0-13}, $\overline{DMS}$, DMD_{0-15}, $\overline{DMRD}$. $\overline{DMWR}$, TRAP, CLKOUT (78 output pins).

[3]Applies to pins: $\overline{BR}$, $\overline{IRQ}_{0-3}$, DMACK, $\overline{RESET}$, $\overline{HALT}$, CLKIN (9 input-only pins).

[4]Applies to pins: PMA_{0-13}, $\overline{PMS}$, PMD_{0-23} $\overline{PMRD}$. $\overline{PMWR}$, PMDA, DMA_{0-13}, $\overline{DMS}$, DMD_{0-15}, $\overline{DMRD}$. $\overline{DMWR}$(75 tristateable pins).

[5]Applies to pins: PMA_{0-13}, PMDA, DMA_{0-13} (29 tristateable pins w/o pullup).

[6]Applies to pins: PMD_{0-23}, $\overline{PMS}$, $\overline{PMRD}$. $\overline{PMWR}$, DMD_{0-15}, $\overline{DMS}$, $\overline{DMRD}$. $\overline{DMWR}$(46 tristateable pins w/pullup).

[7]Additional Test Conditions: V_{IN} = 0V on $\overline{BR}$ and $\overline{RESET}$, CLKIN active, forces tristate condition.

[8]Additional Test Conditions: Outputs loaded TTL loads w/100pF capacitance, V_{IH} = 2.4V, V_{IL} = 0.4V, clock rate = max.

[9]"Power-down" refers to an idle state. While the processor does not have any special standby or low-power mode, these conditions represent the lowest power consumption state.

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage	−0.3V to +7V
Input Voltage	−0.3V to V_{DD} +0.3V
Output Voltage Swing	−0.3V to V_{DD} +0.3V
Operating Temperature Range (Ambient)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10sec) PGA	+300°C
Lead Temperature (5sec) PQFP	+280°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ORDERING INFORMATION

Part Number	Speed (MHz)	Temperature Range	Package	Package Outline
ADSP-2100JG	6.144	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100KG	8.192	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100AJG	10.24	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100AKG	12.50	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100JP	6.144	0 to +70°C	100-PQFP	P-100
ADSP-2100KP	8.192	0 to +70°C	100-PQFP	P-100
ADSP-2100AJP	10.24	0 to +70°C	100-PQFP	P-100
ADSP-2100AKP	12.50	0 to +70°C	100-PQFP	P-100
ADSP-2100SG	6.144	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100ASG	8.192	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100ATG	10.24	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100SG/883B	6.144	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100ASG/883B	8.192	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100ATG/883B	10.24	−55°C to +125°C	100-Pin Grid Array	G-100A

ADSP-2100/ADSP-2100A Development Tools

Part Number	Description
ADDS-2110	Cross-Software and Simulator (VAX/VMS)
ADDS-2121	Cross-Software (IBM PC/DOS)
ADDS-2122	Simulator (IBM PC/DOS)
ADDS-2123-C	Cross-Software and Simulator (Sun 2/3, Unix BSD 4.2)
ADDS-2130	C Compiler, Cross-Software and Simulator (VAX/VMS)
ADDS-2131	C Compiler, Cross-Software and Simulator (IBM PC/DOS)
ADDS-2133-C	C Compiler, Cross-Software and Simulator (Sun 2/3, Unix BSD 4.2)
ADDS-2150A	ADSP-2100A 8MHz In-Circuit Emulator (110V)
ADDS-2150AE	ADSP-2100A 8MHz In-Circuit Emulator (220V)
ADDS-2151A	ADSP-2100A 8MHz In-Circuit Emulator (110V) with Trace Board
ADDS-2151AE	ADSP-2100A 8MHz In-Circuit Emulator (220V) with Trace Board
ADDS-2161	Trace Board Option for ADDS-2150 or ADDS-2150E
ADDS-2160	ADSP-2100A 8MHz Evaluation Board
ADDS-2169	University Package (ADDS-2131 and ADDS-2160)
ADDS-2190	Three Day ADSP-2100 Workshop

ESD SENSITIVITY

The ADSP-2100 and ADSP-2100A feature proprietary input protection circuitry. Per Method 3015 of MIL-STD-883, the ADSP-2100 has been classified as a Class 1 device and the ADSP-2100A as a Class 2 device.

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*.

SWITCHING CHARACTERISTICS

GENERAL NOTES

Use the exact timing information given. Do not attempt to derive parameters from the addition or subtraction of others. While this addition or subtraction would yield meaningful results for an individual part, the values given in this data sheet reflect statistical variations and worst cases. Consequently, you cannot meaningfully add up parameters to derive or "verify" longer times.

TIMING NOTES

Switching characteristics specify how the processor is switching its signals. The user has no control over this operation. It is dependent on the internal design. Timing requirements specify the timing of signals that the user has control over such as the placement of data on the DMD bus as input for a read operation.

Timing requirements are used by a designer to guarantee that the processor operates correctly with another device while switching characteristics inform the designer what the device is doing under any given circumstance. Switching characteristics are also referenced to ensure that any timing requirement of a device connected to the processors (such as a memory) is satisfied.

SPECIFICATIONS

In this edition of the data sheet a number of specifications have been removed. The old parameter numbering has been retained for continuity. The specifications in this data sheet are the only ones required to design with the ADSP-2100.

MEMORY REQUIREMENTS

This chart links common memory device specification names and ADSP-2100/ADSP-2100A timing parameters for your convenience.

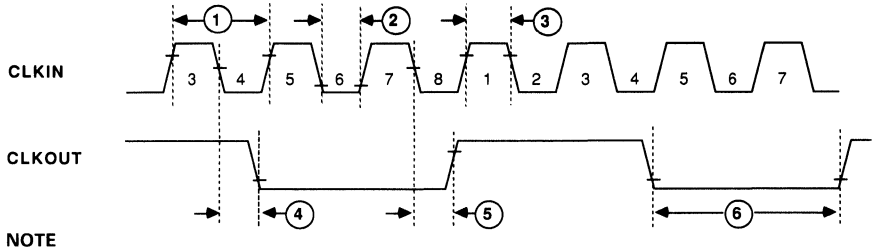
Parameter Number	Parameter Name	Common Memory Device Specification Name
41	PMA Valid to $\overline{\text{PMWR}}$ Low	Address Set Up Time
79	DMA Valid to $\overline{\text{DMWR}}$ Low	Address Set Up Time
42	$\overline{\text{PMWR}}$ High to PMA Invalid	Address Hold Time
80	$\overline{\text{DMWR}}$ High to DMA Invalid	Address Hold Time
55	PMD Out Valid to $\overline{\text{PMWR}}$ High	Data Set Up Time
91	DMD Out Valid to $\overline{\text{DMWR}}$ High	Data Set Up Time
54	$\overline{\text{PMWR}}$ High to PMD Out Invalid	Data Hold Time
90	$\overline{\text{DMWR}}$ High to DMD Out Invalid	Data Hold Time
58	$\overline{\text{PMRD}}$ Low to PMD Input Valid	$\overline{\text{OE}}$ to Data Valid
94	$\overline{\text{DMRD}}$ Low to DMD Input Valid	$\overline{\text{OE}}$ to Data Valid
59	PMA Valid to PMD Input Valid	Address Access Time
95	DMA Valid to DMD Input Valid	Address Access Time

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

	ADSP-2100 Clock Signals	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Timing Requirements										
1	CLKIN Period[1]	A	40.5		30.5		40.5		ns	
2	CLKIN Width Low	A	11		8		11		ns	
3	CLKIN Width High	A	18		12		18		ns	
Switching Characteristics										
4	CLKIN Low (3-4) to CLKOUT Low	B	13	34	13	29	11	34	ns	
5	CLKIN Low (7-8) to CLKOUT High	B	6	24	6	20	5	24	ns	
6	CLKOUT Width Low	A	60		45		60		ns	4

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100A Clock Signals	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Timing Requirements											
1 CLKIN Period[1]	A	24.4		20		30.5		24.4		ns	
2 CLKIN Width Low	A	7		4		8		7		ns	
3 CLKIN Width High	A	9		8		12		9		ns	
Switching Characteristics											
4 CLKIN Low (3-4) to CLKOUT Low	B		24		22		29		24	ns	
5 CLKIN Low (7-8) to CLKOUT High	B		20		18		20		20	ns	
6 CLKOUT Width Low	A	36		28		45		36		ns	4



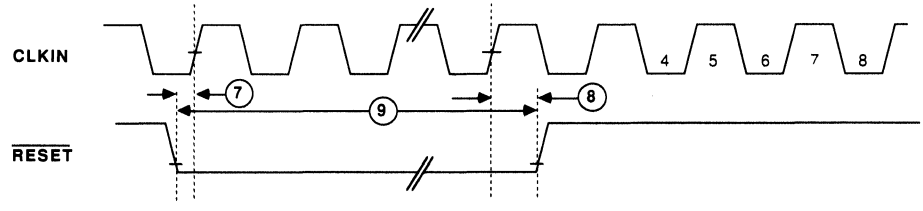
NOTE
The Processor Cycle is Divided into 8 Internal States Determined by the Rising and Falling Edges of CLKIN. CLKOUT is Synchronized to the Processor States as Shown Above.

Figure 8. Clock Signals

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Control Signals	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Timing Requirements									
7 $\overline{\text{RESET}}$ Low to CLKIN High	B	2		2		2		ns	
8 CLKIN High to $\overline{\text{RESET}}$ High	B	6	36	4	26	6	36	ns	2 (max only)
9 $\overline{\text{RESET}}$ Width Low	A	162		122		170		ns	8

ADSP-2100A Control Signals	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Timing Requirements											
7 $\overline{\text{RESET}}$ Low to CLKIN High	B	2		2		2		2		ns	
8 CLKIN High to $\overline{\text{RESET}}$ High	B	4	20	4	16	6	26	4	20	ns	2 (max only)
9 $\overline{\text{RESET}}$ Width Low	A	98		80		128		98		ns	8



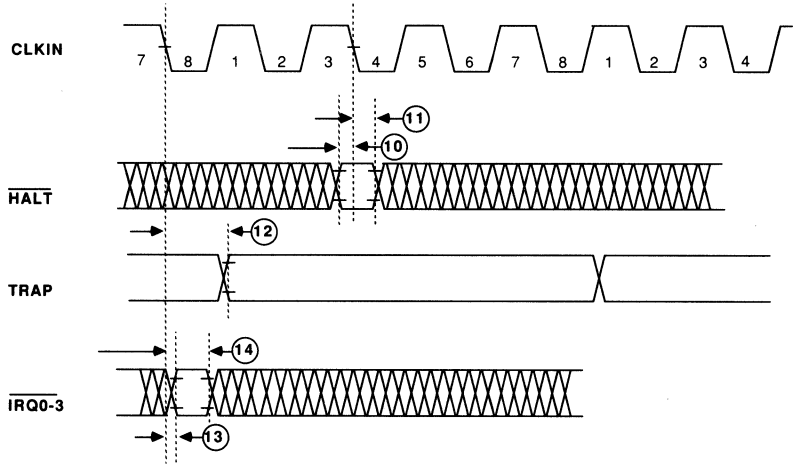
NOTE
The Reset signal determines the phase of the processor cycle.
The processor starts from state 4 after the release of the Reset signal.

Figure 9. $\overline{\text{RESET}}$ Signal

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Control Signals	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Timing Requirements									
10 $\overline{\text{HALT}}$ Valid to CLKIN Low (3-4)	B	0		0		0		ns	
11 CLKIN Low (3-4) to $\overline{\text{HALT}}$ Invalid	B	12		10		12		ns	
Switching Characteristics									
12 CLKIN Low (7-8) to TRAP Valid	B		25		20		25	ns	
Interrupts									
Timing Requirements									
13 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Valid	B		2		2		1	ns	
14 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Invalid	B	21		17		21		ns	

ADSP-2100A Control Signals	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Timing Requirements											
10 $\overline{\text{HALT}}$ Valid to CLKIN Low (3-4)	B	2		2		2		2		ns	
11 CLKIN Low (3-4) to $\overline{\text{HALT}}$ Invalid	B	10		8		10		10		ns	
Switching Characteristics											
12 CLKIN Low (7-8) to TRAP Valid	B		18		16		20		18	ns	
Interrupts											
Timing Requirements											
13 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Valid	B		1		1		1		1	ns	
14 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Invalid	B	14		14		17		14		ns	



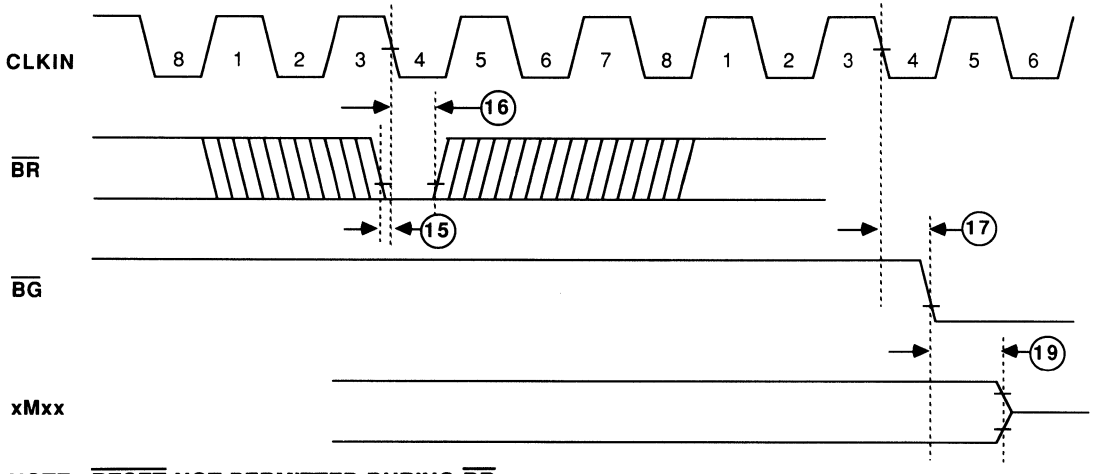
NOTE
The Control Signals are Shown in Relationship to the Processor States in Which They are Recognized or Asserted as Defined by CLKIN. There is No Implied Relationship between HALT, TRAP, and IRQ_{0-3}.

Figure 10. Control Signals

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Bus Request Asserted	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Timing Requirements									
15 $\overline{BR}$ Valid to CLKIN Low (3-4)	B	1		1		1		ns	
16 CLKIN Low (3-4) to $\overline{BR}$ Invalid	B	10		7		10		ns	
Switching Characteristics									
17 CLKIN Low (3-4) to $\overline{BG}$ Low	B		38		30		38	ns	
19 $\overline{BG}$ Low to xMxx Disable[2]	D		22		17		22	ns	

ADSP-2100A Bus Request Asserted	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Timing Requirements											
15 $\overline{BR}$ Valid to CLKIN Low (3-4)	B	4		4		1		4		ns	
16 CLKIN Low (3-4) to $\overline{BR}$ Invalid	B	4		4		7		4		ns	
Switching Characteristics											
17 CLKIN Low (3-4) to $\overline{BG}$ Low	B		26		24		30		26	ns	
19 $\overline{BG}$ Low to xMxx Disable[2]	D		16		16		17		16	ns	



NOTE: $\overline{RESET}$ NOT PERMITTED DURING $\overline{BR}$.

Figure 11. Bus Request Asserted

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Bus Request Negated	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Timing Requirements									
15 $\overline{BR}$ Valid to CLKIN Low (3-4)	B	1		1		1		ns	
16 CLKIN Low (3-4) to $\overline{BR}$ Invalid	B	10		7		10		ns	
Switching Characteristics									
18 CLKIN Low (7-8) to $\overline{BG}$ High	B		31		25		31	ns	
20 xMxx Enable to $\overline{BG}$ High[2]	F		12		10		12	ns	

ADSP-2100A Bus Request Negated	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Timing Requirements											
15 $\overline{BR}$ Valid to CLKIN Low (3-4)	B	4		4		1		4		ns	
16 CLKIN Low (3-4) to $\overline{BR}$ Invalid	B	4		4		7		4		ns	
Switching Characteristics											
18 CLKIN Low (7-8) to $\overline{BG}$ High	B		24		20		25		24	ns	
20 xMxx Enable to $\overline{BG}$ High[2]	F		10		8		10		10	ns	

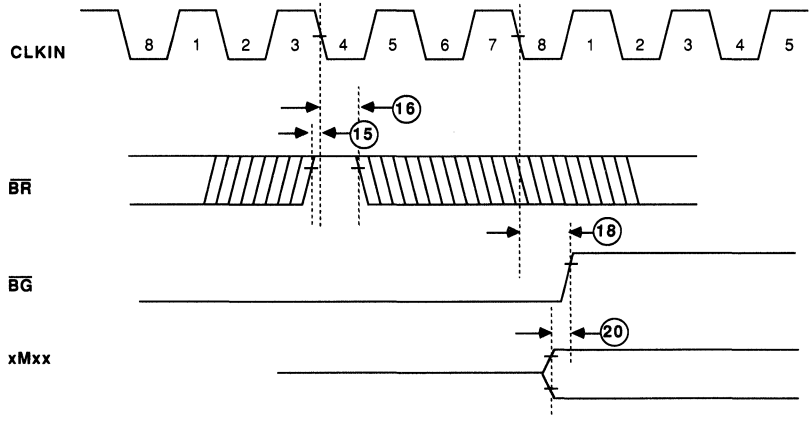
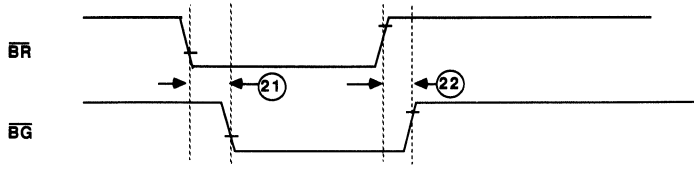


Figure 12. Bus Request Negated

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Bus Request/Grant with $\overline{\text{RESET}}$ Low	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Switching Characteristics									
21 $\overline{\text{BR}}$ Low to $\overline{\text{BG}}$ Low during reset	A		28		23		28	ns	
22 $\overline{\text{BR}}$ High to $\overline{\text{BG}}$ High during reset	A		21		18		21	ns	

ADSP-2100A Bus Request/Grant with $\overline{\text{RESET}}$ Low	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Switching Characteristics											
21 $\overline{\text{BR}}$ Low to $\overline{\text{BG}}$ Low during reset	A		18		16		23		18	ns	
22 $\overline{\text{BR}}$ High to $\overline{\text{BG}}$ High during reset	A		16		14		18		16	ns	



NOTE
During Reset, the Processor Bus Ignores the CLKIN Signal and Therefore the Bus Request/Grant Signals Operate Asynchronously.

Figure 13. Bus Request/Grant with $\overline{\text{RESET}}$ Low

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

	ADSP-2100 Program Memory Read	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
	Switching Characteristics									
31	$\overline{\text{PMRD}}$ Width Low	A	60		45		60		ns	4
32	PMA Valid to $\overline{\text{PMRD}}$ Low	A	18		11		18		ns	3
33	$\overline{\text{PMRD}}$ High to PMA Invalid	A	20		16		20		ns	1
34	PMDA Valid to $\overline{\text{PMRD}}$ Low	A	41		31		41		ns	3
35	$\overline{\text{PMRD}}$ High to PMDA Invalid	A	23		18		22		ns	1
36	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMRD}}$ Low	A	55		40		55		ns	3
37	$\overline{\text{PMRD}}$ High to $\overline{\text{PMS}}$ Invalid	A	16		12		16		ns	1
	Timing Requirements									
58	$\overline{\text{PMRD}}$ Low to PMD Input Valid	A		45		37		45	ns	4
59	PMA Valid to PMD Input Valid	A		57		50		57	ns	7
60	$\overline{\text{PMS}}$ Valid to PMD Input Valid	A		90		65		90	ns	7
97	$\overline{\text{PMRD}}$ High to PMD Input Invalid	A	0		0		0		ns	

	ADSP-2100A Program Memory Read	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
	Switching Characteristics											
31	$\overline{\text{PMRD}}$ Width Low	A	36		28		45		36		ns	4
32	PMA Valid to $\overline{\text{PMRD}}$ Low	A	6		4		14		6		ns	3
33	$\overline{\text{PMRD}}$ High to PMA Invalid	A	8		6		10		8		ns	1
34	PMDA Valid to $\overline{\text{PMRD}}$ Low	A	20		18		24		20		ns	3
35	$\overline{\text{PMRD}}$ High to PMDA Invalid	A	10		10		12		10		ns	1
36	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMRD}}$ Low	A	32		26		40		32		ns	3
37	$\overline{\text{PMRD}}$ High to $\overline{\text{PMS}}$ Invalid	A	8		6		8		8		ns	1
	Timing Requirements											
58	$\overline{\text{PMRD}}$ Low to PMD Input Valid	A		28		20		33		28	ns	4
59	PMA Valid to PMD Input Valid	A		46		32		50		46	ns	7
60	$\overline{\text{PMS}}$ Valid to PMD Input Valid	A		50		45		65		50	ns	7
97	$\overline{\text{PMRD}}$ High to PMD Input Invalid	A	0		0		0		0		ns	

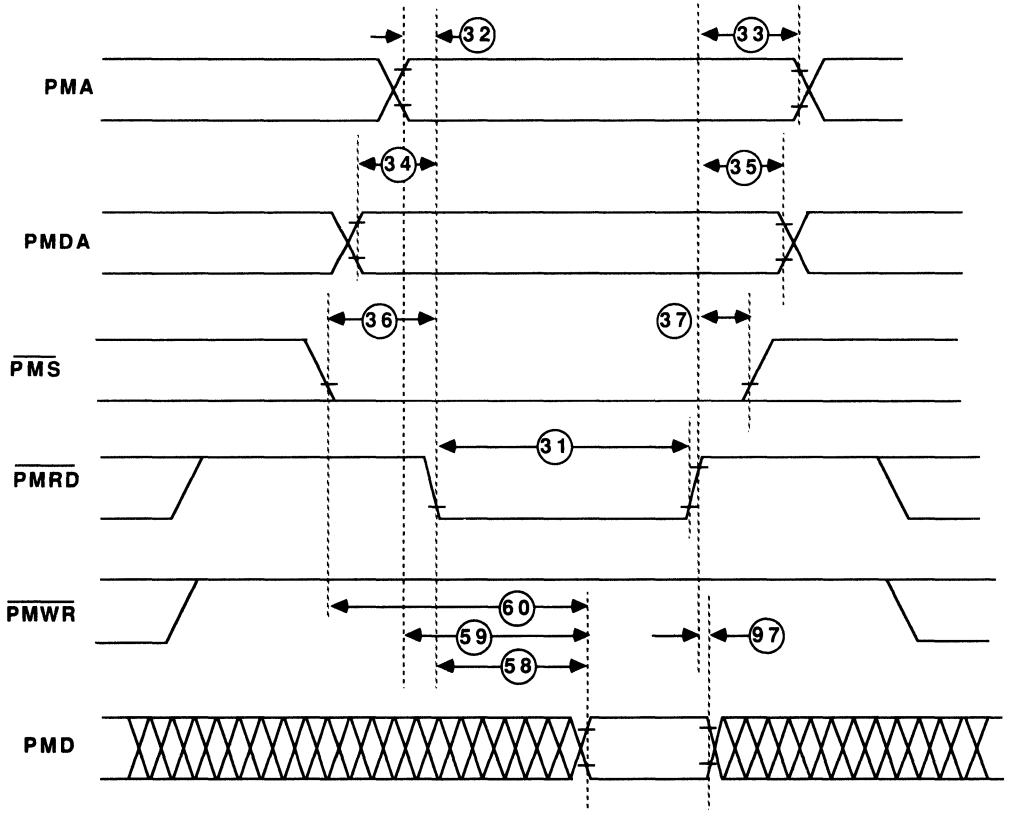


Figure 14. Program Memory Read

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Program Memory Write		Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Switching Characteristics										
40	$\overline{\text{PMWR}}$ Width Low	A	60		45		60		ns	4
41	PMA Valid to $\overline{\text{PMWR}}$ Low	A	16		10		16		ns	3
42	$\overline{\text{PMWR}}$ High to PMA Invalid	A	19		15		19		ns	1
43	PMDA Valid to $\overline{\text{PMWR}}$ Low	A	39		29		39		ns	3
44	$\overline{\text{PMWR}}$ High to PMDA Invalid	A	20		16		21		ns	1
45	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMWR}}$ Low	A	54		40		54		ns	3
46	$\overline{\text{PMWR}}$ High to $\overline{\text{PMS}}$ Invalid	A	15		11		14		ns	1
51	$\overline{\text{PMWR}}$ Low to PMD Out Enable	F	15		10		15		ns	1
52	$\overline{\text{PMWR}}$ High to PMD Out Disable	D		43		37		43	ns	1
53	$\overline{\text{PMWR}}$ Low to PMD Out Valid	A		40		32		40	ns	1
54	$\overline{\text{PMWR}}$ High to PMD Out Invalid	A	23		18		21		ns	1
55	PMD Out Valid to $\overline{\text{PMWR}}$ High	A	33		25		33		ns	3

ADSP-2100A Program Memory Write		Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Switching Characteristics												
40	$\overline{\text{PMWR}}$ Width Low	A	36		28		45		36		ns	4
41	PMA Valid to $\overline{\text{PMWR}}$ Low	A	8		4		12		8		ns	3
42	$\overline{\text{PMWR}}$ High to PMA Invalid	A	8		6		10		8		ns	1
43	PMDA Valid to $\overline{\text{PMWR}}$ Low	A	20		16		28		20		ns	3
44	$\overline{\text{PMWR}}$ High to PMDA Invalid	A	10		8		12		10		ns	1
45	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMWR}}$ Low	A	32		26		40		32		ns	3
46	$\overline{\text{PMWR}}$ High to $\overline{\text{PMS}}$ Invalid	A	6		4		8		6		ns	1
51	$\overline{\text{PMWR}}$ Low to PMD Out Enable	F	8		6		8		8		ns	1
52	$\overline{\text{PMWR}}$ High to PMD Out Disable	D		32		29		38		32	ns	1
53	$\overline{\text{PMWR}}$ Low to PMD Out Valid	A		29		26		32		29	ns	1
54	$\overline{\text{PMWR}}$ High to PMD Out Invalid	A	10		8		12		10		ns	1
55	PMD Out Valid to $\overline{\text{PMWR}}$ High	A	16		13		25		16		ns	3

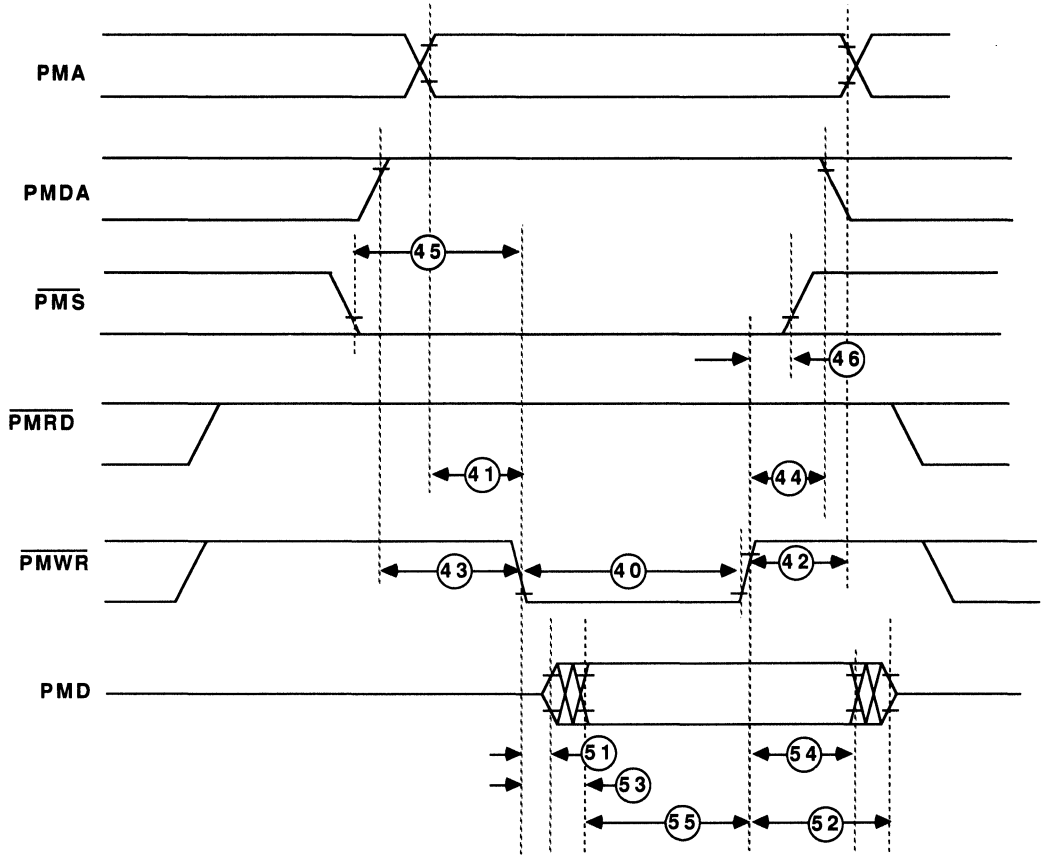


Figure 15. Program Memory Write

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Data Memory Read	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Switching Characteristics									
67 $\overline{\text{DMRD}}$ Width Low	A	60		45		60		ns	4
68 DMA Valid to $\overline{\text{DMRD}}$ Low	A	21		16		21		ns	3
69 $\overline{\text{DMRD}}$ High to DMA Invalid	A	19		15		19		ns	1
70 $\overline{\text{DMS}}$ Valid to $\overline{\text{DMRD}}$ Low	A	35		27		35		ns	3
71 $\overline{\text{DMRD}}$ High to $\overline{\text{DMS}}$ Invalid	A	22		18		21		ns	1
101 DMACK Low to CLKOUT High	A	45		37		45		ns	1
Timing Requirements									
74 $\overline{\text{DMRD}}$ Low to DMACK Valid	A		31		21		31	ns	3
75 DMA Valid to DMACK Valid	A		57		42		57	ns	6
94 $\overline{\text{DMRD}}$ Low to DMD Input Valid	A		57		41		55	ns	4
95 DMA Valid to DMD Input Valid	A		82		61		79	ns	7
96 $\overline{\text{DMS}}$ Valid to DMD Input Valid	A		96		70		96	ns	7
98 $\overline{\text{DMRD}}$ High to DMD Input Invalid	A	0		0		0		ns	
100 DMACK Width	A	81		61		81		ns	4
102 CLKOUT Low to DMACK High	A		28		19		28	ns	3

ADSP-2100A Data Memory Read	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Switching Characteristics											
67 $\overline{\text{DMRD}}$ Width Low	A	36		28		45		36		ns	4
68 DMA Valid to $\overline{\text{DMRD}}$ Low	A	6		4		14		6		ns	3
69 $\overline{\text{DMRD}}$ High to DMA Invalid	A	8		6		10		8		ns	1
70 $\overline{\text{DMS}}$ Valid to $\overline{\text{DMRD}}$ Low	A	18		14		27		18		ns	3
71 $\overline{\text{DMRD}}$ High to $\overline{\text{DMS}}$ Invalid	A	8		6		10		8		ns	1
101 DMACK Low to CLKOUT High	A	36		32		37		36		ns	1
Timing Requirements											
74 $\overline{\text{DMRD}}$ Low to DMACK Valid	A		16		10		21		16	ns	3
75 DMA Valid to DMACK Valid	A		30		20		42		30	ns	6
94 $\overline{\text{DMRD}}$ Low to DMD Input Valid	A		30		20		37		28	ns	4
95 DMA Valid to DMD Input Valid	A		48		32		59		46	ns	7
96 $\overline{\text{DMS}}$ Valid to DMD Input Valid	A		52		45		67		50	ns	7
98 $\overline{\text{DMRD}}$ High to DMD Input Invalid	A	0		0		0		0		ns	
100 DMACK Width	A	50		40		61		50		ns	4
102 CLKOUT Low to DMACK High	A		17		12		19		17	ns	3

NOTE ON GENERATING WAIT STATES

Figures 16a and 17a show the timing of DMACK relative to the data memory bus and control signals. If DMACK is not asserted in this time frame, a wait state will result. Figures 16b and 17b provide additional timing for DMACK with respect to CLKOUT so that any number of additional wait states can be introduced. Since CLKOUT is the only output active during a wait state, it can be used as a cycle counter to determine when the appropriate number of wait states has elapsed. DMACK can be latched for the appropriate number of cycles or a counter can be used to count CLKOUT cycles.

Specification #101 shows the time from the assertion of DMACK until the rising edge of CLKOUT. DMACK should be held low at least this amount of time if the rising edge of CLKOUT is used to latch the DMACK signal into your wait state logic.

Specification #102 indicates the maximum amount of time from the falling edge of CLKOUT to when DMACK must be brought high to terminate the wait state condition. The falling edge of CLKOUT can be used to clear your wait state logic.

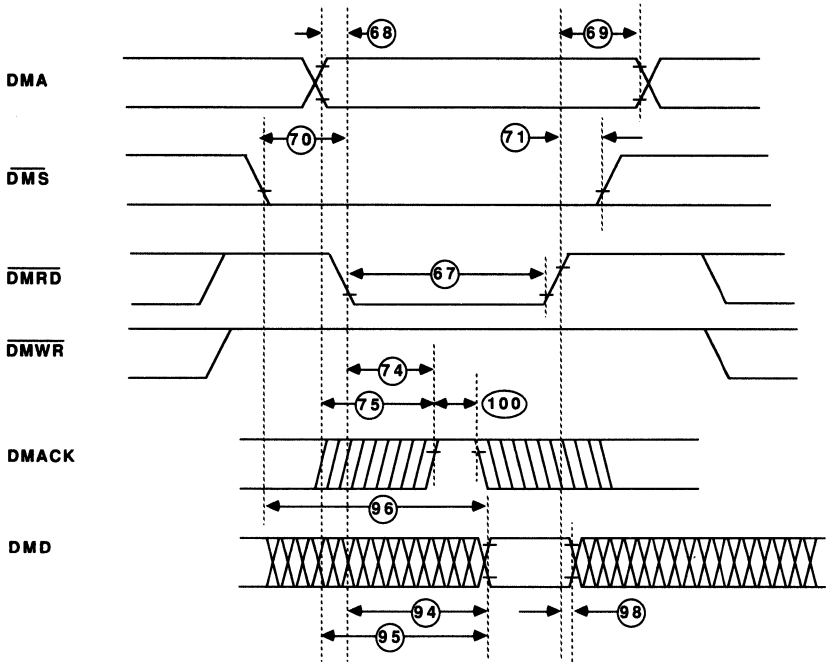


Figure 16a. Data Memory Read

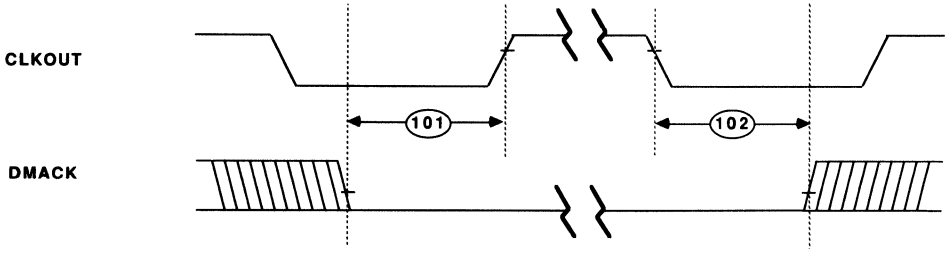


Figure 16b. Data Memory Wait States Extended with DMACK

Notes 1 and 2 and information about the Derating Factors and Test Codes appear on page 2–50.

ADSP-2100 Data Memory Write	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units	Derating Factor
Switching Characteristics									
78 $\overline{\text{DMWR}}$ Width Low	A	60		45		60		ns	4
79 DMA Valid to $\overline{\text{DMWR}}$ Low	A	24		17		24		ns	3
80 $\overline{\text{DMWR}}$ High to DMA Invalid	A	20		15		19		ns	1
81 $\overline{\text{DMS}}$ Valid to $\overline{\text{DMWR}}$ Low	A	37		28		37		ns	3
82 $\overline{\text{DMWR}}$ High to $\overline{\text{DMS}}$ Invalid	A	22		19		22		ns	1
87 $\overline{\text{DMWR}}$ Low to DMD Out Enable	F	14		9		14		ns	1
88 $\overline{\text{DMWR}}$ High to DMD Out Disable	D		40		35		40	ns	1
89 $\overline{\text{DMWR}}$ Low to DMD Out Valid	A		38		32		38	ns	1
90 $\overline{\text{DMWR}}$ High to DMD Out Invalid	A	21		16		19		ns	1
91 DMD Out Valid to $\overline{\text{DMWR}}$ High	A	33		21		33		ns	3
101 DMACK Low to CLKOUT High	A	45		37		45		ns	1
Timing Requirements									
75 DMA Valid to DMACK Valid	A		57		42		57	ns	6
99 $\overline{\text{DMWR}}$ Low to DMACK Valid	A		31		21		31	ns	3
100 DMACK Width	A	81		61		81		ns	4
102 CLKOUT Low to DMACK High	A		28		19		28	ns	3

ADSP-2100A Data Memory Write	Test Code	AJ Grade Min	AJ Grade Max	AK Grade Min	AK Grade Max	AS Grade Min	AS Grade Max	AT Grade Min	AT Grade Max	Units	Derating Factor
Switching Characteristics											
78 $\overline{\text{DMWR}}$ Width Low	A	36		28		45		36		ns	4
79 DMA Valid to $\overline{\text{DMWR}}$ Low	A	8		4		17		8		ns	3
80 $\overline{\text{DMWR}}$ High to DMA Invalid	A	8		6		10		8		ns	1
81 $\overline{\text{DMS}}$ Valid to $\overline{\text{DMWR}}$ Low	A	20		16		28		20		ns	3
82 $\overline{\text{DMWR}}$ High to $\overline{\text{DMS}}$ Invalid	A	6		4		8		6		ns	1
87 $\overline{\text{DMWR}}$ Low to DMD Out Enable	F	8		6		8		8		ns	1
88 $\overline{\text{DMWR}}$ High to DMD Out Disable	D		32		29		38		32	ns	1
89 $\overline{\text{DMWR}}$ Low to DMD Out Valid	A		29		26		32		29	ns	1
90 $\overline{\text{DMWR}}$ High to DMD Out Invalid	A	10		8		12		10		ns	1
91 DMD Out Valid to $\overline{\text{DMWR}}$ High	A	18		13		25		16		ns	3
101 DMACK Low to CLKOUT High	A	36		32		37		36		ns	1
Timing Requirements											
75 DMA Valid to DMACK Valid	A		30		20		42		30	ns	6
99 $\overline{\text{DMWR}}$ Low to DMACK Valid	A		16		10		20		16	ns	3
100 DMACK Width	A	50		40		61		50		ns	4
102 CLKOUT Low to DMACK High	A		17		12		19		17	ns	3

NOTE ON GENERATING WAIT STATES

Figures 16a and 17a show the timing of DMACK relative to the data memory bus and control signals. If DMACK is not asserted in this time frame, a wait state will result. Figures 16b and 17b provide additional timing for DMACK with respect to CLKOUT so that any number of additional wait states can be introduced. Since CLKOUT is the only output active during a wait state, it can be used as a cycle counter to determine when the appropriate number of wait states has elapsed. DMACK can be latched for the appropriate number of cycles or a counter can be used to count CLKOUT cycles.

Specification #101 shows the time from the assertion of DMACK until the rising edge of CLKOUT. DMACK should be held low at least this amount of time if the rising edge of CLKOUT is used to latch the DMACK signal into your wait state logic.

Specification #102 indicates the maximum amount of time from the falling edge of CLKOUT to when DMACK must be brought high to terminate the wait state condition. The falling edge of CLKOUT can be used to clear your wait state logic.

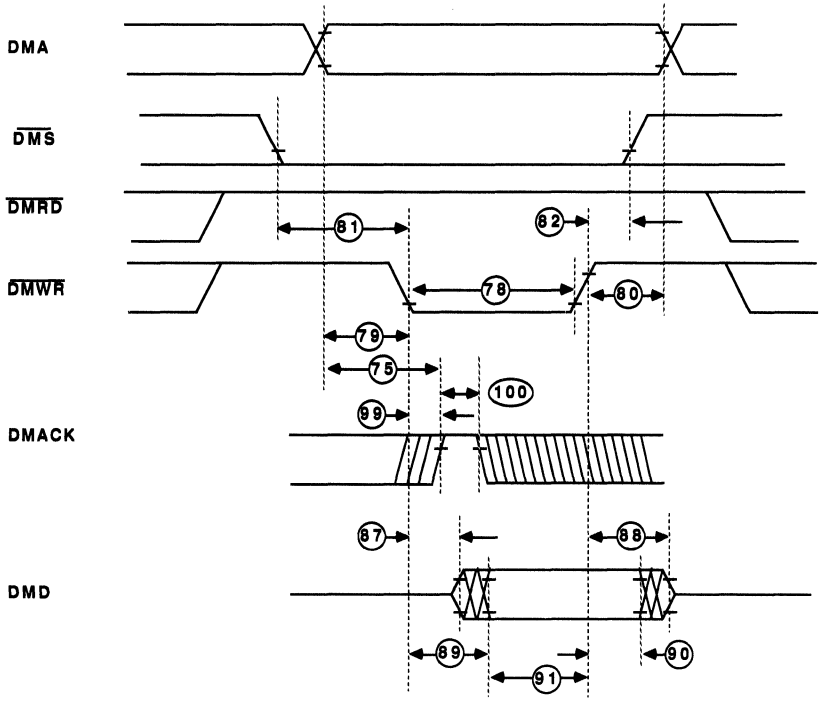


Figure 17a. Data Memory Write

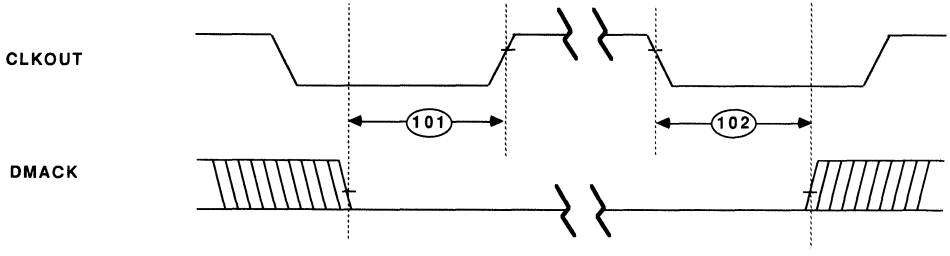


Figure 17b. Data Memory Wait States Extended with DMACK

NOTES

[1]Rise and fall times ≤4ns for ADSP-2100A, 5ns for ADSP-2100.

[2]"xMxx" refers to PMA_{0-13}, $\overline{PMS}$, $\overline{PMRD}$, $\overline{PMWR}$, PMDA, DMA_{0-13}, $\overline{DMS}$, $\overline{DMRD}$ and $\overline{DMWR}$.

TEST CODES

Code	Test Type	Level Reference
A	Inputs, Outputs	Low = 0.8V, High = 2.0V
B	CLKIN *to/from* Inputs, Outputs	1.5V Low = 0.8V, High = 2.0V
D	Output *to* Output Disable	Low = 0.8V, High = 2.0V Low = V_{OL} + 0.5V, High = V_{OH} − 0.5V
F	Output *to/from* Output Enable	Low = 0.8V, High = 2.0V Low = VT − 0.1V, High = VT + 0.1V

VT = 1.5V, the voltage to which tristated outputs are forced.

DERATING FACTOR

The value **N** in the Derating Column shows, for each timing parameter affected, how many of the eight internal clock states are used by this timing parameter; **N**, therefore, ranges between 1 and 8. The formula for changing any individual parameter T uses timing parameter number one, CLKIN Period, shown as P#1:

$$T_{new} = T_{old} + N\ ((P\#1_{new} - P\#1_{old})\ /2)$$

You determine the new value of P#1 based on the derating you wish to accomplish. If no **N** value is given for derating, that timing parameter does not change with clock changes.

CAPACITANCE IN PGA PACKAGE

Input capacitance	C_{IN}	10pF typical
Output capacitance	C_{OUT}	10pF typical

Note that output-only pads (PMA_{13-0}, PMDA and DMA_{13-0}) and bidirectional pads (PMD_{23-0} and DMD_{15-0}) have 50kΩ (typical) pull-up resistors between the output and V_{DD} present when the output driver is off.

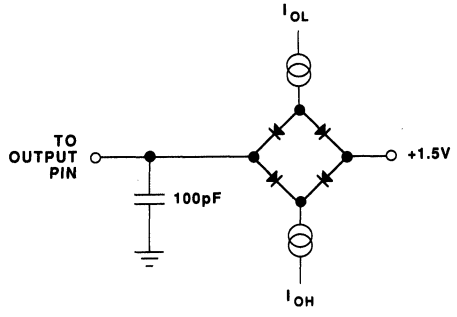
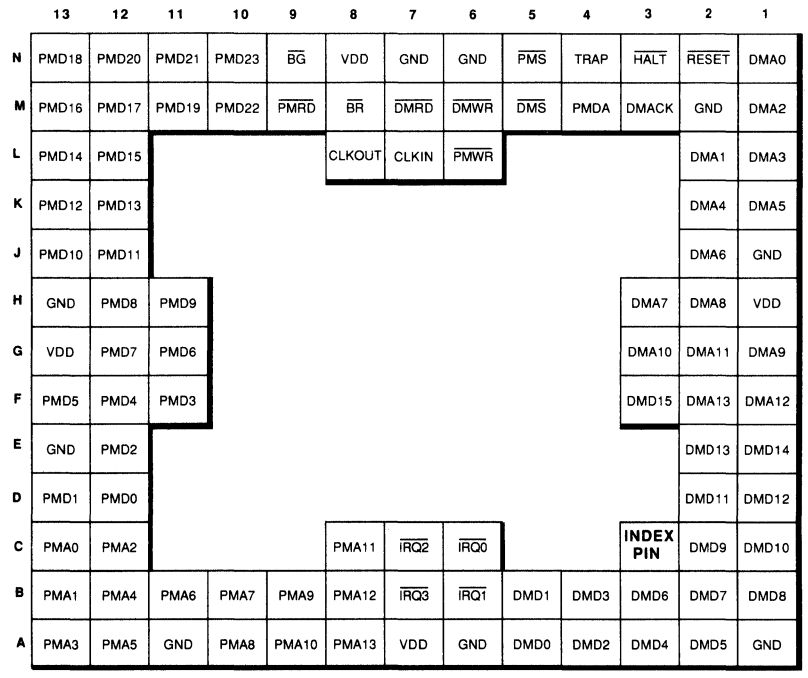


Figure 18. Normal Load for ac Measurements



	13	12	11	10	9	8	7	6	5	4	3	2	1
N	PMD18	PMD20	PMD21	PMD23	$\overline{BG}$	VDD	GND	GND	$\overline{PMS}$	TRAP	$\overline{HALT}$	$\overline{RESET}$	DMA0
M	PMD16	PMD17	PMD19	PMD22	$\overline{PMRD}$	$\overline{BR}$	$\overline{DMRD}$	$\overline{DMWR}$	$\overline{DMS}$	PMDA	DMACK	GND	DMA2
L	PMD14	PMD15				CLKOUT	CLKIN	$\overline{PMWR}$				DMA1	DMA3
K	PMD12	PMD13										DMA4	DMA5
J	PMD10	PMD11										DMA6	GND
H	GND	PMD8	PMD9								DMA7	DMA8	VDD
G	VDD	PMD7	PMD6								DMA10	DMA11	DMA9
F	PMD5	PMD4	PMD3								DMD15	DMA13	DMA12
E	GND	PMD2										DMD13	DMD14
D	PMD1	PMD0										DMD11	DMD12
C	PMA0	PMA2				PMA11	$\overline{IRQ2}$	$\overline{IRQ0}$			INDEX PIN	DMD9	DMD10
B	PMA1	PMA4	PMA6	PMA7	PMA9	PMA12	$\overline{IRQ3}$	$\overline{IRQ1}$	DMD1	DMD3	DMD6	DMD7	DMD8
A	PMA3	PMA5	GND	PMA8	PMA10	PMA13	VDD	GND	DMD0	DMD2	DMD4	DMD5	GND

Figure 19. ADSP-2100 Pins, Top View, Pins Down

Function	Location	Function	Location	Function	Location	Function	Location
V_{DD}	A7	PMA1	B13	PMD12	K13	DMA9	G1
V_{DD}	G13	PMA2	C12	PMD13	K12	DMA10	G3
V_{DD}	H1	PMA3	A13	PMD14	L13	DMA11	G2
V_{DD}	N8	PMA4	B12	PMD15	L12	DMA12	F1
GND	A1	PMA5	A12	PMD16	M13	DMA13	F2
GND	A6	PMA6	B11	PMD17	M12	DMD0	A5
GND	A11	PMA7	B10	PMD18	N13	DMD1	B5
GND	E13	PMA8	A10	PMD19	M11	DMD2	A4
GND	H13	PMA9	B9	PMD20	N12	DMD3	B4
GND	J1	PMA10	A9	PMD21	N11	DMD4	A3
GND	M2	PMA11	C8	PMD22	M10	DMD5	A2
GND	N6	PMA12	B8	PMD23	N10	DMD6	B3
GND	N7	PMA13	A8	$\overline{PMS}$	N5	DMD7	B2
CLKIN	L7	PMD0	D12	$\overline{PMWR}$	L6	DMD8	B1
CLKOUT	L8	PMD1	D13	$\overline{PMRD}$	M9	DMD9	C2
$\overline{BR}$	M8	PMD2	E12	PMDA	M4	DMD10	C1
$\overline{BG}$	N9	PMD3	F11	DMA0	N1	DMD11	D2
$\overline{IRQ0}$	C6	PMD4	F12	DMA1	L2	DMD12	D1
$\overline{IRQ1}$	B6	PMD5	F13	DMA2	M1	DMD13	E2
$\overline{IRQ2}$	C7	PMD6	G11	DMA3	L1	DMD14	E1
$\overline{IRQ3}$	B7	PMD7	G12	DMA4	K2	DMD15	F3
$\overline{RESET}$	N2	PMD8	H12	DMA5	K1	$\overline{DMS}$	M5
TRAP	N4	PMD9	H11	DMA6	J2	$\overline{DMWR}$	M6
$\overline{HALT}$	N3	PMD10	J13	DMA7	H3	$\overline{DMRD}$	M7
INDEX PIN	NC	PMD11	J12	DMA8	H2	DMACK	M3
PMA0	C13						

Table V. ADSP-2100 Pins by Function – G-100A

PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION
1	PMD6	21	PMA10	41	DMD9	61	DMA2	81	$\overline{BG}$
2	V_{DD}	22	PMA11	42	DMD10	62	DMA1	82	$\overline{PMRD}$
3	PMD5	23	PMA12	43	DMD11	63	DMA0	83	PMD23
4	PMD4	24	PMA13	44	DMD12	64	GND	84	PMD22
5	PMD3	25	$\overline{IRQ3}$	45	DMD13	65	$\overline{RESET}$	85	PMD21
6	GND	26	$\overline{IRQ2}$	46	DMD14	66	DMACK	86	PMD20
7	PMD2	27	V_{DD}	47	DMD15	67	$\overline{HALT}$	87	PMD19
8	PMD1	28	GND	48	DMA13	68	PMDA	88	PMD18
9	PMD0	29	$\overline{IRQ1}$	49	DMA12	69	TRAP	89	PMD17
10	PMA0	30	$\overline{IRQ0}$	50	DMA11	70	$\overline{DMS}$	90	PMD16
11	PMA1	31	DMD0	51	DMA10	71	$\overline{PMS}$	91	PMD15
12	PMA2	32	DMD1	52	DMA9	72	$\overline{PMWR}$	92	PMD14
13	PMA3	33	DMD2	53	V_{DD}	73	$\overline{DMWR}$	93	PMD13
14	PMA4	34	DMD3	54	DMA8	74	GND	94	PMD12
15	PMA5	35	DMD4	55	DMA7	75	$\overline{DMRD}$	95	PMD11
16	PMA6	36	DMD5	56	GND	76	CLKIN	96	PMD10
17	GND	37	DMD6	57	DMA6	77	GND	97	PMD9
18	PMA7	38	GND	58	DMA5	78	V_{DD}	98	PMD8
19	PMA8	39	DMD7	59	DMA4	79	$\overline{BR}$	99	GND
20	PMA9	40	DMD8	60	DMA3	80	CLKOUT	100	PMD7

Table VI. ADSP-2100 Pins by Function – P-100

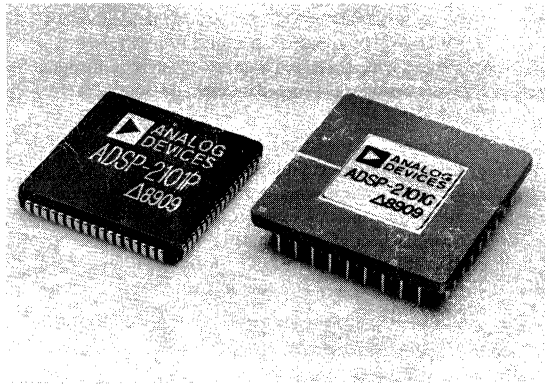
ANALOG DEVICES

12.5MIPS DSP Microcomputer

ADSP-2101/ADSP-2102

FEATURES

Complete DSP Microcomputer
ADSP-2100 Code & Function Compatible
2K Words of Program Memory RAM
ADSP-2102 Version Has Up to 2K Words of Mask Programmable Program Memory
1K Words of Data Memory RAM
Separate Program and Data Buses On-Chip
Dual Purpose Program Memory for Both Instruction and Data Storage
Three Independent Computational Units: ALU, Multiplier/Accumulator and Barrel Shifter
Two Independent Data Address Generators
Powerful Program Sequencer
Zero Overhead Looping
Conditional Arithmetic Instruction Execution
Two Double-Buffered Serial Ports with Companding Hardware and Automatic Data Buffering
Programmable Interval Timer
Programmable Wait State Generation
Automatic Booting from Byte-Wide External Memory, e.g., EPROM
Provisions for Multiprecision Computation and Saturation Logic
Single-Cycle Instruction Execution
Multifunction Instructions
Three Edge- or Level-Sensitive External Interrupts
80ns Cycle Time
80mW Maximum Power Dissipation in Standby Mode
68-Pin PGA and 68-Lead PLCC



GENERAL DESCRIPTION

The ADSP-2101/ADSP-2102 is a single-chip microcomputer optimized for digital signal processing (DSP) and other high speed numeric processing applications. Its instruction set is a fully compatible superset of the ADSP-2100 instruction set. It combines the complete ADSP-2100 architecture (three computational units, data address generators and a program sequencer) with two serial ports, a programmable timer, extensive interrupt capabilities and on-chip program and data memory SRAM (or RAM and ROM in ADSP-2102). The ADSP-2101/ADSP-2102 surpasses other single-chip DSP microcomputers in both performance and ease of design and development.

Fabricated in a high speed 1.0 micron double-layer metal CMOS process, the ADSP-2101/ADSP-2102 operates at 12.5MHz. Every instruction executes in a single cycle, resulting in a 12.5 MIPS processor. Fabrication in CMOS results in low power requirements. The ADSP-2101/ADSP-2102 dissipates less than 1W under all conditions and no more than 80mW under standby conditions.

The ADSP-2101 is a RAM based microcomputer with 1K words of (16-bit) data memory and 2K words of (24-bit) program memory. The ADSP-2102 is a mask programmable version allowing any RAM location to be changed to ROM. In this data sheet, all references to the ADSP-2101 are applicable to the ADSP-2102 except where noted.

The ADSP-2101's flexible architecture and comprehensive instruction set support a high degree of operational parallelism. In one cycle the ADSP-2101 can:

- generate the next program address
- fetch the next instruction
- perform one or two data moves
- update one or two data address pointers
- perform a computational operation
- receive and transmit data via the two serial ports.

DEVELOPMENT SYSTEM

The ADSP-2101 is supported by a complete set of tools for software and hardware system development. The cross-software system is a set of modules. The System Builder provides a high level method for defining the architecture of systems under development. The Assembler produces object code and the Linker combines object modules and library calls into an executable file. The Simulator provides an interactive instruction level simulation with a reconfigurable user interface. A PROM splitter generates PROM burner compatible files. The C Compiler generates ADSP-2101 assembly source code. An Emulator will be available for hardware debugging of ADSP-2101 systems.

ADDITIONAL INFORMATION

For additional information on the architecture and instruction set of the processor, refer to the *ADSP-2101/ADSP-2102 User's Manual*. For more information about the development aystem and ADSP-2101 programmer's reference information, refer to the *ADSP-210X Cross-Software Manual* and the (forthcoming) *ADSP-2101/ADSP-2102 Emulator Manual*.

ARCHITECTURE OVERVIEW

Figure 1 is an overall block diagram of the ADSP-2101. For compatibility with the ADSP-2100 processor, the additional features of the ADSP-2101 appear in the form of new mode controls, new processor registers and a group of memory mapped control registers residing between data memory addresses H#3FE0 and H#3FFF.

The processor contains three independent computational units: the ALU, the multiplier/accumulator (MAC) and the shifter. The computational units process 16-bit data directly and have provisions to support multiprecision computations. The ALU performs a standard set of arithmetic and logic operations; division primitives are also supported. The MAC performs single cycle multiply, multiply/add and multiply/subtract operations. The shifter performs logical and arithmetic shifts, normalization, denormalization, and derive exponent operations. The shifter can be used to efficiently implement numeric format control including multiword floating point representations.

The internal result (R) bus directly connects the computational units so that the output of any unit may be the input of any unit on the next cycle.

A powerful program sequencer and two dedicated data address generators ensure efficient use of these computational units. The sequencer supports conditional jumps, subroutine calls and returns in a single cycle. With internal loop counters and loop stacks, the ADSP-2101 executes looped code with zero overhead; no explicit jump instructions are required to maintain the loop.

The data address generators (DAGs) handle address pointer updates. Each DAG keeps track of four address pointers. Whenever the pointer is used to access data (indirect addressing), it is post-modified by the value of a specified modify register. A length value may be associated with each pointer to implement automatic modulo addressing for circular buffers. With two independent DAGs, the processor can generate two addresses simultaneously for dual operand fetches. The circular buffering feature is also used by the serial ports for automatic data transfers; these are described in the section on serial ports.

Efficient data transfer is achieved with the use of five internal buses.

- Program Memory Address (PMA) bus
- Program Memory Data (PMD) bus
- Data Memory Address (DMA) bus
- Data Memory Data (DMD) bus
- Result (R) bus

The two address buses (PMA and DMA) share a single external address bus, and the two data buses (PMD and DMD) share a single external data bus. The $\overline{\text{BMS}}$, $\overline{\text{DMS}}$ and $\overline{\text{PMS}}$ signals indicate which memory space for which the external buses are being used.

As in the ADSP-2100, program memory can store both instructions and data, permitting the ADSP-2101 to fetch two operands in a single cycle, one from program memory and one from data memory. Because the on-chip program memory is so fast, the ADSP-2101 can fetch an operand from program memory and the next instruction in the same cycle. (This eliminates the need for the cache memory found on the ADSP-2100, as well as any overhead cycles that were associated with initial loading of the cache.)

The memory interface supports slow memories and memory-mapped peripherals with programmable wait state generation. External devices can gain control of buses with bus request/grant signals ($\overline{\text{BR}}$ and $\overline{\text{BG}}$). One execution mode allows the

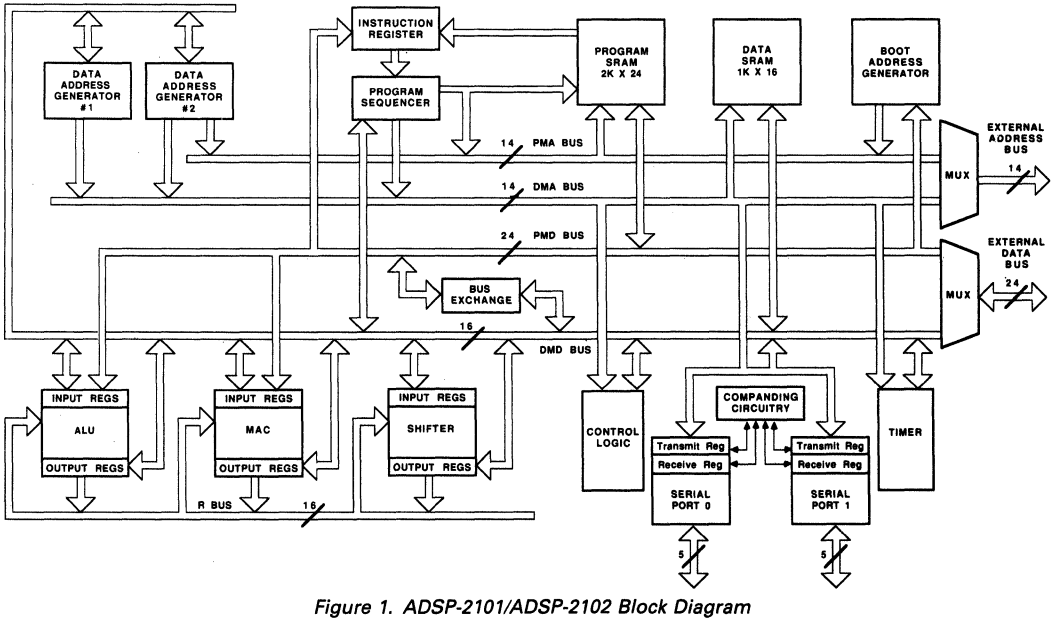


Figure 1. ADSP-2101/ADSP-2102 Block Diagram

PIN DESCRIPTION

The ADSP-2101 is available in a 68-pin PGA and a 68-lead PLCC.

Pin Group Name	# of Pins	Function
Address	14	Address output for program, data and boot memory spaces
Data	24	Data I/O pins for program and data memories. Input only for boot memory space, with two MSBs used as boot space addresses.
$\overline{\text{RESET}}$	1	Processor reset input
$\overline{\text{IRQ2}}$	1	External interrupt request #2 input
$\overline{\text{BR}}$	1	External bus request input
$\overline{\text{BG}}$	1	External bus grant output
$\overline{\text{PMS}}$	1	External program memory select
$\overline{\text{DMS}}$	1	External data memory select
$\overline{\text{BMS}}$	1	Boot memory select
$\overline{\text{RD}}$	1	External memory read enable output
$\overline{\text{WR}}$	1	External memory write enable output
MMAP	1	Memory map select
CLKIN, XTAL	2	External clock or quartz crystal input
CLKOUT	1	Processor clock output
SPORT0	5	Serial Port 0 I/O pins
SPORT1	5	Serial Port 1 I/O pins
	or	
$\overline{\text{IRQ1}}$	1	External interrupt request #1 input
$\overline{\text{IRQ0}}$	1	External interrupt request #0 input
SCLK	1	Programmable clock output
FO	1	Flag output pin
FI	1	Flag input pin
GND	4	Ground pins
V_{DD}	3	Power Supply

Table I. ADSP-2101 Pin List

ADSP-2101 to continue running while the buses are granted to another master as long as an external memory operation is not required. The other execution mode requires the processor to halt while buses are granted.

The ADSP-2101 can respond to six interrupts. There can be up to three external interrupts, configured as edge- or level-sensitive. Internal interrupts can be generated by the Timer and the Serial Ports ("SPORTS"). There is also a master $\overline{\text{RESET}}$ signal.

The two serial ports provide a complete serial interface with companding in hardware and a wide variety of framed and frameless data transmit and receive modes of operation. Each port can generate an internal programmable serial clock or accept an external serial clock.

Boot circuitry provides for loading on-chip program memory automatically from byte-wide external memory. After $\overline{\text{RESET}}$ three wait states are automatically generated. This allows, for example, an 80ns ADSP-2101 to use an external 250ns EPROM as boot memory. Multiple programs can be selected and loaded from the EPROM with no additional hardware.

The ADSP-2101 instruction set provides flexible data moves and multifunction (one or two data moves with a computation) instructions. Every instruction can be executed in a single processor cycle. The ADSP-2101 assembly language uses an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

Arithmetic/Logic Unit

Figure 2 shows the Arithmetic/Logic Unit (ALU).

The ALU provides a standard set of arithmetic and logic functions: add, subtract, negate, increment, decrement, absolute value, AND, OR, Exclusive OR and NOT. Two divide primitives are also provided. The ALU takes two 16-bit inputs, X and Y, and generates one 16-bit output, R. It accepts the carry (AC) bit in the arithmetic status register (ASTAT) as the carry-in (CI) bit. The carry-in feature enables multiword computations. Six arithmetic status bits are generated: AZ (zero), AN (negative), AV (overflow), AC (carry), AS (sign) and AQ (quotient). These status bits are latched in ASTAT.

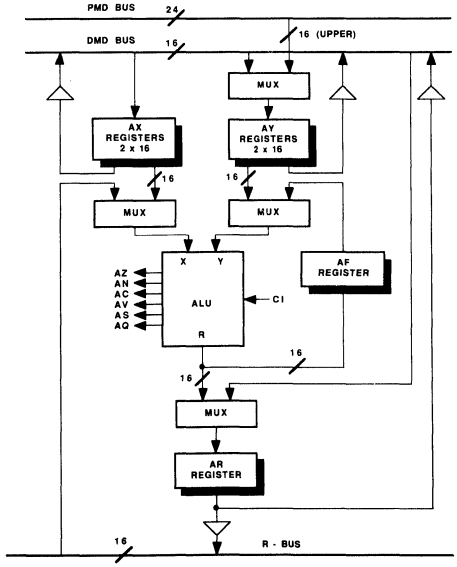


Figure 2. ALU Block Diagram

The X input port can be fed by either the AX register set or any result register via the R bus (AR, MR0, MR1, MR2, SR0, or SR1). The AX register set contains two registers, AX0 and AX1. The AX registers can be loaded from the DMD bus. The Y input port can be fed by either the AY register set or the ALU feedback (AF) register. The AY register set contains two registers, AY0 and AY1. The AY registers can be loaded from either the DMD bus or the PMD bus.

The register outputs are dual-ported so that one register can provide input to the ALU while either one simultaneously drives the DMD bus. The ALU output can be loaded into either the AR register or the AF register.

The AR register has a saturation capability; it can be automatically set to plus or minus the maximum value if an overflow or underflow occurs. The saturation mode is enabled by a bit in the mode status register (MSTAT). The AR register can drive both the R bus and the DMD bus and can be loaded from the DMD bus.

The ALU contains a duplicate bank of registers shown in Figure 2 as a "shadow" behind the primary registers. The secondary set contains all the registers described above (AX0, AX1, AY0, AY1, AF, AR). Only one set is accessible at a time. The two sets of registers allow fast context switching, such as for interrupt servicing. The active set is determined by a bit in MSTAT.

Multiplier/Accumulator

The multiplier/accumulator (MAC) implements high speed multiply, multiply/add and multiply/subtract operations. Figure 3 shows a block diagram of the MAC section.

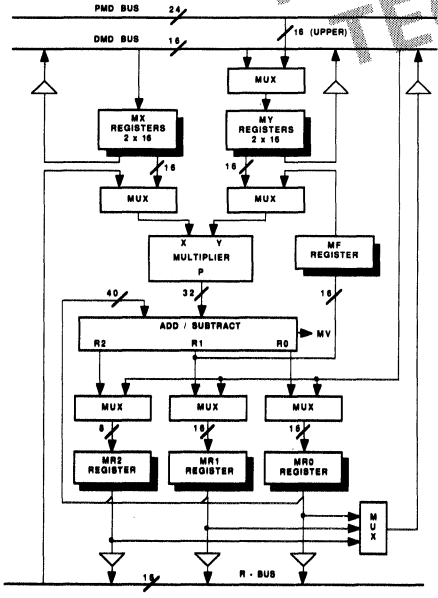


Figure 3. MAC Block Diagram

The multiplier takes two 16-bit inputs, X and Y, and generates one 32-bit output, P. The 32-bit output is routed to a 40-bit accumulator which can add or subtract the P output from the value in MR. MR is a 40-bit register which is divided into three sections: MR0 (Bits 0–15), MR1 (Bits 16–31), and MR2 (Bits 32–39). The result of the accumulator is either loaded into the MR register or into the 16-bit MAC feedback (MF) register. The multiplier accepts the X and Y inputs in either signed or unsigned formats.

In the default operation (ADSP-2100 mode) the result is shifted one bit to the left to remove the redundant sign bit for fractional justification; an optional mode on the ADSP-2101 inhibits this shift for integer operations. The accumulator generates one status bit, MV, which is set when the accumulator result overflows the 32-bit boundary. A saturate instruction is available to change the contents of the MR register to the maximum or minimum 32-bit value if MV is set. The accumulator also has the capability for rounding the 40-bit result at the boundary between Bit 15 and Bit 16.

The MAC and ALU registers are similar. The X input port can be fed by either the MX register set (MX0, MX1) or any result register via the R bus (AR, MR0, MR1, MR2, SR0 or SR1). The MX register set is readable and loadable from the DMD bus and has dual ported outputs.

The Y input port can be fed by either the MY register set (MY0, MY1) or the MF register. The MY register set is readable from the DMD bus and readable and loadable from both the DMD and the PMD bus. Its outputs are also dual ported. The accumulator output can be loaded into either the MR register or the MF register. The MR register is connected to both the R bus and the DMD bus. Like the ALU section, the MAC section contains two complete banks of registers (MX0, MX1, MY0, MY1, MF, MR0, MR1, MR2) to allow fast context switching.

Shifter

The shifter gives the ADSP-2101 its unique capability to handle data formatting and numeric scaling. Figure 4 shows a block diagram of the shifter.

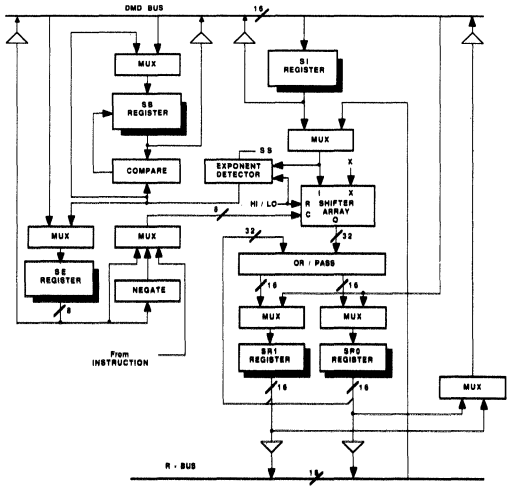


Figure 4. Shifter Block Diagram

The shifter can be divided into the following components: the shifter array, the OR/PASS logic, the exponent detector and the exponent compare logic. These components give the shifter its six basic functions: arithmetic shift, logical shift, normalization, denormalization, derive exponent and derive block exponent.

The shifter array is a 16×32 barrel shifter. It accepts a 16-bit input and can place it anywhere in the 32-bit output field, from

off scale right to off scale left. The shifter can perform arithmetic shifts (shifter output is sign extended to the left) or logical shifts (shifter output is zero filled to the left). The placement of the 16-bit input is determined by the control code (C) and the HI/LO reference signal. The control code can come from one of three sources: directly from the instruction (immediate arithmetic or logical shift), from the SE register (denormalization), or the negated value of the SE register (normalization). The shifter input can come from either the 16-bit SI register or any result register via the R bus. The 32-bit output of the shifter array is fed to the OR/PASS circuit. The result can be either logically ORed with the current contents of the SR register or passed directly to the SR register. The SR register is divided into two 16-bit sections: SR0 (Bits 0–15) and SR1 (Bits 16–31).

The shifter input is also routed to the exponent detector circuitry. The exponent detector generates a value to indicate how many places the input must be up shifted to eliminate all but one of the sign bits. This value is effectively the base 2 exponent of the number. The result of the exponent detector can be latched into the SE register (for a normalize operation) or can be sent to the exponent compare logic. The exponent compare logic compares the derived exponent with the value in the SB register and updates the SB register only when the derived exponent value is larger than the current value in the SB register. Therefore, the exponent compare logic can be used to find the largest exponent value in an array of shifter inputs.

The shifter includes the following registers: the SI register, the SE register, the SB register and the SR register. All these registers are readable and loadable from the DMD bus. The SR register can also drive the R bus. Like the ALU and MAC, the shifter contains two complete banks of registers for context switching. Each set contains all the registers described above, but only one set is accessible at a time. The active set is determined by a bit in MSTAT.

Data Address Generators

Figure 5 shows a block diagram of a data address generator.

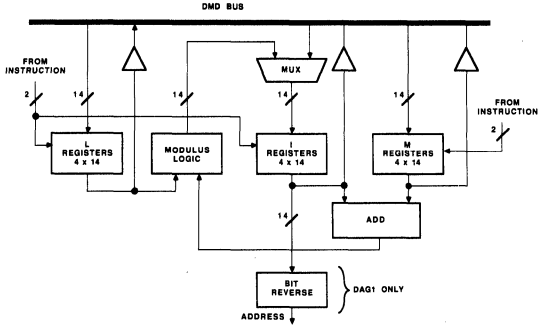


Figure 5. Data Address Generator Block Diagram

The data address generators (DAGs) provide indirect addressing for data stored in the program and data memory spaces. The processor contains two independent DAGs so that two data operands (one in program memory and one in data memory) can be addressed simultaneously. The two data address generators are identical except that DAG1 has a bit-reversal option on the output and can only generate data memory addresses, while DAG2 can generate both program and data memory addresses but has no bit-reversal capability. Both DAGs can also be used for serial port autobuffering.

There are three register files in each DAG: the modify (M) register file, the index (I) register file and the length (L) register file. Each of these register files contains four 14-bit registers which are readable and loadable from the DMD bus. The I registers hold the actual addresses used to access external memory. When using the indirect addressing mode, the selected I register content is driven onto either the PMA or DMA bus. This value is post-modified by adding the (signed) contents of the selected M register. The modified address is passed through the modulus logic.

Associated with each I register is an L register which contains the length of the buffer addressed by the I register. The L register and the modulus logic together enable circular buffer addressing with automatic wraparound at the buffer boundary. Automatic wraparound is also used by the serial ports to generate the serial port interrupt when operating in autobuffering mode. The modulus logic is disabled by setting the L register to zero.

PMD-DMD Bus Exchange

The PMD-DMD bus exchange circuit couples the PMD and DMD buses. The PMD bus is 24 bits wide and the DMD bus is 16 bits wide. The upper 16 bits of PMD are connected to the DMD bus. An 8-bit register (PX) allows transfer of the full width of the PMD bus. When data (as distinct from an instruction) is read from the PMD bus, the lower 8 bits of the PMD bus are loaded into PX. When writing to the PMD bus, the contents of PX are appended to the upper 16 bits, forming a 24-bit value. The PX register is also readable and loadable from the DMD bus.

Program Sequencer

The program sequencer incorporates powerful and flexible mechanisms for program flow control such as zero overhead looping, single cycle branching (both conditional and unconditional) and automatic interrupt processing. Figure 6 shows a block diagram of the program sequencer.

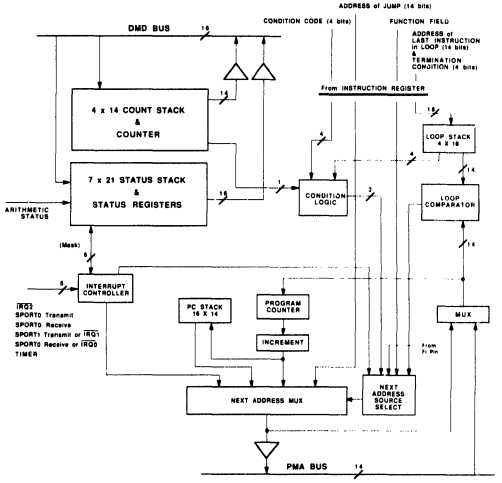


Figure 6. ADSP-2101 Program Sequencer

The sequencing logic controls the flow of the program execution. It outputs a program memory address onto the PMA bus from one of four sources: the PC incrementer, PC stack, instruction register or interrupt controller. The next address source selector controls which of these four sources are selected based on the current instruction word and the processor status. A fifth possible source for the next program memory address is provided by DAG2 when a register indirect jump is executed.

The program counter (PC) is a 14-bit register which contains the address of the currently executing instruction. The PC output goes to the incrementer. The incremented output is selected as the next program memory address if program flow is sequential. The PC value is pushed onto the 16×14 PC stack when a CALL instruction is executed or when an interrupt is processed. The PC stack is popped when the return from a subroutine or interrupt is executed. The PC stack is also used in zero overhead looping.

The program sequencer section contains six status registers. These are the Arithmetic Status register (ASTAT), the Stack Status register (SSTAT), the Mode Status register (MSTAT), the Interrupt Control register (ICNTL), the Interrupt Mask register (IMASK) and the Interrupt Force and Clear register (IFC).

Interrupts

The interrupt controller allows the processor to respond to the six possible interrupts with a minimum of overhead. Individual interrupt requests are logically ANDed with the bits in IMASK; the highest priority unmasked interrupt is then selected.

The interrupt control register, ICNTL, allows each interrupt to be set as either edge or level sensitive. Depending on a bit in ICNTL, interrupt routines can either be nested with higher priority interrupts taking precedence or processed sequentially with only one interrupt service active at a time.

The 12-bit interrupt force and clear register, IFC, contains a force bit and a clear bit for each of the six possible interrupts.

When responding to an interrupt, the status registers ASTAT, MSTAT, IMASK are pushed onto the status stack and the PC counter is loaded with the appropriate vector address. The status stack is seven levels deep to allow interrupt nesting. The stack is automatically popped when a return from the interrupt is executed.

The vector addresses for each interrupt are fixed. In the ADSP-2101 each vector location identifies a block of four instructions. Short service routines can be executed without an additional JUMP, minimizing overhead.

IMASK

IMASK is six bits wide and allows the interrupt inputs to be individually enabled or disabled. The bits in IMASK are:

0 Timer interrupt enable
1 $\overline{IRQ0}$ or SPORT1 receive interrupt enable
2 $\overline{IRQ1}$ or SPORT1 transmit interrupt enable
3 SPORT0 receive interrupt enable
4 SPORT0 transmit interrupt enable
5 $\overline{IRQ2}$ interrupt enable.

The bits are all positive sense (0 = disabled, 1 = enabled). IMASK is set to zero upon a processor reset so that all interrupts are disabled initially.

ICNTL

ICNTL is a 5-bit register configuring the interrupt modes of the processor. The bits in ICNTL are:

0 $\overline{IRQ0}$ or SPORT1 receive sensitivity
1 $\overline{IRQ1}$ or SPORT1 transmit sensitivity
2 $\overline{IRQ2}$ sensitivity
3 Zero
4 Interrupt Nesting Mode.

The sensitivity bits determine whether a given interrupt input is edge- or level-sensitive (0 = level-sensitive, 1 = edge-sensitive).

The interrupt nesting mode determines whether nesting of interrupt service routines is allowed. When set to zero, all IMASK bits are automatically set to zero when an interrupt service routine is entered. Previous IMASK values are pushed on the stack. When set to one, IMASK is set so that equal and lower priority interrupts are masked, permitting higher priority interrupts to interrupt the current interrupt service routine.

Edge-triggered interrupts are automatically cleared when the interrupt service routine is called. They can also be cleared by writing a one to the appropriate IFC bit.

The timer and serial port interrupts act as edge-sensitive interrupts which can be masked, cleared or forced with software. If you force a level-sensitive interrupt in software, it is automatically cleared. For proper operation, the SPORT1 sensitivity bits must be set to edge-sensitive.

IFC

The IFC register is twelve bits wide and contains a bit for clearing and a bit for forcing each of the six possible interrupts in the ADSP-2101. The bits in IFC are defined as follows.

Bit 0 Timer interrupt clear
Bit 1 SPORT1 receive or $\overline{IRQ0}$ interrupt clear
Bit 2 SPORT1 transmit or $\overline{IRQ1}$ interrupt clear
Bit 3 SPORT0 receive interrupt clear
Bit 4 SPORT0 transmit interrupt clear
Bit 5 $\overline{IRQ2}$ interrupt clear

Bit 6 Timer interrupt force
Bit 7 SPORT1 receive or $\overline{IRQ0}$ interrupt force
Bit 8 SPORT1 transmit or $\overline{IRQ1}$ interrupt force
Bit 9 SPORT0 receive interrupt force
Bit 10 SPORT0 transmit interrupt force
Bit 11 $\overline{IRQ2}$ interrupt force.

Pending edge-sensitive interrupts can be cleared by writing a one to the appropriate clear Bit (0-5) in IFC. Edge-triggered interrupts are normally cleared automatically when the corresponding interrupt service routine is called.

Interrupts can be forced under program control by writing a one to the force Bit (6–11) corresponding to the desired interrupt. This causes the interrupt to be serviced once, unless masked. The timer and SPORT interrupts behave like edge-sensitive interrupts and can be masked, cleared and forced.

Loop Mechanisms

The DO UNTIL instruction executes a zero overhead loop using the loop stack and the loop comparator. For a DO UNTIL instruction, a 14-bit termination address and a 4-bit termination condition are pushed onto the 18-bit loop stack. The address of the next instruction (which identifies the top of the loop) is pushed onto the PC stack. The loop comparator continuously

compares the current PC value against the termination address on the top of the loop stack. When the termination address is detected, the processor checks if the termination condition is met. If the termination condition is not met, then the top of the PC stack is used as the next PC address, returning program flow to the beginning of the loop. If the termination condition is met, then the PC stack is popped, the current PC is incremented by one and program flow falls out of the loop. The loop stack is four levels deep, permitting four levels of zero overhead loop nesting.

The down counter and the count stack also support this powerful looping mechanism. The down counter is a 14-bit register with auto decrement capability. It is loaded from the DMD bus with the loop count. The count is decremented every time the counter value is checked; when the count expires, the counter expired (CE) flag is set. The count stack allows the nesting of loops by storing temporarily dormant loop counts. When a new value is loaded into the counter from the DMD bus, the current counter value is automatically pushed onto the count stack, as program flow enters a loop. The count stack is automatically popped whenever the CE flag is tested and is true, thereby resuming execution of the code outside the loop.

Status Registers

The ADSP-2101 maintains six status registers, which can be accessed over the DMD bus (one is read-only and one is write-only, however). These registers are:

ASTAT	Arithmetic Status Register	
SSTAT	Stack Status Register	*(Read-Only)*
MSTAT	Mode Status Register	
ICNTL	Interrupt Control Register	
IMASK	Interrupt Mask Register	
IFC	Interrupt Force and Clear.	*(Write-Only)*

The interrupt registers are described in a previous section; the other three are discussed below.

ASTAT

ASTAT is 8 bits wide and holds the status information generated by the computational sections of the processor. The bits in ASTAT are defined as follows:

0	AZ	(ALU Result Zero)
1	AN	(ALU Result Negative)
2	AV	(ALU Overflow)
3	AC	(ALU Carry)
4	AS	(ALU X Input Sign)
5	AQ	(ALU Quotient Flag)
6	MV	(MAC Overflow)
7	SS	(Shifter Input Sign).

The bits are positive sense (1 = true, 0 = false). They are automatically updated when a new status is generated by the arithmetic operations affecting them, as defined by the following table:

Status Bit	*Updated On*
AZ, AN, AV, AC	Any ALU operation except division
AS	ALU absolute value operation
AQ	ALU divide operations
MV	Any MAC operation except saturate MR
SS	Shifter exponent detect operation.

SSTAT

SSTAT is 8 bits wide and holds the status of the four internal stacks. The bits in SSTAT are:

0	PC Stack Empty
1	PC Stack Overflow
2	Count Stack Empty
3	Count Stack Overflow
4	Status Stack Empty
5	Status Stack Overflow
6	Loop Stack Empty
7	Loop Stack Overflow.

All of the bits are positive sense (1 = true, 0 = false). The empty status bits indicate that the stack is empty. The overflow status bits indicate that the stack has overflowed. Since the stack overflow status bits "stick" once they are set, subsequent pop operations have no effect on them. This means that the stack can be both overflowed and empty under certain circumstances. A processor reset or a software reboot must be executed to clear the stack overflow status.

MSTAT

MSTAT is a 7-bit register that defines various operating modes of the processor. The mode control instruction enables or disables the operating modes. The bits in MSTAT are:

0	Data Register Bank Select
1	Bit-Reverse Mode (DAG1 Only)
2	ALU Overflow Latch Mode
3	AR Saturation Mode
4	MAC Result P Placement Mode
5	Timer Enable
6	Go Mode.

The data register bank select bit determines which set of data registers is currently active (0 = primary, 1 = secondary). The data registers include all of the result and input registers to the ALU, MAC and shifter (AX0, AX1, AY0, AY1, AF, AR, MX0, MX1, MY0, MY1, MF, MR0, MR1, MR2, SB, SE, SI, SR0 and SR1). At $\overline{\text{RESET}}$, the data register bank select bit is cleared.

The bit-reverse mode, when enabled, bit-wise reverses all addresses generated by DAG1. This is most useful for reordering the input or output data in a radix-2 FFT algorithm.

The ALU overflow latch mode causes the AV (ALU overflow) status bit to "stick" once it is set. In this mode, when an ALU overflow occurs, AV will be set and remain set, even if subsequent ALU operations do not generate overflows. AV can then only be cleared by writing a zero into it from the DMD bus.

The AR saturation mode, when set, causes ALU results to be saturated to the maximum positive (H#7FFF) or negative (H#8000) values when an ALU overflow or underflow occurs.

The MAC Result P Placement bit, when set to 0, results in the ADSP-2100 result placement of the multiplier product in the MR register (one bit shift). When this bit is 1, no shift occurs.

The Timer Enable bit, when set to 1, enables the timer decrement mechanism.

The Go Mode bit, when set to 1, allows the processor to continue operations internally (when possible) while the external address and data buses are tristated during a bus grant.

CONDITION CODES

The condition codes are used to determine whether a conditional instruction, such as a jump, trap, call, return, MAC saturation or arithmetic operation, is performed. The 16 basic composite status conditions and their derivations are shown in Table II. Since arithmetic status is latched into ASTAT at the end of a processor cycle, the condition logic represents conditions generated on the previous cycle.

Code	*Status Condition*	*True If*
EQ	ALU Equal Zero	AZ = 1
NE	ALU Not Equal Zero	AZ = 0
LT	ALU Less Than Zero	AN .XOR. AV = 1
GE	ALU Greater Than or Equal Zero	AN .XOR. AV = 0
LE	ALU Less Than or Equal Zero	(AN .XOR. AV) .OR. AZ = 1
GT	ALU Greater Than Zero	(AN .XOR. AV) .OR. AZ = 0
AC	ALU Carry	AC = 1
NOT AC	Not ALU Carry	AC = 0
AV	ALU Overflow	AV = 1
NOT AV	Not ALU Overflow	AV = 0
MV	MAC Overflow	MV = 1
NOT MV	Not MAC Overflow	MV = 0
NEG	ALU X Input Sign Negative	AS = 1
POS	ALU X Input Sign Positive	AS = 0
NOT CE	Not Counter Expired	CE = 0
FOREVER	Always	Always True

Table II. Condition Codes

In addition to the basic 16 conditions, the JUMP and CALL instructions also support the use of the FI pin as a conditional flag. This pin is one of the five used for serial port 1. It is available if serial port 1 is not configured.

FLAG_IN	FI Pin Last Sampled 1
NOT FLAG_IN	FI Pin Last Sampled 0

Table III. Additional Condition Codes For JUMP and CALL

Timer

A programmable interval timer can generate periodic interrupts. When the decrementing mechanism is enabled, a 16-bit count register (TCOUNT) is decremented every *n* cycles, where *n–1* is a scaling value stored in an 8-bit register (TSCALE). When the value of the count register reaches zero, an interrupt is generated and the count register is reloaded from a 16-bit period register (TPERIOD). Timer interrupts can be masked, cleared and forced in software if desired.

The ADSP-2101 8-bit prescaler allows periodic interrupts over a wide range of possible times. In a processor with an 80ns cycle time, for example, the timer interrupt could occur as infrequently as every 1.34 seconds if a maximum scaling value is used. With a minimum scaling value a maximum period of 5.24ms can be timed.

SERIAL PORTS

The ADSP-2101 incorporates two complete serial ports (SPORT0 and SPORT1) for serial communications and multiprocessor coordination.

Each serial port has a 5-pin interface consisting of the following signals.

Signal Name	*Function*
SCLK	Serial Clock I/O
RFS	Receive Frame Synch I/O
TFS	Transmit Frame Synch I/O
DR	Serial Data Receive
DT	Serial Data Transmit.

Here is a brief list of the capabilities of the ADSP-2101 SPORTs. Figure 7 shows a simplified block diagram of a single SPORT.

- Bidirectional: each SPORT has a separate transmit and receive section.
- Double buffered: each SPORT section (both receive and transmit) has a data register accessible to the user and an internal transfer register. The double buffering provides additional time to service the SPORT.
- Flexible clocking: each SPORT can use an external serial clock (up to the full processor cycle rate) or generate its own (from 94Hz up to one half the processor cycle rate).
- Flexible framing: each SPORT section (receive and transmit) can run in an unframed mode; with internally generated or externally generated frame synch signals; with active high or inverted frame signals; with either of two pulse widths/timings. Framing for the receive and transmit sections is independent but shares the same serial clock.
- Flexible word length: each SPORT supports serial data word lengths from three to sixteen bits.
- Companding in hardware: each SPORT provides optional A-law and μ-law companding according to CCITT recommendation G.711. Different companding can be used for each SPORT, for example, A-law for SPORT0 and μ-law for SPORT1.
- Flexible interrupt scheme: each SPORT section (receive and transmit) can generate a unique interrupt upon completing a data word transfer or after transferring an entire buffer (see next item).
- Autobuffering with single cycle overhead: using the ADSP-2101 DAGs, each SPORT can receive and/or transmit an entire circular buffer of data with an overhead of only one cycle per data word. Transfers to and from the SPORT and the circular buffer are automatic in this mode and do not require additional programming. An interrupt is generated only when pointer wraparound occurs in the circular buffer.
- Multichannel capability: SPORT0 provides a multichannel interface for selective receipt and transmission of arbitrary data channels from a 24- or 32- word, time division multiplexed, serial bitstream. This is especially useful for T1 or CEPT interfaces or as a network communication scheme for multiple processors.
- Alternate configuration: SPORT1 can be configured as two external interrupt inputs (IRQ0 and IRQ1) and the Flag In and Flag Out signals. The internally generated serial clock may still be used in this configuration.

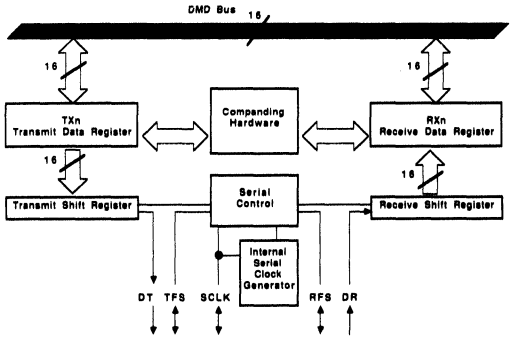


Figure 7. Serial Port Block Diagram

SPORT OPERATION

Each SPORT has a receive and a transmit register; SPORT0's registers are RX0 and TX0; SPORT1's are RX1 and TX1. Companding (a contraction of COMpressing and exPANDing) is the process of logarithmically encoding data to minimize the number of bits that must be sent. Both SPORTs share the companding hardware: one expansion and one compression operation can occur in each processor cycle. In the event of contention, SPORT0 has priority. The ADSP-2101 supports both of the widely used algorithms for companding: A-law and μ-law. The type of companding can be independently selected for each SPORT.

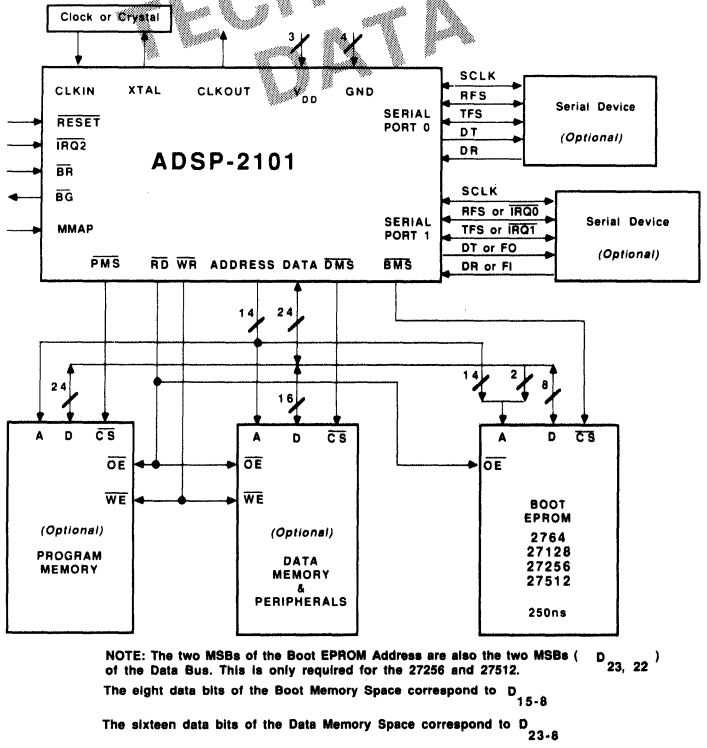
The TXn and RXn registers are identified by name in the ADSP-2101 assembly language, not memory-mapped. TXn and RXn can be read and written (like other non-data registers) with the following instructions: read/write to data memory (direct address), load non-data immediate, and internal (register-to-register) moves. They cannot be accessed by instructions that require indirect addressing, i.e., addresses generated by the DAGs.

There are two ways to generate the SPORT interrupts after the transmission or receipt of (1) each data word or (2) each complete buffer of data words.

Normal (Word by Word) Operation

Writing to the TXn register readies the SPORT for transmission; the TFS signal initiates it. The value in TXn is shifted into the internal transmit register, and after framing synchronization has occurred (if required), the bits are sent, MSB first.

When the first bit has been transferred, the SPORT generates the transmit interrupt. TXn is now available for the next piece of data, even though the transmission of the first is not complete.



NOTE: The two MSBs of the Boot EPROM Address are also the two MSBs ($D_{23, 22}$) of the Data Bus. This is only required for the 27256 and 27512.

The eight data bits of the Boot Memory Space correspond to D_{15-8}

The sixteen data bits of the Data Memory Space correspond to D_{23-8}

Figure 8. ADSP-2101 Basic System Configuration

In the receiving section, bits accumulate as they are received in an internal receive register. When a complete word has been received, it is shifted into the RXn register and the receive interrupt for that SPORT is generated. RXn may then be read.

Autobuffered Operation

In autobuffered operation, the interrupt is not generated until a complete buffer of data words has been received or transmitted. To do this, the user sets up a circular buffer in data memory and identifies the I and M registers in the DAG used to point to this buffer. The SPORT automatically transfers each data word to or from the buffer, stealing a single cycle for each word. (For example, to buffer 16 words of data would require just 16 additional cycles.) When the modulus logic detects buffer wraparound, the SPORT interrupt is generated. Transmitting in autobuffered mode must be started by explicitly writing the first word of the transmit buffer to TXn. The transmission of this word starts the automatic cycling through the transmit buffer.

These serial port features, in conjunction with other features of the ADSP-2101, make it possible to interface to most codecs, A/Ds, DACs and to additional ADSP-2101s with no additional hardware and limited software overhead.

SYSTEM INTERFACE

Figure 8 shows a basic system configuration with the ADSP-2101, two serial codecs, a boot EPROM and optional external program and data memories. Up to 16K words of data memory and 16K words of program memory can be supported. Programmable wait state generation allows the processor to interface easily to slow memories.

The ADSP-2101 also provides one external interrupt and two serial ports or three external interrupts and one serial port.

Clock Signals

The ADSP-2101 takes a TTL-compatible clock signal, CLKIN, running at the instruction rate. Because the ADSP-2101 contains an internal oscillator, an external crystal may be used in place of an external clock oscillator. A clock output (CLKOUT) signal is generated by the processor synchronized to the processor's internal cycles. The rising edge of CLKOUT is aligned with the rising edge of CLKIN. CLKIN may not be halted, changed during operation or operated below the specified frequency.

Bus Interface

The ADSP-2101 can relinquish control of the data and address buses to an external device. When the external device requires access to memory, it asserts the bus request ($\overline{BR}$) signal. After completing the current instruction, the processor halts program execution, tristates the data and address bus, the $\overline{PMS}$, $\overline{DMS}$, $\overline{BMS}$, $\overline{RD}$, $\overline{WR}$ output drivers and asserts the bus grant ($\overline{BG}$) signal. When the $\overline{BR}$ signal is released, the processor releases the $\overline{BG}$ signal, re-enables the output drivers and continues program execution from the point where it stopped.

If the Go mode is set, the processor continues execution (from internal memory) while the bus is granted. In this mode, while $\overline{BG}$ is asserted, the processor only halts if an external memory access is required.

Wait States

The ADSP-2101 can be easily interfaced to slow memories using its programmable wait state generation capability. Three registers control wait state generation for the boot, program and data memory interface. Wait states for boot memory default to 3 cycles at $\overline{RESET}$, while program and data memory each default to 7 cycles. You can specify 0 to 7 wait states for each memory interface.

PROGRAM MEMORY INTERFACE

The program memory address bus (PMA) and the program memory data bus (PMD) are multiplexed with DMA and DMD, sharing the external data and address bus. The 14-bit address bus directly addresses up to 16K words of which 2K is on chip. The data bus is bidirectional and 24 bits wide to external program memory.

There is no placement restriction for instruction code and data in the program memory space, except for the locations used for interrupt and restart vectors.

The program memory data lines are bidirectional. The program memory select ($\overline{PMS}$) signal indicates access to the program memory and can be used as a chip select signal. The write ($\overline{WR}$) signal indicates a write operation and can be used as a write strobe. The read ($\overline{RD}$) signal indicates a read operation and can be used as a read strobe or output enable signal.

Although the processor internal data bus is only 16 bits, the ADSP-2101 can write to the full 24-bit program memory using the PX register.

Program Memory Maps

Program memory can be mapped in two ways, depending on the state of the MMAP pin. Figure 9 shows the two configurations. When MMAP = 0, internal RAM occupies 2K words beginning at address 0000; external program memory uses the remaining 14K words beginning at address H#0800. In this configuration, the boot loading sequence (described below) is automatically initiated when $\overline{RESET}$ is released.

When MMAP = 1, 14K words of external program memory begin at address 0000 and internal RAM is located in the upper 2K words, beginning at address H#3800. In this configuration, program memory is not loaded although it can be written to and read from under program control.

ADSP-2102 ROM Memory

In the ADSP-2102 ROM-based system, both program memory maps are available, selected by the MMAP pin as above. Automatic boot loading is optional.

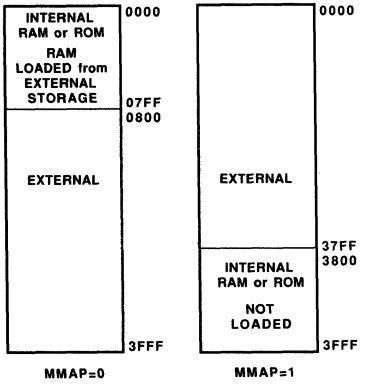


Figure 9. ADSP-2101/ADSP-2102 Program Memory Maps

Boot Memory Interface

The boot memory space consists of an external 64K by 8 space,

divided into eight separate 8K by 8 pages. Three bits in the system control register select which page is loaded by the boot memory interface. Another bit in the system control register allows the user to force a boot loading sequence under software control. The boot loading after $\overline{\text{RESET}}$ is initiated if MMAP = 0.

The boot memory interface defaults to three wait states after $\overline{\text{RESET}}$ and can be set to any value in the range 0 to 7.

The $\overline{\text{BMS}}$ and $\overline{\text{RD}}$ signals are used to select and strobe the boot memory interface. Only 8-bit data is read over the data bus. To accommodate up to eight pages of boot memory, the two MSBs of the data bus are used in the boot memory interface as the two MSBs of the boot space address.

At $\overline{\text{RESET}}$, the ADSP-2101 generates three wait states while booting. This allows a 12.5MHz processor to use a slow (250ns), low-cost EPROM for program storage. Program memory is loaded a byte at a time and converted to 24-bit words.

$\overline{\text{BR}}$ is recognized during the booting sequence. The bus is granted after the completion of loading the current byte. $\overline{\text{BR}}$ during booting may be used to implement booting under the control of a host processor.

The ADSP-210X Assembler and Linker support the creation of programs and data structures requiring multiple boot pages during execution. Table IV shows the state of various processor registers after $\overline{\text{RESET}}$ and after a software forced boot.

2

Control Field	Description	$\overline{\text{RESET}}$	Reboot
Data Registers			
PX	PMD-DMD Bus exchange	undefined	undefined
All others		unchanged	unchanged
Status Registers			
IMASK	Interrupt service enables	0	0
ASTAT	Arithmetic status	undefined	no change
MSTAT	Mode status	0	no change
SSTAT	Stack status	H#55	H#55
Control Registers (Memory-Mapped)			
BWAIT	Boot memory wait states	3	no change
BPAGE	Boot page	0	no change
SPORT 1 configure	Configuration	1	no change
SPE0	SPORT0 enable	0	no change
SPE1	SPORT1 enable	0	no change
DWAIT0–4	Data memory wait states	7	no change
PWAIT	Program memory wait	7	no change
TCOUNT	Timer count register	undefined	no change
TPERIOD	Timer period register	undefined	no change
TSCALE	Timer scale register	undefined	no change
Serial Port Control Registers (Memory-Mapped, One Set Per SPORT)			
ISCLK	Internal serial clock	0	no change
RFSR, TFSR	Frame sync required	0	no change
RFSW, TFSW	Frame sync width	0	no change
IRFS, ITFS	Internal frame sync	0	no change
INVRFS, INVTFS	Invert frame sense	0	no change
SLEN	Serial word length	0	no change
MCE	Multichannel enable	0	no change
MCL	Multichannel length	0	no change
SCLKDIV	Serial clock divide	undefined	no change
RFSDIV	RFS divide	undefined	no change
Multichannel word enable bits		undefined	no change
FO	Flag Out value	undefined	no change
RBUF, TBUF	Autobuffering enable	0	0

Table IV. $\overline{\text{RESET}}$ and Software Boot Machine State

Data Memory Interface
The data memory address bus (DMA) and the data memory data bus (DMD) are multiplexed with PMA and PMD, sharing the external data and address bus. The 14-bit address bus directly addresses up to 16K words of data. The data bus is bidirectional and 16 bits wide.

The data memory select ($\overline{DMS}$) signal indicates access to the data memory and can be used as a chip select signal. The write ($\overline{WR}$) signal indicates a write operation and can be used as a write strobe. The read ($\overline{RD}$) signal indicates a read operation and can be used as a read strobe or output enable signal.

The ADSP-2101 supports memory-mapped I/O, with the peripherals memory mapped into the data memory address space and accessed by the processor in the same manner as data memory.

Data Memory Map
The on-chip data memory RAM resides in the 1K words of data memory beginning at address H#3800, as shown in Figure 10. In addition, data memory locations from H#3C00 to the end of data memory at H#3FFF are reserved. Control registers for the system, timer, wait state configuration and serial port operations are located in this region of memory.

The remaining 14K of data memory is external. External data memory is divided into five zones associated with five different wait states. This allows slower peripherals to be mapped into zones of data memory with more wait states. Figure 10 shows these zones.

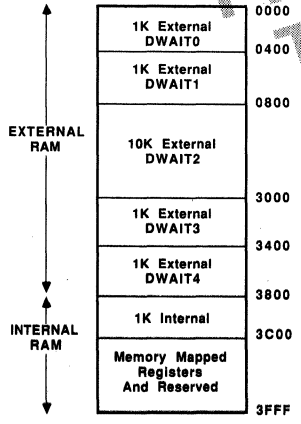


Figure 10. ADSP-2101 Data Memory Map

Interrupt Handling
The ADSP-2101 provides up to three external interrupt input pins, $\overline{IRQ0}$ to $\overline{IRQ2}$. $\overline{IRQ2}$ is always available as a dedicated pin; $\overline{IRQ1}$ and $\overline{IRQ0}$ may be alternately configured as part of Serial Port 1. Each interrupt pin corresponds to a particular interrupt priority level from 2 (highest) to 0 (lowest).

The ADSP-2101 also supports internal interrupts from the timer and the two serial ports. The interrupt levels are internally prioritized and individually maskable. These input pins can be programmed to be either level- or edge-sensitive. The priorities of all six interrupts are shown in Table V.

The ADSP-2101 supports a vectored interrupt scheme: when an interrupt is acknowledged, the processor shifts program control to the interrupt vector address corresponding to the interrupt level. Interrupts can optionally be nested so that a higher priority interrupt can pre-empt the currently executing interrupt service routine. Each interrupt vector location is four instructions in length, so that simple service routines can be coded entirely in this space. Longer routines require an additional JUMP or CALL.

Source of Interrupt	Interrupt Vector
$\overline{IRQ2}$ (external pin)	0004 (*highest priority*)
SPORT0 Transmit (internal)	0008
SPORT0 Receive (internal)	000C
SPORT1 Transmit (internal) or $\overline{IRQ1}$ (external)	0010
SPORT1 Receive (internal) or $\overline{IRQ0}$ (external)	0014
Timer (internal)	0018 (*lowest priority*)

Table V. Interrupts & Interrupt Vector Addresses

$\overline{RESET}$ Signal
The $\overline{RESET}$ signal initiates a master reset of the ADSP-2101. The $\overline{RESET}$ signal must be asserted after the chip is powered up to assure proper initialization. If $\overline{RESET}$ follows initial power-up, it must be held long enough to allow the internal clock to stabilize. If $\overline{RESET}$ is activated subsequently, the clock continues and does not require this stabilization time. The master reset performs the following:

1. Initialize internal clock circuitry, if necessary
2. Reset all internal stack pointers to empty stack condition
3. Mask all interrupts
4. Clear MSTAT register
5. When $\overline{RESET}$ is released, if there is no pending bus request, execute the boot-loading sequence (if configured)
6. Drive PMA with the restart vector, H#0000.

Interprocessor Communication
The serial ports provide a way to bidirectionally link two ADSP-2101s in a system with no additional hardware required. Figure 11 shows a typical system configuration with two ADSP-2101 processors.

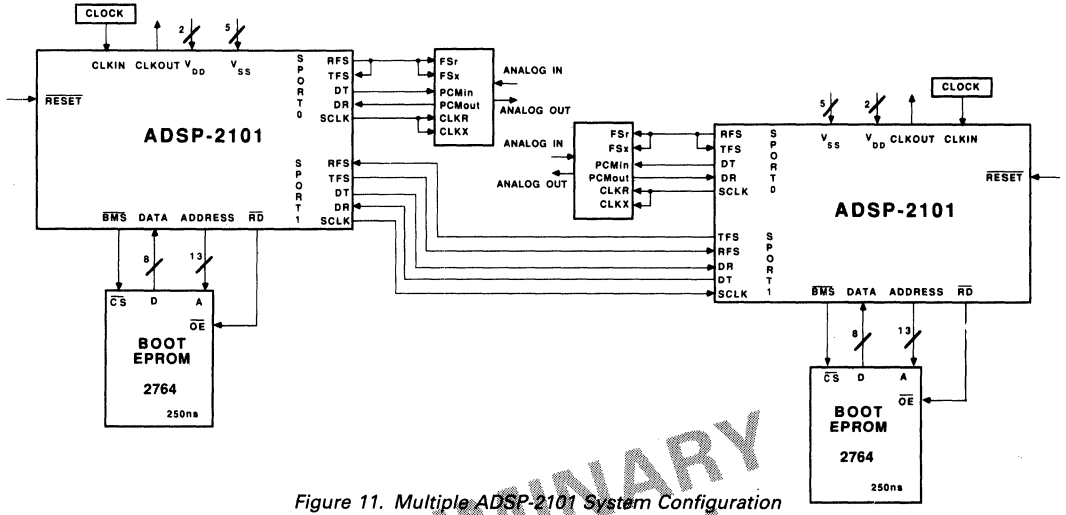


Figure 11. Multiple ADSP-2101 System Configuration

INSTRUCTION SET DESCRIPTION

The ADSP-2101 assembly language, like the ADSP-2100's, uses an algebraic syntax for ease of coding and readability. The sources and destinations of computations and data movements are written explicitly in each assembly statement, eliminating cryptic assembler mnemonics. Every instruction assembles into a single 24-bit word and executes in a single cycle. The instructions encompass a wide variety of instruction types along with a high degree of operational parallelism. There are five basic categories of instructions: data move instructions, computational instructions, multifunction instructions, program flow control instructions and miscellaneous instructions. Each of these instruction types is described briefly. The complete instruction set is summarized at the end of this section. The *ADSP-2101 User's Manual* gives an overview and the *ADSP-210X Cross-Software Manual* contains a complete reference to the instruction set.

ADSP-2100 Compatibility

ADSP-2101 source code is a superset of the ADSP-2100 instruction set. The ADSP-2101 is source and object code compatible with the ADSP-2100. An ADSP-2100 program may need to be relocated to utilize internal memory and conform to the new interrupt vector placement.

The TRAP instruction, however, is not supported since the ADSP-2101 does not have the TRAP/$\overline{\text{HALT}}$ signals.

Data Move Instructions

Table VI gives a list of all registers that are accessible using the data move instructions. This set of registers is denoted as *reg* in the instruction set summary given in Table IX at the end of this publication. A subset of the *reg* group associated with the computational units, which generally hold data as opposed to address or status information, are denoted as *dreg*. Memory-mapped control registers are treated as data memory locations, not as registers.

The data move instructions include transfers between internal registers, between data memories and internal registers, between program memories and internal registers, and immediate value loading of registers and data memories. The content of every *reg* can also be loaded to any other *reg*.

Two addressing modes are supported for data memory transfers: direct addressing and indirect addressing. In direct addressing, the memory address is supplied from the instruction word. In indirect addressing, one of the data address generators provides the address. Using direct addressing, the contents of a data memory location can be written and read by any *reg*. Using indirect addressing, the contents of a data memory location can only be written and read by a *dreg*. Immediate data load to data memory is permitted with indirect addressing. Only the indirect addressing mode is supported for program memory data transfers and contents of a program memory location can be read and written to any *dreg*.

AX0, AX1	Data Registers (*dreg*)
AY0, AY1	
AR	
MX0, MX1	
MY0, MY1	
MR0, MR1, MR2	
SI	
SE	
SR0, SR1	
SB	Accessible Registers (*reg*)
PX	
I0, I1, I2, I3, I4, I5, I6, I7	
M0, M1, M2, M3, M4, M5, M6, M7	
L0, L1, L2, L3, L4, L5, L6, L7	
CNTR	
ASTAT	
MSTAT	
SSTAT	
IMASK	
ICNTL	
RX0, TX0	
RX1, TX1	

Table VI. Register Classification

Memory-Mapped Registers

In addition to the registers listed in the table above, the ADSP-2101 provides a set of memory-mapped registers for controlling system features, serial ports and the timer. The table below summarizes these registers.

Memory Location	Register Use
3FFF	System control
3FFE	Data memory wait state control register
3FFD	Timer period
3FFC	Timer count
3FFB	Timer scaling factor
3FFA & 3FF9	SPORT0 multichannel receive word enables
3FF8 & 3FF7	SPORT0 multichannel transmit word enables
3FF6	SPORT0 control
3FF5	SPORT0 serial clock divide modulus
3FF4	SPORT0 receive frame sync divide modulus
3FF3	SPORT0 autobuffer control
3FF2	SPORT1 control
3FF1	SPORT1 serial clock divide modulus
3FF0	SPORT1 receive frame sync divide modulus
3FEF	SPORT1 autobuffer control

Table VII. Memory-Mapped Registers

Computational Instructions

There are three types of operations associated with the computational units: ALU operations, MAC operations and shifter operations. With few exceptions, all these computational instructions can be made conditional. (The permissible conditions are specified in Table II.) Each computational unit has a set of input registers and output registers. A list of permissible input operands and result registers for each of the units is given in Table VIII.

ALU

Source for X Input (xop)	*Source for Y Input (yop)*	*Destination for Output Port R*
AX0, AX1	AY0, AY1	AR
AR	AF	AF
MR0, MR1, MR2		
SR0, SR1		

MAC

Source for X Input (xop)	*Source for Y Input (yop)*	*Destination for Output Port R*
MX0, MX1	MY0, MY1	MR (MR2, MR1, MR0)
AR	MF	MF
MR0, MR1, MR2		
SR0, SR1		

Shifter

Source for Shifter Input (xop)	*Destination for Shifter Output*
SI	SR (SR1, SR0)
AR	
MR0, MR1, MR2	
SR0, SR1	

Table VIII. Computational Input/Output Registers

Multifunction Instructions

Multifunction instructions execute one computational operation with one or two data moves. All of the multifunction instructions utilize various combinations of the computational and data move operations described above. Since the instruction word is only 24 bits wide, only certain combinations are valid. In general, the following rules are followed.

1. Computation must be unconditional.
2. Any memory transfer must use the indirect addressing mode.
3. Data move operations can only use data registers (*dregs*).

Program Flow Control Instructions

Program flow control instructions include JUMP, CALL, return from subroutine, return from interrupt, DO UNTIL, SET, CLEAR and TOGGLE the FLAG_OUT, and IDLE. All except the IDLE and FLAG_OUT instructions can be made conditional. The JUMP and CALL instructions support both direct addressing, with the destination address specified by the instruction word, and indirect addressing, with the destination address specified by one of the I registers in DAG2. JUMP and CALL also accept the additional condition based on the state of the FI (Flag In) pin with direct addressing.

IDLE puts the processor into a low-power, wait-for-interrupt mode of operation.

Miscellaneous Instructions

Miscellaneous instructions include indirect register modify, stack control, mode control and NOP operations. Mode control allows the user to enable or disable bit-reversal (DAG1), ALU overflow latching, AR register saturation, use of secondary register set, Go mode, MAC format adjust mode and the timer.

Table IX Instruction Set Summary

The following conventions are used in this table:

1. All keywords are shown in capital letters.
2. Brackets enclose optional parts of the syntax.
3. Vertical lines indicate that one parameter must be chosen from those enclosed.
4. Table VI defines the set of registers for *dreg* and *reg*.
5. Table VIII defines the set of registers for *xop* and *yop*.
6. Tables II and III define the conditions for <condition>.
7. <data> represents an immediate value or a pointer to (^) or length of (%) operator used with an identifier.
8. <address> may be an immediate value or label.
9. <comp>, in a multifunction instruction, represents all legal ALU, MAC or shifter operations (the restrictions are detailed in the *ADSP-210X Cross-Software Manual*).

MULTIFUNCTION INSTRUCTIONS

```
| <ALU†>  |, | AX0 | = DM( | I0 |, | M0 |), | AY0 | = PM( | I4 |, | M4 |);
| <MAC>   |  | AX1 |       | I1 |, | M1 |     | AY1 |       | I5 |, | M5 |
                | MX0 |       | I2 |, | M2 |     | MY0 |       | I6 |, | M6 |
                | MX1 |       | I3 |, | M3 |     | MY1 |       | I7 |, | M7 |

| AX0 | = DM( | I0 |, | M0 |), | AY0 | = PM( | I4 |, | M4 |);
| AX1 |       | I1 |, | M1 |     | AY1 |       | I5 |, | M5 |
| MX0 |       | I2 |, | M2 |     | MY0 |       | I6 |, | M6 |
| MX1 |       | I3 |, | M3 |     | MY1 |       | I7 |, | M7 |

| <ALU>     | , dreg = | DM( | I0 |, | M0 | );
| <MAC>     |          |     | I1 |, | M1 |
| <SHIFT†>  |          |     | I2 |, | M2 |
                          |  I3 |, | M3 |
                          |  I4 |, | M4 |
                          |  I5 |, | M5 |
                          |  I6 |, | M6 |
                          |  I7 |, | M7 |
                      PM( | I4 |, | M4 | )
                          |  I5 |, | M5 |
                          |  I6 |, | M6 |
                          |  I7 |, | M7 |

| DM( | I0 |, | M0 | ) | = dreg; | <ALU>  | ;
|     | I1 |, | M1 |   |         | <MAC>
|     | I2 |, | M2 |   |         | <SHIFT>
|     | I3 |, | M3 |   |
|     | I4 |, | M4 |   |
|     | I5 |, | M5 |   |
|     | I6 |, | M6 |   |
|     | I7 |, | M7 |   |
| PM( | I4 |, | M4 | ) |
|     | I5 |, | M5 |   |
|     | I6 |, | M6 |   |
|     | I7 |, | M7 |   |

| <ALU>  | , dreg = dreg;
| <MAC>
| <SHIFT>
```

†All computation is unconditional; ALU division and shift immediate operations prohibited.

ALU INSTRUCTIONS

```
[IF condition]  | AR |  =  xop  | + yop     |  ;
                | AF |          | + C       |
                                 | + yop + C |

[IF condition]  | AR |  =  xop  | - yop     |  ;
                | AF |          | - yop + C - 1 |

[IF condition]  | AR |  =  yop  | - xop     |  ;
                | AF |          | - xop + C - 1 |

[IF condition]  | AR |  =  xop  | AND  yop  |  ;
                | AF |          | OR        |
                                 | XOR       |

[IF condition]  | AR |  =  PASS | xop |        ;
                | AF |          | yop |
                                 | 0   |
                                 | 1   |

[IF condition]  | AR |  =  -    | xop |        ;
                | AF |          | yop |

[IF condition]  | AR |  =  NOT  | xop |        ;
                | AF |          | yop |

[IF condition]  | AR |  =  ABS  xop            ;
                | AF |

[IF condition]  | AR |  =  yop  + 1            ;
                | AF |

[IF condition]  | AR |  =  yop  - 1            ;
                | AF |

DIVS yop, xop ;
DIVQ xop ;
```


MAC INSTRUCTIONS

```
[IF condition]  | MR |  =  xop * yop  (      | SS  |  ) ;
                | MF |                       | SU  |
                                             | US  |
                                             | UU  |
                                             | RND |

[IF condition]  | MR |  =  MR + xop * yop  ( | SS  |  ) ;
                | MF |                       | SU  |
                                             | US  |
                                             | UU  |
                                             | RND |

[IF condition]  | MR |  =  MR - xop * yop  ( | SS  |  ) ;
                | MF |                       | SU  |
                                             | US  |
                                             | UU  |
                                             | RND |

[IF condition]  | MR |  =  0;
                | MF |

[IF condition]  | MR |  =  MR [ (  RND ) ];
                | MF |

IF MV SAT MR;
```

SHIFTER INSTRUCTIONS

```
[IF condition]  SR   = [SR OR] ASHIFT xop   ( HI | LO );
[IF condition]  SR   = [SR OR] LSHIFT xop   ( HI | LO );
[IF condition]  SR   = [SR OR] NORM xop     ( HI | LO );
[IF condition]  SE   = EXP xop              ( HI | LO | HIX );
[IF condition]  SB   = EXPADJ xop;
SR              = [SR OR] ASHIFT xop BY <data>  ( HI | LO );
SR              = [SR OR] LSHIFT xop BY <data>  ( HI | LO );
```

MOVE INSTRUCTIONS

```
reg             = reg;
reg             = DM (<address>) ;
dreg            = DM ( I0 , M0 | I1 , M1 | I2 , M2 | I3 , M3 | I4 , M4 | I5 , M5 | I6 , M6 | I7 , M7 );

DM ( I0 , M0 | I1 , M1 | I2 , M2 | I3 , M3 | I4 , M4 | I5 , M5 | I6 , M6 | I7 , M7 ) = dreg | <data> ;

DM (<address>)  = reg;
reg             = <data>;
dreg            = PM ( I4 , M4 | I5 , M5 | I6 , M6 | I7 , M7 );

PM ( I4 , M4 | I5 , M5 | I6 , M6 | I7 , M7 ) = dreg ;
```

PROGRAM FLOW CONTROL INSTRUCTIONS

```
[IF condition]   JUMP                    | (I4)          |  ;
                                         | (I5)          |
                                         | (I6)          |
                                         | (I7)          |
                                         | <address>     |

[IF condition]   CALL                    | (I4)          |  ;
                                         | (I5)          |
                                         | (I6)          |
                                         | (I7)          |
                                         | <address>     |

IF              | FLAG_IN       |        CALL  <address>  ;
                | NOT FLAG_IN   |

IF              | FLAG_IN]      |        JUMP  <address>  ;
                | NOT FLAG_IN   |

[IF condition]   RTS                                      ;

[IF condition]   RTI                                      ;

DO <address>     [UNTIL termination]                      ;

IDLE ;

| SET    |
| CLEAR  |  FLAG_OUT                                      ;
| TOGGLE |
```

MISCELLANEOUS INSTRUCTIONS

```
NOP;

        | PUSH | STS [, POP CNTR] [, POP PC] [, POP LOOP]      ;
        | POP  |

| ENA | | BIT_REV   |  [ , . . . ]                          ;
| DIS | | AV_LATCH  |
        | AR_SAT    |
        | SEC_REG   |
        | TIMER     |
        | GO_MODE   |

MODIFY ( | I0 | , | M0 | );
         | I1 | , | M1 |
         | I2 | , | M2 |
         | I3 | , | M3 |
         | I4 | , | M4 |
         | I5 | , | M5 |
         | I6 | , | M6 |
         | I7 | , | M7 |
```

Table IX. Instruction Set Summary

Microcoded Support Components
Contents

GENERAL INFORMATION

Support for microcoded systems is provided by the Word-Slice® family of microcoded building blocks. The current members of the Word-Slice family of microcode components include the ADSP-1401 and ADSP-1402 Program Sequencers, the ADSP-1410 Address Generator and the ADSP-3128A Flexible Register File. The program sequencers and address generator feature the Look-Ahead™ pipeline which eliminates the need for external microcode pipeline registers by internally latching instructions and addresses. The ADSP-3128A Register File has five ports and fast access times to maximize computational throughput of microcoded systems using high speed floating-point components. The ADSP-3128A also provides flexible shared memory for multiprocessing systems using the ADSP-2100 family of microprocessors. The ADSP-1401 and ADSP-1410 are fabricated in a fast 1.5μm CMOS process. The ADSP-1402 and ADSP-3128A are fabricated in a 1μm CMOS. These components improve performance, reduce board space and ease development compared to bit-slice and byte-slice solutions.

ADSP-1401 and ADSP-1402 PROGRAM SEQUENCERS

The ADSP-1401 and ADSP-1402 are 16-bit microprogram sequencers with many high performance features such as single-cycle branching. This makes them ideal for the demanding sequencing tasks found in digital signal processors and high speed, general purpose computers. In addition to high speed, these sequencers feature on-chip storage and control of ten prioritized and maskable interrupts; four decrementing event counters; absolute, relative and indirect addressing capability; and a dynamically configurable 64-word RAM. The ADSP-1402 is fully code-compatible with the ADSP-1401 but offers higher speed and more I/O pins for interrupts, traps and reset than the ADSP-1401. The ADSP-1402 can support 20MIPS (million instructions per second) operation. The ADSP-1401 can be used in 11.1MIPS systems.

ADSP-1410 ADDRESS GENERATOR

The ADSP-1410 is a fast, flexible address generator that rapidly generates the data memory addresses required by operations such as digital filters, FFTs, matrix operations and DMAs. The ADSP-1410 features a 16-bit ALU, a comparator and thirty 16-bit registers. In a single cycle the ADSP-1410 can output a 16-bit memory address, modify this address and detect when the value has crossed a preset boundary and conditionally loop back to the top of a circular buffer.

A 255-page *Word-Slice User's Manual* covers all aspects of programming with the ADSP-1401, ADSP-1402 and ADSP-1410. Third-party support is available in the form of meta-assemblers, development systems and behavioral models. Contact Analog Devices for further information on third-party support.

ADSP-3128A MULTIPORT REGISTER FILE

This 128×16 or 64×32 register file provides high speed local storage for our floating-point components and microprocessors while also providing flexibility in operand data transfers with its five-port structure. The register file is fast enough to provide full computational throughput rates for our latest 1.0μm floating-point parts. The ADSP-3128A contains on-chip latches for a multitude of system interfacing configurations without the need for external glue logic. On-chip multiplexers automatically sequence double-precision data transfers.

Selection Guide

ADSP-1410 ADDRESS GENERATOR

Grade		Data Memory Address Size Single-Precision	Data Memory Address Size Double-Precision	Clock-to-Address Valid Delay[1]	Minimum Cycle Time	# of Address Registers	Total # of Registers	I_{DD}[2]	Package Options[3]	Process	Logic Type
Commercial	J	16 Bits / 64K Words	30 Bits / 1 Gigaword	35	100ns	16	30	75mA	D, N, P	CMOS	TTL
	K	16 Bits / 64K Words	30 Bits / 1 Gigaword	30	90ns	16	30	75mA	D, N, P	CMOS	TTL
Military	S	16 Bits / 64K Words	30 Bits / 1 Gigaword	45	125ns	16	30	100mA	D	CMOS	TTL
	T	16 Bits / 64K Words	30 Bits / 1 Gigaword	35	100ns	16	30	100mA	D	CMOS	TTL

NOTES
[1]ns, max @ +70°C commercial, +125°C MIL.
[2]mA maximum, f_{CLK}=max, over full V_{DD} range, @ +70°C commercial,+125°C MIL.
[3]D=ceramic 48-pin DIP, N=plastic 48-pin DIP, P=52-contact PLCC.

ADSP-1401/ADSP-1402 PROGRAM SEQUENCERS

Model	Program Address Size	Clock-to-Address Valid Delay[1] Comm	Clock-to-Address Valid Delay[1] MIL	Minimum Cycle Time Comm	Minimum Cycle Time MIL	Number of Interrupts	I_{DD}[2] Comm	I_{DD}[2] MIL	No. of Pins	Package Options[3]	Process	Logic Type
ADSP-1402	16 Bits / 64K Words	K=17[4]	(Note 5)	K=50ns	(Note 5)	10	(Note 5)	(Note 5)	84	G	CMOS	TTL
ADSP-1401	16 Bits / 64K Words	J=35 K=25	S=45 T=35	J=90ns K=70ns	S=110ns T=90ns	10	75	100	48 52	D, N P	CMOS	TTL

NOTES
[1]ns, max @ +70°C commercial, +125°C MIL.
[2]mA max, f_{CLK}=max, over full V_{DD} range, @ +70°C commercial, +125°C MIL.
[3]D=ceramic 48-pin DIP, N=plastic 48-pin DIP, P=52-contact PLCC.
[4]Preliminary specification.
[5]Contact factory.

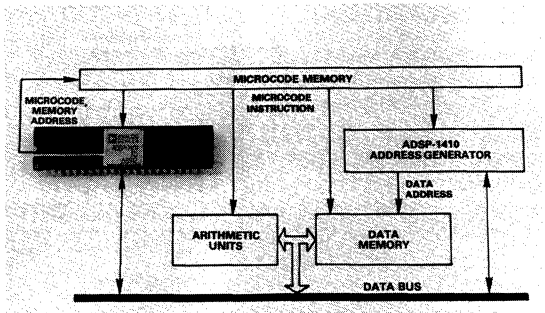
ANALOG DEVICES

Word-Slice Program Sequencer

ADSP-1401

FEATURES

16-Bit Microcode Addressing Capability
Look-Ahead™ Pipeline
Extensive Interrupt Processing, With Ten On-Chip Interrupt Vectors
70ns Cycle Time; 25ns Clock-to-Address Delay
64-Word RAM for Storing:
Subroutine Linkage
Jump Addresses
Counters
Status Register
375mW Maximum Power Dissipation with CMOS Technology
48-Pin Ceramic or Plastic DIP and 52-Lead Plastic Leaded Chip Carrier



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1401

3

GENERAL DESCRIPTION

The ADSP-1401 is a high-speed microprogram controller optimized for the demanding sequencing tasks found in digital signal processors and general purpose computers. In addition to high speed (25ns clock-to-address delay) and large addressing range (64K of program memory), this Word-Slice® component has unique features that make it highly versatile:

- on-chip storage and control of ten prioritized and maskable interrupts
- four decrementing event counters
- absolute, relative and indirect addressing capability
- download capability (writeable control store) and
- a dynamically configurable 64-word RAM.

The ADSP-1401 microprogram sequencer's main task is to provide the appropriate microprogram addressing to support programming requirements (e.g., looping, jumping, branching, subroutines, condition testing and interrupts). An internal Look-Ahead pipeline, controlled by both phases of the clock, allows the ADSP-1401 to satisfy these requirements at very high speed.

During each micro-instruction, the ADSP-1401 monitors the conditions and instructions to determine the next microprogram address. This address can come from one of several sources: the stack, the jump address space in the RAM, the data port, the interrupt vectors, or the microprogram counter. An extensive set of conditional instructions are also available, including jumps, branches, subroutines, interrupts, and writeable control store.

The ADSP-1401's internal 64-word RAM is user-configurable into three regions; subroutine stack, register stack and indirect jump address space. The subroutine stack is used for linking interrupts and subroutines and, during their execution, allow storage of system states. The register stack allows association of unique jump addresses with various levels of interrupts and subroutines (both local and global stacks are provided). Indirect jump capability is also supported, addressing for which is provided at the data port.

Interrupts are handled entirely on chip. The ADSP-1401's internal interrupt control logic includes registers for eight external (user) interrupt vectors, a mask register, and a priority decoder. Two additional vectors are reserved for internally-generated interrupts resulting from counter underflow and stack limit violation. A stack limit violation is caused by stack overflow, underflow or collision. A mechanism is provided for recovering from stack violations.

The ADSP-1401's four decrementing 16-bit counters are used to track loops and events. These counters generate a signal when negative. This negative condition is used by several conditional instructions and can also trigger an internal interrupt.

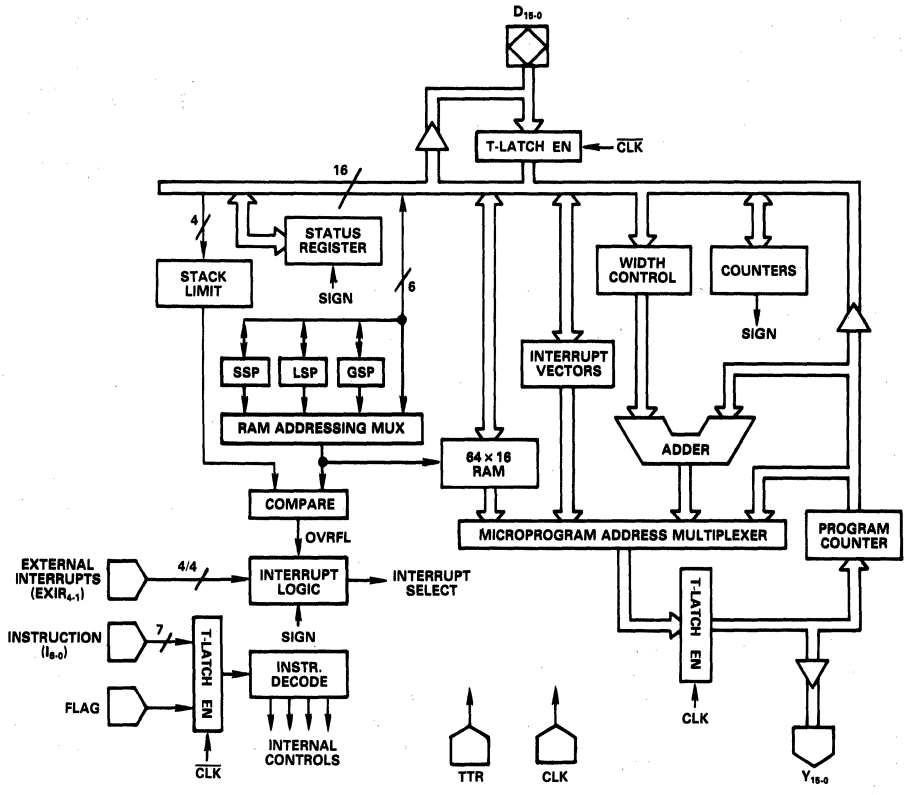


Figure 1. ADSP-1401 Block Diagram

ADDRESSING MODES

Direct: both absolute and relative
Indirect: from internal RAM

HARDWARE FEATURES

Instruction Port
Bidirectional Data Port
Four Input Address Multiplexer
Three Stack Pointers
Four Event Counters
Condition Flag
Eight Prioritized and Maskable User Interrupts
TTR Pin:
- Trap
- Three-State
- Reset

INSTRUCTION TYPES

Jumps and Branches
Stack Operations
Status Register Operations
Counter Operations
Interrupt Control
Relative Address Width Controls
Instruction Hold Control
Writeable Control Store
Dedicated Counter Underflow Interrupt
Dedicated Stack Overflow Interrupt

ADSP-1401 PIN ASSIGNMENTS

Pin Name	Description
I_6-I_0	The 7-bit microinstruction controlling the ADSP-1401.
$Y_{15}-Y_0$	Output bus which provides addresses to the microprogram memory.
$D_{15}-D_0$	Bidirectional Data bus for transferring data to or from the ADSP-1401.
$EXIR_{4-1}$	Four external interrupt request lines. Note that internal circuitry supports 8 interrupts with the aid of an external 2 to 1 multiplexer.
CLK	External clock input
FLAG	An input used for conditional instructions. Its source is usually a condition multiplexer.
TTR	A multi-purpose pin accommodating traps, output disable and reset.
V_{DD}	+5 Volt supply.
GND	Ground.

1.0 ARCHITECTURE

1.1 Look-Ahead Pipeline

Logically, the Look-Ahead pipeline is split into two halves: the first, located at the instruction and data ports; and the second, located at the address port. Each half of the pipeline (input vs. output) has a transparent latch which operates out of phase with the other; the address latch is transparent during the first half of the cycle (clock HI), while the input latches (instruction and data) are transparent during the second half of the cycle (clock LO). This complementary arrangement allows new instructions to be decoded (in preparation for the following cycle) while the program address for the current cycle is held steady.

1.2 Instruction Port

The instruction port receives 7-bit instructions defining the next operation to perform from microcode. The ADSP-1401 has a built-in Look-Ahead pipeline latch, eliminating the need for an external microcode latch to hold instructions. This implementation has the further benefit of allowing instruction "look-ahead"; the sequencer is able to decode the next instruction during execution of the current cycle. During the "look-ahead" period, the sequencer precalculates the next address, allowing its output as early as possible in the next cycle.

External instructions are internally latched during clock HI, and passed directly to the instruction decoder during clock LO (transparent phase); thus, implementing the first half of the Look-Ahead pipeline latch.

The use of the instruction hold mode (see: Instruction Set Description, 2.7; and Instruction Hold Control, appendix 4.1) allows an instruction to be held in the instruction latch for execution over several cycles (freeing microcode for use by other devices).

1.3 Address Port and Multiplexer Sources

The address port provides 16-bit program addresses with three-state drivers designed for driving large microcode memories. Addresses come from a four-to-one microprogram address multiplexer. Between the multiplexer and output port is a transparent latch which is transparent during clock HI and latched during clock LO, permitting addresses to be output as early as possible during phase one (clock HI) while holding the address constant during phase two (clock LO) – implementing the second half of the Look-Ahead pipeline latch.

Inputs to the microprogram address multiplexer are the:

- 16-Bit Program Counter
- 16-Bit Adder
- Interrupt Vector File and
- Internal 64-Word RAM.

Addressing Modes

The ADSP-1401 supports two addressing modes: direct and indirect. The direct addressing mode uses the internal adder to generate either absolute addresses from the data port (without modification) or relative addresses from the program counter (with or without extension: see Status Register, 1.4.4). The indirect addressing mode uses the lower order bits at the data port to access the contents of internal RAM for output.

Output Drivers

The address port output drivers are always active unless placed in the high-impedance state by the IDLE instruction or appropriately asserting the TTR pin (see TTR Pin, 1.7). This allows other devices to supply microcode addresses, which is particularly useful in multi-tasking or context switching applications where several ADSP-1401s may be sharing common microcode memory.

1.3.1 Program Counter

The program counter (PC) consists of a 16-bit incrementing counter. For most instructions, the PC is incremented by the end of the cycle (post-increment) as follows:

$$PC <= \text{output address} + 1.$$

1.3.2 Adder and Width Control

For absolute jumps, data from the data port is passed unchanged through the adder directly to the microprogram address port. For relative jumps, a twos complement offset is supplied from the data port and added with the 16-bit PC. Since the PC normally points to the next instruction, the jump distance is (offset + 1) from the jump instruction. See Status Register (1.4.4) for more details.

The width control block permits microcode width to be reduced in systems not requiring full, 16-bit jump distances. Internal width control logic sign-extends reduced offsets of 8- and 12-bits to full 16-bit precision, accommodating jumps in either direction (positive or negative displacement).

1.3.3 Interrupt Vector File

Ten prioritized interrupt vectors may be stored in the interrupt vector file. The associated interrupts are internally latched and may be individually masked or entirely disabled by the "Disable Interrupts" (DISIR) instruction. The highest priority interrupt vector displaces the usual address on the next cycle following its detection. See Interrupts (1.4.3) for more details.

1.3.4 Internal RAM

Any of the 64 words of RAM may be output on the address port. Four distinct address sources may access the RAM:

- Local Stack Pointer
- Global Stack Pointer
- Subroutine Stack Pointer and
- Lower Order Data Port Bits.

The use of internal RAM and its various address sources are described in section 1.4.2.

1.4 Bidirectional Data Port

The 16-bit bidirectional data port (D_{15-0}) supplies direct or indirect jump addresses and permits loading or dumping of all internal registers. The input data latch freezes incoming data (for counter or register writes executed during that cycle) during the first half-cycle (clock HI) and is transparent for the remainder of the cycle. The output data driver asserts output data *only* during the first half-cycle of a data output instruction and is independent of the address port drivers. This complementary I/O arrangement permits data to be output from the sequencer (as in a read register instruction) during the first half-cycle while accommodating external data setups (for the next cycle) during the second half-cycle.

Direct addressing via the data port may be either relative or absolute. For indirect addressing, the six LS data bits (D_{5-0}) are used to address internal RAM, containing the desired jump address (see Internal RAM, 1.4.2).

1.4.1 Counters

Four independent 16-bit counters are provided for maintaining loops and event tracking. These counters hold twos complement values that may be decremented or preloaded through dedicated instructions. The sign bit associated with the most recently used counter, prior to its decrement, is always saved in the status register (SR_1). Simultaneously, the sign bit is also made available to control various conditional instructions or for asserting the lowest priority interrupt, IR_0, reserved for counter underflow (see: Instruction Set Description, 2.0; and Interrupts, 1.4.3).

Note that interrupt IR_0 is primarily used for ending writeable control store downloads (see Instruction Set Description – WCS, 2.7). Use of IR_0 in the context of a "Decrement Counter and Interrupt on Underflow" operation represents the worst case instruction and flag setup times because of the additional overhead in processing the interrupt after determining whether the counter was underflowed. These setup times are specified two ways:

1. all conditions and
2. IR_O masked.

The source of SIGN (applied to the condition test) depends upon the type of instruction used (see Instruction Set Description, 2.1). Two possibilities exist:

1. If an *explicit* counter is selected, then the sign applied is that of the counter, prior to the decrement.
2. If *no counter* is selected, then the sign applied is implicitly that of the *status register*, SR_1.

1.4.2 Internal RAM

The ADSP-1401's internal 64-word RAM implements two distinct stacks: a Subroutine Stack (SS) and a Register Stack (RS). The subroutine stack has a dedicated, Subroutine Stack Pointer (SSP), while the register stack shares two pointers: the Local Stack Pointer (LSP) and the Global Stack Pointer (GSP). The three stack pointers are each held in 6-bit, preloadable, up/down counters.

Upon reset, (TTR pin held HI for three cycles, see TTR Pin, 1.7) the SSP is initialized to 0 (top of RAM). The RS pointers (LSP and GSP) are typically configured as shown in Figure 2 using the "Write RSP" instruction (WRRSP). The SSP pushes down while the RS pointers push up. Selection of the active RS pointer (LSP or GSP) is made in the status register.

Stack overflow detection is provided via a stack limit register to protect software integrity and allow stack expansion (see Instruction Set Description – SLRIVP, 2.5).

Each RS pointer may be explicitly initialized by performing the "Write RS Pointer" (WRRSP) instruction. The LSP should be located above the GSP, allowing the local stack to grow upwards as the level of nested subroutines increases. Finally, indirect jump address space (as needed) should be reserved below the global stack.

The sequencer will generate a stack underflow interrupt whenever RAM location zero is popped. This facility may be used in support of stack paging. IV_9 should be masked if not using stack paging, allowing location zero to be used as the first stack location without interrupting. When using paged stacking, location zero must be reserved as an underflow buffer to avoid a subsequent stack POP (which may otherwise occur, depending upon the next instruction) prior to the interrupt routine saving the stack.

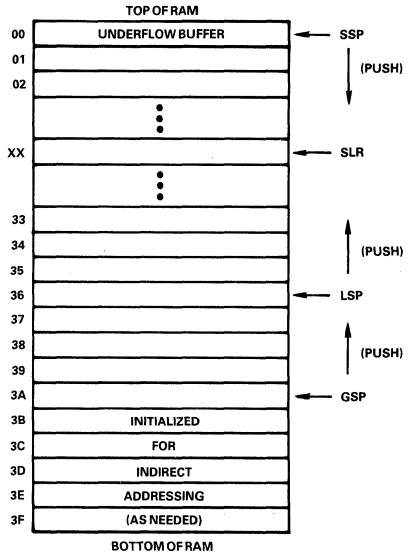


Figure 2. Typical RAM Initialization

Register Stack Pointers (LSP and GSP)

Upon entering a routine, up to four jump addresses may be pushed onto the register stack. A Push onto the register stack first decrements the RS pointer (either LSP or GSP, depending upon the status register) and then writes the appropriate data to RAM. A Pop from the register stack first reads the RAM location and then increments the RS pointer (LSP or GSP).

Four registers are available within context of any routine which are addressed relative to the stack pointer (LSP or GSP) by the two LSBs of the relevant instruction. For example, the instruction:

IF CONDITION, JMP R_2

accesses the location (LSP+2 or GSP+2) in RAM as the conditional address source. Prior to exiting a routine, local or global registers can be effectively removed from the RS by the "ADD i TO RSP" (AIRSP) instruction (see Instruction Set Description, 2.2).

Often, the same set of jump addresses are used by several different routines. The GSP is available for addressing these common registers — conserving RAM space and eliminating repeated stack pushes and pops. Global registers can be pushed, popped, and used by conditional instructions in the same way that local registers are handled. In addition, the GSP can itself be pushed and popped to/from the subroutine stack, allowing different routines to access different subsets of the global stack area.

Subroutine Stack Pointer (SSP)

A Push onto the SS (jump subroutine or interrupt) first increments the SSP and then writes the return address to RAM. A pop from the SS first reads the return location and then decrements the SSP, effectively removing the data from the stack (although the data remains in RAM). For interrupts, the return address is the one that would have been output in the cycle when the

interrupt vector was output. For subroutine jumps, the return address is the instruction immediately following the subroutine call. For further information, see: Return from Interrupt with Pending Interrupt, appendix 4.2; and the Instruction Set Description, 2.0.

The subroutine stack can also be used to save key program parameters such as the status register, GSP, or counter values. After entering a new routine, critical parameters from the calling routine are pushed onto the stack, thus freeing the associated hardware for use by the new routine. Prior to the end of the routine, the original parameters are restored with their former values for continued use by the calling routine.

The Stack Usage Example (appendix 4.3) illustrates the state of RAM after three subroutine calls.

Stack Limit Register and Stack Overflow

The preloadable Stack Limit Register (SLR) and associated circuitry warns the user of impending stack overflows, permitting stack overflow recovery. The highest priority interrupt, IR_9, is assigned to stack overflow, although it may be masked. A stack overflow interrupt will occur under any of the following three circumstances:

- a push causing the SSP to increment to the value in the stack limit register
- a pop from SS location 00 (underflow)
- a push causing the RS pointer (LSP or GSP) to decrement to the value in the stack limit register +3.

The three location buffer between the SLR and the RS pointer allows for three extra pushes that may occur (in a worst case) prior to entering the stack overflow service routine. These pushes would be:

1. the push causing the initial overflow
2. a possible push operation while IV_9 is output and
3. the IR_9 return address push.

See: Interrupts, 1.4.3; and Three Stack Pushes on Stack Overflow (appendix 4.2.5) for more details.

The SLR is only 4-bits wide and is compared to the 4 MS bits of the 6-bit RAM address. Therefore, stack limits may only be set at integer multiples of 2^2, i.e., RAM locations 0, 4, 8, 12, . . . , 60. The SLR is right-filled the additional two bits with zeros or ones, depending upon the direction of the push being performed ('00' for SS pushes and '11' for RS pushes, see Instruction Set Description – SLRIVP, 2.5). In the cycle following a stack overflow, the highest priority interrupt vector IRV_9 (also used for trapping; see TTR Pin, 1.7) is output. To determine the cause of this interrupt, both SS and RS pointers must be tested in the first several cycles of the service routine. Prior to returning from the overflow interrupt routine, the SLRIVP instruction must be executed, to clear the calling IR_9 from the interrupt latch.

1.4.3 Interrupts

The ADSP-1401 processes eight external and two internal interrupts. All external interrupts are level sensitive (positive logic: see IR Latch, this section) and are processed by the interrupt logic block. The block elements (see Figure 4) are comprised of an interrupt de-multiplexer followed by an interrupt latch, masking logic and priority decoder for selecting the most urgent interrupt (IR_9 having the highest priority, and IR_0 the lowest), and special one-shot to override the address multiplexer with the interrupt vector (IV_{9-0}) on the cycle following the interrupt request.

The external interrupts (IR_{8-1}) may be used for any purpose, however, unused inputs *must not* be left floating (i.e., tie them to logic LO so as to preclude the associated interrupt). Two additional interrupts which are internal are reserved for stack overflow — IR_9 (see Stack Limit Register and Stack Overflow, 1.4.2) and counter underflow — IR_0 (see Counters, 1.4.1). See Counters (1.4.1) for implications of using IR_0 for other than writable control store downloading.

Interrupt vectors are always output (assuming interrupts are enabled and the associated interrupt is not masked) on the cycle immediately following the acceptance of the interrupt request. Contextual saves (stacking and storing) should be made immediately upon entering the interrupt service routine and restored immediately prior to its exit.

Up to four external interrupts may be connected directly to the external interrupt pins, $EXIR_{4-1}$, and are treated as interrupts IR_{8-5}, respectively. Lower priority interrupts, IR_{4-1}, must be masked out in this case.

Up to eight external interrupts may be accommodated using time-division multiplexing. An external 2:1 multiplexer reduces the eight external interrupts to two groups of four (see Figure 3). An internal de-multiplexer automatically restores the external interrupts back to eight.

The interrupt vector file may be directly read and written via the data bus with the aid of the Interrupt Vector Pointer (see Instruction Set Description, Interrupts, 2.5).

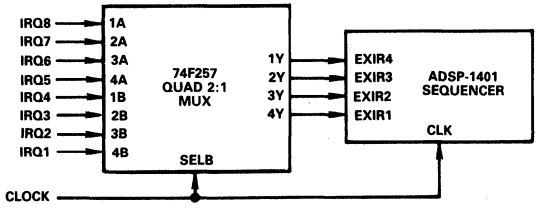


Figure 3. Expanding External Interrupts

IR Latch

Interrupt requests IR_{8-5} are latched during the first half-cycle (clock HI), while IR_{4-1} are latched during the second half-cycle (clock LO). Once latched, external interrupt requests are held until processed, even if the external request signal goes away. This latching technique allows removal of external interrupt sources after they have been recognized by the sequencer.

Latched user interrupt requests (IR_{8-1}) are held until: i) the interrupt is processed and a "Return from Interrupt" (RTNIR) instruction is executed; ii) the interrupt service routine executes a "Clear Current Interrupt" instruction (allowing nested interrupts); or, iii) a "Clear All Interrupts" instruction is executed. Reserved interrupts (IR_9 and IR_0) are cleared from the interrupt latch by utilizing the SLRIVP and CLRS instructions, respectively. See Internal IR Control Logic (1.4.3) for details.

The user may bypass the interrupt latch with the "Select Transparent Interrupts" (STIR) instruction (setting status register bit SR_0). In the transparent mode, the interrupting device must assert the interrupt request until the interrupt service routine resets the request source.

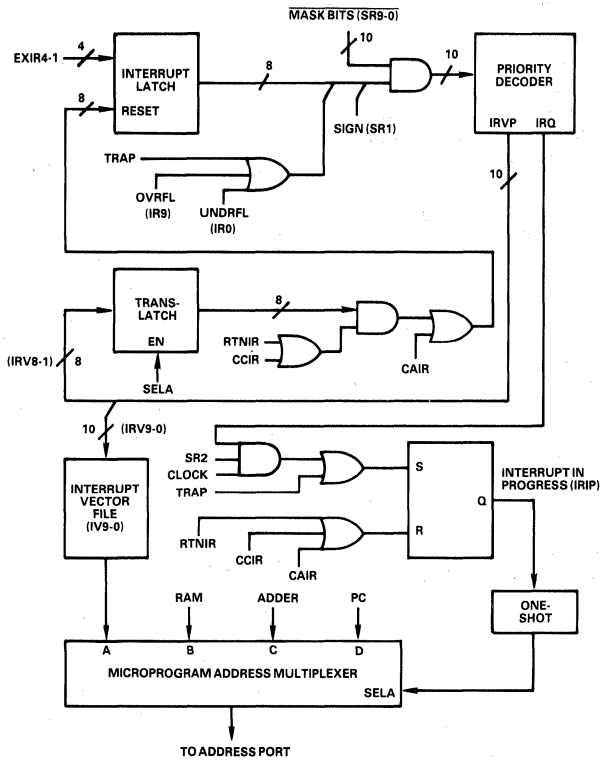


Figure 4. Internal Interrupt Control Logic

IR Mask

All ten interrupts may be independently masked using status register bits SR_{15-6} (corresponding to interrupts IR_{9-0}). Setting a particular mask bit prevents the interrupt from being executed. Note that the status register may be read or written via the Data port, and also pushed and popped to/from the subroutine stack, allowing nesting and servicing of interrupts in any desired order (see: Internal IR Control Logic, 1.4.3; and Status Register, 1.4.4).

Two instructions allow bitwise clearing or setting of the interrupt mask. "IR Mask Bit Clear" (IRMBC) will clear those mask bits for which the corresponding data bits (D_{15-6}, as applied to IR_{9-0}) are set, while "IR Mask Bit Set" (IRMBS) will set those mask bits for which the corresponding data bits are set. In both cases, zeros in the data field will preserve the corresponding mask bit. See Instruction Set Description – Status Register, 2.3.

IR Priority Decoder

Unmasked interrupts are passed to the priority decoder which determines the most urgent, valid interrupt and generates an internal Interrupt Request Signal (IRS). The corresponding vector is then fetched from the interrupt vector file and passed to the address port.

Minimum IR Servicing Requirements

Interrupt vectors are output on the cycle following the acceptance of an interrupt request. Interrupt jumps differ from subroutine jumps in that subroutine jumps push the return address in the *same* cycle as the jump address is output, whereas interrupt return addresses are not pushed until the *following* cycle. This is because the instruction executing while the interrupt vector is output may be utilizing RAM and must complete its execution prior to pushing the interrupt return address. Thus, the PC (interrupt return address) is pushed automatically in the first cycle of the interrupt service routine, i.e., *the cycle following the interrupt request acceptance.*

For this reason, the first instruction of any interrupt service routine is always ignored; it *must* be a no-op (CONT). Note that a minimum interrupt service routine would be a CONT followed by a RTNIR.

Internal IR Control Logic

The interrupt enable bit of the status register, SR_2, *must* be set for interrupt servicing to occur. Interrupt servicing may be inhibited by clearing this bit, although external interrupt requests will continue to be latched.

Only one interrupt is ever active at a time. Additional interrupts are "locked out" by an internal "Interrupt In Progress" signal (IRIP) during interrupt servicing (except for TRAP), although they continue to be latched. The IRIP signal is automatically reset upon the "Return from Interrupt" (RTNIR) instruction which pops the return address from the subroutine stack to the PC.

Normally, multiple interrupts are accumulated in the interrupt latch. Whenever a valid interrupt is pending, the internal signal "Interrupt Request" (IRQ) is asserted. Upon each RTNIR, the highest priority, unmasked, pending interrupt is serviced.

Nested interrupts are supported with two instructions: "Clear Current Interrupt" (CCIR) or "Clear All Interrupts" (CAIR). The CCIR instruction clears the IRIP signal and interrupt latch bit for the interrupt in progress. This action re-enables interrupting, relegating the interrupt in progress to a subroutine status. If an external interrupt is pending, the associated IR vector will be output on the cycle following CCIR. To cancel all pending interrupt requests, the CAIR instruction clears the IRIP signal and the entire interrupt latch.

Normally, it is good practice to convert interrupts to subroutines. This can be done by executing the "Clear Current Interrupt" (CCIR) instruction (resetting IRIP) and should be done as early as possible in the interrupt service routine. There are two reasons for changing the status of an interrupt to that of a subroutine. Firstly, if IRIP is allowed to remain active throughout the interrupt service routine, then the occurrence of either internal interrupt (stack overflow or counter underflow, IR_9 or IR_0, respectively) will remain undetected until the current interrupt concludes; the user will be unaware of these interrupt requests.

When using the TRAP capability (see TTR Pin, 1.7), there is a second reason to clear IRIP. Because TRAP must have the highest priority, interrupt IR_9 (when invoked by a TRAP request) is not locked out by IRIP. This allows TRAP to displace an interrupt in progress, but also means that upon completion of the trap service routine, IRIP will be cleared by the RTNIR instruction; re-enabling interrupting in spite of the incomplete interrupt which TRAP displaced.

Either of these instructions (CCIR or CAIR) require an "extra" cycle before a pending interrupt vector may be output. A typical scenario being an interrupt in progress, IR_n (containing a CCIR instruction), with a interrupt pending, IR_m:

CCIR Example

µCode Location	Instruction Executing	Output Address	Comments
n	IR_n Routine	n+1	IR_m Pending
n+1	CCIR	n+2	Clear IRIP
n+2	IR_n Routine	IV_m	IR_m Recognized
IV_m	IR_m Routine	IV_m+1	. . .

1.4.4 Status Register

The ADSP-1401 has a 16-bit status register for storing various operational modes. The ten MS bits of this register (SR_{15-6}) comprise the interrupt mask for interrupts IR_{9-0}, respectively. The remaining six LS bits (SR_{5-0}) control the operational modes as shown below.

Status Register Bit Assignments

Bit#	Function (HI/LO)
SR_{15}	IR_9 Mask Bit
.	.
.	.
.	.
SR_6	IR_0 Mask Bit
SR_{5-4}	Relative Jump Width Selection: '00' = 16-bit relative address width '01' = 8-bit width '10' = IHC Mode (8-bit width) '11' = 12-bit width
SR_3	Select GSP/LSP
SR_2	Enable/Disable Interrupts
SR_1	Set/Clear Sign Bit
SR_0	Select Transparent/Latched Interrupts

The status register can be directly read and written via the data port and also pushed and popped to/from the subroutine stack. In addition, status register bits SR_{15-6} (the interrupt mask) may be bitwise cleared or set with dedicated instructions. See: Instruction Set Description – Status Register, 2.3; and Interrupts – IR Mask, 1.4.3.

1.5 Clock

The input clock employs both HI and LO levels to control the various transparent latches throughout the device. Generally, the clock should be symmetric; however, in some instances the clock may be stretched during the second half-cycle (LO) to accommodate unusual circumstances such as a cache memory miss (see: TTR Pin – Trap, 1.7).

1.6 External Flag

The external flag input may be used to control conditional instructions. FLAG is latched similarly to instructions (latched during clock HI and transparent during clock LO), but requires less setup time. Two instructions make explicit use of FLAG as their condition (JPCOF and JPCNF), while others employ a condition mode selection (UNCONDITIONAL, NOT FLAG, FLAG, or SIGN; see Instruction Set Description, 2.0) to be specified as part of their opcode.

1.7 TTR Pin (Trap, Three-State and Reset)

The Trap, Three-State and Reset pin (TTR) is a time-multiplexed, three-purpose pin used to

- provide program trap capability
- control the address port output drivers and
- reset the ADSP-1401.

If the TTR pin is held HI for an entire cycle, the RESET sequence begins and TTR must be held HI for at least two more complete cycles (RESET requires three cycles to complete). If trap and three-state control capabilities are also needed, the combination of the 1401's internal circuits and the external circuitry shown in Figure 5 can be used to effectively time-multiplex the TTR pin.

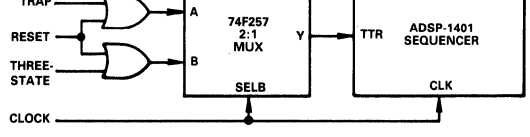


Figure 5. External Logic for TTR Pin

Trap

For a trap to occur, the TTR pin must be asserted during clock LO only. The primary reason to invoke a trap is in support of cache memory systems, or in case of system emergencies. Cache memory systems generally utilize a large microcode memory space, of which only a small area (that currently under execution) is comprised of high-speed RAM (the balance consisting of slower, less costly memory). The high-speed RAM is directly accessible by the sequencer, whereas the bulk of (slow) memory is usually accessible indirectly (via a cache memory controller which controls downloads of code to the cache memory area).

In a cache-based system, microcode is generally executed from the high-speed cache. If an access is attempted to code not resident in the cache area, the cache memory controller must detect the discrepancy and generate an exception to the access (a "cache miss"). Then, the missing code segment must be down-loaded to the cache memory area (see: Instruction Set Description – Writeable Control Store, 2.7).

When a cache miss occurs, the cache memory control logic asserts the TTR pin while stretching the system clock LO. Upon detecting the trap request, the sequencer immediately generates the highest priority interrupt, IR_9, replacing the current address (that causing the cache miss). The cache miss address is pushed on the subroutine stack and popped after the interrupt service routine has reloaded the cache area with the missing code segment.

Note: Trap requests which occur on the first cycle of an interrupt service routine are *not* recognized. The ADSP-1401 always executes a CONT instruction in this cycle, and ignores its instruction port and therefore trap requests as well.

The trap interrupt differs from the standard interrupt protocol in three ways:

1. The interrupt vector, IV_9, is output asynchronously, i.e., it occurs t_{TRAD} after asserting the Trap signal and must occur *before* the next cycle! To accomplish this, a clock stretch cycle may be needed to allow enough time to fetch the new instruction.
2. The *current address* is pushed onto the SS for later restoration (after the cache miss is resolved), whereas standard interrupts push the *current address + 1*.
3. Trap interrupts *cannot* be masked or disabled. Note that if IR_9 is also used for stack overflow and underflow, the service routine must discriminate which actually occurred.

Caution: because trapping is asynchronous, spikes on the TTR pin wider than 3ns during clock LO may initiate inadvertent trapping.

Three-State
The address port is placed in a high-impedance state when the TTR pin is HI during clock HI and LO during clock LO. The TTR signal is latched during clock LO and transparent during clock HI. This facilitates full cycle, three-state control. (Note that the IDLE instruction can also place the address port in a high-impedance state.)

Reset
The TTR pin may be used to initialize the ADSP-1401 by asserting it (HI for both clock phases) for at least three full cycles. Use of the reset operation alone does not require the multiplexing described above. However, if the trap and/or three-state controls are also needed, they must not occur in the same cycle (this would be an abnormal situation), as this constitutes a reset. The RESET signal forces a zero output address, places the data port in the high-impedance state, and resets internal registers as follows:

Sequencer Status after RESET Operation

Parameter	Reset Condition
Program Counter	μCode Location 0000_{16}
Subroutine Stack Pointer (SSP)	RAM Location 00_{10}
Local Stack Pointer (LSP)	Undefined
Global Stack Pointer (GSP)	Undefined
Stack Limit Register (SLR)	RAM Location 32_{10}
RAM Data	No Change
Counters	No Change
Interrupt Mask (SR_{15-6})	All Bits to '0' (Unmasked)
Interrupt Vector File	No Change
Interrupt Vector Pointer (IVP)	Undefined
SR_{5-4}	'00' (16-Bit Relative Offsets)
SR_3	'0' (LSP Selected)
SR_2	'0' (Interrupts Disabled)
SR_1	'0' (Sign Bit Cleared)
SR_0	'0' (Latched Interrupt Mode)
Writeable Control Store Mode	Cleared

NOTE:
The first instruction (microcode location 0000_{16}) must be a "CONT".

2.0 INSTRUCTION SET DESCRIPTION

The instruction set is divided into seven categories pertaining to generic operation (see data sheet outline or Mnemonics and Opcodes, 4.5).

Several instructions employ two instruction bits (I_1 and I_0) to specify a counter (C_{3-0}) and/or a local register (R_{3-0}, relative to the RSP) as arguments. Nine of the conditional instructions use another two instruction bits (I_3 and I_2) to select one of the four condition modes:

'00' UNCONDITIONAL
'01' NOT FLAG
'10' FLAG
'11' SIGN

The sign bit of the status register, SR_1, may also be used to (implicitly or explicitly) store an external condition. This is useful if the condition results from an operation performed in the middle of a loop, but is not tested until the end; the loop is exited with an "If Sign: Jump" instruction. Recall that any subsequent counter operations will overwrite SR_1.

2.1 Jump and Branch Instructions

Jump and branch instructions provide flow control of microcode execution, offering three-way branches, jumps, subroutine calls, returns, and addressing mode selection (see Figure 6). These instructions support conditional control, allowing addressing from the register stack, the data port, or the indirect jump address space in the RAM. Generally, they are of the form:

If Condition: Do Operation; Else, Continue.

JPCOF IF FLAG: JUMP PC

The address is not incremented while the flag is at a logic HI, i.e., PC<=PC. If the flag is LO, the next address is (PC+1).

JPCNF IF NOT FLAG: JUMP PC

The address is not incremented while the flag is at a logic LO, i.e., PC<=PC. If the flag is HI, the next address is (PC+1).

JTWO IF CONDITION: JUMP PC+2

If the condition specified is met, this instruction causes the next sequential microprogram address to be skipped. This instruction allows single instruction bypassing or interleaving without need to provide explicit addressing.

JDA IF CONDITION: JUMP DATA, ABSOLUTE

If the specified condition is met, this instruction causes a jump to the absolute address at the data port. If the condition is not met, the next sequential instruction will be executed.

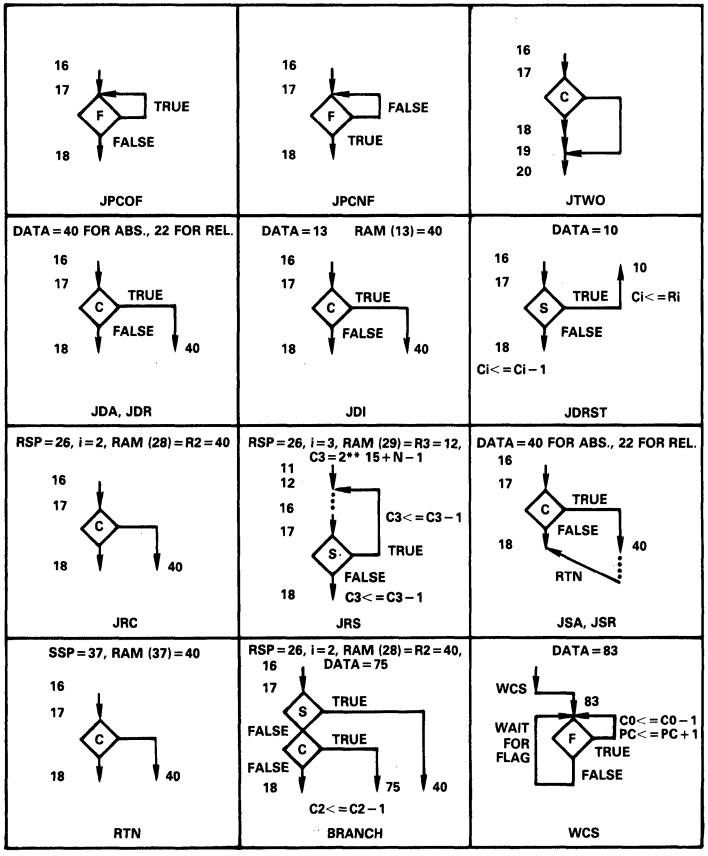


Figure 6. Instruction Flow Charts

JDR IF CONDITION: JUMP DATA, RELATIVE

If the condition specified is met, the address at the data port will be added to the PC and output (jump distance is offset plus one). The offset width is determined by the address width selection (8, 12, or 16-bits). If the condition is not met, the next sequential instruction will be executed.

JDI IF CONDITION: JUMP DATA, INDIRECT

If the condition specified is met, this instruction will output the address stored in the RAM address given by bits D_{5-0} of the data port. If the condition is not met, the next sequential instruction will be executed.

JDRST IF SIGN OF C_i: JUMP DATA, $C_i <= R_i$; ELSE, $C_i <= C_i - 1$

This instruction first tests the sign of the counter, C_i. If negative, the address at the data port is output and the counter is re-initialized (reset) with the data in the register pointed to by (RSP + i). If the sign is positive, the counter is decremented and the next sequential address is output. The register and counter use the same subscript, i.

JRC IF CONDITION: JUMP R_i. (COND ≠ SIGN)

If the condition specified is met, output the address in RAM at the location (RSP + i), where i is given by I_{1-0} of the instruction. The selected condition may not be SIGN, as this is the JRS instruction. The PC may be pushed on the register stack and referenced as a register thus allowing a "jump to stack" instruction which is useful for looping.

JRS IF SIGN OF C_i: JUMP R_i, $C_i <= C_i - 1$; ELSE, $C_i <= C_i - 1$

This instruction first tests the sign of counter, C_i. If negative, output the address in RAM at location (RSP + i). If the sign is positive, the next sequential microprogram address is output. The counter is always decremented after the test.

JSA IF CONDITION: JUMP SUBROUTINE, ABSOLUTE

If the condition specified is met, the 16-bit absolute address at the data port is output and the PC will be pushed onto the subroutine stack. If the condition is not met, the next sequential instruction will be executed.

JSR IF CONDITION: JUMP SUBROUTINE, RELATIVE

If the condition specified is met, the address at the data port is added to the PC and output (jump distance is offset plus one) and the PC is pushed onto the subroutine stack. The offset width is determined by the address width selection (8, 12, or 16-bits). If the condition is not met, the next sequential instruction will be executed.

RTN IF CONDITION: RETURN FROM SUBROUTINE

This instruction is used to return from subroutines. If the condition specified is met, the subroutine stack is POPped, which outputs the return address and decrements the SSP. If the condition is not met, the next sequential instruction will be executed.

BRANCH IF SIGN OF C_i: JUMP R_i, $C_i <= C_i - 1$;
ELSE, IF CONDITION:
JUMP DATA, $C_i <= C_i - 1$;
ELSE, $C_i <= C_i - 1$ (COND ≠ SIGN)

This instruction implements a three-way branch with the address source from the data port, register R_i, or the PC. The instruction first tests the sign bit of the counter C_i; if negative, the output address is given by R_i, i.e., RSP + i. If the sign was not true, but the specified condition is true, the address source is the data port. If the sign was not true and the condition is not met, the next sequential instruction is executed.

The counter and the register use the same subscript value i. The counter is *always* decremented. Note that this instruction uses only absolute data addresses; relative addressing is not available with the three-way branch instruction.

2.2 Stack Operations

Subroutine Stack
Subroutine Stack Pointer (SSP) instructions are used for maintaining the subroutine stack. These instructions may also be used to upload or download the entire RAM for examination, stack expansion or context switches.

PSDSS PUSH DATA ONTO SS

Increments the stack pointer and then loads the RAM location specified by the SSP with the data at the data port.

PPSSD POP SS TO DATA PORT

Transfers the contents of the stack location given by the stack pointer to the data port and decrements the stack pointer.

WRSSP WRITE SSP

Loads the SSP with bits D_{5-0} of the data port.

RDSSP READ SSP

Read the 6-bit subroutine stack pointer. This allows the value of the stack pointer to be saved or examined. Bits D_{5-0} of the data port correspond to bits 5-0 of the SSP. The 10 MSB's of the data port (D_{15-6}) are undefined.

DSSP DECREMENT SSP

Decrements the stack pointer without reading.

Register Stack
Register Stack Pointer (RSP) instructions are used to upload and download the entire RAM for initialization, examination, or context switching and to maintain the RAM space allocated to local and global jump registers. As previously discussed, register stack instructions refer to either the Local Stack Pointer (LSP) or the Global Stack Pointer (GSP), depending upon the status register (SR_3). If SR_3 is LO, register stack instructions pertain to the LSP. If SR_3 is HI, register stack instructions pertain to the GSP.

SGSP SELECT GSP

Select the Global Register Stack Pointer. Set Status bit SR_3 (HI).

SLSP SELECT LSP

Select the Local Register Stack Pointer. Clear Status bit SR_3 (LO).

RDRSP READ RSP

Transfers the RSP to the data port bits D_{5-0} for examination or storage. The 10 MSBs (D_{15-6}) of the D port are undefined.

WRRSP WRITE RSP

Preload the selected RSP (LSP or GSP) with bits D_{5-0} of the data port.

PSPC PUSH PC ONTO RS

Decrements the RSP and writes the PC to the register stack. This instruction may be used to set up a JRC loop (IF CONDITION: JUMP R_i = PC).

PSGSP PUSH GSP ONTO SS

Increment the SSP and write the GSP onto the subroutine stack.

PPGSP POP GSP FROM SS

Write the subroutine stack to the GSP and decrement the SSP.

PSDRS PUSH DATA ONTO RS

Decrement the RSP and then write the data at the data port into the location specified by the updated RSP.

PPRSD POP RS TO DATA PORT

Transfers RAM data pointed to by the RSP to the data port and then increments the RSP.

AIRSP ADD i TO RSP

Add i to the register stack pointer. Note that i = 0, 1, 2, or 3 in this instruction corresponds to 4, 1, 2, or 3, respectively. This instruction effectively removes up to four registers from the stack.

S1RSP SUBTRACT ONE FROM RSP

Subtract 1 from the RSP without a write. This instruction is used to modify the RSP without explicitly reloading it.

S4RSP SUBTRACT FOUR FROM RSP

Subtract four from the RSP without a write. This instruction may be used to modify the RSP without explicitly reloading it.

2.3 Status Register Operations

The status register bits, SR_{15-0}, contain ten mask bits, SR_{15-6}, for masking interrupts IR_{9-0}, and six control bits, SR_{5-0} (see

Bidirectional Data Port, 1.4). The entire status register can be read or written via the data port, or pushed or popped to/from the subroutine stack. Upon RESET, the entire status register is initialized to zero.

RDSR READ SR

The entire status register (SR_{15-0}) is output over the data port (D_{15-0}).

WRSR WRITE SR

Write the data port (D_{15-0}) to the status register (SR_{15-0}).

PSSR PUSH SR ONTO SS

Increment the SSP and then write the status register to the subroutine stack.

PPSR POP SR FROM SS

The top of the subroutine stack is written into the status register, and then the SSP is decremented.

2.4 Counter Operations

Counters may be pushed and popped to/from the subroutine stack or loaded directly from the data port. The counters may be read externally by pushing the counters onto the subroutine stack then popping the subroutine stack to the data port. The device has four counters, denoted C_i, which are indexed by the two LSBs of the instruction.

If a jump is required *after* N events (until sign), the counter should be loaded with two less than the number of events desired (N − 2). If a jump is required *for* N events (while sign), the counter is loaded with $2^{15} + N - 2 = 8000_{16} + N - 2$.

Care must be taken when using the counter underflow interrupt (IR_0, see 1.4.3) to clear the sign bit *before* the IR_0 mask bit is cleared.

WRCNTR WRITE C_i

Write to the selected counter, C_i, from the data port.

CLRS CLEAR SIGN BIT

Clear status register bit SR_1.

SETS SET SIGN BIT

Set status register bit SR_1.

PSCNTR PUSH C_i ONTO SS

Increment the SSP and write the specified counter onto the subroutine stack.

PPCNTR POP C_i FROM SS

Transfer the data from the subroutine stack to the counter specified by the instruction, then decrement the SSP.

DCCNTR DECREMENT C_i

Unconditionally decrement counter C_i.

IFCDEC IF CONDITION: DECREMENT C_0

Decrement counter C_0 on condition. If the sign condition is selected, the sign is taken from the status register bit SR_1, rather than from the counter sign (which normally provides the sign condition).

Normally, if the counter underflow interrupt (IR_0) is enabled, it is activated by the counter sign bit going HI. However, if IFCDEC is used to decrement C_0, the IR_0 interrupt is activated by the SR_1 bit, rather than the sign bit of C_0. Since the SR_1 bit goes HI only after C_0 has underflowed, IFCDEC must be executed once more after the C_0 underflow to generate the IR_0 interrupt. Alternatively, the preloaded value of C_0 may be reduced by one.

2.5 Interrupt Control

Detailed interrupt operation is described in the Interrupts section (1.4.3). Here, specific interrupt operations such as interrupt clearing, IRV read/write, interrupt mask manipulation, etc., are described.

CCIR CLEAR CURRENT INTERRUPT

Allows nesting of user interrupts IR_{8-1} on subsequent instructions by clearing both the interrupt latch bit currently being serviced and the interrupt in progress signal (IRIP), re-enabling interrupts. If an external interrupt is pending, the associated IR vector will not be output until the cycle following CCIR. Internal interrupts (IR_9 and IR_0) are *not* cleared by CCIR and must be explicitly cleared through the SLRIVP and CLRS instructions, respectively.

CAIR CLEAR ALL INTERRUPTS

Clears external interrupt latches IR_{8-1}, and re-enables the interrupt interface (IRIP cleared LO). The next sequential instruction will be executed prior to the jump to a pending interrupt. Internal interrupts (IR_9 and IR_0) are *not* cleared by CAIR and must be explicitly cleared through the SLRIVP and CLRS instructions, respectively.

RTNIR RETURN FROM INTERRUPT

Clears the current interrupt latch for IR_{8-1}, re-enables interrupts (IRIP cleared LO), and pops the return address from the subroutine stack. The next sequential instruction will be executed prior to the jump to a pending interrupt routine. Internal interrupts are *not* cleared and the IR_9 and IR_0 interrupt latches must be cleared explicitly through the SLRIVP and CLRS instructions, respectively.

RDIV READ IRV AND INCREMENT IVP

Outputs the interrupt vector currently pointed to by IVP to the data port and then increments the IVP. Interrupts should be disabled when writing or reading interrupt vectors.

WRIV WRITE IRV AND INCREMENT IVP

Writes the interrupt vector currently pointed to by the IVP from the data port and then increments the IVP. Interrupts should be disabled when writing or reading interrupt vectors.

IRMBC IR MASK BITWISE CLEAR

Allows selected IR mask bits to be cleared. Data port bits D_{15-6} are applied to status register bits SR15-6 (corresponding to mask bits for IR_{9-0}). Those data bits which are HI will clear the mask bit, while those data bits which are LO will leave the mask bit intact. Data port bits D_{5-0} are ignored.

IRMBS IR MASK BITWISE SET

Allows selected IR mask bits to be set. Data port bits D_{15-6} are applied to status register bits SR_{15-6} (corresponding to mask bits for IR_{9-0}). Those data bits which are HI will set the mask bit, while those data bits which are LO will leave the mask bit intact. Data port bits D_{5-0} are ignored.

DISIR DISABLE INTERRUPTS

Disables the execution of all further interrupts by clearing the enable interrupt flag (SR_2). External interrupts continue to be latched.

ENAIR ENABLE INTERRUPTS

Enables execution of interrupts by setting the enable interrupt flag (SR_2).

SLIR SELECT LATCHED INTERRUPTS

Places the interrupt request latches in the latched mode for interrupts IR_{8-1} (SR_0 LO). Interrupts are latched if they are valid at the appropriate clock edge. Interrupts IR_{8-5} are latched at the positive going clock edge while IR_{4-1} are latched at the negative going clock edge.

STIR SELECT TRANSPARENT INTERRUPTS

Places the interrupt request latches in the transparent mode (SR_0 HI) for interrupts IR_{8-1}. The interrupt request is only valid while the external interrupt inputs are high. Interrupts are still processed on the next cycle, so long as they meet the minimum interrupt setup specification. Note that selecting transparent interrupting will clear any pending interrupts stored in the interrupt latch.

SLRIVP WRITE SLR WITH D_{5-2}, AND IVP WITH D_{15-12}

Loads the 4-bit stack limit register (SLR) and the 4-bit interrupt vector pointer (IVP) from the data port. This instruction also clears the stack overflow interrupt request IR_9.

For stack overflow detection, the active 6-bit stack pointer (SSP, LSP or GSP) is compared to a 6-bit word comprised of the 4-bit SLR (MSBs) and the two LSBs determined by the instruction type, as follows:

'00' for subroutine stack push (PSDSS); or,
'11' for register stack push (PSDRS).

For example, if a stack limit of 36_{10} and positioning of the IVP at IRV_7 is desired, the value '0111xxxxxx1001xx' is provided at the data port. Note that the SLR and IVP cannot be read.

The interrupt vector pointer (IVP) addresses the vector file for reading or writing interrupt vectors. To write interrupt vectors IRV_{9-0}, the IVP must first be initialized by SLRIVP. The WRIV instruction (see above) is then used to write the interrupt vector pointed to by the IVP, which is then incremented automatically.

2.6 Relative Address Width Controls

The width control instructions allow reduction of microcode when Jump Data Relative and Jump Subroutine Relative instructions need less than the full, 16-bit range. Use these instructions to sign extend the 8, 12 or 16-bit wide jump data presented at the data port. The jump width may be selected by the explicit instructions or by directly setting the status register bits SR_{5-4} as described below. Any of these three instructions will reset the Instruction Hold Control mode (see Misc. Instructions – IHC, 2.7).

Note that selection of 8-bit width can be made with or without IHC. For all relative jumps, the jump distance is the offset +1.

REL16 SELECT 16-BIT RELATIVE JUMPS

Select the 16-bit relative jump. This adds D_{15-0} at the data port to the PC to obtain the jump address. The status bits SR_{5-4} are set to '00'.

REL12 SELECT 12-BIT RELATIVE JUMPS

Selects the jump data from D_{11-0}. The offset is sign-extended allowing relative jumps in the range +2047 to −2048. The status bits SR_{5-4} are set to '11'.

REL8 SELECT 8-BIT RELATIVE JUMPS

Selects the jump data from D_{7-0}. The offset is sign-extended allowing relative jumps in the range +127 to −128. The status bits SR_{5-4} are set to '01'.

2.7 Miscellaneous Instructions

CONT CONTINUE

Increment and output the next location in microcode memory without any other changes. Allows straight line microcode execution.

IDLE DISABLE OUTPUTS AND JUMP PC

Places the address port into the high-impedance state, inhibiting program counter (PC) increments. Useful in applications where multiple sequencers share a common microcode address bus.

This instruction causes the ADSP-1401 to behave as if the clock had stopped. The IDLE instruction may be latched internally by using IHC, freeing microcode for use by another device.

External interrupt requests must be inhibited during IDLE. If interrupts are not inhibited, the ADSP-1401 will attempt to process an interrupt that goes active. However, it will be unable to output an interrupt vector because the IDLE instruction places the address port in the high-impedance state; more importantly, it will set its IRIP flag, which will inhibit further interrupt processing even after the IDLE state is exited.

Interrupts can be inhibited using the interrupt mask or the DISIR instruction. While inhibited, interrupt requests will still be latched in the interrupt latch.

IHC ENABLE INSTRUCTION HOLD CONTROL

Sets SR_{5-4} to '10' and redefines the function of IR_1 to allow a subsequent instruction to be held for repeated execution, regardless of the instruction port. Use of the IHC mode requires that the mask bit for IR_1 be set. See Instruction Hold Control, appendix 4.1 for more details.

While in the IHC mode, asserting IR_1 HI (prior to the second half-cycle of any instruction) will hold that instruction and disable *all* interrupts (although they continue to be latched) until IR_1 is brought LO again (again, prior to the second half-cycle of any instruction).

It is recommended that IR_1 be dedicated to control of the IHC mode (if needed). However, if it must also be used for subsequent interrupting, then the CAIR instruction should be executed before unmasking IR_1 (to clear the interrupt request resulting from use of IR_1 as the IHC control).

Use of IHC is constrained to 8-bit relative addressing (see Relative Address Width Controls, 2.6) and clearing IHC is accomplished by executing any of the relative address width control instructions (changing status register bits SR_{5-4}).

WCS WRITE CONTROL STORE

Provides sequential addressing during microcode downloads to a RAM based microcode store. The instruction may be interpreted as:

JUMP DATA;
IF FLAG: DECREMENT C_0 AND CONTINUE UNTIL INTERRUPTED.

Upon initiation of the WCS instruction, the sequencer outputs the address found at the data port (that of the first instruction to be downloaded). The external flag is then used to gate subsequent sequential addressing for the download and decrementing of counter C_0. This action continues until an interrupt is detected (from either a C_0 underflow, externally or the chip is RESET). Instructions at the instruction port are *ignored* during WCS, until the interrupt or reset occurs.

The external flag allows synchronization of an external memory with the sequencer. FLAG should be asserted HI as each new μcode word is made available for writing to μcode memory.

Notes on Using a Writeable Control Store:

- If a counter interrupt is desired, counter C_0 must be initialized with *two* less than the length of microcode segment to be downloaded.
- If counter interrupting is to be used to exit the WCS mode, IRV_0 should be unmasked and initialized with the address of the instruction to be executed upon WCS completion (see Interrupts, 1.4.3 for timing).
- Since interrupting is used to exit the WCS mode, the last address downloaded is pushed onto the SS stack as an interrupt return address. However, because it is not actually a return address, the SS should be popped immediately by decrementing the SSP (DSSP) to clear it of this last address.
- Since FLAG is used to gate the download, it should not become active until after the WCS instruction is executed.

See application note "Writeable Control Store using the ADSP-1401."

3.0 SPECIFICATIONS

This section describes the ADSP-1401's performance parameters. The Specifications Table lists the device's relevant electrical and switching characteristics, while Figure 7 presents the corresponding timing diagram.

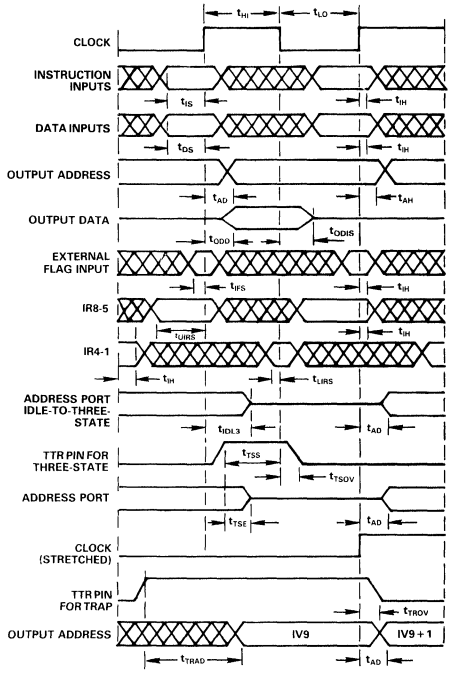


Figure 7. ADSP-1401 Timing Diagram

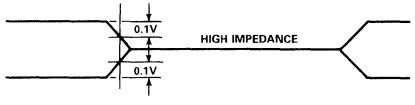


Figure 8. Three-State Reference Levels

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1401JN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1401KN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1401JP	0 to +70°C	52-Lead PLCC	P-52
ADSP-1401KP	0 to +70°C	52-Lead PLCC	P-52
ADSP-1401JD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1401KD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1401SD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1401TD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1401SD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1401TD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		J & K Grades Min	J & K Grades Max	S & T Grades[2] Min	S & T Grades[2] Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_{AMB}	Ambient Operating Temp.	0	70	−55	125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	J & K Grades Min	J & K Grades Max	S & T Grades[2] Min	S & T Grades[2] Max	Unit
V_{IH}	Hi-Level Input Voltage	V_{DD} = max	2.0		2.0		V
V_{IHC}	Clock Input Hi-Level Input Voltage	V_{DD} = max	3.0		3.5		V
V_{IL}	Lo-Level Input Voltage	V_{DD} = min		0.8		0.8	V
V_{OH}	Hi-Level Output Voltage	V_{DD} = min, I_{OH} = −1mA	2.4		2.4		V
V_{OL}	Lo-Level Output Voltage	V_{DD} = min, I_{OL} = 3mA		0.6		0.6	V
I_{IH}	Hi-Level Input Current, All Inputs	V_{DD} = max, V_{IN} = 5V		10		10	μA
I_{IL}	Lo-Level Input Current, All Inputs	V_{DD} = max, V_{IN} = 0V		10		10	μA
I_{OZH}	Three-State Leakage Current	V_{DD} = max, V_{IN} = max		50		50	μA
I_{OZL}	Three-State Leakage Current	V_{DD} = max, V_{IN} = 0		50		50	μA
I_{DD}	Supply Current	max clock rate, TTL inputs		90		115	mA
I_{DD}	Quiescent Supply Current	V_{IN} = 2.4V		50		65	mA

ABSOLUTE MAXIMUM RATINGS

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

ESD SENSITIVITY

The ADSP-1401 features proprietary input protection circuitry. Per Method 3015 of MIL-STD-883, the ADSP-1401 has been classified as a Class 1 device.

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*.

SWITCHING CHARACTERISTICS[3]

Parameter		J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade[2] Min	S Grade[2] Max	T Grade[2] Min	T Grade[2] Max	Unit
t_{HI}	Clock HI	50		40		60		50		ns
t_{LO}	Clock LO	40		30		50		40		ns
t_{IS}	Instruction Setup Time	36		30		45		40		ns
t_{DS}	Data Setup Time	10		*		15		15		ns
t_{IH}	Input Signal Hold Time	3		*		*		*		ns
t_{AD}	Address Delay[4] (C = 50pF)		35		25		45		35	ns
t_{AH}	Address Hold Time	3		*		1		1		ns
t_{ODD}	Output Data Delay (C = 30pF)		50		35		60		45	ns
t_{ODIS}	Output Data Disable Time		20		15		25		20	ns
t_{IFSM}	Input Flag Setup Time (IR0 masked)	15		10		20		15		ns
t_{IFSU}	Input Flag Setup Time (no constraints)	30		26		35		30		ns
t_{UIRS}	Upper Interrupts (IR_{8-5}) Setup Time	30		25		35		30		ns
t_{LIRS}	Lower Interrupts (IR_{4-1}) Setup Time	20		15		25		20		ns
t_{TSS}	Three-State (TTR) Setup Time	10		*		15		15		ns
t_{TSOV}	Three-State (TTR) Overlap Time (With Trap)		13		13		5		5	ns
t_{TSE}	Three-State (TTR) Disable Delay		20		15		25		20	ns
t_{IDL3}	IDLE-to-Three-State Disable Delay		20		15		25		20	ns
t_{TROV}	Trap (TTR) Overlap Time (With Three-State)		10		8		10		10	ns
t_{TRAD}	Trap (TTR) to Address Delay		60		45		70		55	ns

NOTES
*Specifications same as J grade.
[1]All specifications are over the recommended operating conditions.
[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-1401 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 three-state reference levels, which are shown in Figure 8. For capacitive loads greater than 100pF, we recommend the use of external buffers.
[4]Address delays may be derated from the specified 50pF test loading shown in Figure 12 by adding 7ns/50pF for increased capacitive loading.
Specifications subject to change without notice.

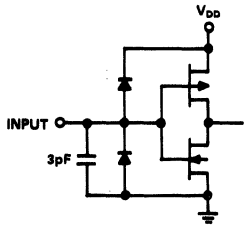


Figure 9. Equivalent Input Circuit

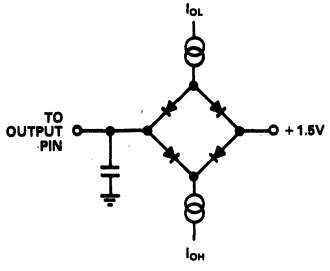


Figure 10. Normal Load for ac Measurements

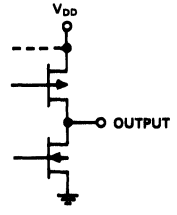


Figure 11. Equivalent Output Circuit

4.0 APPENDICES

4.1 Instruction Hold Control (IHC)

The IHC function allows external microcode width to be reduced by allowing the 1401's microcode field to be shared with another device. This sharing is accomplished by temporarily latching an instruction that is used repetitively within the ADSP-1401 and re-directing its microcode to a different device. Control of the latching is accomplished by the IHC instruction, which re-assigns the function of interrupt signal IR_1, becoming the latch/unlatch control line.

To use this mode, execute the IHC instruction, which sets status register bits SR_{5-4} to '10'. Interrupt line IR_1 now controls the instruction hold mode (not interrupt), so IR_1 must be masked. The shared signal, IR_5 (recall, IR_{8-5} and IR_{4-1} share the same pins), is still used normally, since it is active during clock low.

To initiate an instruction hold, execute the instruction to be repeated, while asserting IR_1 (HI) prior to the clock falling edge of the same cycle. For so long as IR_1 is kept high (on the falling edge of the clock), the instruction will repeat. All interrupts are automatically disabled while the instruction is held.

When IR_1 is needed for interrupts (instead of controlling the instruction hold mode) the IHC mode may be disabled by: executing one of the relative jump width control instructions; or, by changing status register bits SR_{5-4} directly. Prior to unmasking IR_1, execute the CAIR (clear all interrupts) instruction to clear the interrupt latch.

4.2 Programming Examples

The following examples are given to illustrate some fine points of programming the ADSP-1401.

4.2.1 Jump Register (See Figure 13a)

In this example, three jump registers (R_{3-1}) are loaded with external data and one (R_0) is loaded with the program counter, enabling a jump to the top-of-stack.

Current Address	Instruction Executed	Output Address	Comments	RSP
20	PSDRS	21	Push R_3	57
21	PSDRS	22	Push R_2	56
22	PSDRS	23	Push R_1	55
23	PSPC	24	Push PC (R_0 = 24)	54
24	Start of Loop . . .	25		
. . .	. . .	. . .		
30	. . .	. . .		
31	JRC (R_0)	32/24	Cond. Jump to [R_0] = 24	
32/24	. . .			

4.2.2 Return from Interrupt with Pending Interrupt (See Figure 13b)

This example shows the program flow when two interrupts occur in the same cycle or an interrupt is latched while another interrupt is being executed. The "Return from Interrupt" instruction (RTNIR) will execute one instruction of the mainline routine before servicing a pending interrupt since interrupts are not re-enabled until the end of the cycle. Here, $IV_7 = 60$ and $IV_3 = 21$.

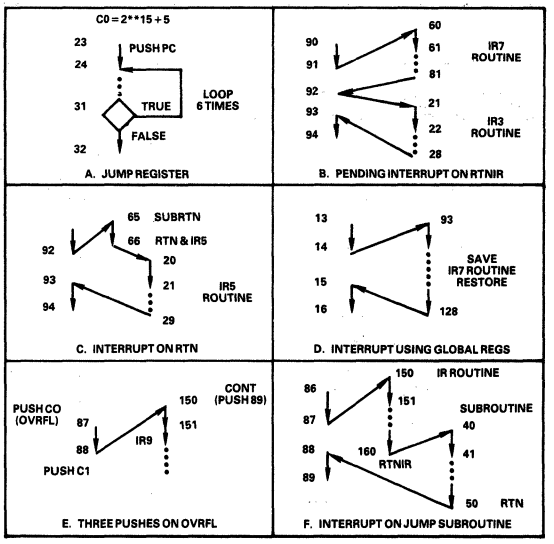


Figure 13. Programming Examples

Current Address	Instruction Executed	Output Address	Comments
89	. . .	90	. . .
90	. . .	91	Interrupts I_7 & I_3 valid.
91	. . .	60	IV_7 output. Instruction 91 still executed.
60	CONT	61	92 is pushed on stack.
61	. . .	62	. . .
. . .	. . .	. . .	
81	RTNIR	92	92 popped and interrupts re-enabled.
92	. . .	21	IV_3 output. Instruction 92 still executed.
21	CONT	22	93 pushed on stack.
22	. . .	23	. . .
. . .	. . .	. . .	. . .
28	RTNIR	93	93 is popped from stack.
93	. . .		

4.2.3 Interrupt on a Return from Subroutine (See Figure 13c)

If an interrupt occurs on a subroutine return, no instructions in the main program are executed prior to servicing the interrupt routine. Here, $IV_5 = 20$.

Current Address	Instruction Executed	Output Address	Comments
91	. . .	92	. . .
92	JSR	65	Jump to 65. 93 pushed.
65	. . .	66	IR_5 becomes valid.
66	RTN	20	IV_5 address output. 93 popped.
20	CONT	21	93 pushed.
. . .	. . .	. . .	. . .
29	RTNIR	93	93 popped.
93	. . .		

4.2.4 Interrupt Routine using Global Registers (See Figure 13d)

Current Address	Instruction Executed	Output Address	Comments
12	. . .	13	Mainline . . .
13	. . .	14	IR_7 occurs here.
14	CONT	93	Output IV_7.
93	PSSR	94	Push status register.
94	PSCNTR (C_3)	95	Save previous values . . .
95	PSCNTR (C_1)	96	. . .
96	PSGSP	97	. . .
97	WRSR	98	Write new values . . .
98	WRCNTR (C_3)	99	. . .
99	WRCNTR (C_1)	100	. . .
100	WRRSP	101	. . .
101	. . .	102	Begin interrupt servicing . .
. . .	. . .	. . .	. . .
123	. . .	124	End of interrupt service routine.
124	PPGSP	125	Pop in reverse order of pushes . . .
125	PPCNTR (C_1)	126	. . .
126	PPCNTR (C_3)	127	. . .
127	PPSR	128	. . .
128	RTNIR	15	Jump back to mainline.
15	. . .	16	. . .

4.2.5 Three Stack Pushes on Stack Overflow (See Figure 13e)
The four register buffer between the subroutine stack and the register stack will be filled with three values whenever the stack push that caused the overflow is followed by another instruction that causes a stack push. The second stack push occurs since the instruction that is interrupted (the second stack push) must complete internally to preserve the correct state of the ADSP-1401 after the interrupt. The third push occurs to provide the return address to the main program. The sequence is illustrated below. Assume that the address of the stack overflow service routine (IV_9) is at 150.

Current Address	Instruction Executed	Output Address	Comments
86	. . .	87	
87	PSCNTR (C_0)	88	The push causes a stack overflow.
88	PSCNTR (C_1)	150	The interrupted instruction executes.
150	CONT	151	89 is pushed onto the stack.
151	. . .		

4.2.6 Interrupt on Jump Subroutine Instruction (See Figure 13f)

Current Address	Instruction Executed	Output Address	Comments
86	. . .	87	Interrupt occurs to location 150
87	JSA (40)	150	
150	CONT	151	40 Pushed on stack
. . .	. . .	. . .	. . .
. . .	. . .	160	. . .
161	RTNIR	40	Return from interrupt
40	. . .	41	

4.3 Use of RAM by Multiple Subroutines
This diagram (Figure 14) shows the state of RAM after three nested subroutine calls.

Prior to the first subroutine call, the RSP was used to preload the bottom portion of the RAM with indirect jump addresses. Next, global jump registers were preloaded. In the mainline program, only global jump registers are used.

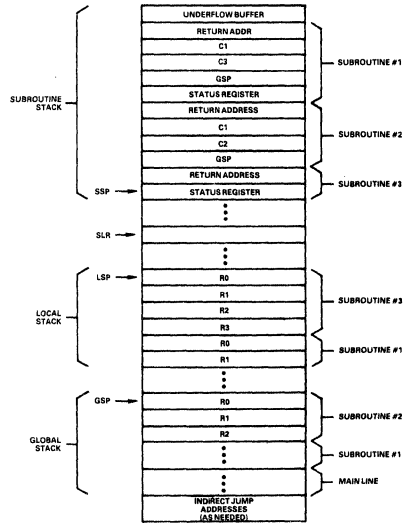


Figure 14. RAM Status after Subroutine Calls

The instruction calling the first subroutine pushes the return address of the main program onto the subroutine stack. The values of counters 1 and 3 are also pushed onto the stack to free counters 1 and 3 for use in subroutine #1. The GSP is saved since different routines will require different GSPs. Similarly, the status register of the main program is saved. As shown, routine #1 uses both global and local jump registers. It selects the GSP or LSP at the appropriate times in the routine by executing SGSP or SLSP instructions.

Routine #2 saves the return address, some counters, and the GSP for routine #1. Since no local registers are used in routine #2, none are loaded.

Routine #3 saves the return address and the status register. Since the GSP and counters are not used in this routine, they are not saved. After the new status register is loaded (selecting the LSP), local registers are pushed onto the stack.

4.4 Bus Drive Considerations with the Word-Slice Family
The various members of Analog Devices' Word-Slice family are designed with high-speed drivers on all output pins. This capability means that large peak currents may pass through the ground and V_{DD} pins when all the bus lines are simultaneously charging their load capacitance from LO to HI, or *vice versa*.

To calculate the peak current for a typical family member (such as the ADSP-1401 Program Sequencer), we assume that all output drivers are switching from a HI to a LO state. From a

fall time and capacitance measurement, we can determine that the peak current in each driver is:

$$I_{peak} = C_{load} \cdot \Delta V/\Delta t,$$

where $\Delta V/\Delta t$ is the initial slew rate.

In the case of the program sequencer, for an external load capacitance of 50pF and a measured slew rate of 0.6V/ns, the peak current will be about 30mA. Since there are 16 such drivers, the total peak current may approach 480mA!

The internal ground and supply lines may undergo a large disturbance during this transition unless the ADSP-1401 is tied to a solid ground plane and good high frequency decoupling is used (0.1μF ceramic between GND and V_{DD} as close as possible to the device). Otherwise, is it possible that internal data in the ADSP-1401 may be lost.

4.5 Mnemonics and Opcodes

Opcode bits "ii" select the relevant register (R_{3-0}) and/or counter (C_{3-0}). Opcode bits "cc" select the condition to be applied:

'00' UNCONDITIONAL
'01' NOT FLAG
'10' FLAG
'11' SIGN

The SIGN condition is precluded from instructions prefixed with "*".

Mnemonic	Opcode (I_{6-0})	Description
Jump and Branch Instructions:		
JPCOF	001 0101	IF FLAG: JUMP PC (*self*)
JPCNF	011 0101	IF NOT FLAG: JUMP PC (*self*)
JTWO	101 cc01	IF COND: JUMP PC+2 (*skip*)
JDA	111 cc11	IF COND: JUMP DATA, ABSOLUTE
JDR	111 cc01	IF COND: JUMP DATA, RELATIVE
JDI	101 cc10	IF COND: JUMP DATA, INDIRECT
JDRST	100 11ii	IF SIGN OF C_i: JUMP DATA, $C_i <= R_i$; ELSE, $C_i <= C_i - 1$
*JRC	110 ccii	IF COND: JUMP R_i
JRS	110 11ii	IF SIGN OF C_i: JUMP R_i, $C_i <= C_i - 1$
JSA	111 cc00	IF COND: JUMP SUB, ABSOLUTE
JSR	111 cc10	IF COND: JUMP SUB, RELATIVE
RTN	101 cc11	IF COND: RETURN FROM SUB
*BRANCH	100 ccii	IF SIGN OF C_i: JUMP R_i; ELSE, $C_i <= C_i - 1$, IF COND: JUMP DATA
Stack Operations:		
Subroutine Stack		
PSDSS	001 1110	PUSH DATA ONTO SS
PPSSD	011 1110	POP SS TO DATA PORT
WRSSP	000 1110	WRITE SSP
RDSSP	010 1100	READ SSP
DSSP	000 0010	DECREMENT SSP
Register Stack		
SGSP	000 0111	SELECT GSP
SLSP	000 0110	SELECT LSP
RDRSP	010 1111	READ RSP
WRRSP	000 1100	WRITE RSP
PSPC	010 0011	PUSH PC ONTO RS
PSGSP	000 0101	PUSH GSP ONTO SS
PPGSP	000 0100	POP GSP FROM SS
PSDRS	001 1111	PUSH DATA ONTO RS
PPRSD	011 1111	POP RS TO DATA PORT
AIRSP	010 10ii	ADD i TO RSP
S1RSP	000 1111	SUBTRACT 1 FROM RSP
S4RSP	011 1100	SUBTRACT 4 FROM RSP

Status Register Bit Assignments

Bit#	Function (HI/LO)
SR_{15}	IR_9 Mask Bit
•	•
•	•
•	•
SR_6	IR_0 Mask Bit
SR_{5-4}	Relative Jump Width Selection: '00' = 16-bit relative address width '01' = 8-bit width '10' = IHC Mode (8-bit width) '11' = 12-bit width
SR_3	Select GSP/LSP
SR_2	Enable/Disable Interrupts
SR_1	Set/Clear Sign Bit
SR_0	Select Transparent/Latched Interrupts

Status Register Operations:		
RDSR	010 1110	READ SR
WRSR	001 1100	WRITE SR
PSSR	010 0001	PUSH SR ONTO SS
PPSR	010 0010	POP SR FROM SS
Counter Operations:		
WRCNTR	011 10ii	WRITE C_i
CLRS	001 0100	CLEAR SIGN BIT
SETS	011 0100	SET SIGN BIT
PSCNTR	000 10ii	PUSH C_i ONTO SS
PPCNTR	001 10ii	POP C_i FROM SS
DCCNTR	011 00ii	DECREMENT C_i
IFCDEC	101 cc00	IF COND: DECREMENT C_0
Interrupt Control:		
CCIR	001 0001	CLEAR CURRENT INTERRUPT
CAIR	000 0001	CLEAR ALL INTERRUPTS
RTNIR	000 0011	RETURN FROM INTERRUPT
RDIV	010 1101	READ INTERRUPT VECTOR AND INCREMENT IVP
WRIV	000 1101	WRITE INTERRUPT VECTOR AND INCREMENT IVP
IRMBC	001 0011	IR MASK BITWISE CLEAR
IRMBS	001 0010	IR MASK BITWISE SET
DISIR	001 0110	DISABLE INTERRUPTS
ENAIR	011 0110	ENABLE INTERRUPTS
SLIR	001 0111	SELECT LATCHED INTERRUPTS
STIR	011 0111	SELECT TRANSPARENT INTERRUPTS
SLRIVP	001 1101	WRITE $SLR <= D_{5-2}$ AND $IVP <= D_{15-12}$
Relative Address Width Controls:		
REL16	010 0100	SELECT 16-BIT RELATIVE ADDRESSING
REL12	010 0111	SELECT 12-BIT RELATIVE ADDRESSING
REL8	010 0110	SELECT 8-BIT RELATIVE ADDRESSING
Miscellaneous Instructions:		
CONT	000 0000	CONTINUE
IDLE	001 0000	IDLE
IHC	010 0101	ENABLE INSTRUCTION HOLD CONTROL
WCS	010 0000	WRITE CONTROL STORE

ADSP-1401 PIN CONFIGURATIONS

DIP
D-48A
N-48A

PIN	FUNCTION	PIN	FUNCTION
1	D7	48	D6
2	D8	47	D5
3	D9	46	D4
4	D10	45	D3
5	D11	44	D2
6	D12	43	D1
7	D13	42	D0
8	D14	41	CLK
9	D15	40	FLAG
10	EXIR1	39	I6
11	EXIR2	38	I5
12	GND	37	V_{DD}
13	EXIR3	36	I4
14	EXIR4	35	I3
15	TTR	34	I2
16	Y15	33	I1
17	Y14	32	I0
18	Y13	31	Y0
19	Y12	30	Y1
20	Y11	29	Y2
21	Y10	28	Y3
22	Y9	27	Y4
23	Y8	26	Y5
24	Y7	25	Y6

PLCC
P-52

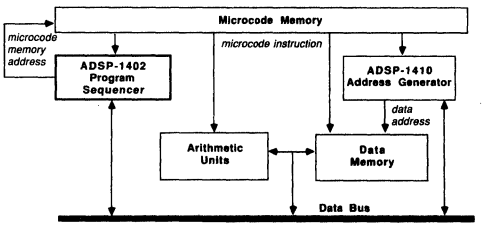
PIN	FUNCTION	PIN	FUNCTION
1	D7	52	D6
2	D8	51	D6
3	D9	50	D4
4	D10	49	D3
5	D11	48	D2
6	D12	47	D1
7	GND	46	GND
8	D13	45	D0
9	D14	44	CLK
10	D15	43	FLAG
11	EXIR1	42	I6
12	EXIR2	41	I5
13	GND	40	V_{DD}
14	EXIR3	39	I4
15	EXIR4	38	I3
16	TTR	37	I2
17	Y15	36	I1
18	Y14	35	I0
19	Y13	34	Y0
20	GND	33	GND
21	Y12	32	Y1
22	Y11	31	Y2
23	Y10	30	Y3
24	Y9	29	Y4
25	Y8	28	Y5
26	Y7	27	Y6

ANALOG DEVICES

Word-Slice Program Sequencer

ADSP-1402

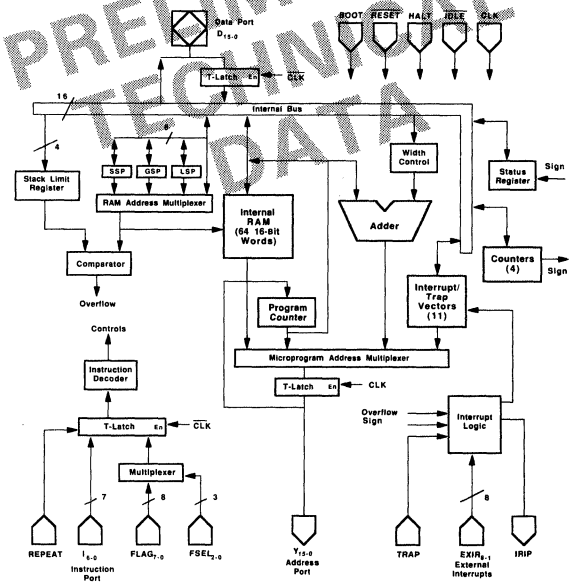
FEATURES
16-Bit Microcode Addressing Capability
Look-Ahead™ Pipeline
Extensive Interrupt Processing with Eleven On-Chip Interrupt Vectors
Four Event Counters to Support Looping
Absolute, Relative and Indirect Addressing
50ns Cycle Time
64-Word RAM for Storing:
Subroutine Linkage
Jump Addresses
Counters
Status Register
1μm CMOS Technology
84-Pin Grid Array Package



Word-Slice® Microcoded System with ADSP-1402

GENERAL DESCRIPTION

The ADSP-1402 Program Sequencer is an instruction-compatible upgrade to the ADSP-1401. It can be used with high speed arithmetic units and provides many features to simplify the design of microcoded systems. Among the devices it supports are the ADSP-3212 Floating-Point Multiplier, the ADSP-3222 Floating-Point ALU and the ADSP-3128A Register File. The ADSP-1402 is functionally identical to the ADSP-1401, except for the changes described in this section. A block diagram of the ADSP-1402 is shown below. For a detailed description of the architecture and instruction set of the ADSP-1402, see the *Word-Slice® User's Manual* and the *ADSP-1401 Data Sheet.*



ADSP-1402 Block Diagram

Look-Ahead is a trademark of Analog Devices, Inc.
Word-Slice is a registered trademark of Analog Devices, Inc.

The ADSP-1402 is a high speed microprogram controller optimized for the demanding sequencing tasks found in digital signal processors and general purpose computers. In addition to high speed and large addressing range (64K of program memory), this Word-Slice component has unique features that make it highly versatile:

- On-chip storage and control of ten prioritized and maskable interrupts plus a nonmaskable trap,
- Four decrementing event counters,
- Absolute, relative and indirect addressing capability,
- Download capability (writeable control store) and
- A dynamically reconfigurable 64-word RAM.

The ADSP-1402 microprogram sequencer's main task is to provide the appropriate microprogram addressing to support programming requirements, such as looping, jumping, branching, subroutines, condition testing and interrupts. An internal Look-Ahead pipeline, controlled by both phases of the clock, allows the ADSP-1402 to satisfy these requirements at very high speed.

During each microinstruction, the ADSP-1402 monitors the conditions and instructions to determine the next microprogram address. This address can come from one of several sources: the stack, the jump address space in the on-chip RAM, the data port, the interrupt vectors or the microprogram counter. In all cases, the next address is available in a single cycle. An extensive set of conditional instructions is also available, including jumps, branches, subroutines, interrupts and writeable control store. Eight multiplexed flag inputs can be used as external conditions for these instructions.

The ADSP-1402's internal 64-word RAM is user configurable into three regions: subroutine stack, register stack and indirect jump address space. The subroutine stack is used for linking interrupts and subroutines and, during their execution, allowing the storage of system states. The register stack can be used to store sets of jump addresses; each set can be associated with a particular level of interrupt or subroutine (both local and global stacks are provided). Indirect jump capability is also supported, addressing for which is provided at the data port.

Interrupts are handled entirely on chip. The ADSP-1402's internal interrupt control logic includes registers for eight external (user) interrupt vectors, a mask register and a priority decoder. Two additional vectors are reserved for internally generated interrupts resulting from counter underflow and stack limit violation, and a special vector is provided for the nonmaskable trap interrupt. A stack limit violation is caused by stack overflow, underflow or collision. A mechanism is provided for recovering from stack violations. Trap interrupts have the highest priority of all interrupts, and the stack limit violation interrupt has the second highest priority.

The ADSP-1402's four decrementing 16-bit counters are used to track loops and events. These counters generate a signal when negative. This negative condition is available to several conditional instructions and can also trigger an internal interrupt.

CHANGES FROM THE ADSP-1401

TTR Input

The ADSP-1401 TTR (Trap/Tristate/Reset) input is eliminated in the ADSP-1402. In its place are separate $\overline{\text{RESET}}$ and TRAP control pins. The tristate function is implemented with the the $\overline{\text{IDLE}}$ pin as described under *Idle and Halt,* below.

Reset

The default reset function in the ADSP-1402 is similar to that of the ADSP-1401. While $\overline{\text{RESET}}$ is LO and $\overline{\text{IDLE}}$ is HI, the ADSP-1402 outputs H#0000 on its address port. When $\overline{\text{RESET}}$ goes HI, the address port remains at H#0000 for one clock cycle (the first cycle of normal operation). As with the ADSP-1401, the first ADSP-1402 instruction must be a CONT instruction.

The ADSP-1402 also provides an alternate reset function in which the address port is placed in a high impedance state. If the $\overline{\text{IDLE}}$ input is asserted LO during reset, the address port is tristated rather than outputting H#0000. Asserting $\overline{\text{IDLE}}$ during reset does not affect internal operation, only the address port. When $\overline{\text{RESET}}$ goes HI, the ADSP-1402 outputs H#0000 for one clock cycle.

Boot (WCS)

The ADSP-1401 and ADSP-1402 implement a WCS (Writeable Control Store) instruction. This instruction places the ADSP-1401 or ADSP-1402 in a mode in which an active FLAG input increments the program counter (PC), decrements the C_0 counter and outputs the PC to the address port. This operation is used to synchronize address sequencing for downloading microcode from a host. The usual way to exit this mode is by an interrupt, from either an external interrupt or the internal counter underflow (of C_0 in this case).

The ADSP-1402 also provides a pin that allows external hardware control of a download. The BOOT input of the ADSP-1402 controls the operation for downloading microcode in much the same way as the WCS instruction. The boot operation, although slightly more restricted compared to the WCS operation, requires no external circuitry.

Note: $\overline{\text{IDLE}}$ must be HI and TRAP must be LO while the boot function is being used. $\overline{\text{RESET}}$ must be active when BOOT is asserted and remain active until BOOT is deasserted.

In the cycle that BOOT is asserted, the ADSP-1402 outputs H#0000 on the address port and sets the PC to H#0000. When $FLAG_0$ is asserted, the PC is incremented and its new value is output on the address port. The ADSP-1402 remains in this mode until the BOOT pin is deasserted. Thus, no interrupt is required to end the download.

The system clock must be stable and $\overline{\text{RESET}}$ must be asserted for a minimum number of cycles before BOOT is asserted and after BOOT is deasserted. $\overline{\text{IDLE}}$ must be HI and TRAP must be LO for the entire time that BOOT is asserted. When BOOT is active, $FLAG_0$ is edge-sensitive (therefore, it cannot be asserted more than every other cycle). $FLAG_0$ must also meet minimum setup and hold times.

Trap
The ADSP-1402 trap function is controlled by the TRAP input. The TRAP signal must be asserted at least t_{TS} before the next rising clock edge and must be held at least t_{TH} after the rising clock edge. In addition, TRAP must not change state (HI or LO) t_{TS} before or t_{TH} after the rising clock edge, and it must meet a minimum pulse width specification.

The nonmaskable TRAP input on the ADSP-1402 has a dedicated interrupt vector (IV_{10}) that is separate from the IR_9 (stack over/underflow) vector (unlike the ADSP-1401, in which Trap and IR_9 share the same vector). As with the ADSP-1401, the TRAP signal may require a clock skip to allow time to fetch a new instruction. A block diagram of an example circuit for implementing a clock skip is shown in Figure 1. TRAP aborts the current instruction and pushes its address onto the subroutine stack.

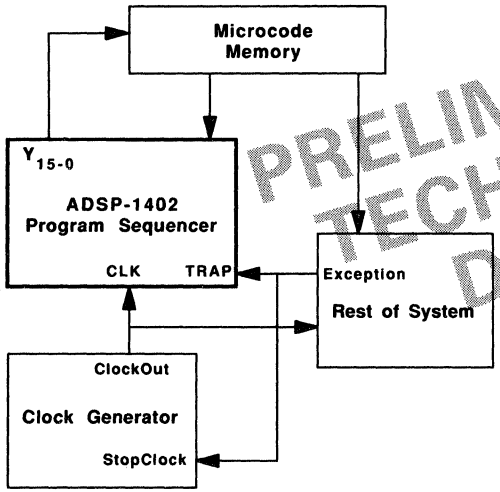


Figure 1. Example Clock Skip Circuit

External Interrupts
The eight external interrupts of the ADSP-1402 are input on eight separate pins, $EXIR_{8-1}$ (whereas the eight external interrupts of the ADSP-1401 are time-multiplexed into four inputs). All external interrupts are latched on the rising edge of the clock. The ADSP-1402 outputs the interrupt vector address in the same clock cycle in which the interrupt is latched.

Interrupt masking and enabling in the ADSP-1402 is the same as in the ADSP-1401. Interrupts on the ADSP-1402 are prioritized in descending numerical order; Trap has the highest priority, IR_9 has the next highest, and IR_0 has the lowest.

Interrupt In Progress (IRIP)
The ADSP-1402 has an internal Interrupt In Progress (IIP) bit that indicates when it is processing an interrupt (IR_9–IR_0). The ADSP-1402 also has an internal Trap In Progress (TIP) bit that indicates when a trap is being processed. The IRIP output flag is the logical OR of the IIP and TIP bits.

If TIP is set, the CCIR (Clear Current Interrupt) and RTNIR (Return From Interrupt) instructions reset TIP without affecting IIP. If TIP is not set, however, then executing one of these instructions resets IIP. Executing the CAIR (Clear All Interrupts) instruction resets both TIP and IIP. Thus, unlike in the ADSP-1401, a trap service routine can be nested inside an interrupt service routine; the return from the trap service routine will not eliminate the Interrupt In Progress status.

Flag Inputs
The ADSP-1402 has eight external flag inputs ($FLAG_{7-0}$). These eight input flags are multiplexed on-chip into one signal that is equivalent to the FLAG input on the ADSP-1401. Three external control bits select one of the eight input flags. The multiplexed flag signal is latched during clock HI and transparent during clock LO. During a Boot or Writeable Control Store operation, the multiplexer automatically selects $FLAG_0$.

Idle and Halt
The ADSP-1402 has two controls for stopping internal operation, one which tristates the address and data ports ($\overline{IDLE}$) and one which does not (HALT). Both perform functions similar to that of the ADSP-1401 IDLE instruction, which is functional but obsolete on the ADSP-1402.

Note: The IDLE instruction must *not* be input to the ADSP-1402 with either $\overline{IDLE}$ or HALT asserted; otherwise, the ADSP-1402 will not function properly.

The $\overline{IDLE}$ pin is useful for implementing multitasking in systems with multiple sequencers. $\overline{IDLE}$ removes the ADSP-1402 from the address and data buses, allowing another sequencer to drive them.

$\overline{IDLE}$ requires a minimum setup and hold time to the rising clock edge. When $\overline{IDLE}$ is asserted, the ADSP-1402 finishes executing the current instruction, and then the internal clock of the ADSP-1402 is stopped, freezing internal operation. At the next rising edge of the CLK input, both the address port and the data port are tristated, and the next instruction is latched but not executed. Fetching and execution of new instructions are inhibited until $\overline{IDLE}$ is deasserted. Interrupts are not latched, and traps are ignored as well. When $\overline{IDLE}$ is deasserted, normal operation continues at the next rising clock edge with the previously latched instruction.

The ADSP-1402 HALT input can be used to stretch the internal ADSP-1402 clock. HALT is primarily intended to implement wait states or to be used in conjunction with TRAP to handle exceptions.

HALT stops internal operation without tristating the address and data ports. It halts internal operation at the next rising edge of the CLK input after HALT is asserted. The address port and the data port are not updated; both ports maintain the states current at a time when HALT is asserted. No new instruction is latched. During HALT, fetching and execution of new instructions are inhibited. Interrupts are not latched; however, unlike during IDLE, active TRAP inputs *are* recognized and processed. The ADSP-1402 latches and executes its next instruction and updates the address and data ports at the next rising clock edge after HALT is deasserted.

Repeat

The REPEAT input causes the ADSP-1402 to repeat the next instruction (the one being set up at the same time as REPEAT) for one clock cycle. This input performs the same function as the ADSP-1401 IR_1 input in IHC (Instruction Hold Control) mode. The ADSP-1402 repeats the instruction as long as REPEAT stays asserted.

Interrupts cannot be serviced while the REPEAT pin is active because the ADSP-1402 ignores its instruction port; however, interrupt requests are still latched. Because TRAP is not latched, it should not be used while REPEAT is active.

The REPEAT input is dedicated to the repeat function; the ADSP-1402 has no IHC mode. In the ADSP-1401, the IHC instruction activates the IHC mode and selects an 8-bit relative jump offset width. For compatibility, the IHC instruction in the ADSP-1402 also selects an 8-bit relative jump offset width (the same effect as the REL8 instruction).

Data Port

The ADSP-1402 has a full-cycle data port rather than the half-cycle data port of the ADSP-1401. Instructions that write data out of the ADSP-1402 drive the bus for a full cycle. Instructions that read data into the ADSP-1402 require data to be valid a specified time before and after the clock rising edge. To avoid bus contention, therefore, an ADSP-1402 instruction that outputs data on the data port cannot be followed by an instruction that reads data from the port; a NOP cycle must occur between the two instructions.

The data port output drivers are tristated unless a data output is being performed.

Power and Ground

The ADSP-1402 has nine power pins and nine ground pins.

PIN LIST

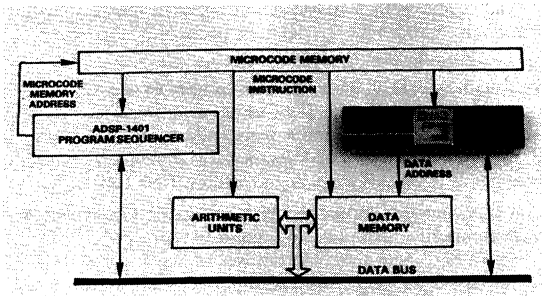
Name	Type	Function
D_{15-0}	Bidirectional	Data Port
Y_{15-0}	Output	Address Port
IRIP	Output	Interrupt in Progress
I_{6-0}	Input	Instruction Port
$EXIR_{8-1}$	Input	External Interrupts
$FLAG_{7-0}$	Input	Flags
$FSEL_{2-0}$	Input	Flag Select
CLK	Input	Clock
TRAP	Input	Trap
$\overline{\text{RESET}}$	Input	Reset
$\overline{\text{IDLE}}$	Input	Idle
REPEAT	Input	Repeat Instruction
HALT	Input	Halt
BOOT	Input	Boot (WCS) Mode

ANALOG DEVICES

Word-Slice Address Generator

ADSP-1410

FEATURES
16-Bit Addresses with Higher Precision Options
30ns Address Output Delay @ 11.1MHz Operation
Look-Ahead™ Pipeline
Versatile Addressing Hardware:
30 16-Bit Registers
16-Bit ALU with Left/Right Shift & Carry I/O
Comparator
Bit Reverser
Dual Ports
Powerful Single-Cycle Looping Instructions
375mW Maximum Power Dissipation with CMOS Technology
48-Pin DIP, 52-Lead PLCC



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1410

3

GENERAL INFORMATION

The ADSP-1410 is a fast, flexible address generator optimized for digital signal/array processors and other high-performance computers. This low-power CMOS device rapidly generates the data memory addresses required by routines such as digital filters, FFTs, matrix operations, and DMAs. With its 16-bit architecture, registers, dual ports, and speed, this Word-Slice® component improves performance and reduces board space substantially relative to bit-slice solutions.

The ADSP-1410's architecture features a 16-bit ALU, a comparator, and 30 16-bit registers. The registers are organized into four files: sixteen address (R) registers, six offset (B) registers, four compare (C) registers, and four initialization (I) registers.

The ADSP-1410 rapidly executes key address generating operations. In a single instruction cycle, the device can:

- output a 16-bit memory address;
- modify this memory address; and,
- detect when the address value has moved to or beyond a pre-set boundary and conditionally loop back to the top of a circular buffer.

Consequently, circular buffers and modulo addressing for data memories can be implemented without overhead.

The ADSP-1410's 10-bit microcode instructions include commands for looping, register read/writes, internal data transfers, and logical/shift operations. Instructions are normally supplied from an external source. However, an internal Alternate Instruction Register (AIR) can provide the instruction under external control, allowing microcode to be conserved in many applications.

The ADSP-1410 has a 16-bit address (Y) port for outputting addresses and a 16-bit data (D) port for I/O between internal and external registers. Also, an internal path allows external data, provided via the D port, to serve as an ALU source and/or to be directly output over the Y port for a DMA capability.

Double-precision (30-bit), single-cycle addressing can be performed by cascading two ADSP-1410's, with the MSB of each chip's D and Y port dedicated to interchip communication. Alternatively, a single AG can provide double-precision addresses at a rate of one per two clock cycles.

The Look-Ahead pipeline eliminates the need for an external microcode pipeline register by internally latching instructions and addresses; microcode bits may be directly routed to the ADSP-1410 from microcode memory. Logically, the Look-Ahead pipeline is split into two halves: the first, located at the instruction (and data) port; and the second, located at the address port. Each half of the pipeline (input vs. output) has a transparent latch which operates out of phase with the other: the address latch is transparent during the first half of the cycle (clock HI), while the input latches (instruction and data) are transparent during the second half of the cycle (clock LO). This complementary arrangement allows new instructions to be decoded (in preparation for the following cycle) while the program address for the current cycle is held steady.

ADSP-1410 OVERVIEW

Digital Signal Processing (DSP) and array processing systems require fast, flexible address generation circuitry. An Address Generator (AG) supplies the address of a location in data or coefficient memory. The value residing at the specified address is fetched and fed to an arithmetic unit for processing. The AG must then modify the address pointer in anticipation of the next data fetch. For algorithms that repetitively loop through data buffers, the AG may need to compare the address to a buffer end and conditionally loop back to the top of the buffer. Finally, to maximize throughput, an AG must perform its addressing tasks rapidly and without overhead.

With the ADSP-1410, 16-bit pointers to memory are stored in an address (R) register file. Since an AG must track several pointers concurrently, sixteen R registers, denoted R_n, are provided. If we denote Y as the address port, the operation "$Y \leftarrow R_n$" corresponds to the AG supplying an address from register R_n.

After supplying an address, the AG must update the pointer for the next memory fetch. The updating may be as simple as an increment but, more generally, involves adding or subtracting an arbitrary offset value. Also, algorithms generally access several different offset values. To this end, the AG provides six offset registers, denoted B_m, and can execute in a single-cycle the core operation:

$$Y \leftarrow R_n;\ R_n \leftarrow R_n + B_m.$$

In DSP applications, data arrays are often addressed as circular buffers. That is, when addressing reaches the buffer end, it wraps back to the beginning of the buffer. To implement this looping, the AG compares the supplied address to one of four compare registers, denoted C_j. If the address has moved to or beyond the end of the boundary ($R_n \geq C_j$), the device can transfer an initialization register value, denoted I_j, to the register ($R_n \leftarrow I_j$); otherwise, it is updated in normal fashion ($R_n \leftarrow R_n + B_m$). To minimize overhead, the AG can execute normal updates while also performing conditional re-initializations; again, in one core operation:

$$Y \leftarrow R_n;\ \text{IF } (R_n \geq C_j)\text{: } R_n \leftarrow I_j;\ \text{ELSE } R_n \leftarrow R_n + B_m.$$

Since the above instruction handles the looping required of circular buffer addressing, it is termed a looping instruction. To a large extent, the ADSP-1410's architecture and instruction set revolve around efficient implementation of this instruction. However, many variations of this instruction are supported on the device and spelled out in the following sections.

ADDRESS SOURCES
- Sixteen internal R registers
- External data provided over the D port

OFFSET SOURCES
- Six internal B registers
- Data Port

OFFSET OPERATIONS
- Increment ($R_n \leftarrow R_n + 1$)
- Decrement ($R_n \leftarrow R_n - 1$)
- Add Offset ($R_n \leftarrow R_n + B_m$)
- Subtract Offset ($R_n \leftarrow R_n - B_m$)
- Single-Bit Left/Right Shifts
- Logical Operations (AND, OR, XOR)

CONDITIONAL RE-INITIALIZATION
- Independent Inhibit/Enable for each of four initialization registers
- Conditional AIR execution (used for true modulo addressing)

OUTPUT/UPDATE SEQUENCE
- Normal (Pre-Update) Mode (output the address before update)
- Post-Update Mode (output the address after update)

PRECISION
- Single chip supplies 16-bit addresses
- Two chips cascaded provide 30-bit addresses
- One chip provides 30-bit addresses in two cycles

ADSP-1410 PIN ASSIGNMENTS

PIN NAME	DESCRIPTION
$Y_{15} - Y_0$	The address (Y) output port. In single-chip/double-precision mode, the MSB (Y_{15}) indicates whether the supplied address is the MSW or LSW (see Precision Modes). In two-chip/double-precision mode, the MSB conveys the carry/shift bit from the Least Significant (LS) to the Most Significant (MS) chip.
$D_{15} - D_0$	The bi-directional data (D) port. In two-chip/double-precision addressing mode, the MSB (D_{15}) of this port conveys CMP status from the partner chip.
$I_9 - I_0$	The instruction port.
CMP/Z	A dual function pin. Looping instructions, which compare address register values to compare register values, assert this pin HI to convey CMP status if i) $R \geq C$ for positive offsets, or ii) $R \leq C$ for negative offsets. Logical/Shift instructions assert this pin HI to convey the ZERO status of the result.
DSEL	Data Select control. Asserting this control HI causes data set up on the data port to substitute for the R value specified in the instruction.
AIR Enable	Alternate Instruction Register control. Asserting this control HI causes the device to execute an instruction stored in the internal AIR, rather than the instruction set up on the instruction port.
CLK	Clock
V_{dd}	+5 Volt Power Supply
GND	Ground

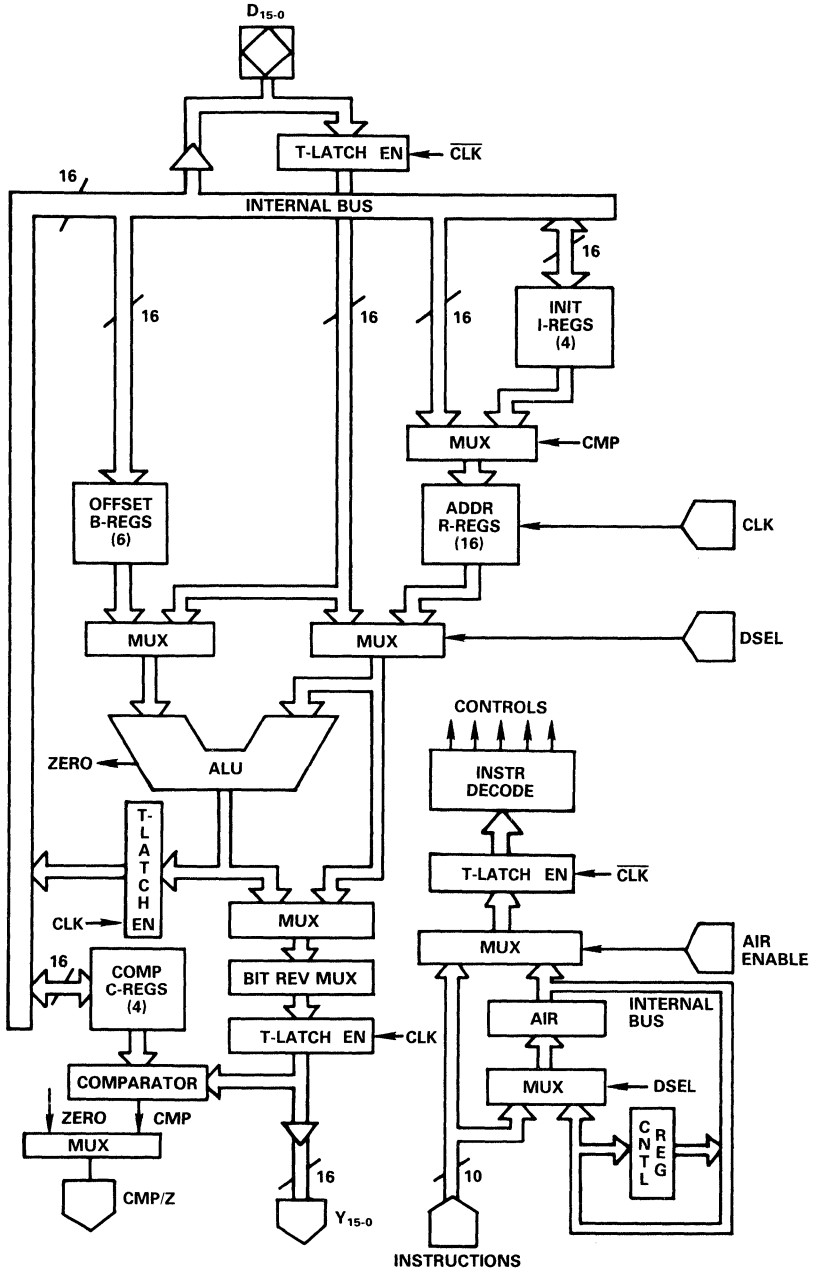


Figure 1. ADSP-1410 Functional Block Diagram

ARCHITECTURE

After discussing the architecture of the ADSP-1410, different operating modes of the ADSP-1410 are detailed, followed by a description of the ADSP-1410's method of operation: including timing concerns and instructions. Brief applications information is then presented, and the data sheet concludes with a section on MNEMONICS AND OPCODES.

The ADSP-1410's architecture (Figure 1) features four register files, an ALU, a Comparator, an Alternate Instruction Register (AIR), and a Control register. External interfaces include a 10-bit instruction port, a 16-bit data (D) and address (Y) ports, a DSEL (Data Select) control pin, an AIR Enable control pin, and a status flag.

Instruction Port

The microcode controlling the ADSP-1410 is supplied over the 10-bit wide INSTRUCTION PORT. The instruction word, I_{9-0}, is latched prior to the instruction decoder during phase one (clock HI) and is passed during phase two (clock LO). In addition to the microcode, two dedicated control pins affect the device's operation: the DSEL pin (see Y Port, D Port, and DSEL Control Pin); and the AIR Enable pin (see Alternate Instruction Register and AIR Enable). These pins are considered instruction bits, and latched as described above.

Y Port, D Port, and DSEL Control Pin

The ADSP-1410 has two 16-bit ports: a DATA (D) PORT and an ADDRESS (Y) PORT. The output drivers of both ports are three-state disabled unless an instruction specifies an output.

Addresses supplied to external data memory are output over the unidirectional Y port. The address supplied may come from one of three sources: an internal address (R) register, the data (D) port, or the ALU. The DSEL (Data Select) pin controls whether an R register (DSEL LO) or external data (DSEL HI) is the address source. The address source can either be directly output over the Y port, or passed through the ALU for modification prior to output (see Pre-Update Mode versus Post-Update Mode). Hardware three-state output control of the Y port is possible (see note in "Alternate Instruction Register and AIR Enable" section). Finally, the address being output (direct or modified source) may be bit-reversed (see Bit Reverser).

The Y port has two modes of operation (see Transparent Mode versus Latched Mode). In the more commonly used latched mode, addresses are latched during phase two (clock LO). The transparent mode disables the output latch and may be used in conjunction with stopping the clock LO, allowing data to be passed through (directly, or modified by the ALU) the AG without performing updates.

Any internal register may be read or written via the ADSP-1410's D port. Also, external data can be supplied to the chip over this port for immediate addressing purposes.

Note:
The ADSP-1410 may power-up driving the data bus. Caution should be used to avoid creating a bus contention with other devices which may be sharing this bus. To prevent bus contention, the CLK input may be forced LO during power-up (disabling the output data drivers). During this time, a RESET instruction should be setup at the instruction port to be executed as the first operation when the clock starts up.

Registers

The ADSP-1410 has 30 16-bit registers, organized into four banks. Single-cycle transfers between certain register banks are supported.

Sixteen ADDRESS (R) REGISTERS hold memory address pointers. In the same cycle that a 16-bit R value is output over the address (Y) port, it may be incremented, decremented, offset, modified by a logical operation, or left/right shifted by one bit. The updated value is then written back into the original R location (pre-update mode). In post-update mode, the address is output after being modified. Any R value (or data, using DSEL) may be bit-reversed on output.

Six OFFSET (B) REGISTERS furnish a second operand to the ALU (the other, provided by an R register or the data bus) for modifying the address to be output. The B registers are partitioned into two, user-selectable (see Control Register: B Bank Select) banks and external data can substitute as an offset value whenever B_3 (bank one) or B_7 (bank two) is used (see Table IV).

Four COMPARE (C) REGISTERS supply one source to the on-chip comparator, whose other source is the address being output. When an address moves to or beyond a boundary set by the C value, the CMP flag goes active (HI).

Four INITIALIZATION (I) REGISTERS can—conditional on the CMP flag going active—overwrite any R value, allowing overhead-free branches to the top of an addressing loop. Note that I and C registers are always paired. Conditional re-initializing of R registers may be independently inhibited for individual I registers (see Control Register CR_{3-0}).

ALU and Shifter

The ADSP-1410's 16-BIT ALU performs adds, subtracts, and logical operations. Usually, one source is an offset (B) register, while the other is an address (R) register. However, external data provided via the D port may substitute either for an R register (under the control of the DSEL pin), or a B register (using B3 or B7).

For two-chip/double-precision ALU operations, CARRIES into the MS chip and out of the LS chip (CS_{in} and CS_{out}) are conveyed via the Y_{15} pin (see Precision Modes).

The ALU also contains the logic required for single-bit SHIFTS of a supplied R register. Left shifts are logical, while right shifts are arithmetic. In two-chip/double-precision shift operations, the Y_{15} pin conveys the shifted bit. In single-precision operation, the carry/shift status of the device cannot be monitored.

The destination of an ALU or shift result is always the source R register location specified in the instruction—even if external data is the source. If the post-update mode is used, the ALU/shift result is sent directly over the address (Y) port on the current cycle (in addition to being returned to the source R location).

Alternate Instruction Register and AIR Enable

The ALTERNATE INSTRUCTION REGISTER (AIR) is a 10-bit register which may be loaded with any instruction. On any cycle that the AIR Enable pin is asserted, the device will execute the instruction held in the AIR, rather than the instruction set up on the instruction pins (except for the RST instruction).

The AIR's principal purpose is to conserve microcode. One way to conserve microcode is to load a frequently-used instruction (e.g., a looping instruction) into the AIR. Then, this instruction is executed simply by asserting the device's AIR Enable pin—temporarily suspending the need for external microcode.

The AIR can also conserve microcode in applications using multiple AGs (e.g., double-precision or high-throughput systems). If the AGs generally execute identical instructions, external microcode may be significantly reduced if they share a common microcode instruction field. During some cycles, however, it may be crucial for an AG to execute an instruction different from the common instruction—something which the AIR and its enable pin allow. For example, a NOP instruction can be loaded into an AG's AIR; anytime the AIR Enable pin is asserted, the AG will be selectively "put to sleep" (I/O pins three-state disabled; no change in internal state).

The AIR register may be read over the data port (D_{9-0}) in a single cycle. As Table I shows, the AIR may be written via the data port (D_{9-0}) or the instruction port. If the instruction written into the AIR is provided via the instruction port, two cycles are required. This method allows the AIRs of two or more AGs sharing microcode to be selectively loaded by differentially asserting their DSEL pins. Note that if the DSEL pin is LO during the *entire* second phase (clock LO) of the LDA instruction, no AIR loading occurs. This implicitly requires that DSEL be setup accordingly prior to the start of the LDA instruction, as it is latched during phase one (clock HI).

INSTRUCTION LOADED INTO THE AIR VIA THE:	
DATA PORT	**INSTRUCTION PORT**
1. Execute "Write AIR" instr.	1. Execute "Load AIR" instr. 2. Provide instr. on instr. port and assert DSEL pin.

Table I. Options for Reading and Writing the AIR

A second method exists for executing the instruction in the AIR. Looping instructions compare an address (R) value to a compare (C) value and, if the address has moved to or beyond a pre-set boundary, the CMP flag goes HI. If CR_{10} (see Control Register and Conditional AIR Execute Mode) is set, a true comparison causes the device to execute its next instruction from the AIR (see Table III.) This capability facilitates no-overhead modulo addressing (see application note: Modulo Addressing).

Note:
The AIRE pin may be used to control the Y port output drivers by loading a NOP into the AIR register; the AIRE pin becomes dedicated to three-state control of the Y port. This technique supports connection of multiple address sources to the same bus.

Flags and Comparator
The ADSP-1410 has two internal flags—CMP and ZERO—that share the external CMP/Z pin. The CMP flag, set by the comparator, is affected by looping instructions. The ZERO flag is set whenever a Logical/Shift instruction has a zero result. In cycles that do not affect the CMP or ZERO flag, the CMP/Z flag pin defaults LO.

As Table II shows, the CMP flag goes HI whenever the supplied address moves to or beyond a boundary set by the specified C register. The address that is compared to the C value is always the address that is output—even in post-update mode. R, C, and B values are treated as unsigned integers by the Comparator.

Twos-Complement Offsets
Negative offsets are generally handled by the $R \leftarrow R - B$ instruction. However, if for some reason the user is interpreting offset values as negative twos complement numbers, the instruction $R \leftarrow R + B$ will cause the comparator to sense whether $R \geq C$ (when the condition $R \leq C$ is of interest). The user may account for this reversal (e.g., by monitoring for the CMP flag going LO, rather than HI), but looping instructions cannot be fully utilized.

ARITHMETIC OPERATION	CMP FLAG HIGH IF:
$R_n \leftarrow R_n + 1$ (YINC instruction)	$R_n \geq C_j$
$R_n \leftarrow R_n - 1$ (YDEC instruction)	$R_n \leq C_j$
$R_n \leftarrow R_n + B_m$ (YADD instruction)	$R_n \geq C_j$
$R_n \leftarrow R_n - B_m$ (YSUB instruction)	$R_n \leq C_j$

Table II. CMP Flag Truth Table

Alternating Offsets
If the microprogram switches between different offsets and the AG is in the normal, pre-update mode, the comparator logic may produce seemingly erroneous results because comparisons are not made until the cycle following the update. In pre-update mode, when a routine switches between positive and negative offsets, the comparator will check for wrong condition because the comparison is not made until the following cycle. The value in the compare register must anticipate the comparator sense reversal by one cycle.

Bit Reverser
Addresses can be bit-reversed as they are output, which is useful in algorithms such as the Fast Fourier Transform. The bit-reverse mapping is as follows, where K_j and Y_j denote the j^{th} bit of K (either an address register or the data bus) and Y (the address port), respectively.

$K_0 \rightarrow Y_{15}$
$K_1 \rightarrow Y_{14}$
•
•
•
$K_{15} \rightarrow Y_0$

Bit reversal only affects the value that appears on the address port; it does not affect the value returned to the R register location. The hardware bit reverser operates only on single-precision, 16-bit addresses. For details on software reversal of N-bit ($N<16$) fields, see the application note: Variable-Width Bit Reversing.

Control Register
The ADSP-1410's 11-bit CONTROL REGISTER (CR_{10-0}) may be read or written via the device's data port, D_{10-0}. Dedicated instructions are used to read or write the entire control register, or to set and clear individual bits (see Instruction Group 4). On power-up, the RST instruction clears the control register to all zeros automatically.

The following list shows the control register organization. If the bit(s) is set (HI), the specified mode is operative.

CR	Bit Assignment
3–0	*Re-Initialization Mask*: For looping instructions, enables conditional re-initialization of R registers with I registers. For example, setting bit 2 of the CR allows I_2 to re-initialize the selected R register if the address has moved to or beyond the boundary set by C_2.
5–4	*Precision Select*: 00 = single-precision mode; 01 = double-precision mode, LS chip; 10 = double-precision mode, MS chip; 11 = double-precision mode, single-chip.
6	*Transparent Mode*: Sets the address (Y) port to the transparent mode: otherwise, the Y port is latched during phase two.
7	*R Bank Select*: Selects the upper eight R registers as address sources for the YADD and YSUB instructions.
8	*B Bank Select*: Selects the upper four B registers as offset sources for all instructions.
9	*Post-Update Mode*: Sets the post-update mode (addresses supplied after updating).
10	*Conditional AIR Execute*: Sets the conditional AIR mode: allowing looping instructions to (conditional upon the CMP status going true) be fetched from the AIR on the next instruction, rather than the instruction port. Using this mode disables conditional re-initialization (of R by I on CMP) and forces the default update of R.

ADSP-1410 OPERATING MODES

The flexibility of the ADSP-1410 is enhanced by several optional modes of operation. These modes, governed by the control register, are discussed in detail in this section.

Precision Modes

Typically, the ADSP-1410 provides single-precision (16-bit) addresses. If greater addressing range is needed, double-precision (30-bit) addresses can be supplied. Two double-precision modes are supported—one with two chips cascaded and the other with a single chip. Specific instructions set these modes. Double-precision (single- or two-chip) bit-reversing is not supported.

Two-Chip/Double-Precision

(CR_{5-4} = "01" for LS chip; "10" for MS chip). In this mode, two ADSP-1410's are cascaded to generate double-precision addresses at a rate of one per cycle. Each address may be output, incremented, decremented or modified by an offset value, compared to a double-precision value, and conditionally re-initialized by a double-precision word. Alternately, double-precision logical/shift operations may be performed.

The Y and D ports of each chip are restricted to the lower 15 bits, freeing the MSBs of both devices to convey carry/shift and CMP status, respectively (see Figure 2). For double-precision adds/subtracts, the LS chip sends carry/borrow status over the Y_{15} pin; the MS chip uses Y_{15} to accept carry/borrow status from the LS. For left (right) shifts, the LS (MS) conveys the shifted bit over the Y_{15} pin.

Double-precision, conditional re-initializations are implemented by dedicating the D_{15} pin on each device's data port to receive the CMP status from the other. When performing a looping instruction, the MS chip generates a valid CMP flag on its CMP/Z output. For a logical or shift instruction, the CMP/Z outputs from both the LS and MS chips must be ANDed to produce a single valid ZERO flag. To ensure that this flag is valid on the next low-to-high transition of the clock, the output of the AND gate should be latched as shown in Figure 2. The ZERO flag is latched on the falling edge of the clock and held by the latch until the next falling edge.

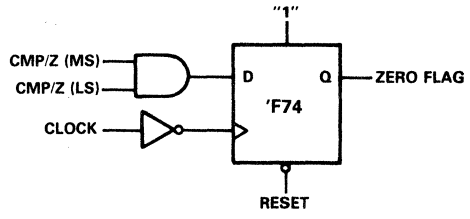


Figure 2. Valid Two-Chip Double-Precision ZERO (Logical Instructions)

In this mode, all values are 15-bit words. The 30-bit address is supplied in two 15-bit words over the Y_{14-0} pins of the two devices. Internally, the MS bit of each operand is zeroed prior to ALU operations, the MSB of the result then becoming the carry/shift bit. External data provided over the D port must be segmented with the 15 LSBs going to the LS chip and the 15 MSBs to the MS chip.

In two-chip/double-precision mode, both chips may share the same microcode instruction. The only complication to this sharing is in differentially initializing the MS and LS chips. Internal logic allows this initialization to be accomplished. Both chips are fed the instruction designating it as the LS chip. The assertion of DSEL on the intended MS chip during the SETP instruction reverses the two LS instruction bits (those defining the chip configuration to the control register), allowing both MS and LS designations to be performed simultaneously.

Single-Chip/Double-Precision

(CR_{5-4} = "11"). In this mode, double-precision (30-bit) addresses are generated at a rate of one every two cycles. Each address may be output, incremented or decremented, and compared to a double-precision compare (C) value. Logical/shift operations are also supported. Conditional re-initialization with I registers and the conditional AIR mode are not supported.

LSW operations are executed first, followed by MSW operations (with the exception of right shifts). Even-numbered R registers are reserved for LSWs, while odd registers are assumed to be MSWs. No such restrictions apply to B or C registers; MS *or* LS words may be held in *any* B or C register, but such allocation must be tracked by the user. After an operation involving LSW registers, the device stores the carry/shift bit (as appropriate) needed to complete the double-precision operation. On the next operation involving MSW registers, this intermediate value is utilized. Storage of the carry/shift bit occurs only on LSW operations, except for double-precision right shifting, which starts with the MSW. If non-addressing operations intervene, the intermediate value is not disturbed. The comparator will generate a meaningful CMP signal after each MSW operation.

In this mode, only the 15 LSBs of any register are used. The LSW and MSW addresses that are supplied are both 15-bit words. The Y_{15} (MSB) pin of the 16-bit address port designates

whether the address is the LSW (=0) or MSW (=1), and may be used to control an external mux. Note that the MSB of values provided via the data (D) port is not meaningful in this mode.

Transparent Mode
(CR_6 HI). In this mode, the address port is made transparent during the entire cycle, rather than only phase one. The transparent mode may also be used in conjunction with stopping the clock (LO), in which case the entire device behaves asynchronously and no updates are written internally.

Latched Mode
(CR_6 LO). In latched mode, output values are enabled during phase one and latched at the address (Y) port during phase two.

Use of the latched mode guarantees that outputs remain stable throughout the current cycle regardless of changes at the instruction port. This, in contrast to the transparent mode, in which such changes may occur quickly enough to alter the output before cycle end.

Post-Update Mode
(CR_9 HI). Addresses are output after the update operation. The delay between the start of phase one and output of a valid address is extended in this mode to allow for updating. The addresses output are equivalent to the values written back into the specified address (R) register. In this mode, external data may be brought on chip, modified and output—in a single clock cycle.

Pre-Update Mode
(CR_9 LO). This is the normal update mode in which addresses are output over the address (Y) port prior to update operations (increment, decrement, offset, shift, and logical)—allowing addresses to be generated at maximal speed. Note however, that this mode requires two cycles to bring external data on chip, modify it, and supply it as an address.

Conditional AIR Execute Mode
(CR_{10} HI). In this mode, a valid CMP flag on looping instructions causes the next instruction to be executed from the AIR. The MODULO ADDRESSING section highlights a particularly valuable use of this mode.

Note that conditional re-initialization of address registers is disabled when using the conditional AIR execute mode. The default (ELSE clause) is performed unconditionally whether or not the instruction is from the instruction port or the AIR.

(CR_{10} LO). Conditional AIR execution is disabled. Conditional re-initialization is fully operational, contingent upon the re-initialization mask (CR_{3-0}).

Table III summarizes the different ways the CMP status affects operation of the AG as a function of the conditional AIR execute mode control bit, CR_{10}, and the re-initialization mask, CR_{3-0}.

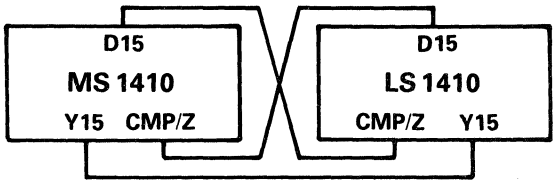


Figure 3. Two-Chip/Double Precision Handshaking

CMP STATUS	CR_{10} LO		CR_{10} HI
	CR_j LO	CR_j HI	
LO	No Effect	No Effect	No Effect
HI	CMP/Z goes HI	CMP/Z goes HI; $R_n \leftarrow I_j$	CMP/Z goes HI; Next instr. executed from AIR

Table III. Effect of Compare (CMP) Status for Looping Instructions; Note: j=3–0, the Re-Initialization Mask.

INSTRUCTION SET DESCRIPTION

The ADSP-1410's instruction set is partitioned into six groups, which are discussed below. First, however, issues spanning several instruction groups are discussed.

Most of the instruction groups contain instructions using one of the chip's six offset (B) registers. Without exception, these instructions have just two bits available for selecting the B register. Consequently, offset registers are partitioned into two banks. The upper/lower bank selection is maintained in the control register (CR_8) and is set or cleared by dedicated instructions. Whenever the "fourth" B register of either bank is specified (B_3 or B_7), the ALU's offset source becomes external data (see Table IV).

CR_8 & TWO-BIT OFFSET (B) REGISTER FIELD	OFFSET SOURCE
0 00	B0
0 01	B1
0 10	B2
x 11	Data Port*
1 00	B4
1 01	B5
1 10	B6
x 11	Data Port*

Table IV. Offset Value Structure

*Explicit use of DSEL is unnecessary when using B_3 or B_7 offsets; the offset data is sourced from the data bus by default.

In several instruction groups (see mnemonics and opcodes for details), address (R) registers are used. In all cases, asserting the DSEL pin allows external data to be substituted for an R value as both output and update data.

Two instruction groups (looping and logical/shift) both supply and update the address. Normally, addresses are supplied prior to updating (pre-update). In post-update mode however, the addresses are output after the update operation is performed. CR_9 controls this mode of operation.

For all instructions accessing an offset register, the MS bit of the three-bit offset register address (B, of Bbb) is fetched from the control register and is programmed by the SELB instruction. This is also the case for the YADD and YSUB instructions (group 1) as pertains the MS bit of the four-bit address register address (R, of Rrrr), programmed by the SELR instruction. In both cases, it is incumbent upon the programmer to ensure the appropriate register bank is selected.

The Y port is only driven on output instructions (mnemonic form Yxxx, see MNEMONICS AND OPCODES). Otherwise, the Y port defaults to a high-impedance state.

Instruction Group 1: Looping

Instructions in the looping group supply the contents of a selected address (R) register to the address (Y) port and then overwrite the R location with an updated value.

All instructions in this group generate an internal CMP status indicating whether the supplied address has moved to or beyond the boundary specified by the compare register. This status may be monitored externally via the CMP/Z pin. Internal to the chip, the CMP status can i) be ignored, ii) be used to control re-initialization of the R register value with a selected I register value (e.g., to restart an addressing loop), or iii) control execution of an instruction located in the AIR on the next cycle. Individual control register bits determine which option is enforced (see Control Register).

YINC Output & Increment/Init.

Pre-Update Mode: $Y \leftarrow R_n$;
IF ($R_n \geq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n + 1$.
Post-Update Mode: $Y \leftarrow R_n + 1$;
IF ($Y \geq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n + 1$.

Output an address (R) register on the address (Y) port and compare it to one of the compare (C) registers. If the address is less than C_j, the R location is simply updated with an incremented value. However, if $R_n \geq C_j$, CMP status goes HI and the R register is re-initialized with the I_j value, provided the initialization mask (CR_{3-0}) is enabled for I_j. Note that other modes of operation allow CMP status to be ignored (e.g., the instruction executed is simply "$Y \leftarrow R_n$; $R_n \leftarrow R_n + 1$") or to cause the AIR instruction to execute on the next cycle.

YDEC Output & Decrement/Init.

Pre-Update Mode: $Y \leftarrow R_n$;
IF ($R_n \leq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n - 1$.
Post-Update Mode: $Y \leftarrow R_n - 1$;
IF ($Y \leq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n - 1$.

Same as above except the R value is decremented instead of incremented; CMP is valid if the R value is less than or equal to the C value.

YADD Output & Add Offset/Init.

Pre-Update Mode: $Y \leftarrow R_n$;
IF ($R_n \geq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n + B_m$.
Post-Update Mode: $Y \leftarrow R_n + B_m$;
IF ($Y \geq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n + B_m$.

Same as YINC except the R value is summed with the contents of a selected offset (B) register.

The R register bank select bit (CR_7) is used in both the YADD and YSUB (offset) instructions.

YSUB Output & Subtract Offset/Init.

Pre-Update Mode: $Y \leftarrow R_n$;
IF ($R_n \leq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n - B_m$.
Post-Update Mode: $Y \leftarrow R_n - B_m$;
IF ($Y \leq C_j$):
THEN $R_n \leftarrow I_j$,
ELSE $R_n \leftarrow R_n - B_m$.

Same as YADD except the selected offset (B) register is subtracted from the R value.

Instruction Group 2: Register Transfers

Instructions in the register transfer group support internal register transfers, as well as transfers between internal and external registers. Internally, any I or B register may be written directly to any R register. Also, any R register may simultaneously be output and written directly to a B or C register. For an R-to-R transfer, the source R register can first be written to a B register, followed by a write of the B register to an R register on the next cycle.

Internal registers are read or written externally via the bi-directional data port. There are explicit instructions to read any of these registers; however, only the I registers have an explicit Write instruction. The R, B, and C registers may be written with external data by executing a transfer instruction (YRTR, YRTB, and YRTC) and asserting the DSEL pin, substituting the external data for the designated R value.

YRTR Output & Transfer Addr. Reg. to Self

$Y \leftarrow R_n$

Outputs selected address (R) register over the address (Y) port. When DSEL is asserted, data port values are output and, in the same cycle, written into the selected R register.

YRTB Output & Transfer Addr. Reg. to Base Reg.

$Y \leftarrow R_n$; $B_m \leftarrow R_n$

Outputs selected R register over the Y port and copies it into a selected B register. When DSEL is asserted, data port values are output and, in the same cycle, written into the selected B register.

YRTC Output & Transfer Addr. Reg. to Comp. Reg.

$Y \leftarrow R_n$; $C_j \leftarrow R_n$

Same as above, except that values are written to a C register.

DTI Transfer Data Bus to Init. Reg.
$I_j \leftarrow D$

Loads selected I register from data (D) port.

ITR Transfer Init. Reg. to Addr. Reg.
$R_n \leftarrow I_j$

Selected R register is loaded from an I register, allowing a microprogram to restart a loop at any time.

BTR Transfer Base Reg. to Addr. Reg.
$R_n \leftarrow B_m$

Loads an R register from a B register. Once in the R register, the B value may be modified and then returned to the B file (using a YRTB instruction). Recall, use of B_3 or B_7 will access the data port as the offset source, allowing R registers to be initialized directly from the data port.

RTD Transfer Addr. Reg. to Data Bus
$D \leftarrow R_n$

Supplies selected R register to data (D) port.

CTD Transfer Comp. Reg. to Data Bus
$D \leftarrow C_j$

Supplies selected C register to data (D) port.

BTD Transfer Base Reg. to Data Bus
$D \leftarrow B_m$

Supplies selected B register to data (D) port.

ITD Transfer Init. Reg. to Data Bus
$D \leftarrow I_j$

Supplies selected I register to data (D) port.

Instruction Group 3: Logical & Shift
Instructions in the logical/shift group supply a value from a selected address (R) register to the address (Y) port and then unconditionally overwrite the selected R location with a modified version of the output. Modify operations include logical (AND, OR, and XOR) and shift (one-bit left/right) operations. All instructions in this group affect the ZERO flag, which goes HI if the result of the modification is zero. The ZERO flag status is available externally over the CMP/Z pin.

YOR Output & Logical OR to Addr. Reg.
$Y \leftarrow R_n; R_n \leftarrow (R_n \text{ OR } B_m)$

Selected R register is supplied to the address (Y) port; the specified R location is then overwritten with the logical OR of the B register and original R value.

YAND Output & Logical AND to Addr. Reg.
$Y \leftarrow R_n; R_n \leftarrow (R_n \text{ AND } B_m)$

Same as above, except that a logical AND is performed.

YXOR Output & Logical XOR to Addr. Reg.
$Y \leftarrow R_n; R_n \leftarrow (R_n \text{ XOR } B_m)$

Same as above, except that a logical XOR is performed.

YASR Output & Arithmetic Right Shift to Addr. Reg.
$Y \leftarrow R_n; R_n \leftarrow \text{ASR}(R_n)$

Selected R register is supplied to the address (Y) port; the specified R location is then overwritten with the original R value arithmetically shifted right (ASR) by one bit (the MSB is repeated).

YLSL Output & Logical Left Shift to Addr. Reg.
$Y \leftarrow R_n; R_n \leftarrow \text{LSL}(R_n)$

Selected R register is supplied to the address (Y) port; the specified R location is then overwritten with the original R value logically shifted left (LSL) by one bit (the LSB is zero-filled).

Instruction Group 4: Control Register
Instructions in the control register group reset, read, and write the entire control register or individual control register bits (see Control Register).

Note the use of "x" and "pp" to denote values supplied within the opcode field (see MNEMONICS AND OPCODES). A positive logic convention is used throughout.

RST Reset Control Reg.
$CR \leftarrow 0$

Clears the entire control register (CR_{10-0}). The RST instruction has dedicated decoding logic so that it takes precedence even over the second instruction of a conditional AIR sequence.

DTCR Transfer Data Bus to Control Reg.
$CR \leftarrow D$

Writes the entire control register (CR_{10-0}) from the data port, D_{10-0}.

CRTD Transfer Control Reg. to Data Bus
$D \leftarrow CR$

Outputs the entire control register (CR_{10-0}) over the data port, D_{10-0}.

SETI Set/Clear Conditional Init. on CMP Flag
$CR_{jj} \leftarrow x$

Enables conditional re-initialization of an R location, subject to CMP status (see Control Register). This instruction loads the x value into the control register bit specified by jj. Conditional re-initialization of address registers by the C_{jj}/I_{jj} pair is inhibited if the corresponding CR_{jj} is cleared.

SETP Set Chip precision
$CR_{5-4} \leftarrow pp$

Loads a 2-bit code (pp) into control register bits 5 and 4, specifying the addressing mode of the device:
00 = single-precision mode;
01 = double-precision mode, LS chip (10 if DSEL);
10 = double-precision mode, MS chip;
11 = double-precision mode, single-chip.

If the instruction "SETP, 01" is supplied and the MS chip's DSEL pin is asserted, the CR_{5-4} bits are reversed, i.e., the MS chip is loaded with "10", not "01" (see Precision Modes). This is useful if the MS and LS chips share a common instruction bus.

SETY Set Y Port to Transparent/Latched Mode
$CR_6 \leftarrow x$

Uses the LS instruction bit to set the address (Y) port to the transparent (HI) or latched (LO) mode. This status is maintained in control register bit 6.

SELR Select Upper/Lower Addr. Reg. Bank
$CR_7 \leftarrow x$

The LS bit of this instruction provides the missing Address (R) register select bit required by the YADD and YSUB instructions. This selection is maintained in control register bit 7.

SELB Select Upper/Lower Base Reg. Bank
$CR_8 \leftarrow x$

The LS bit of this instruction provides the missing B register select bit required by all instructions utilizing offset (B) registers. This selection is maintained in control register bit 8.

SETU Set Update Mode (Post/Pre)
$CR_9 \leftarrow x$

Setting this bit causes the chip to output address values after updating them (post-update mode). The LS bit of this instruction determines the value of control register bit 9.

SETA Set/Clear Conditional AIR Execute Mode
$CR_{10} \leftarrow x$

Setting this bit causes Looping instructions—conditional on CMP status being HI—to execute the following instruction from the AIR on the next cycle. In this mode, conditional re-initialization of R by I on CMP is inhibited. The LS bit of this instruction determines the value of control register bit 10.

Instruction Group 5: AIR Control
Instructions in the AIR group write and read the Alternate Instruction Register (AIR). The AIR may be written or read over the data bus in one cycle or written via the instruction port in two cycles (see Table I). The instruction contained in the AIR is executed whenever the AIR Enable pin is asserted or on the next cycle in the conditional AIR execute mode.

WRA Write AIR with Data Bus
AIR $\leftarrow$ D

Write the AIR from the data (D) bus (D_{9-0}).

RDA Read AIR at Data Bus
D $\leftarrow$ AIR

Read the AIR over the data (D) bus (D_{9-0}).

LDA Load AIR from Instruction Port on Next Cycle (Requires DSEL HI)
AIR $\leftarrow$ Instruction Port

This instruction is the first of a two-cycle sequence that loads the AIR via the instruction port. On the cycle following the execution of LDA, the instruction at the instruction port is loaded into the AIR (and not executed). DSEL must be asserted with the LDA instruction (meeting the same setup and hold time requirements); otherwise, the AIR is not loaded. In systems with multiple ADSP-1410s sharing microcode instructions, this feature allows you to select particular devices for AIR loading.

Instruction Group 6: Miscellaneous

YDTY Pass Data Bus to Y Port
Y $\leftarrow$ D

Data (D) port values are supplied directly to the address (Y) port. Note that internal address (R) registers are not affected by this instruction.

YREV Output Addr. Reg. in Bit-Reversed Format
$Y \leftarrow YREV(R_n)$; $R_n \leftarrow R_n + B_m$

The selected address (R) register is bit reversed at the output port. The original (unreversed) R value is added to the selected offset (B) register, and written back into the specified R location. Condition testing is not performed. Bit reversing affects only output data, not register contents.

NOP No Operation

Prevents any changes to the internal conditions of the AG. All I/O pins go to the three-state disable mode.

ADDRESS GENERATOR APPLICATIONS

The ADSP-1410 has a wide range of uses in high-speed digital signal processing and general purpose computer applications. In particular, this AG can be used in implementing the following:

Circular Data Buffers
- FIR filter tapped delay lines
- Correlator delay lines
- Image processing delay lines
- Recirculated data I/O for transient data capture or stimulus source

Memory Management
- Fast Fourier Transform data and twiddle factors
- Matrix computations

Table Look-Ups
Masking and table address mapping with AND/OR and bit reverse capabilities.

Variable-Width Bit Reversing
The internal bit-reversing multiplexer of the AG accommodates only full, 16-bit addresses (64K FFTs). For smaller FFTs, (utilizing a right-justified subset of the 16-bit address field), a zero-overhead software approach may be employed. The details of this approach may be found in the application note: "Variable-Width Bit Reversing with the ADSP-1410 Address Generator." Essentially, the technique is this: an R register is intialized with the bit-reversed value of the 16-bit starting address (a "pre-reversed" version of the first data point location) and a B register with the value $K \cdot 2^{16-N}$, where K is the step size between samples and N is the order of the FFT. Now, repeated execution of the YREV instruction will output the appropriate bit-reversed addresses; updating the R register each time.

Multi-Tasking Operations
Context switching allowed by large number of on-chip registers or by instructions allowing all registers to be saved and restored.

16-Bit ALU/Accumulator
By substituting external data for a B register and operating in post-update mode, ALU operations can be performed at high speed. ALU sources are the external data and any one of sixteen internal R registers. Results are stored on-chip in these R registers. Two chips may be cascaded for double-precision operations.

Unclocked (Flow-Through) Applications
When operating in transparent and post-update modes with the DSEL line asserted, the device serves as an unclocked ALU.

Digital Differential Analyzer
- Sine and cosine generation
- Graphics/Line drawing
- Control and guidance

Modulo Addressing
Hardware on the ADSP-1410 allows the addressing of circular buffers to be implemented without overhead. The Conditional re-initialization structure handles the simple case of returning to the top of a loop.

Some applications require robust modulo addressing of a circular buffer with an arbitrary starting point, ending point, and increment between addresses. To implement true modulo addressing with the ADSP-1410, consider a buffer of length L. First, R_n is initialized with the start address n, B_m is initialized with m, a constant increment or step between addresses, and B_o is initialized with (L − m), to implement a modulo jump to the beginning of the buffer. Then a compare register C_j is loaded with the value (n + L − 2m) for pre-update mode, or the value (n + L − m) for post-update mode. Bit CR_{10} of the Control Register is set to enable conditional AIR execution. The instruction "YADD R_n C_j B_m" is then executed repeatedly, from the instruction port. This outputs R_n and updates it (for pre-update mode – or updates R_n and then outputs it for post-update mode) by summing it with the offset B_m. The comparator monitors whether $R_n \geq (n + L - 2m)$, for pre-update mode, or $R_n \geq (n + L - m)$, for post-update mode. When such an event occurs, the instruction in the AIR is executed in the next cycle. This should be the negative offset instruction "YSUB R_n C_j B_o" which updates R_n with a negative offset of (L − m), causing a modulo L jump back to the beginning of the buffer. In this fashion, true modulo addressing can be implemented for arbitrary buffer boundaries and offsets.

SPECIFICATIONS

The specification tables contain the electrical and switching characteristics of the ADSP-1410. Figure 7 is the accompanying timing diagram for the device.

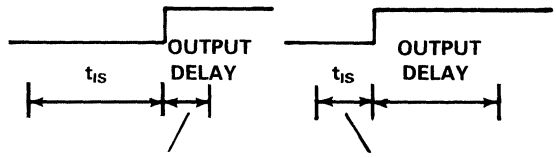
The clock input to the ADSP-1410 is a single, two-phase clock with cycle time: t_{CY}.

The setup and hold times for the instruction inputs are t_{IS} and t_{IH}, respectively. Input instructions consist of the 10-bit microcode instruction, the DSEL control, and the AIRE control: all of which are latched during phase one (clock HI).

The timing of internal register reads from the data port is specified by t_{ODD} and t_{DDIS}. Assuming a data output instruction is executing, the data drivers are activated *only* during phase one (clock HI). Therefore, output data becomes valid t_{ODD} into phase one (clock HI) and remains valid for a portion of phase two (clock LO). t_{DDIS} specifies how long into phase two the data drivers take to disable. If data outputs are followed by data inputs, t_{DDIS} establishes the timing required to avoid bus contention.

If the device is in the transparent mode, the DSEL pin may be asserted to open the path between the data port and the address port. Assuming data is properly setup on the D port, t_{TAn} or t_{TAp} (for pre- or post-update modes, respectively) specifies the interval from DSEL assertion to a valid address appearing at the Y port. Note that changes on the DSEL pin (or *any* instruction pin) are not recognized during phase one (clock HI).

Latched Mode Parameters have a Sliding Window
Output delays for addresses and the CMP/Z flag depend upon whether the device is in the pre-update (normal) mode or post-update mode and upon the use of a latches vs. transparent mode of operation. In the latched mode, a "sliding window" effect is apparent, resulting from the internal Look-Ahead pipeline (see Figure 3). The sliding window effect is described to facilitate



a. Minimum Output Delay *b. Minimum Setup Time*

Figure 4. Boundary Cases of "Sliding Window" Effect: Minimum Output vs. Minimum Setup

exact calculation of guaranteed Clock-to-Output delays as a function of faster or slower instruction setup times. Latched mode guaranteed Clock-to-Output delays are given as a min/max pair. The user may vary the output delays within these limits by adjusting the instruction setup time.

As the instruction setup time is increased beyond the minimum ($t_{IS} \geq \min[t_{IS}]$), the corresponding guaranteed Clock-to-Output delay will be reduced (see Figure 3a) toward its minimum value. Conversely, as the instruction setup time is reduced toward its minimum ($t_{IS} \rightarrow \min[t_{IS}]$), the corresponding Clock-to-Output delay will increase (see Figure 3b) toward its maximum value.

The required instruction setup time for the fastest latched output delay is simply the difference between the minimum and maximum guaranteed Clock-to-Output specifications plus the minimum instruction setup time, e.g., an instruction setup time of $|\max[t_{LAn}] - \min[t_{LAn}] + \min[t_{IS}]|$ is required to realize $\min[t_{LAn}]$.

For intermediate cases (in which neither min/max limits apply), output delays may be calculated by subtracting the *actual* instruction setup time from the *sum* of the minimum instruction setup time and the maximum guaranteed Clock-to-Output specifications, as the following example shows (in which $t_{ISmin} = 15ns$ and $30ns \leq t_{LAn} \leq 35ns$):

Actual t_{IS}	Guaranteed Clock-to-Output Delay ($\max[t_{LAn}] + \min[t_{IS}] - t_{IS}$)	
5	n/a	Invalid (minimum t_{IS} violated)
10	n/a	Invalid (minimum t_{IS} violated)
15	35	Minimum Setup, Maximum Delay
16	34	Sliding Window Dominant
17	33	Sliding Window Dominant
18	32	Sliding Window Dominant
19	31	Sliding Window Dominant
20	30	Maximum Usable Setup, Minimum Delay
25	30	Minimum t_{LAn} Dominant
30	30	Minimum t_{LAn} Dominant
etc.	etc.	etc.

Transparent Mode Parameters
The transparent mode of operation is entirely dissociated from clock edges. Hence, the relevant parameters are referenced to the instruction becoming valid rather than the clock edge; only *maximum* Valid-instruction-to-Output Delay specifications pertain.

(continued on page 3–42)

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		J & K Grades Min	J & K Grades Max	S & T Grades[2] Min	S & T Grades[2] Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_{AMB}	Ambient Operating Temp.	0	70	−55	125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	J & K Grades Min	J & K Grades Max	S & T Grades[2] Min	S & T Grades[2] Max	Unit
V_{IH}	Hi-Level Input Voltage	V_{DD} = max	2.0		2.0		V
V_{IHc}	Clock Input Hi-Level Input Voltage	V_{DD} = max	3.0		3.5		V
V_{IL}	Lo-Level Input Voltage	V_{DD} = min		0.8		0.8	V
V_{OH}	Hi-Level Output Voltage	V_{DD} = min, I_{OH} = −1mA	2.4		2.4		V
V_{OL}	Lo-Level Output Voltage	V_{DD} = min, I_{OL} = 3mA		0.6		0.6	V
I_{IH}	Hi-Level Input Current	V_{DD} = max, V_{IN} = 5V		10	10		μA
I_{IL}	Lo-Level Input Current	V_{DD} = max, V_{IN} = 0V		10		10	μA
I_{IH}	Clocks & Control Inputs Hi-Level Input Current	V_{DD} = max, V_{IN} = 5V		10		10	μA
I_{IL}	Clocks & Control Inputs Lo-Level Input Current	V_{DD} = max, V_{IN} = 0V		10		10	μA
I_{OZH}	Three-State Leakage Current	V_{DD} = max, V_{IN} = max		50		50	μA
I_{OZL}	Three-State Leakage Current	V_{DD} = max, V_{IN} = 0		50		50	μA
I_{DD}	Supply Current	max clock rate, TTL inputs		75		100	mA
I_{DD}	Quiescent Supply Current	V_{IN} = 2.4V		35		50	mA

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	−0.3V to 7V
Input Voltage	−0.3V to V_{DD}
Output Voltage Swing	−0.3V to V_{DD}
Load Capacitance	200pF
Operating Temperature Range (Ambient)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10 Seconds)	300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1410JN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1410KN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1410JP	0 to +70°C	52-Lead PLCC	P-52
ADSP-1410KP	0 to +70°C	52-Lead PLCC	P-52
ADSP-1410JD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1410KD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1410SD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1410TD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1410SD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1410TD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A

ESD SENSITIVITY

The ADSP-1410 features proprietary input protection circuitry. Per Method 3015 of MIL-STD-883, the ADSP-1410 has been classified as a Class 1 device.

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*.

SWITCHING CHARACTERISTICS[3]

Parameter		J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade[2] Min	S Grade[2] Max	T Grade[2] Min	T Grade[2] Max	Unit
t_{IS}	Instruction Setup Time[4]	20		15		30		20		ns
t_{IH}	Instruction Hold Time	5		4		5		4		ns
t_{CY}	Instruction Cycle Time	100		90		125		100		ns
t_{IDS}	Input Data Setup Time	10		10		10		10		ns
t_{IDH}	Input Data Hold Time	5		5		6		6		ns
t_{ODD}	Guaranteed Clock-to-Data Delay[5]	35	55	30	50	45	70	40	60	ns
t_{DENA}	Output Data Enable Time[5]	30	50	25	45	40	65	35	55	ns
t_{DDIS}	Output Data Disable Time		20		20		25		20	ns
t_{ADIS}	Output Address Disable Time		30		25		45		40	ns
	Latched Mode, Guaranteed Clock-to-Output Delays:									
t_{LAn}	Pre-Update Address Delay[5]	35	45	30	35	40	55	35	45	ns
t_{LFn}	Pre-Update CMP/Z Flag Delay[5] (C = 25pF)	45	55	35	45	60	75	45	60	ns
t_{LAp}	Post-Update Address Delay[5]	35	60	30	50	40	75	35	55	ns
t_{LFp}	Post-Update CMP/Z Flag Delay[5] (C = 25pF)	45	70	35	55	60	95	45	75	ns
	Transparent Mode, Valid-Instruction-to-Output Delays:									
t_{TAn}	Pre-Update Address Delay		50		45		65		55	ns
t_{TFn}	Pre-Update CMP/Z Flag Delay (C = 25pF)		65		55		90		70	ns
t_{TAp}	Post-Update Address Delay		75		65		95		80	ns
t_{TFp}	Post-Update CMP/Z Flag Delay (C = 25pF)		90		75		115		95	ns
	Supplemental Parameters for Double-Chip/Double-Precision Operation[6]:									
t_{CSD}	Valid Instruction-to-Carry/Shift Output Delay		65		57		80		72	ns
t_{CSS}	Carry/Shift Input Setup Time	35		28		40		35		ns
t_{MSD}	Carry/Shift Input to Valid MS Address (Post-Update Only)		45		40		55		48	ns
t_{CZDc}	Valid Instruction to MS CMP/Z (Compare) Flag Delay		115		105		120		115	ns
t_{CZI}	Clock High to CMP/Z (Compare) Invalid Delay	4		4		4		4		ns
t_{CZDz}	Valid Instruction to CMP/Z (Zero) Flag Delay		80		70		85		80	ns
t_{IID}	Instruction Invalid to CMP/Z (Zero) Invalid Delay	10		10		10		10		ns

NOTES
[1]All specifications are over the recommended operating conditions.
[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-1410 can be found in Analog Devices' *Military Products Databook*.
[3]Input levels are GND and 3V. Rise times are 5ns. Input timing reference levels and output timing reference levels are 1.5V. For capacitive loads greater than 100pF, we recommend the use of external buffers.
[4]Instruction setups beyond the clock LO period (into the previous cycle) will not be recognized, regardless of latched *or* transparent mode, as the instruction latch is *always* frozen during clock HI. Also, the clock HI period must always exceed the guaranteed Clock-to-Output/Data delay.
[5]Minimum specifications pertain to maximum usable instruction setups, while maximum specifications pertain to absolute minimum instruction setups. See discussion of "sliding window" under Specifications.
[6]The Instruction Cycle Time, t_{CY}, does not apply to DCDP operation. Clock HI and LO relationships for DCDP operation are described in the Specifications .text under DCDP Parameters: t_{HI} is derived from the t_{CSD}, t_{CSS}, t_{MSD}, and t_{IS} parameters, and the inequality, $t_{LO} \geq t_{IS}$, must also hold.
Specifications subject to change without notice.

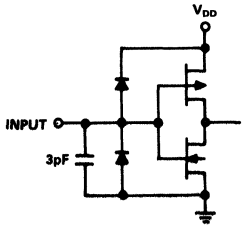


Figure 5. Equivalent Input Circuits

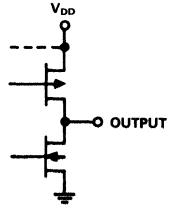


Figure 6. Equivalent Output Circuits

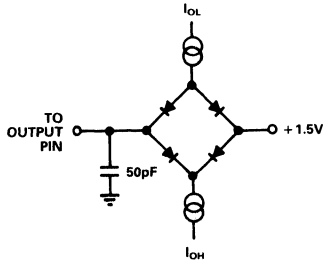


Figure 7. Normal Load for ac Measurement

(continued from page 3–39)

Data Output Parameters

Data output parameters are independent of operating modes. Data drivers are asserted *only* during phase one (clock HI). The guaranteed Clock-to-Data Delay is, again, subject to the sliding window phenomenon; the min/max specifications pertain to maximum usable and absolute minimum instruction setup times, respectively.

Double-Chip/Double-Precision Parameters

The double-chip/double-precision (DCDP) mode of operation utilizes the Y_{15} pin to commute the interchip carry/borrow/shift information, and D_{15}, the CMP/Z status (see Figure 2).

Pre-Update DCDP

Normally, (as is the case with *any* pre-update operation) pre-update DCDP operations have only to output the previously calculated result. However, because the carry/shift output delay is asynchronous, the clock cycle time becomes a function of how soon the instruction is valid; increasing DCDP instruction setups decreases the required clock cycle time.

The carry/shift output delay, t_{CSD}, is referenced to the valid instruction, while the carry/shift setup time, t_{CSS}, is referenced to the clock falling edge. Together, they comprise the minimum time required from the valid instruction to the falling edge of the clock. Therefore, the sum of the carry/shift I/O operations (t_{CSD} + t_{CSS}) less the instruction setup time, t_{IS}, defines the minimum clock HI period; $t_{HI} \geq (t_{CSD} + t_{CSS}) - t_{IS}$, as referenced in footnote 6 of the switching characteristics table. The clock LO duration must accommodate the instruction setup time; $t_{LO} \geq t_{IS}$.

Post-Update DCDP

Because post-update DCDP operation of the ADSP-1410 requires calculation of the address prior to its output, the additional parameter for the MS word output delay, t_{MSD}, is necessary in specifying this mode. In post-update DCDP mode, t_{MSD} supplants t_{CSS} for Clock HI determination; $t_{HI} \geq (t_{CSD} + t_{MSD}) - t_{IS}$.

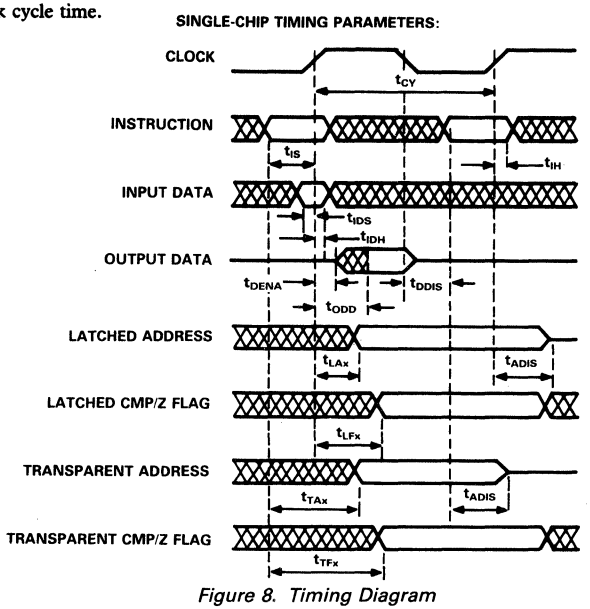


Figure 8. Timing Diagram

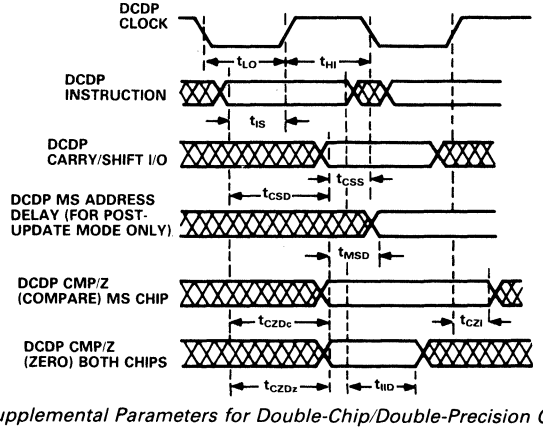


Figure 9. Supplemental Parameters for Double-Chip/Double-Precision Operation

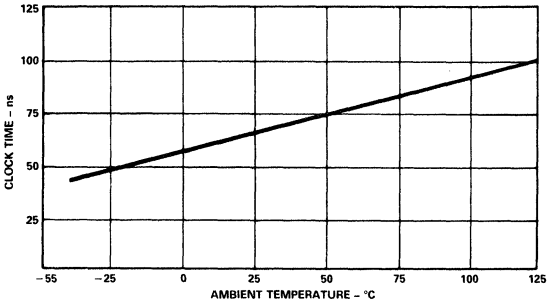


Figure 10. Clock Cycle Time vs. Temperature

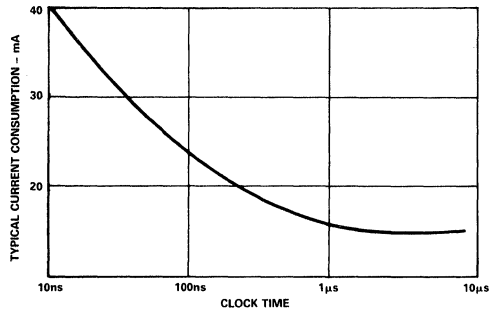


Figure 11. Typical I_{DD} vs. Frequency of Operation

3

MNEMONICS AND OPCODES

The following list gives the instruction mnemonics and opcodes. Various parameters are substituted by the user, defining register numbers or control bits. The notation convention is this:

R	=	Address register
B	=	Base (offset) register
C	=	Compare register
I	=	Initialization register
D	=	Data bus
CR	=	Control register
rrrr	=	Four-bit address register number
rrr	=	Three-bit address register number
bb	=	Two-bit base (offset) register number
cc	=	Two-bit comparison register number
ii	=	Two-bit initialization register number
pp	=	Two-bit precision code
x	=	One-bit control bit

*External data may substitute for R using DSEL.
†Operable in either pre- or post-update mode.

Instr.	Opcode (I_{9-0})	Description
Looping Instructions		
YINC*†:	1011ccrrrr	output & increment/init
YDEC*†:	1010ccrrrr	output & decrement/init
YADD*†:	11ccbb1rrr	output & add offset/init
YSUB*†:	11ccbb0rrr	output & subtract offset/init
Register Transfer Instructions		
YRTR*:	000101rrrr	output & xfr R to R
YRTB*:	0011bbrrrr	output & xfr R to B
YRTC*:	0010ccrrrr	output & xfr R to C
DTI:	00001111ii	xfr D to I
ITR:	1000iirrrr	xfr I to R
BTR:	0100bbrrrr	xfr B to R
RTD:	000100rrrr	xfr R to D
CTD:	00001100cc	xfr C to D
BTD:	00001101bb	xfr B to D
ITD:	00001110ii	xfr I to D
Logical and Shift Instructions		
YOR*†:	0111bbrrrr	output & OR B with/to R
YAND*†:	0110bbrrrr	output & AND B with/to R
YXOR*†:	0101bbrrrr	output & XOR B with/to R
YASR*†:	000111rrrr	output & arith SR R to R
YLSL*†:	000110rrrr	output & logical SL R to R
Control Register Instructions		
RST:	0000000001	reset CR
DTCR:	0000101110	xfr D to CR
CRTD:	0000101111	xfr CR to D
SETI:	0000100iix	set cond re-init on CMP mode
SETP:	00001010pp	set chip precision
SETY:	000001001x	set Y port to trans/latched mode
SELR:	000001101x	select upper/lower R bank
SELB:	000001100x	select upper/lower B bank
SETU:	000001011x	set post/pre update mode
SETA:	000001010x	set cond AIR mode
AIR Instructions		
WRA:	0000101100	write AIR with D
RDA:	0000101101	read AIR at D
LDA:	0000011110	load AIR on next cycle
Misc. Instructions		
YDTY:	0000011111	pass D to Y port
YREV*†:	1001bbrrrr	output R in bit-reverse format
NOP:	0000000000	no operation

ADSP-1410 PIN CONFIGURATIONS

DIP
D-48A
N-48A

PIN	FUNCTION	PIN	FUNCTION
1	I4	48	I5
2	I3	47	I6
3	I2	46	I7
4	I1	45	I8
5	I0	44	I9
6	CLK	43	DSEL
7	CMP/Z	42	AIRE
8	Y15	41	D15
9	Y14	40	D14
10	Y13	39	D13
11	Y12	38	D12
12	GND	37	V_{DD}
13	Y11	36	D11
14	Y10	35	D10
15	Y9	34	D9
16	Y8	33	D8
17	Y7	32	D7
18	Y6	31	D6
19	Y5	30	D5
20	Y4	29	D4
21	Y3	28	D3
22	Y2	27	D2
23	Y1	26	D1
24	Y0	25	D0

PLCC
P-52

PIN	FUNCTION	PIN	FUNCTION
1	GND	52	I5
2	I4	51	I6
3	I3	50	I7
4	I2	49	I8
5	I1	48	I9
6	I0	47	DSEL
7	CLK	46	AIRE
8	CMP/Z	45	D15
9	Y15	44	D14
10	Y14	43	D13
11	Y13	42	D12
12	Y12	41	V_{DD}
13	GND	40	V_{DD}
14	GND	39	D11
15	Y11	38	D10
16	Y10	37	D9
17	Y9	36	D8
18	Y8	35	D7
19	Y7	34	D6
20	Y6	33	D5
21	Y5	32	D4
22	Y4	31	D3
23	Y3	30	D2
24	Y2	29	D1
25	Y1	28	D0
26	Y0	27	GND

ANALOG DEVICES

Multiport Register File

ADSP-3128A

FEATURES

128×16 or 64×32 Register File Organization
Flexible "Crossbar" Data Routing via Five Ports
- **Two Input**
- **Two Output**
- **One Bidirectional**

Cascadable Horizontally and Vertically
Supports 20MHz Operation from Single 1×Clock
- **18ns Clock-to-Valid Output (Registered)**
- **35ns Address-to-Valid Output (Transparent)**

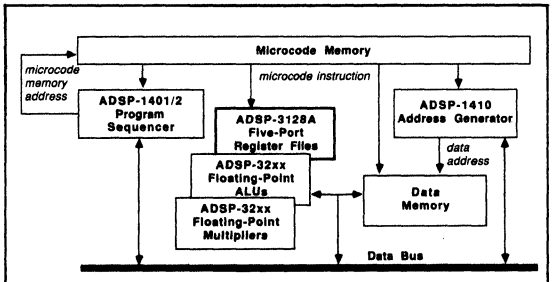
Flexible Latching Modes at Address and Data Ports:
- **Transparent, Latched, Registered**

Prioritized Write Ports
Write Inhibit Control on Each Write Port
Automatically Pipelined Bank Select and Port Select
Register-to-Register Transfers
Three-State Outputs
Fully Static Operation
145-Pin Grid Array

APPLICATIONS

High Speed Temporary Data Storage in
- **Digital Signal Processing**
- **Numeric ProcessingGraphics**

Floating-Point and Fixed-Point



Word-Slice® Floating-Point Microcoded System with ADSP-3128A Multiport Register Files

GENERAL DESCRIPTION

The ADSP-3128A Multiport Register File is a versatile data storage component that can greatly expand the computational bandwidth of a fast-arithmetic processor. (See Figure 1 for the ADSP-3128A's Functional Block Diagram.) The ADSP-3128A also simplifies processor design by permitting flexible data routing through its five 16-bit data ports: two input ports, two output ports and a bidirectional port. This register file complements the floating-point and fixed-point multipliers and ALUs available from Analog Devices. Because of its flexibility, however, it has application in a broad range of processor designs. The ADSP-3128A is a higher speed, pin-compatible upgrade from the ADSP-3128.

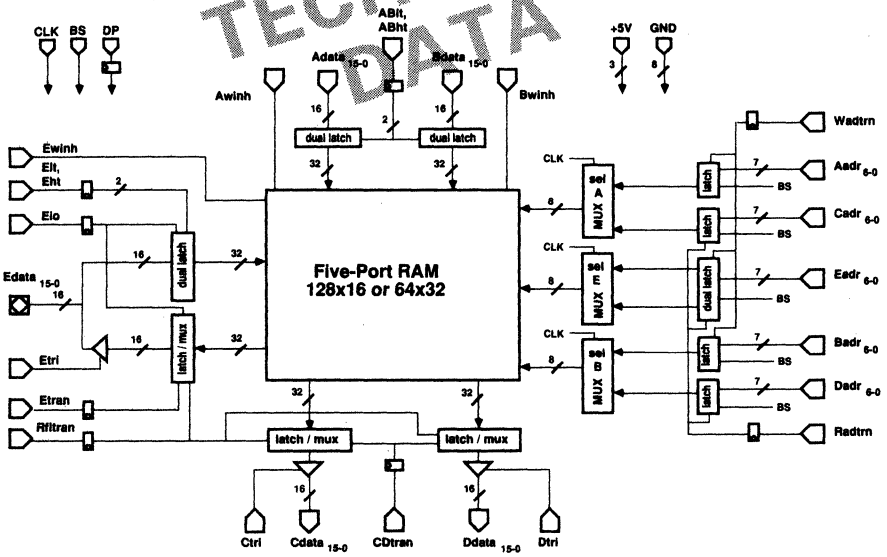


Figure 1. ADSP-3128A Multiport Register File Functional Block Diagram

The ADSP-3128A is configurable via a control pin as either a 128×16 register file or a 64×32 register file. In the Single-Precision 128×16 configuration, the ADSP-3128A is best suited for fixed-point and single-precision (32-bit) floating-point data storage. For single-precision floating-point, two register files should be used "horizontally" yielding 128 words of 32-bit storage. The 64×32 Double-Precision configuration is intended for double-precision (64-bit) floating-point, again with two register files in a horizontal architecture. In this Double-Precision mode, the register files will each transfer 16-bits in each phase of the clock, 32-bits of data per port in a one-cycle write or read operation. Microcode need only be applied to the register file at the system's 1×clock rate.

To accommodate critical system timing requirements, the ADSP-3128A offers a variety of latching modes on both data and address ports. The prioritized write data ports have control lines that define the input data latching mode for Single-Precision as (a) latched on clock HI, (b) transparent or (c) registered on the clock's falling edge. However loaded, data can also be held at the input latches for subsequent cycles.

In *Single-Precision mode*, the Multiport Register File's five ports allow five 16-bit data transfer operations per cycle. The input and output latches transfer data to and from the ADSP-3128A's RAM using 16-bit internal buses. The bidirectional Edata-Port can be directly controlled to either write or read. Normal operation allows up to three 16-bit writes in clock HI and three 16-bit reads in clock LO per cycle. Register-to-register transfers are made via the bidirectional Edata-Port (which can be accomplished in two sequential clock phases by following a read with a write). See the Applications Note, "Register-to-Register Transfers with the ADSP-3128A."

In *Double-Precision mode,* the Multiport Register File's five ports allow five 32-bit data transfer operations per cycle for a total bandwidth of 160 bits per cycle. The input and output latches transfer data to and from the RAM via 32-bit internal buses. The input data latching modes allow either an early input or a late input mode. With early input, the Y_Word (Y_W) is presented to the input data latches in clock HI and the X_Word (X_W) in clock LO. With late input, the Y_Word is presented to the input latches in clock LO and the X_Word in clock HI of the next cycle. For data transfers with a slower system bus, the Edata-Port allows both input and output values to be transferred more slowly than the ADSP-3128A's clock rate (Edata Slow Input and Edata Slow Read). Register-to-register transfers are made via the bidirectional Edata-Port.

Each *write data* port of the ADSP-3128A has an independent write-inhibit control that disables the write operation that normally occurs during clock HI. Write-inhibit allows cancelling a write based on an external condition.

The *read data* ports have control lines that define the output data latching mode for Single-Precision as (a) registered on the clock's rising edge or (b) transparent. In Double-Precision mode, the output data latching modes allow either an early read or a late read. With early read, the Y_Word can be output in clock LO and the X_Word in clock HI of the next cycle. With late read, the Y_Word can be output in clock HI and the X_Word in clock LO of the same cycle. Each read data port has an independent tristate control that allows putting that output port into a high impedance state.

The 7-bit *write address* latches corresponding to the write ports can be mutually defined to latch addresses in one of two ways. Either (a) write addresses are latched to the address latches on clock HI, or (b) the address latches are transparent. The 7-bit *read address* latches can be mutually defined to latch addresses in one of two different ways. Either (a) read addresses are registered to the address latches on the clock's rising edge, or (b) the address latches are transparent. In Double-Precision mode, there are half as many words that are twice as wide. For Double-Precision addressing, the (unneeded) highest order address bits function as Port Select lines. Port Select (the most significant address bit) enables or disables individual ports consistent with their pipelines.

Bank Select enables or disables an entire ADSP-3128A consistent with all read and write pipelines. Bank Select and Port Select allow the user to expand register file storage "vertically" for more than 128 single-precision or 64 double-precision data words.

The ADSP-3128A is fabricated in double-metal 1.0μm CMOS. Each chip consumes significantly less power than comparable bipolar solutions.

The ADSP-3128A is available for both commercial and extended temperature ranges. Extended temperature range parts are available processed fully to MIL-STD-883, Class B. The ADSP-3128A is packaged in a ceramic 145-lead pin grid array.

TABLE OF CONTENTS **PAGE**

ADSP-3128A MULTIPORT REGISTER FILE PIN LIST (POSITIVE TRUE LOGIC CONVENTION)

Pin Name	Description
DATA PORTS	
$Adata_{15-0}$	Write Adata-Port Input Data
$Bdata_{15-0}$	Write Bdata-Port Input Data
$Cdata_{15-0}$	Read Cdata-Port Output Data
$Ddata_{15-0}$	Read Ddata-Port Output Data
$Edata_{15-0}$	Bidirectional Edata-Port Input and Output Data
ADDRESS PORTS	
$Aadr_{6-0}$	Address Port for Adata-Port Writes
$Badr_{6-0}$	Address Port for Bdata-Port Writes
$Cadr_{6-0}$	Address Port for Cdata-Port Reads
$Dadr_{6-0}$	Address Port for Ddata-Port Reads
$Eadr_{6-0}$	Address Port for Edata-Port Writes and Reads and for Register-to-Register Transfers
GENERAL CONTROLS	
BS	Bank Select (registered or asynchronous, depending on address port Latches)
DP	Double-Precision Mode (registered)
ADDRESS LATCH CONTROLS	
Wadtrn	Write Address Latch Transparent (registered)
Radtrn	Read Address Latch Transparent (registered)
DATA INPUT AND WRITE CONTROLS	
ABlt, ABht	Input Latch Controls for Both Adata-Port and Bdata-Port (registered)
Elt, Eht	Input Latch Controls for Edata-Port (registered)
Awinh	Inhibit Write to RAM from Adata-Port Input Latches (asynchronous)
Bwinh	Inhibit Write to RAM from Bdata-Port Input Latches (asynchronous)
Ewinh	Inhibit Write to RAM from Edata-Port Input Latches (asynchronous)
DATA READ AND OUTPUT CONTROLS	
CDtran	Output Latch Controls (Make Transparent) for Both Cdata-Port and Ddata-Port (registered)
Etran	Output Latch Controls (Make Transparent) for Edata-Port (registered)
Rfltran	Clock-On-Rising/Falling Select for Slow Inputs in Double-Precision Mode (registered)
Eio	Edata-Port Slow Read Control in Double-Precision Mode (registered)
Ctri	Cdata-Port Three-State Control (asynchronous)
Dtri	Ddata-Port Three-State Control (asynchronous)
Etri	Edata-Port Three-State Control (asynchronous)
MISCELLANEOUS	
CLK	Clock
GND	Ground (Eight Lines)
V_{DD}	+5V Power Supply (Three Lines)

FUNCTIONAL DESCRIPTION

The ADSP-3128A Multiport Register File consists of a high speed static RAM (configurable as either 128×16 or 64×32) surrounded by the latches and control logic needed for simple system interfacing (see Figure 1). Six internal data paths, all 32-bits wide, connect this RAM with multiplexers (muxes) and latches. Three are read data paths; three are write data paths. Three 7-bit internal address paths connect this RAM with muxes and address latches. These three address paths are internally time-multiplexed to allow the presentation of six addresses to the RAM per cycle.

Three addresses are presented to RAM in clock HI from the Aadr, Badr and Eadr address latches. These are RAM write addresses. They are prioritized in case of conflict. Three addresses are presented to RAM in clock LO from the Cadr, Dadr and Eadr address latches. These are RAM read addresses. Three simultaneous reads, even from the same RAM location, are possible for clock LO reads. The Eadr–Port feeds both a write (clock HI) address latch and a read (clock LO) address latch, which can be independently set to latched or transparent modes.

Writes to the RAM occur in clock HI when Awinh and/or Bwinh and/or Ewinh are LO. Note that data writen in clock HI is available to be read in the same clock cycle.

The DP control determines whether the Register File is in Double-Precision mode (HI) or Single-Precision mode (LO). In Single-Precision mode, all data paths between RAM and data latches behave as if they were 16 bits. The data latches also behave like 16-bit latches. The register file is organized 128×16 in Single-Precision mode, and each location is addressed with seven bits. DP can be changed dynamically, consistent with the constraints imposed in the timing diagrams (Figures 4 through 13).

In Double-Precision mode, the Register File is organized 64×32, and each location is addressed with six bits. In Double-Precision mode, all data paths between RAM and data latches are 32 bits, as are the data latches. Writes (32-bit) to the RAM occur in clock HI and reads (32 bit) from the RAM occur in clock LO. Multiplexers between the latches and the 16-bit data ports alternately select Y_Word and X_Word. Note that when ADSP-3128A Register Files are configured in horizontal pairs for Double-Precision operation, the Y_Words from the pair will make up half the external 64-bit double-precision word and the X_Words the other half. See Figures 14 and 15.

In Single-Precision mode, the input latches can be configured to latch input data at clock HI, register input data on the falling clock edge, be made transparent, or hold the most recent data. The output latches can be configured to register data from the RAM on the rising clock edge or to be transparent clock LO and latched clock HI. The bidirectional Edata-Port can be configured to do either one read or one write each cycle. Each read port has an independent three-state enable control.

In Double-Precision mode, the input latches can be configured for an early input, a late input, a slow input on the Edata-Port (for transfers from slow devices), or a hold of the most recent data on the A&Bdata-Ports. Early and late inputs are distinguished by a one-half clock cycle difference between when the Y_Word and X_Word are written to the input latches. The output latches can be configured for an early read, a late read or a slow read on the Edata-Port (for transfers to slow devices). Early and late reads are distinguished by a one-half clock cycle difference between when the Y_Word and X_Word are read from the output latches. To accomplish late inputs and early reads, the latches are transparent for 16 bits of the data transfer, allowing either a direct write of the X_Word to RAM or a direct read of the Y_Word from RAM, respectively.

The write address latches can be made transparent or latched in clock HI. The read address latches can be made transparent or registered with the clock's rising edge. In Double-Precision mode, the unused high-order address bit is interpreted as Port Select. Port Select and Bank Select (BS) are treated as part of the address field so that their write-disable and three-state effects properly track the selected pipeline delays.

CONTROLS

The ADSP-3128A Register File has 18 control lines. Their functional descriptions are summarized in mode Tables I through III.

Most control lines are registered, as indicated in the "Pin List" and in Figure 1. All registered controls meet the timing requirements of Figure 2. The timing requirements for the three asynchronous three-state controls, Ctri, Dtri and Etri, are shown in Figure 3. The timing for the remaining asynchronous controls are illustrated in timing diagrams Figures 2 through 13.

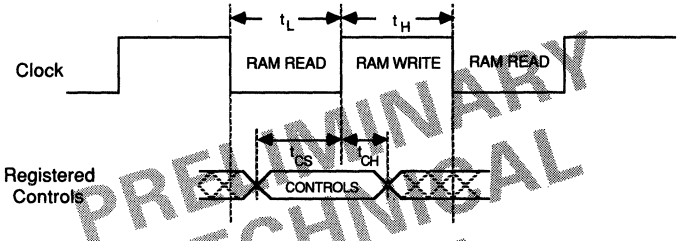


Figure 2. ADSP-3128A Registered Controls Timing

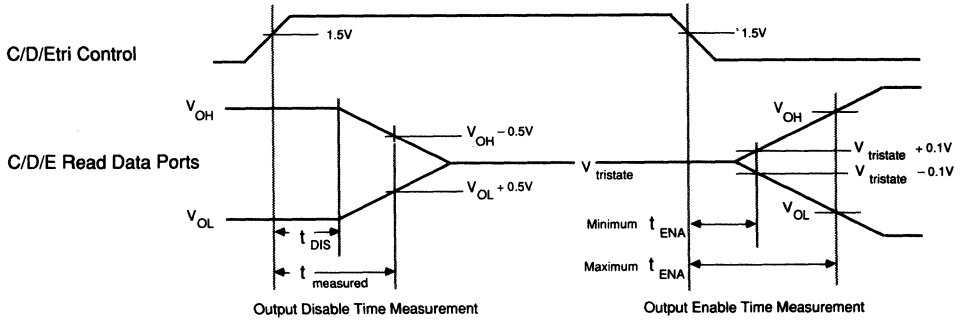


Figure 3. ADSP-3128A Three-State Disable and Enable Timing

BS	DP	AB&Elt	AB&Eht	A&B&Einh	Rfltran	Description
0	X	X	X	X	X	Disable chip (consistent with pipelines) but advance pipelines with clock cycle
1	0	**0**	**0**	X	X	Register write data at A&B or Edata input latches on falling edge
1	0	**0**	**1**	X	X	Hold most recent data at A&B or Edata input latches for the next cycle
1	0	**1**	**0**	X	X	Latch write data at A&B or Edata input latches at clock HI
1	0	**1**	**1**	X	X	Make transparent A&B or Edata input latches
1	X	X	X	**0**	X	Allow write to RAM from the A, B and Edata input latches
1	X	X	X	**1**	X	Inhibit write to RAM from the A, B and Edata input latches
1	1	**0**	**0**	X	X	Early Input to A&B or Edata input latches: register Y_W on falling edge to input latches and latch X_W to input latches in clock HI
1	1	**0**	**1**	X	X	Late Input to A&B or Edata input latches: latch Y_W to input latches in clock HI and make input latches transparent for X_W in clock HI
1	1	**1**	X	X	X	Hold most recent data at A&B input latches for the next cycle
1	1	1	**1 → 0**	X	**0**	Edata Slow Input: register Y_W to Edata input latch on next falling edge (Eht only)
1	1	1	**0 → 1**	X	**0**	Edata Slow Input: register X_W to Edata input latch on next falling edge (Eht only)
1	1	**1 → 0**	1	X	**1**	Edata Slow Input: register Y_W to Edata input latch on next rising edge (Elt only)
1	1	**0 → 1**	1	X	**1**	Edata Slow Input: register X_W to Edata input latch on next rising edge (Elt only)

Table I. ADSP-3128A Summary of Data Input and Write Control Modes

BS	DP	CD&Etran	Rfltran	C&D&Etri	Eio	Description
0	X	X	X	X	X	Disable chip (consistent with pipelines) but advance pipelines with clock cycle
1	X	X	X	**0**	X	Drive data from output latches through C or D or Edata-Port
1	X	X	X	**1**	X	Three-state (high impedance) output C or D or Edata-Port
1	0	**0**	X	X	X	Register data from RAM to C&D or Edata output latches on rising edge
1	0	**1**	**0**	X	X	C&D or Edata output latches are transparent clock LO, latched clock HI
1	0	X	X	X	**0**	Edata-Port is configured for one read or one write per cycle
1	1	**0**	0	X	**0**	Configured for Late Read at C&D or Edata-Port: register Y_W & X_W from RAM to output latches on rising edge; output Y_W in clock HI, output X_W on next clock LO
1	1	**1**	0	X	**0**	Configured for Early Read at C&D or Edata-Port: output Y_W from RAM through transparent output latches in clock LO; latch X_W to output latches and output in clock HI
1	1	**0**	0	X	**1**	Configured for Edata Slow Read: hold RAM read data at Edata output latch; output Y_W at clock HI
1	1	**1**	0	X	**1**	Configured for Edata Slow Read: hold RAM read data at Edata Output Latch; output X_W at clock HI
1	**1**	X	**1**	X	X	Defines Clock-On-Rising/Falling mode for Edata Slow Inputs

Table II. ADSP-3128A Summary of Data Read and Output Control Modes

BS	DP	Wadtrn	Radtrn	A/B/C/D/Eadr$_6$ (Port Select)	Description
0	X	X	X	X	Disable chip (consistent with pipelines) but advance pipelines with clock cycle
1	X	0	X	X	Latch A or B or Eadr write addresses at clock HI
1	X	1	X	X	A or B or Eadr write address latches are transparent
1	X	X	0	X	Register C or D or Eadr read address latches on the rising edge
1	X	X	1	X	C or D or Eadr read address latches are transparent
X	1	X	X	0	Disable A/B/C/D/Edata-Port
1	1	X	X	1	Enable A/B/C/D/Edata-Port

Table III. ADSP-3128A Summary of Address Control Modes

ADDRESS LATCHES FOR BOTH SINGLE- AND DOUBLE-PRECISION MODES

The three read (clock HI) address latches and three write (clock LO) address latches hold the seven bits required for Register File addressing, Port Select and Bank Select. Radtrn controls whether the three read address latches are transparent or latched; Wadtrn controls whether the three write address latches are transparent or latched. When Radtrn/Wadtrn is HI, addresses presented at the read/write address ports are transferred directly to the RAM with no pipeline delay. When Radtrn is LO, addresses presented at the read address ports are registered on the rising edge of the clock, to be used during the next clock LO. When Wadtrn is LO, addresses presented at the write address ports are latched on the rising edge of the clock, to be used immediately during the next clock HI.

Both Radtrn and Wadtrn latch controls are registered and affect the configuration of the address latches on the rising clock edge in which they are registered. They remain in effect until the next rising edge.

Transparent addresses must be valid at least t_{AST} before the end of the phase in which they are used. The setup time for latched or registered addresses is t_{ASR}. All addresses must be held valid t_{AH} after the end of the phase in which they are asserted.

Output delays for transparent data reads from transparent addresses are referenced from address valid. However, an address valid prior to the clock LO in which the RAM is read provides no additional benefit. The output delay, t_{ODTT}, is referenced from address valid or the clock falling edge – whichever is *later*. The transparent read address must be held valid throughout the RAM read phase.

SINGLE-PRECISION OPERATION

Single-Precision mode is determined by the registered DP control being LO. Single-Precision mode must be asserted as shown in the timing diagrams to insure that the high-order single-precision address bits are *not* misinterpreted as Double-Precision Port Select bits and that latch controls are given their proper Single-Precision interpretation. A general discussion of dynamic switching between Single- and Double-Precision modes can be found below in "DP/SP Changeover." In Single-Precision mode, the Register File is configured as 128 words that are 16 bits in width. The 128 words are addressed by 7-bit addresses from the five address ports. All data paths and data latches behave as if they were 16 bits wide.

Up to five 16-bit data transfers per cycle are possible in Single-Precision mode. These transfers can be comprised of three writes and two reads, or two writes and three reads.

SP Reads

The operations of transferring data from RAM to a latch and from a latch to the output pins are logically distinct with the ADSP-3128A. Transfers from RAM to latch are called "reads" in this data sheet; transfers from latch to output port are called "outputs."

Read addresses can be transparent or registered (Figure 4). In all timing diagrams, the phase in which an address causes a RAM *read or write* is indicated by a Greek letter. For Figure 4's reads, all addresses shown cause a read in phase α. Not all controls are shown on this or other timing diagrams as explicit waveforms. In Figure 4, for example, the expression "Radtrn = 1" at a rising edge implies that Radtrn was asserted HI before that edge and met the standard setup and hold time requirements of Figure 2 for controls.

The output latches can be set transparent via registered controls CDtran HI and/or Etran HI. Note that one control, CDtran, affects both Cdata-Port and Ddata-Port output latches. From a transparent read address (Radtrn HI), read data when the output latches are transparent will be valid t_{ODTT} after a valid read address or after the clock falling edge – whichever is *later*. From a transparent read address, read data will be valid t_{ODC} after the rising clock edge when the output latches are in registered mode from the C&Ddata-Ports and/or the Edata-Port.

When the read addresses are registered (Radtrn LO), the data output timing is very similar except that the output delay for a transparent read is now referenced from a clock edge rather than address valid. The transparent read data will be valid t_{ODRT} after the falling clock edge.

Note that in all four combinations of address and output latching modes, the read from RAM took place in phase α. Specifying registered output latches simply introduces an additional clock phase of pipelining. Note also that for all Single-Precision reads, the data out is held valid throughout the phase *after* the data became valid. In the case of transparent data reads, the latch is actually holding the data valid for this phase. Data will be held valid t_{ODH} after the clock edge for all reads (in all modes).

Each read port has its own asynchronous three-state control: Ctri, Dtri and Etri. See Figure 3 for enable and disable timing.

SP Writes

Single-Precision mode must be asserted as shown in Figure 5 to insure that the high-order single-precision address bits are *not* misinterpreted as Double-Precision Port Select bits and that latch controls are given their proper Single-Precision interpretation. The operations of transferring data from a port to a latch and from a latch to the RAM are logically distinct with the ADSP-3128A. Transfers from port to latch are called "inputs" in this data sheet; transfers from latch to RAM are called "writes."

Write addresses can be transparent (Wadtrn HI) or registered (Wadtrn LO), exactly as with read addresses (Figure 5).

The Adata-Port and Bdata-Port input latches can be set to transparent, latched or clock-on-falling mode via the ABlt and ABht controls (Table I and Figure 5). The Edata-Port input latch can be set to transparent, latched or clock-on-falling mode via the Elt and Eht controls. When the "lt" and "ht" controls are both asserted HI, the latches are transparent ("t"). When only "lt" is asserted, the latches are in latched mode ("l"). When only "ht" is asserted the latches are in hold mode ("h"). When both controls are LO, the latches are in clock-on-falling mode.

Note that one set of controls, ABlt and ABht, affects both Adata-Port and Bdata-Port input latches. (These controls also permit holding the most recent write data at the input latches. See "SP Input to Input Latches and Hold" below.) These controls are always registered on the rising edge and become effective as of the next falling edge. When the input latches are transparent, write data must be valid t_{DST} before the end of the write phase. When the input latches are in latched mode, write data must be valid t_{DSR} before the beginning of the write phase. When the input latches are in clock-on-falling mode, write data must be valid t_{DSN} before the falling clock edge prior to the write phase. In all cases, the write data presented at write data ports must be held t_{DH} after the next clock edge.

The operations of inputting data to an input latch and writing data from the input latch to RAM are distinct. To write input data to the RAM, the asynchronous Write Inhibit Controls (Awinh, Bwinh, and/or Ewinh) must be LO as shown in Figure 5. Writes should be enabled no later than t_{WEN} before the falling edge.

Note that a write can be enabled later than a write can be inhibited. If you might want to inhibit a write to the Register File as late as the very phase in which a write is attempted, you can keep the A/B/Ewinh controls normally HI, i.e., write inhibited, and bring them LO every time you actually want to write. Alternatively, for simplicity, the A/Bwinh controls can be wired LO (write enable) and dummy writes be performed to an unused RAM location in every clock HI. Write addresses must always be stable, however, whenever the Write Inhibit controls are LO. In general, do not hardwire Ewinh LO; any Edata-Port output data will be written back to unintended RAM locations.

The write ports are prioritized with the Edata-Port of highest priority, followed by the Adata-Port, followed by the Bdata-Port. If writes to the same RAM location are attempted in a given clock HI phase, the data presented at the higher priority enabled write data port will be the data written to RAM.

SP Bidirectional Edata-Port

The Edata-Port will behave like any write port if treated as such according to the timing diagrams. Alternatively, it will also behave like any read port if treated as such. The Edata-Port can be used as a write port in one cycle, a read port in the next and a write port in the third cycle, as long as the Edata-Port is disabled to high impedance before setting up write data.

SP Input to Input Latches and Hold

Data input to the input latches can be held at those latches with the ABlt and ABht and Elt and Eht controls (Table I). These controls are always registered on the rising edge and become effective at the next falling edge. Figure 6 shows how data written to the latches in any of the three input modes can be held at a latch as long as desired. As of the falling edge after hold is asserted, data at the write data port is ignored and will be ignored until the next falling edge after one of the three input modes is asserted. The hold feature allows the input latches to be used for temporary data storage. Examples of using this feature include delaying a write to the RAM to avoid overwriting some data currently in the RAM or writing the same data to multiple RAM locations.

SP Bank Select

Bank Select is treated in exactly the same way in both Single-Precision and Double-Precision modes (Figure 12). The BS control is not registered in general but rather follows the addresses through the address latches (Figure 1). Hence, its setup requirement is t_{AST} and t_{ASR}, the setup requirement for read and write addresses for transparent and latched/registered modes respectively. All applicable requirements must be met. Flowing with addresses allows Bank Select to track all read and write pipelines as shown in Figure 12. When LO, writes will be disabled and output ports put in high impedance.

With Bank Select, the user's register file space can be extended "vertically" beyond 128 single-precision words to whatever register file space is desired. The user would typically use more than seven bits for addressing, decoding the high-order bits to select a horizontal row of ADSP-3128As that produce a single "word" and applying the low-order seven bits to the address ports in all rows.

The only restriction on extending the register file address space using Bank Select is that all reads and writes in a given cycle must be from the same horizontal row of ADSP-3128As. (Port Select removes this restriction for Double-Precision mode). In Single-Precision mode, the user can select/deselect individual *ports*, even if in different rows, using the asynchronous Write Inhibit and Three-State controls. The user would have to apply these with timing based on the latch modes currently selected to properly track the pipelines.

Note that the timing requirements for Bank Select are simple if write addresses are latched but are more complicated for transparent write addresses because of the way BS flows with the write address. For a Bank Deselect, the BS control must be LO in the clock HI write phase β (Figure 12). If writes are currently enabled, BS must be set up in phase α; if they're inhibited, BS is not needed LO until phase β to disable writes. The Write Inhibit controls for the three write data ports are independent. Therefore, if *any* Write Inhibit is LO (write enable) in phase α, BS will have to be LO in phase α to disable *all* writes.

DOUBLE-PRECISION OPERATION

Double-Precision mode is determined by the registered DP control being HI. A general discussion of dynamic switching between Single- and Double-Precision modes can be found below in "DP/SP Changeover." In Double-Precision mode, the Register File is configured as 64 words that are 32 bits in width. The 64 words are addressed by 6-bit addresses from the five address ports. The seventh, high-order bit used in Single-Precision addressing becomes a Port Select bit. All data paths between RAM and data latches are true 32-bit paths. That is, all 32-bit reads from the RAM to the latches and 32-bit writes to the RAM from the latches take place in a single read or write clock phase. The ports, however, are 16-bits wide. Data transfers through the ports are time-multiplexed.

The ADSP-3128A automatically controls the multiplexing through the data ports once the DP control is HI. The user only supplies one address to reference the two 16-bit halves of the data word transferred through the data ports. In Edata-Port Slow Input and Slow Read modes, however, the user has direct control over these multiplexers to allow communication with slower devices.

Up to five 32-bit data transfers per cycle are possible in Double-Precision mode. These five transfers can be comprised of three writes and two reads or two writes and three reads, depending on whether the Edata-Port is used as a read port or a write port.

Double-Precision mode is intended for interfacing to processors that use time-multiplexed 64-bit data, like Analog Devices' ADSP-32XX Floating-Point Multipliers and ADSP-32XX Floating-Point ALUs. Normally, two ADSP-3128A Multiport Register Files would be used "horizontally" to communicate with 32-bit buses.

In the descriptions that follow, one 16-bit half of a given ADSP-3128A's 32-bit word is referenced as an "Y_Word," the other half as an "X_Word." Note that normally a user would put together the Y_Words from two ADSP-3128As to create a 32-bit half of a 64-bit double-precision floating-point number. Similarly, the floating-point number's other 32-bit half would be constituted from the X_Words of two ADSP-3128As.

What is called a "Y_Word" in this data sheet is simply the 16-bit half of a 32-bit field that is written to the Register File first and read from the Register File first. But it is nothing more than a semantic convention; what are called here "Y_Words" can be used used to make up either Most Significant or Least Significant Words, depending on system requirements. The key point is that whichever half is written first will be the half read first.

DP Normal Reads

Double-Precision mode must be asserted as shown in Figure 7 to insure that the Port Select bits are *not* misinterpreted as Single-Precision address bits and that latch controls are given their proper Double-Precision interpretation. Addresses can be transparent or registered (Figure 7), just as in Single-Precision mode.

The two normal read options in Double-Precision mode are Early Read and Late Read. They are controlled via registered controls CDtran and/or Etran, which can make the output latches transparent or latched. The effect in Double-Precision mode is to create two pipelining options. Note that one control, CDtran, affects both Cdata-Port and Ddata-Port output latches.

Early Reads are generated when CDtran and/or Etran are HI. The Y_Word is read transparently from the RAM in phase γ through the output data port with delays, t_{ODRT} and t_{ODTT}, corresponding to registered and transparent read addresses respectively. The X_Word is also read from the RAM in phase γ but is held at the 32-bit output latch to be multiplexed out the output data port in the next phase with output delay t_{ODC}. Data hold times for Early Reads, as for all other kinds, is t_{ODH}. As described in "Address Latches," the transparent address can be set up before the RAM read phase but t_{ODTT} will then be referenced from the falling clock edge rather than address valid.

Late Reads are generated when CDtran and/or Etran are LO. As with Early Reads, both the Y_Word and X_Word are read from the RAM to the 32-bit output latches in phase γ. In the case of Late Read, the Y_Word is held at the output latch until the next phase, when it is driven off chip with delay t_{ODC}. The X_Word follows in the phase after that with the same delay characteristic of registered reads.

Each read port has its own asynchronous three-state control: Ctri, Dtri and Etri. See Figure 3 for enable and disable timing.

DP Writes

Double-Precision mode must be asserted as shown in Figure 8 to insure that the Port Select bits are *not* misinterpreted as Single-Precision address bits and that latch controls are given their proper Double-Precision interpretation. Addresses can be transparent or registered (Figure 8), just as with Double-Precision reads.

The two normal write options in Double-Precision mode are Early Write and Late Write. They are exactly analogous to Early Read and Late Read in that they offer two pipelining options. They are controlled via registered controls ABlt, ABht, Elt and Eht as shown in Figure 8 and Table II. Note that one set of controls, ABlt and ABht, affects both Adata-Port and Bdata-Port input latches. These controls become effective as of the falling edge after they are registered.

In Early Write, both Y_Word and X_Word are input to the 32-bit input latches before they are both written to RAM in phase δ. Both Y_Word and X_Word have the setup time requirement, t_{DSR}, characteristic of latched-mode data inputs. Data hold requirements for Early Write and all other writes is t_{DH}.

With Late Write, the user can input the Y_Word and X_Word into the Register File latches one half cycle later for a write to RAM in the same phase δ. The Y_Word is latched with setup time t_{DSR}. The X_Word, however, is transparently written to RAM in phase δ. Note that the setup requirement on the X_Word is therefore t_{DST}.

The actual write to RAM occurs in the single phase δ. Hence the Write Inhibit controls in Double-Precision work exactly as they do in Single-Precision. To write input data to the RAM, the asynchronous Write Inhibit Controls (Awinh, Bwinh and/or Ewinh) must be LO as shown in Figure 8. Writes should be enabled no later than t_{WEN} before the falling edge.

Note that a write can be enabled later than a write can be inhibited. If you might want to inhibit a write to the Register File as late as the very phase in which a write is attempted, you can keep the A/B/Ewinh controls normally HI, i.e., write inhibited, and bring them LO every time you actually want to write. Alternatively, for simplicity, the A/Bwinh controls can be wired LO (write enable) and dummy writes be performed to an unused RAM location in every clock HI. Write addresses must always be stable, however, whenever the Write Inhibit controls are LO. In general, do not hardwire Ewinh LO; any Edata-Port output data will be written back to unintended RAM locations.

The write ports are prioritized with the Edata-Port of highest priority, followed by the Adata-Port, followed by the Bdata-Port. If writes to the same RAM location are attempted in a given clock HI phase, the data presented at the higher priority enabled write data port will be the data written to RAM.

DP Edata-Port Slow Input and Slow Read

The bidirectional Edata-Port is intended to be the port interfaced to a system bus, which may run more slowly than local buses. To simplify the interface for Double-Precision, the ADSP-3128A provides a mode for loading the Y_Word and X_Word into the input latches over multiple ADSP-3128A clock cycles (Figure 9). Also a mode is provided for multiplexing Y_Word and X_Word read data from the output latches over multiple clock cycles (Figure 10).

For a Slow Input (Figure 9), the input latches are updated when there is a *transition* in a designated control input from one clock rising edge to the next clock rising edge. Both Clock-on-Falling and Clock-on-Rising Slow Input modes are supported. Rfltran LO indicates that data is to be loaded on the clock's falling edge, Rfltran HI indicates rising edge. In the case of Clock-on-Falling, the transition *in Eht* updates the latches while Elt is concurrently HI (Hold mode). Call Eht the "transition control" for Clock-on-Falling and Elt the "background control". Clock-on-Rising reverses the role of these two controls; the transition *in Elt* causes the latches to update while Eht is concurrently HI. In other words, for Clock-on-Rising, Elt becomes the transition control, Eht the background control. Regardless of which clock edge is loading the data, it must be set up to the input latches with set up time t_{DSR} as shown.

When the transition control goes from HI to LO, the external data will be input to the Y_Word position in the Edata input latch and be held there. When the transition control goes from LO to HI, the external data will be input to the X_Word position in the Edata input latch and be held there. A write to RAM can be enabled (with Ewinh LO) at the next clock HI from either latched or transparent Eadr.

For a Slow Read, registered control Eio, when asserted HI in conjunction with Double-Precision (DP HI), configures the Edata-Port for a Slow Read. When Eio goes HI, data at the output latch is held. In Figure 10, this is the 32-bit data read at phase γ. For a Slow Read, output delays will be t_{ODC}. Data will be held t_{ODH} after the clock edges shown in Figure 10. When configured for Slow Read, the ADSP-3128A's registered Etran control becomes a direct, asynchronous controller of the Edata-Port's Double-Precision output multiplexer. When Etran is LO, the Y_Word read from RAM in phase γ will be driven through the Edata-Port (if enabled with Etri). When Etran is HI, the X_Word read from RAM in phase γ will be driven through the Edata-Port (if enabled with Etri). The outputs will be driven as long as Eio is HI and Etran doesn't change.

DP Input to A&B Data-Port Input Latches and Hold

Data input to the A&Bdata-Port input latches can be held at those latches with the ABlt and ABht, controls (Table I). These controls are always registered on the rising edge and become effective as of the next falling edge. Figure 11 shows how data written to the latches in either Early Write or Late Write modes can be held at a latch as long as desired. As of the falling edge after hold is asserted with ABlt HI, data at the write data port is ignored. It will continue to be ignored until the next falling edge after ABlt goes LO. The hold feature allows the input latches to be used for temporary data storage. Note that the Edata-Port supports Input-and-Hold in SP only, since Elt is used in DP for Slow Edata-Port inputs.

DP Bank Select and Port Select

Bank Select is treated in exactly the same way in both Single-Precision and Double-Precision modes (Figure 12). The BS control is not registered in general but rather follows the addresses through the address latches (Figure 1). In Double-Precision, the seventh address bit (not needed for Double-Precision addressing) is redefined to function as Port Select for the ports being addressed. DP must be asserted HI as shown in Figure 13 to insure that these bits are interpreted as Double-Precision Port Selects and not Single-Precision address bits (and that latch controls are given their proper Double-Precision interpretation).

Behaving as addresses, both BS and A/B/C/D/Eadr$_6$ have setup requirements of t_{ASTx} and t_{ASRx}, the setup requirement for read and write addresses for transparent and latched/registered modes, respectively. All applicable requirements must be met. Flowing with addresses allows Bank Select and Port Select to track all read and write pipelines as shown in Figures 12 and 13. When LO, writes will be disabled and output ports put in high impedance.

The only restriction on extending the register file address space using Bank Select is that all reads and writes in a given cycle must be from the same horizontal row of ADSP-3128As. Port Select removes this restriction for Double-Precision mode (only). Like Bank Select, the Port Select controls track the ADSP-3128A's internal pipelines. But since every port can be independently selected or deselected, reads can be made from and writes made to any combination of locations in the user's register file space. They need not be all made from the same horizontal row.

Note that the timing requirements for Bank Select and Port Select are simple if write addresses are latched but are more complicated for transparent write addresses because of the way BS and A/B/Eadr$_6$ flow with the write address. For a Bank or Port Deselect, the BS or A/B/Eadr$_6$ control must be LO in the clock HI write phase β (Figures 12 and 13). If writes are currently enabled, BS or A/B/Eadr if they're inhibited, BS or A/B/Eadr$_6$ is not needed LO until phase β to disable writes. Since the Write Inhibit controls for the three write data ports are independent, if any is enabled in phase α, BS or A/B/Eadr$_6$ will have to be LO in phase α to disable all writes.

DP/SP Changeover

Many controls are interpreted and internal states affected by the DP control. The timing diagrams show when DP must be HI and when it must be LO to accomplish the operation described in each timing diagram. For times when the state of DP is not explicitly shown, it can be changed. That is, the user can dynamically reconfigure the ADSP-3128A from Single-Precision to Double-Precision and conversely as long as these restrictions are observed.

Internal RAM Organization

It may be useful to know that a 32-bit word in Double-Precision mode consists of two 16-bit words that can be addressed in Single-Precision mode with seven bit addresses by the six bit address used in double precision mode (*n*) and that address plus 64 (*n+64*). The Y_Word of the double-precision word will be in *n*; the X_Word in *n+64*. By switching from Double- to Single-Precision, the user can independently access the Y_Word and the X_Word.

DESIGN CONSIDERATIONS

Power Up

At power up, any or all of the three output ports, Edata-Port, Cdata-Port or Ddata-Port, may be driving off chip. Because of pipelining, Bank Select should not be used to serve a reset or "chip select" function unless no other devices on the buses driven by these ports could themselves possibly be driving. Bank Select will tristate these ports, but they cannot be guaranteed to be in a high impedance state until t_{DIS} into the second cycle after the rising edge at which BS is LO (Figure 12).

Any ADSP-3128A output port that shares a buses should be forced into a high impedance state at power up using the Etri/-Ctri/Dtri controls. The bits driving these pins from microcode can be gated with the user's general system reset control.

Power Supply Decoupling

The ADSP-3128A register file is designed with high speed drivers on all output pins. This means that large peak currents may pass through the driver ground and V_{DD} pins, particularly when all output port lines are simultaneously charging their load capacitance in transition, whether from LO to HI or vice versa. These peak currents can cause a large disturbance in the ground and supply lines. To help isolate the effects of this disturbance, the ADSP-3128A provides separate pins for driver GND and V_{DD}s and logic GND and V_{DD}s.

The ADSP-3128A's GND and V_{DD} pins must be tied directly to solid ground and V_{DD}planes and properly bypassed. Lead lengths and trace lengths should be as short as possible. The ground plane should tie to driver GND in particular with a very low inductance path. High frequency bypass capacitors (0.1μF ceramic) should be located as close as possible to the V_{DD} pins. Low frequency bypass capacitors (20μF tantalum) should be located *outside* the chip perimeter (not directly under the chip). System noise immunity can be improved by careful design of V_{DD} and GND planes. See the Applications Note, "Power and Ground Connection Guidelines for Pin Grid Arrays" for layout suggestions.

KEY CHANGES FROM JUNE 1988 ADSP-3128A A PRELIMINARY DATA SHEET

The ADSP-3128A is a pin-compatible speed-upgrade to the ADSP-3128 with the following qualifications:

1. The specification t_{WINH} has been added and the specification t_{WIN} has been redefined to make it easier to use. The Write Inhibit Delay (t_{WIN}) is the maximum time after the rising edge of the clock before the A/B/Ewinh pin must be high to inhibit a write to the register file. The new specification Write Inhibit Control Hold Time (t_{WINH}) is the minimum hold time required after the falling edge of the clock to insure that the enable write or inhibit write has occurred. New versions of Figure 5 and Figure 8 show this timing.
2. The Elt and Eht lines are reversed in Figure 9 and the last two entries of Table I in the June 1988 Data Sheet for Double Precision Clock-on-Rising Slow Inputs to the E-port. Figure 9 and Table I have been corrected. Paragraph two of DP Edata-Port Slow Input and Slow Read on Page 3–54 has also been changed.
3. The specifications t_{AST} and t_{ASR} have been separated for reads and writes. The new specifications are:

Transparent Address Setup – Read	t_{ASTR}
Transparent Address Setup – Write	t_{ASTW}
Registered Address Setup – Read	t_{ASRR}
Registered Address Setup – Write	t_{ASRW}

4. The low-level input voltage level on the Clock line is 0.6V maximum. On all other lines it remains 0.8V maximum.
5. I_{DD} Supply Current is 600mA maximum.
6. The Edata-port can function in any one cycle as either a read port or a write port. It cannot both read and write in one cycle.
7. Extra reads from the C, D and Edata-ports are no longer allowed.
8. The following specifications have been removed:

t_{EDIS}	Three-State E Port Auto-Disable
t_{HIER}	Clock Period HI – Write Plus Extra Read
t_{ODRTH}	Clock Address-to-Transparent Delay – Extra Reads
t_{CLK}	Clock Period – Clocked Reads
t_{CLKS}	Clock Period – Trans Reads
t_{CLKA}	Clock Period – Transparent I/O
t_{ASTBS}	Trans. Clk HI Bank Select Setup
t_{ODCE}	Clk-to-Data Output Delay – Eport

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3128A J and K Grades Min	ADSP-3128A J and K Grades Max	ADSP-3128A S and T Grades[2] Min	ADSP-3128A 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-3128A J and K Grades Min	ADSP-3128A J and K Grades Max	ADSP-3128A S and T Grades[2] Min	ADSP-3128A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0				V
V_{IHA}	High-Level Input Voltage, CLK and All Asynchronous Control Inputs	@ V_{DD} = max	2.2				V
V_{IL}	Low-Level Input Voltage	@ V_{DD} = min		0.8			V
V_{IL}	Low-Level Input Voltage (CLK)	@ V_{DD} = min		0.6			V
V_{OH}	High-Level Output Voltage	@ V_{DD} = min & I_{OH} = −1.0mA	2.4				V
V_{OL}	Low-Level Output Voltage	@ V_{DD} = min & I_{OL} = 4.0mA		0.4			V
I_{IH}	High-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 5.0V		10			μA
I_{IL}	Low-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 0.0V		10			μA
I_{OZ}	Three-State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = 0V or max		50			μA
I_{DD}	Supply Current[3]	@ max Clock Rate; TTL Inputs (CLK = 0, 3V)		600			mA
I_{DDQ}	Supply Current-Quiescent	All V_{IN} = 2.4V		100			mA

3

ORDERING INFORMATION

Part Number	Temperature Range	Package
ADSP-3128AJG	0 to +70°C	144-Pin Grid Array
ADSP-3128AKG	0 to +70°C	144-Pin Grid Array
ADSP-3128ASG	−55 to +125°C	144-Pin Grid Array
ADSP-3128ATG	−55 to +125°C	144-Pin Grid Array
ADSP-3128ASG/883B	−55 to +125°C	144-Pin Grid Array
ADSP-3128ATG/883B	−55 to +125°C	144-Pin Grid Array

SWITCHING CHARACTERISTICS

Parameter		ADSP-3128A J Grades 0 to +70°C Min	Max	K Grades 0 to +70°C Min	Max	S Grades[2] −55 to +125°C Min	Max	T Grades[2] −55 to −125°C Min	Max	Unit
t_L	Clock LO Period			20						ns
t_H	Clock HI Period			22						ns
t_{CS}	Control Setup			10						ns
t_{CH}	Control Hold			1						ns
t_{ASTR}	Transparent Address Setup – Read			18						ns
t_{ASTW}	Transparent Address Setup – Write			30						ns
t_{ASRR}	Registered Address Setup – Read			4						ns
t_{ASRW}	Registered Address Setup – Write			11						ns
t_{AH}	Address Hold			3						ns
t_{ENA}	Three-State Enable Delay			2	21					ns
t_{DIS}	Three-State Disable Delay				11					ns
t_{DISBS}	Three-State Disable Delay – Bank & Port Sel				24					ns
t_{ODTT}	Trans Adr-to-Trans Output Delay				39					ns
t_{ODC}	Clk-to-Data Output Delay – C & Dports				18					ns
t_{ODRT}	Clkd Adr-to-Trans Output Delay				40					ns
t_{ODH}	Output Data Hold			3						ns
t_{DSR}	Latched Data Setup				7					ns
t_{DST}	Transparent Data Setup				18					ns
t_{DSN}	Clock-on-Falling Data Setup				12					ns
t_{DH}	Input Data Hold			1						ns
t_{WEN}	Write Enable Setup			23						ns
t_{WIN}	Write Inhibit Delay				0					ns
t_{ATBE}	Trans Adr to Write Enable			1						ns
t_{WINH}	Write Inhibit Control Hold Time			0						

NOTES

[1]All min and max specifications are over power-supply and temperature range indicated. Input levels are GND and 3.0V. Rise times are 5ns. Input timing reference levels and output reference levels are 1.5V, except for t_{ENA}, t_{DIS} and t_{DISBS} which are as indicated in Figures 3, 12 and 13.

[2]S and T grade parts are available processed in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-3128A can be found in Analog Devices' Military Data Book. Regular S and T grade parts are tested at +125°C.

[3]Worst-case with all outputs switching twice per cycle. (Example: DP Reads)

Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

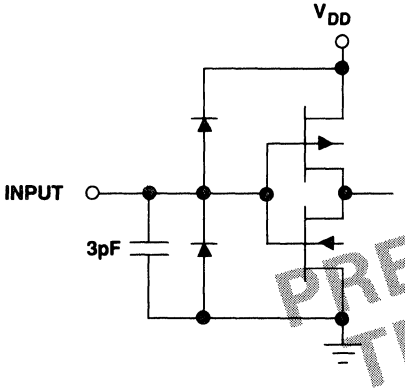
Supply Voltage . . . . . −0.3V to +7V
Input Voltage . . . . . −0.3V to V_{DD} +0.3V
Output Voltage Swing . . . . . −0.3V to V_{DD} +0.3V
Operating Temperature Range (Ambient) . . . . −55°C to +125°C
Storage Temperature Range . . . . . −65°C to +150°C
Lead Temperature (10sec) PGA . . . . . +300°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

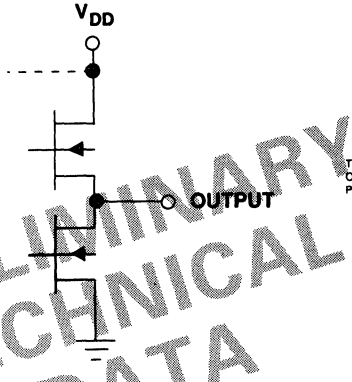
ESD SENSITIVITY

The ADSP-3128A features proprietary input protection circuitry. Per Method 3015 of MIL-STD-883C, the ADSP-3128A has been classified as a Class 1 device.

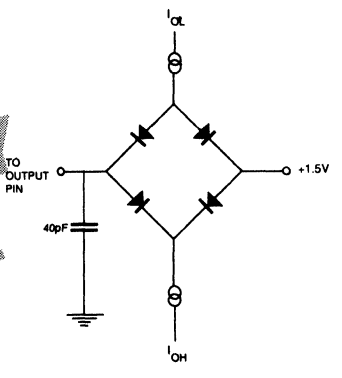
Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should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



Equivalent Input Circuits



Equivalent Output Circuits



Normal Load for ac Measurements

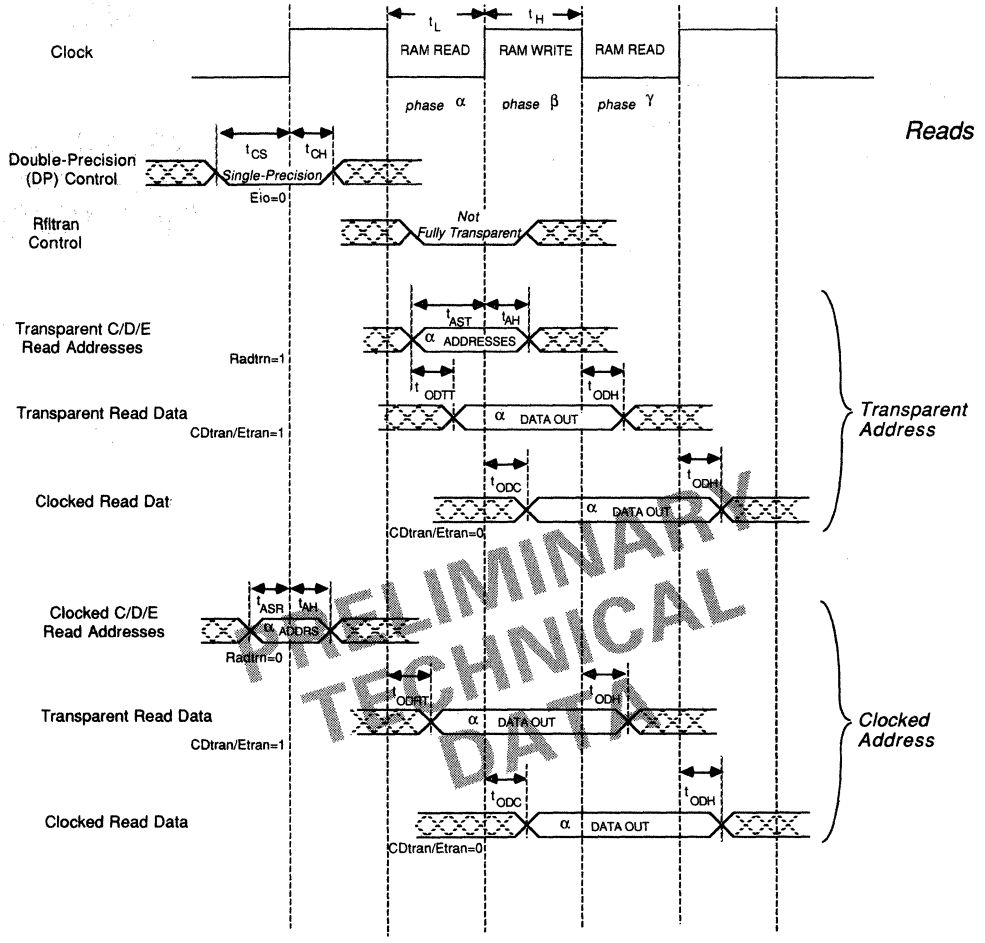


Figure 4. ADSP-3128A Single-Precision Read Output Timing

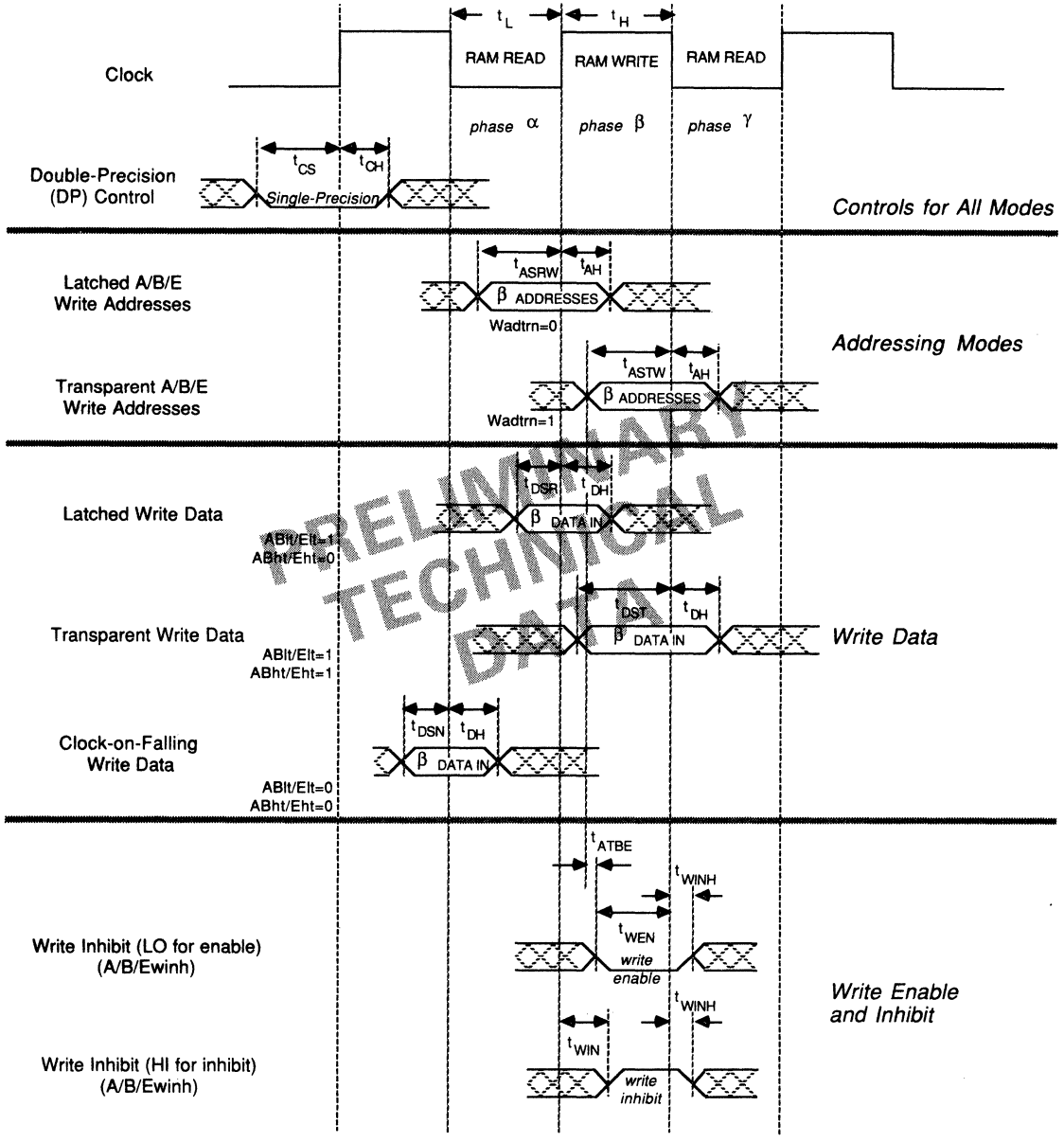


Figure 5. ADSP-3128A Single-Precision Write Input Timing

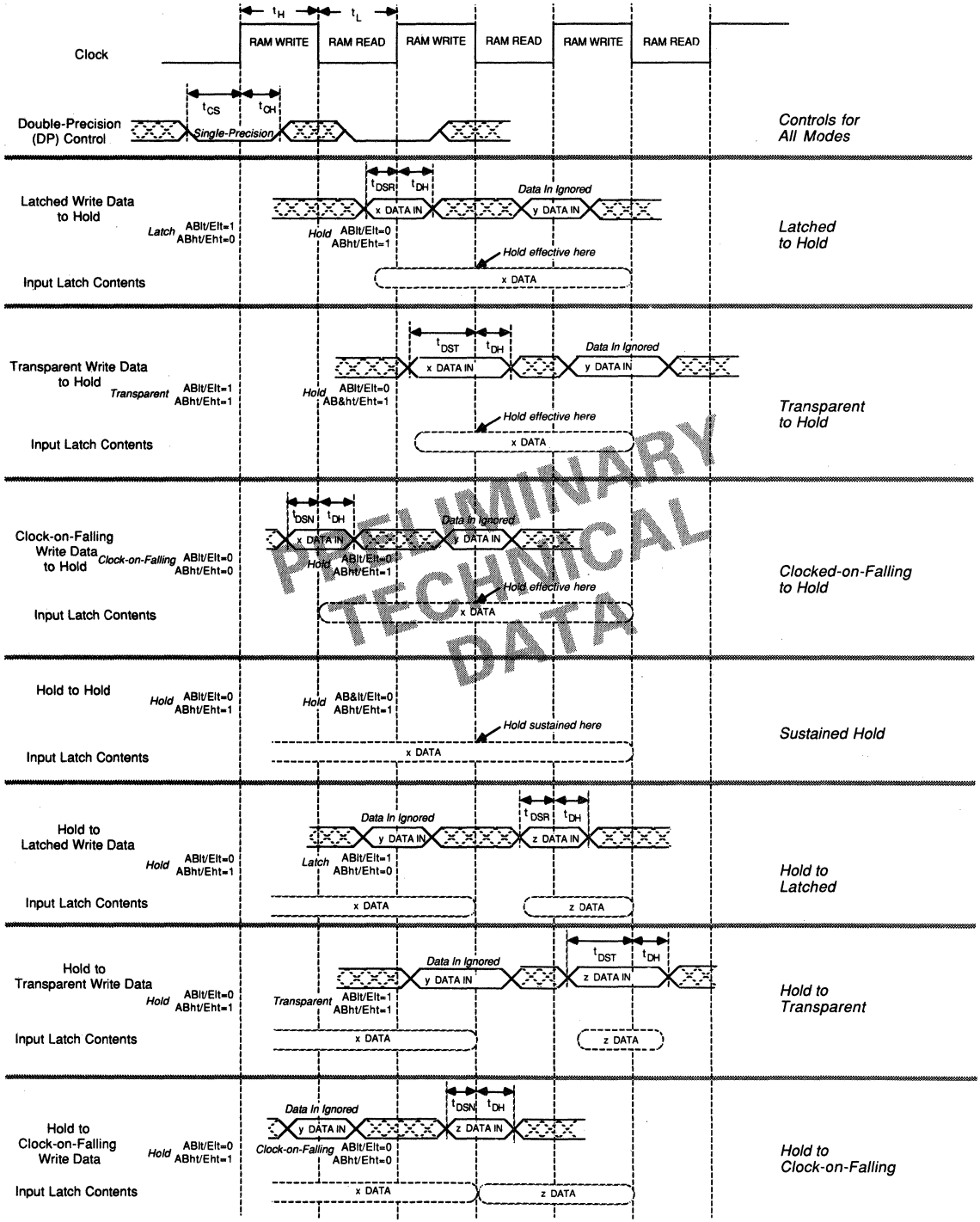


Figure 6. ADSP-3128A Single-Precision Write to Input Latches and Hold Timing

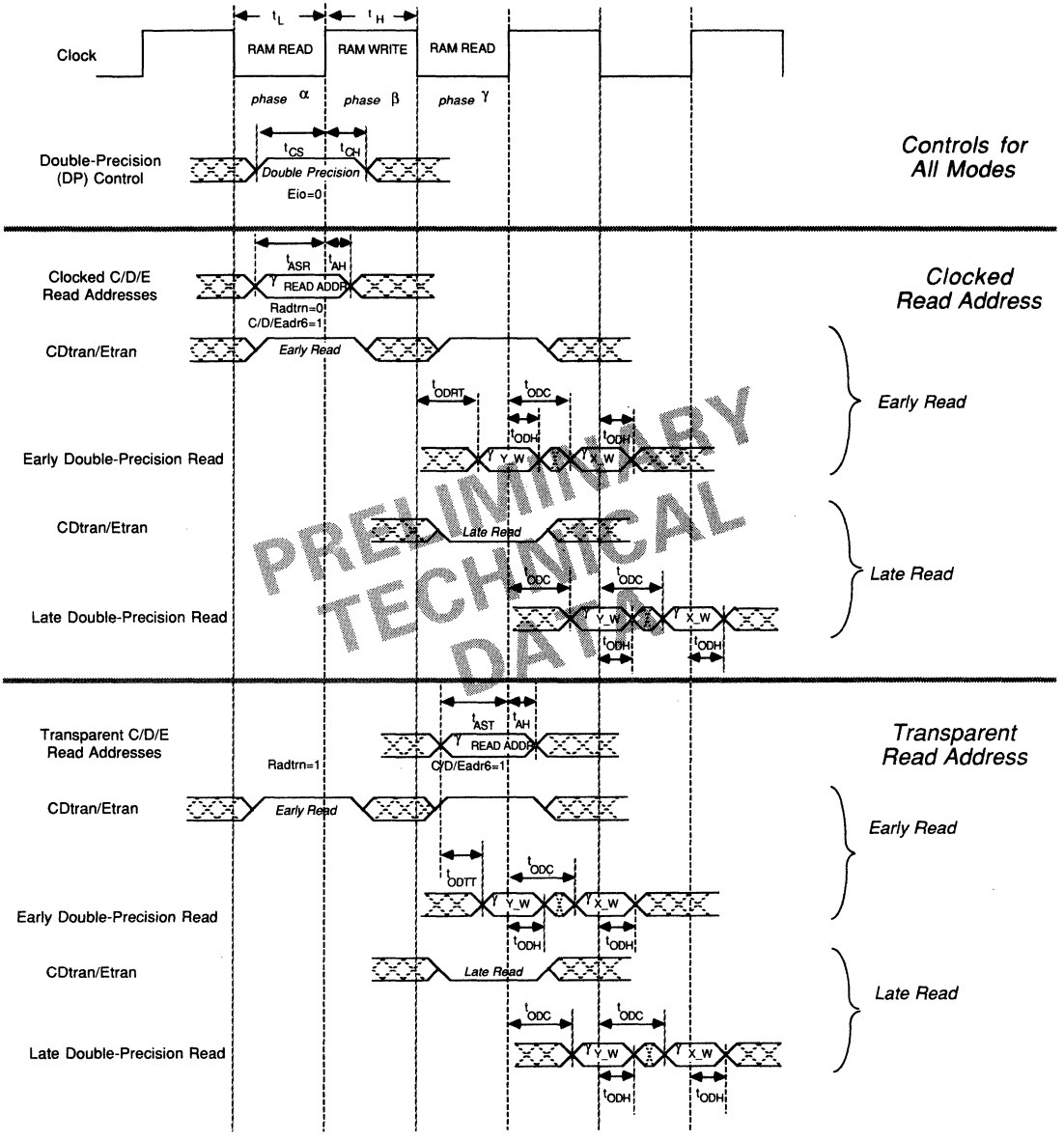


Figure 7. ADSP-3128A Double-Precision Read Output Timing

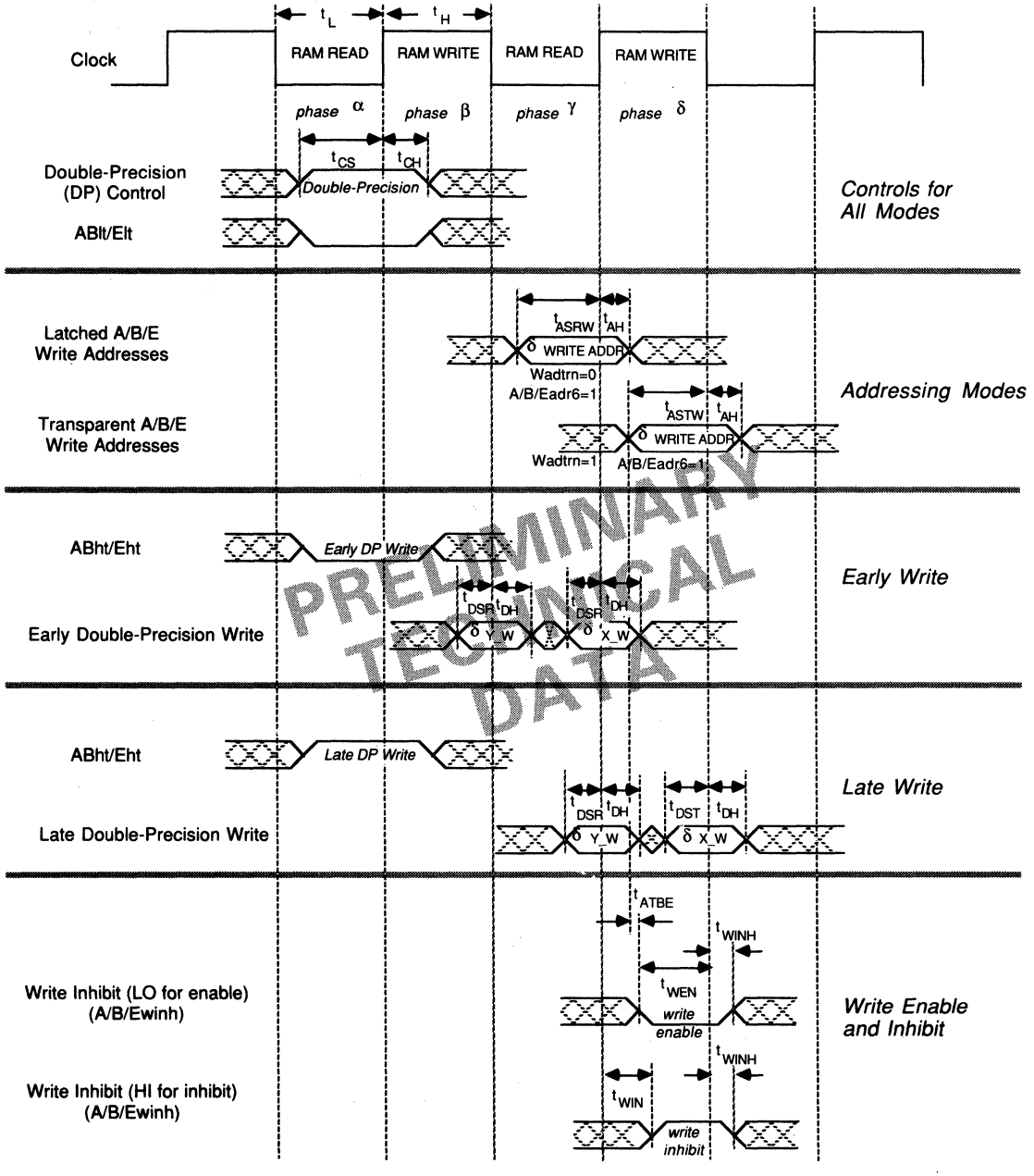
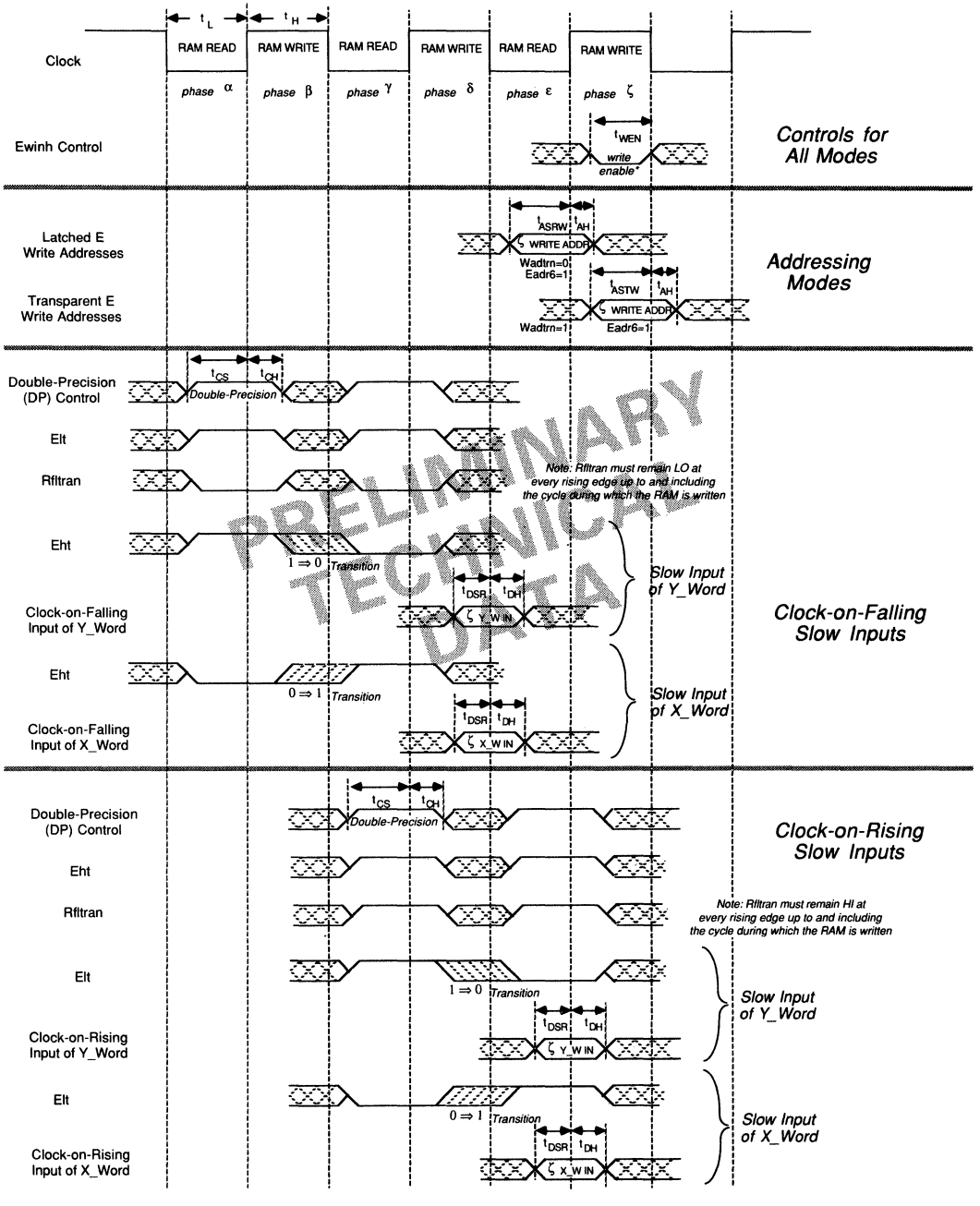


Figure 8. ADSP-3128A Double-Precision Write Input Timing



* See Figure 9 for the complete set of conditions for Ewinh

Figure 9. ADSP-3128A Double-Precision Slow Edata-Port Input Timing

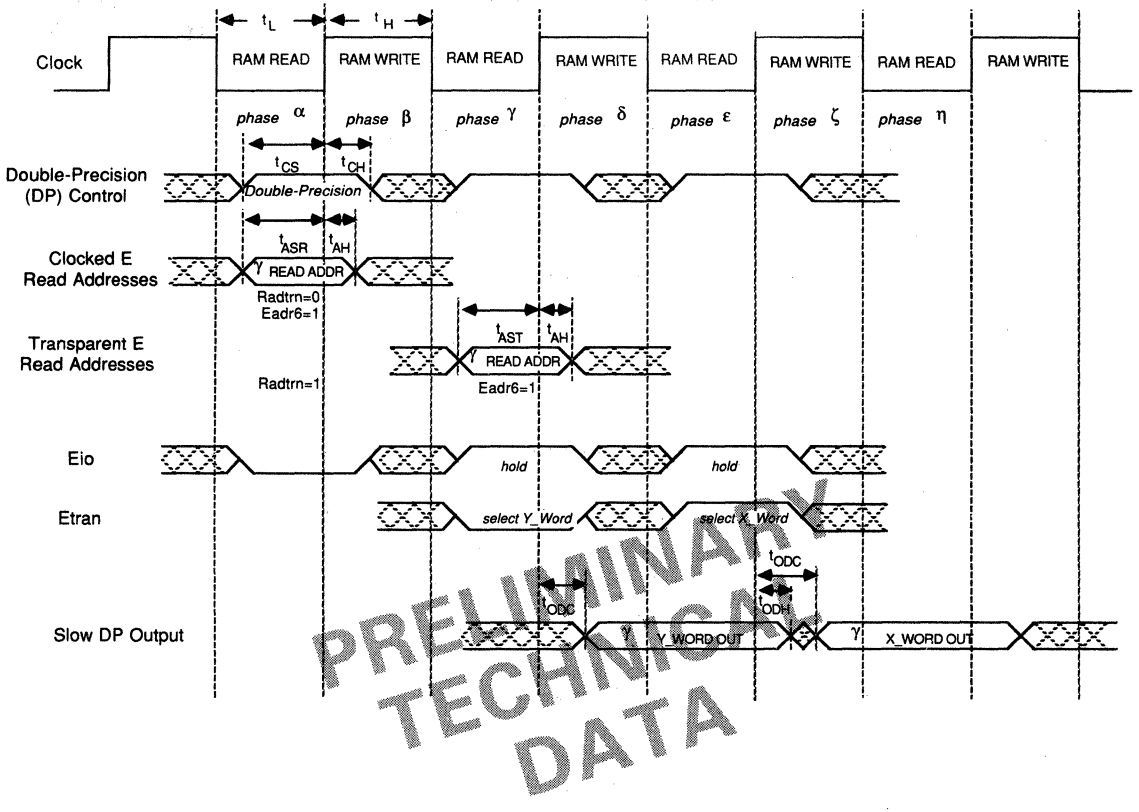


Figure 10. ADSP-3128A Double-Precision Slow Edata-Port Read Output Timing

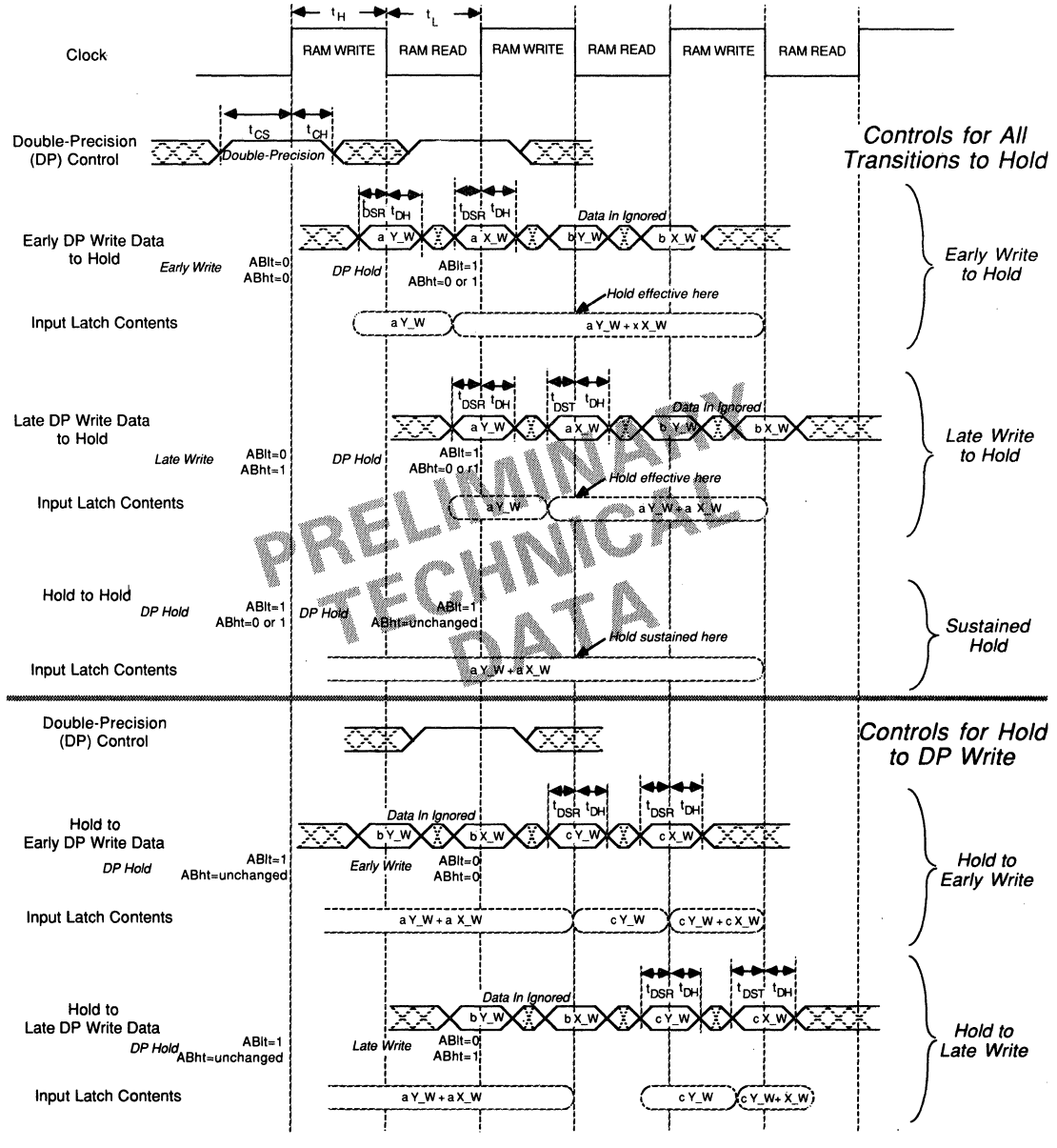


Figure 11. ADSP-3128A Double-Precision Write to Input Latches and Hold Timing (Adata-Port and Bdata-Port only)

This information applies to a product under development. Its characteristics and specifications are subject to change without notice. Analog Devices assumes no obligation regarding future manufacture unless otherwise agreed to in writing.

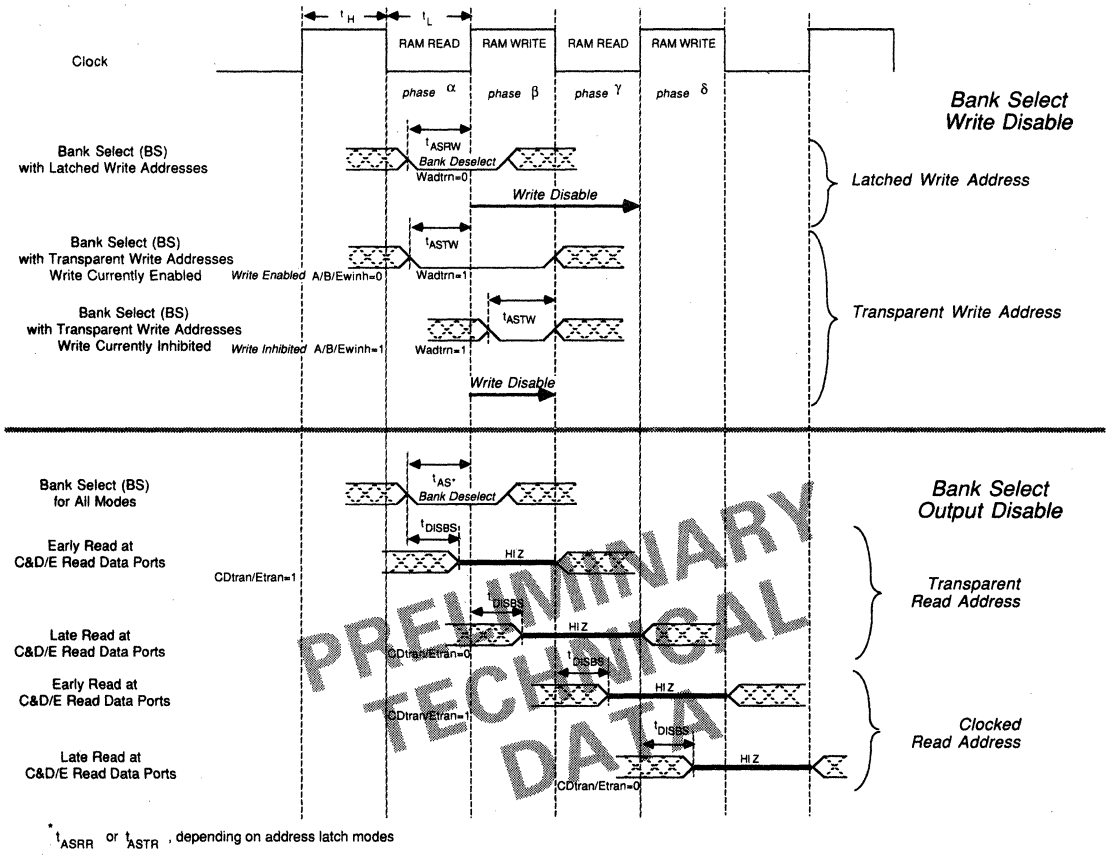


Figure 12. ADSP-3128A Bank Select Timing

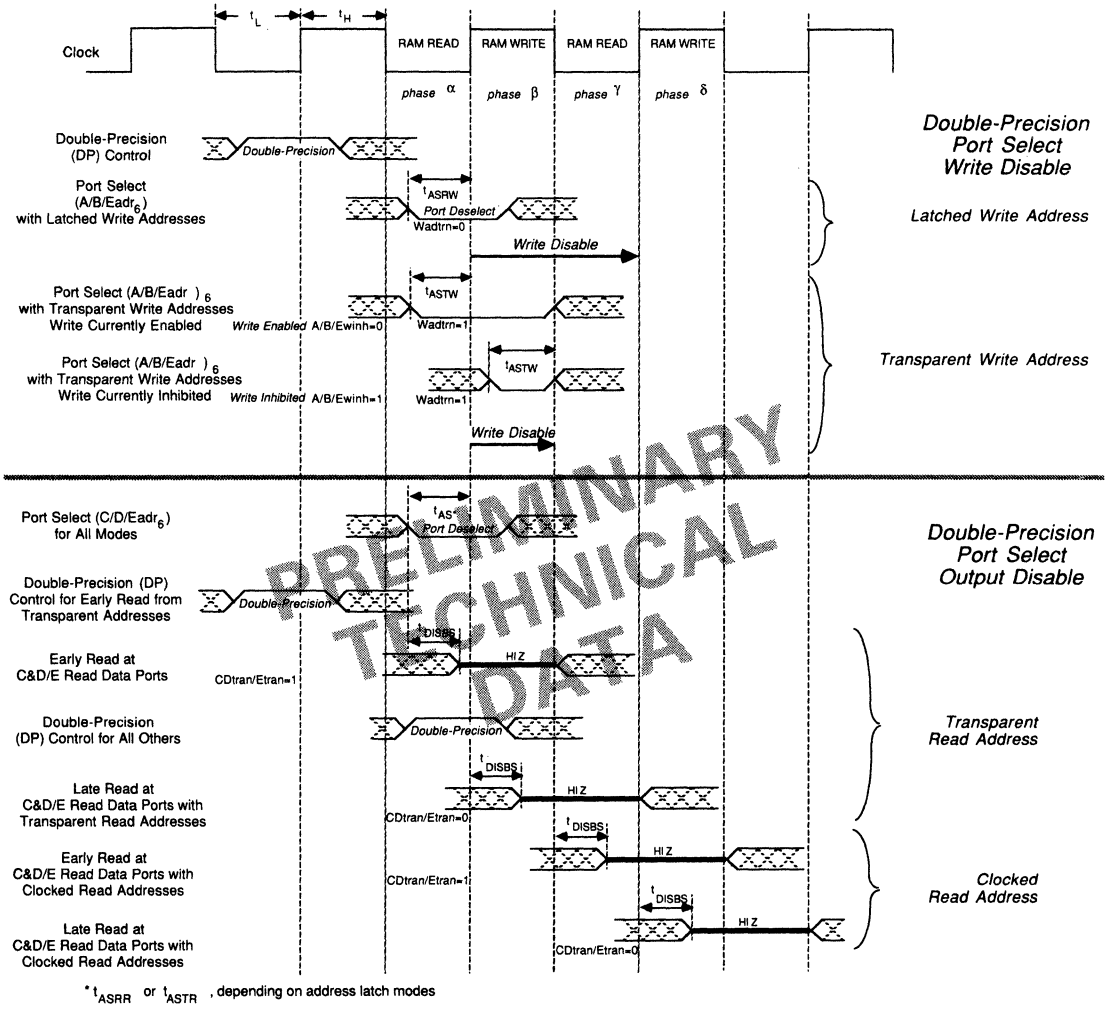


Figure 13. ADSP-3128A Double-Precision Port Select Timing

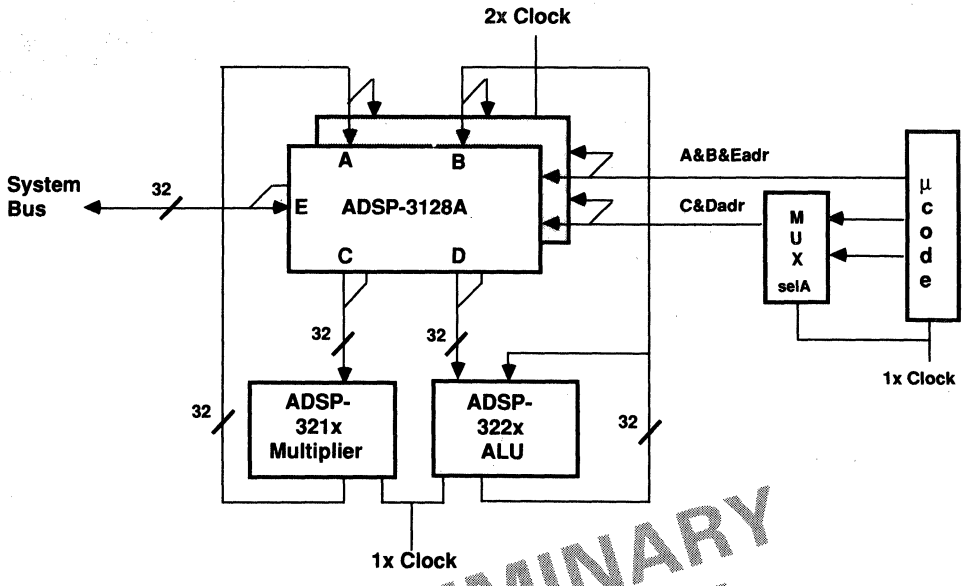


Figure 14. ADSP-3128A Single-Precision Application with ADSP-32XX

Muxing the read addresses allows two reads (at 1×clock) for loading the input ports of both the ADSP-321X and ADSP-322X with two 32-bit words per 32XX cycle (at 1×clock) while still using 1×μcode rates. In this application, write data is latched on clock HI and read data is registered on the rising edge. Write addresses are latched; read addresses are transparent.

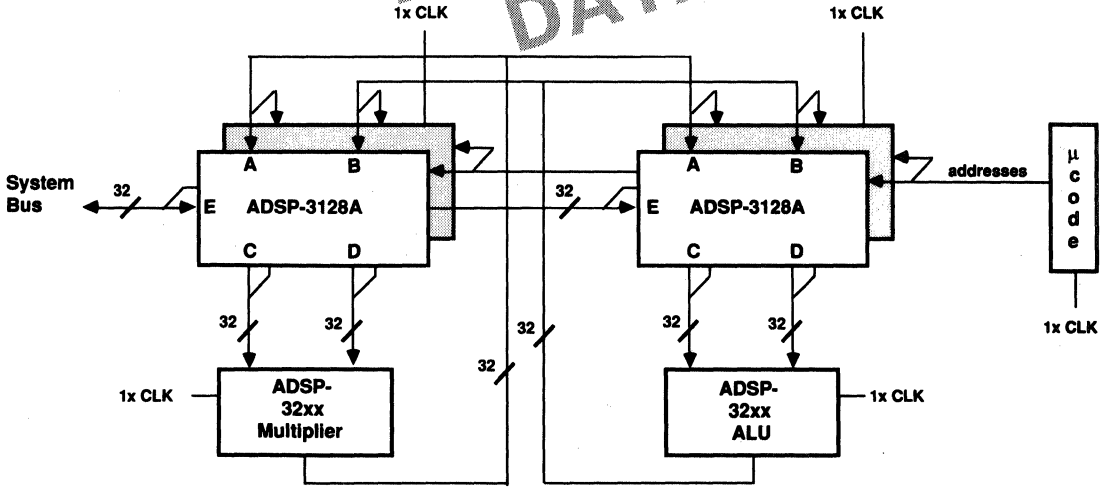


Figure 15. ADSP-3128A Seven-Port Double-Precision Application with ADSP-32XX

Double-Precision mode allows transfer of both MSW and LSW in a single cycle while still using μcode at the same cycle rate. Pairing pairs of ADSP-3128As creates a seven-port register file for unconstrained data transfers. The same data is always written to both the right and left pairs (therefore, the same A, B and Eadrs go to both pairs). In this application, Early Writes are used at the input ports for the simplest interface to the floating-point chipset's output. The data read from the two sides is generally distinct, so the C and Dadrs for each pair are distinct. Late Reads match the input loading requirements of these chips and are, therefore, used on the rightmost pair of ADSP-3128As.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Q	Bdata8	Bdata5	Bdata2	Bdata0	Adata13	Adata10	Adata9	Adata7	Adata4	Adata3	Eht	ABht	ABlt	Awinh	Aadr0	Q
P	Bdata12	Bdata9	Bdata6	Bdata4	Bdata1	Adata14	Adata12	Adata8	Adata2	Adata1	Elt	DP	Bwinh	**internal GND**	Aadr3	P
N	Edata15	Bdata13	Bdata10	Bdata7	Bdata3	Adata15	Adata11	Adata6	Adata5	Adata0	Ewinh	**internal GND**	**internal GND**	Aadr2	Aadr6	N
M	Edata14	Bdata15	Bdata11										Aadr1	Aadr4	Badr1	M
L	Edata12	Edata13	Bdata14										Aadr5	Badr0	Badr4	L
K	Edata8	Edata10	Edata11										Badr2	Badr3	Cadr0	K
J	Edata7	Edata9	Edata6										Badr6	Badr5	Cadr1	J
H	Edata4	Edata3	Edata5										Cadr4	Cadr2	Cadr3	H
G	Edata2	Ddata15	Edata0										Cadr5	Dadr1	Cadr6	G
F	Edata1	Ddata13	Ddata12										Dadr3	Dadr2	Dadr0	F
E	Ddata14	Ddata10	Ddata8										Eadr2	Dadr5	Dadr4	E
D	Ddata11	internal GND	driver Vdd	**driver GND**									Eadr5	Eadr1	Dadr6	D
C	Ddata9	Ddata7	Ddata4	Ddata3	Ddata0	Cdata10	driver GND	driver GND	Cdata1	Etri	Wadtrn	driver Vdd	internal Vdd	Eadr3	Eadr0	C
B	**driver GND**	Ddata5	Ddata1	Cdata15	Cdata12	Cdata9	Cdata8	Cdata4	Cdata0	CDtran	Ctri	Radtrn	Rfltran	Eadr6	Eadr4	B
A	Ddata6	Ddata2	Cdata14	Cdata13	Cdata11	Cdata7	Cdata6	Cdata5	Cdata3	Cdata2	Etran	Dtri	BS	Eio	CLK	A
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

BOTTOM VIEW

ADSP-3128A Pinout

Floating-Point Components
Contents

Since the introduction of our first floating-point chips in 1984, Analog Devices has been a leader in supplying fast floating-point arithmetic units. We currently produce four floating-point chipsets, each consisting of a multiplier and an ALU. All parts implement the IEEE Standard 754 for Binary Floating-Point Arithmetic. All deliver the highest performance in throughput and latency with the advantages of CMOS processing. Our floating-point chips are supported by our Word-Slice product line which includes address generators, microcode program sequencers and a five-port register file, the ADSP-3128A. All of these parts are described in the "Microcode Support Components" section of this databook.

The floating-point units provide performance to 40 MFLOPS and precision to 64-bits. With only one internal pipeline register, all attain high pipelined throughput while minimizing latency. The key advantages of each chipset are summarized below and in the Selection Guide on the next page.

ADSP-3210 & ADSP-3211 DOUBLE-PRECISION MULTIPLIERS
ADSP-3220 & ADSP-3221 DOUBLE-PRECISION ALUs
These chips process operations on three data formats: 32-bit IEEE single-precision, 32-bit fixed-point and 64-bit IEEE double-precision. There are two multipliers and two ALUs in this group; either ALU can be used with either multiplier.

ADSP-3210/ADSP-3211 Multipliers
The ADSP-3211 is a three-port multiplier with an I/O structure identical to the ADSP-3220/ADSP-3221. Throughput for the ADSP-3211LG is 20 MFLOPS single-precision, 5 MFLOPS double-precision and 20 MIPS fixed-point. The ADSP-3211 operates directly on both twos-complement, unsigned-magnitude and mixed-mode fixed-point numbers. The ADSP-3210 offers the capability to conserve on-board space and cost with a two-port structure while still maintaining full pipelined throughput. Throughput with the ADSP-3210 reaches 16.6 MFLOPS single-precision, 4 MFLOPS double-precision and 16.6 MIPS fixed-point. The ADSP-3210's fixed-point computations are twos-complement only.

ADSP-3220/ADSP-3221 ALUs
The ADSP-3220 and ADSP-3221 ALUs both have a three-port structure and attain 10 MFLOPS throughput for single- and double-precision floating-point and 10 MIPS for fixed-point number formats. The ADSP-3221 is pin-compatible with the ADSP-3220 and can compute the IEEE exact division and square root functions completely on-chip.

ADSP-3201/ADSP-3202 SINGLE-PRECISION CHIPSET
The ADSP-3201 Floating-Point Multiplier and the ADSP-3202 Floating-Point ALU offer the capability to build a high-performance, single-precision only system at minimum cost. Both chips offer the same three-port structure as the ADSP-3211/ADSP-3221 and both process 32-bit floating-point and 32-bit fixed-point numbers. The chips reach 10MHz throughput for single and fixed-point operations. The compatibility of the single-precision parts with the ADSP-3211 and ADSP-3221 provides an upgrade path to double-precision.

ADSP-3212 MULTIPLIER & ADSP-3222 ALU
These next generation 1.0μm CMOS upgrades to the ADSP-3211 and ADSP-3221 build on their key features: full IEEE 754 arithmetic, only one internal pipeline register, low power CMOS technology and MIL-STD-883B processing. The one micron process used yields a throughput of 40 MFLOPS. Because of minimal pipelining, latency is about 150ns. Exact division is computed at a 300ns (single-precision) or 600ns (double-precision) rate. Exact square root is also supported.

Selection Guide

FLOATING-POINT COMPONENTS

Part	Grade	Number of Ports	Pipelined Throughput (ns) 32-Bit	Pipelined Throughput (ns) 64-Bit	Latency (ns) 32-Bit	Latency (ns) 64-Bit	IEEE Exact Divide (μs) Single Precision	IEEE Exact Divide (μs) Double Precision	IEEE Exact Square Root (μs) Single Precision	IEEE Exact Square Root (μs) Double Precision
ADSP-3211 Multiplier	L	3	50	200	140	315				
	K	3	100	400	240	590				
	J	3	125	500	300	738				
	U	3	70	280	190	400				
	T	3	125	500	300	738				
	S	3	150	600	360	885				
ADSP-3210 Multiplier	L	2	60	240	190	370				
	K	2	100	400	290	590				
	J	2	125	500	363	738				
	U	2	75	300	238	463				
	T	2	125	500	363	738				
	S	2	150	600	435	885				
ADSP-3212 Multiplier/ Divider	K	3	50	50	130	155	0.3	0.6		
	J	3	60	60	157	187	0.36	0.72		
	T	3	58	58	150	179	0.345	0.69		
	S	3								
ADSP-3221 ALU	K	3	100	100	240	290	1.6	3	2.9	5.8
	J	3	125	125	300	363	2	3.75	3.63	7.25
	T	3	125	125	300	363	2	3.75	3.63	7.25
	S	3	150	150	360	435	2.4	4.5	4.35	8.7
ADSP-3220 ALU	K	3	100	100	240	290				
	J	3	125	125	300	363				
	T	3	125	125	300	363				
	S	3	150	150	360	435				
ADSP-3222 ALU	K	3	50	50	130	155	0.8	1.5	1.45	2.9
	J	3	60	60	157	187	0.96	1.8	1.74	3.48
	T	3	58	58	150	179	0.92	1.725	1.67	3.34
	S	3								
ADSP-3201 Multiplier	K	3	100	240						
	J	3	125	300						
	T	3	125	300						
	S	3	150	360						
ADSP-3202 ALU	K	3	100	240			1.6		2.9	
	J	3	125	300			2		3.63	
	T	3	125	300			2		3.63	
	S	3	150	360			2.4		4.35	

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

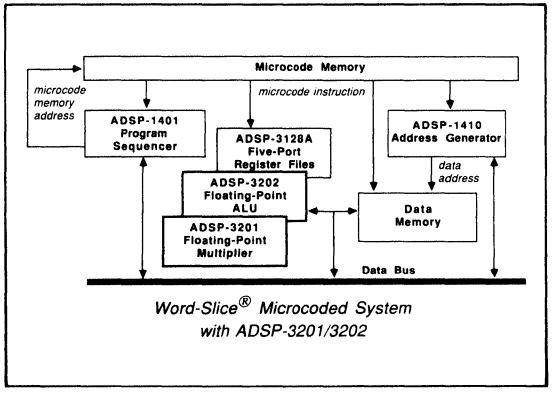
APPLICATIONS

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

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.

Word-Slice is a registered trademark of Analog Devices, Inc.

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**

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.

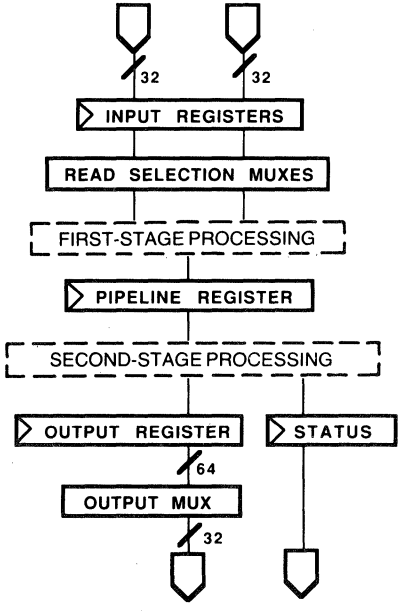


Figure 1. ADSP-3201/3202 Generic Architecture

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.

4

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)	

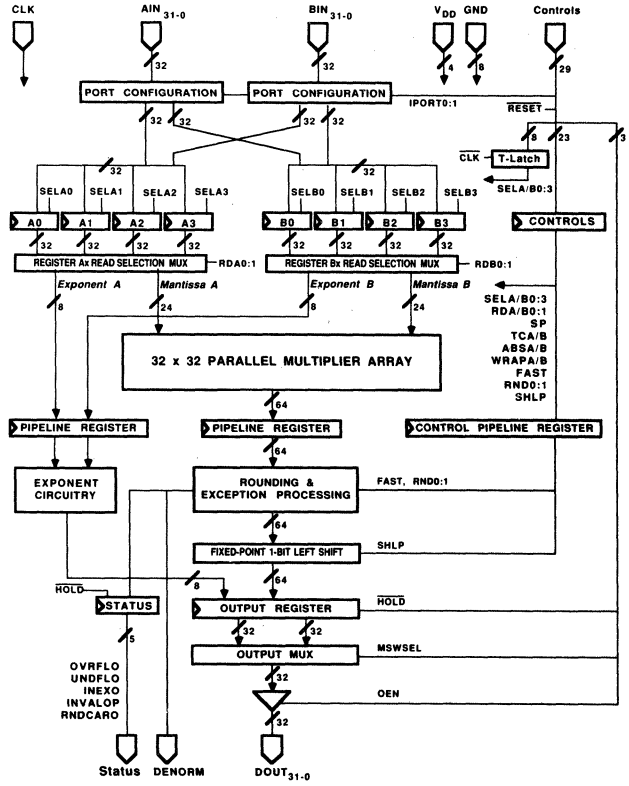


Figure 3. ADSP-3201 Functional Block Diagram

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

PIN NAME	DESCRIPTION	TYPE
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)	

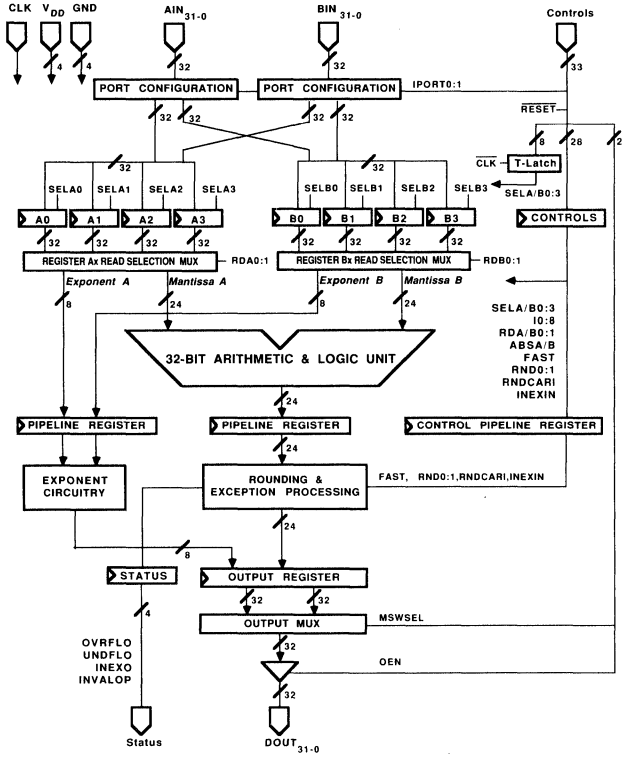


Figure 4. ADSP-3202 Functional Block Diagram

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,

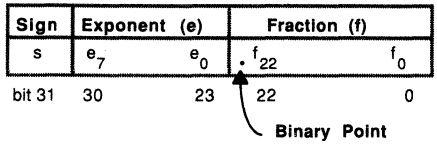


Figure 5. Single-Precision Floating-Point Format

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 $1 \leq e \leq 254$ 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 $-22 \leq e \leq 0$. 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 $-171 \leq e \leq -23$ 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 $-171 \leq e < -128$, 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.

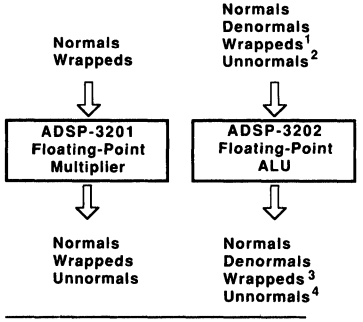
Mnemonic	Exponent	Fraction	Value	Name	IEEE Format?
NAN	255	non-zero	undefined	not-a-number	yes
INF	255	zero	$(-1)^s$(infinity)	infinity	yes
NORM	1 thru 254	any	$(-1)^s(1.f)2^{e-127}$	normal	yes
DNRM	0	non-zero	$(-1)^s(0.f)2^{-126}$	denormal	yes
ZERO	0	zero	$(-1)^s 0.0$	zero	yes
WRAP	−22 thru 0	any	$(-1)^s(1.f)2^{e-127}$	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:

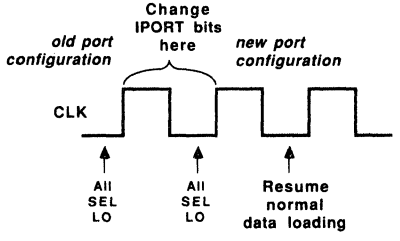


Figure 14. Timing Requirements for Changing the ADSP-3201/3202 Input Port Configurations

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:

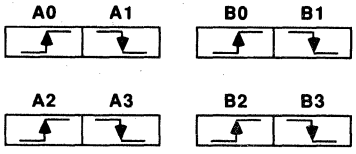


Figure 16. ADSP-3201/3202 Clock Edge for Data Loading – Two Port Configuration

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:

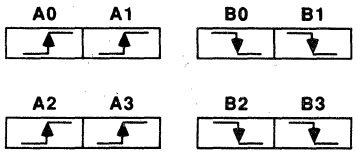


Figure 17. ADSP-3201/3202 Clock Edge for Data Loading – One Port Configuration

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.

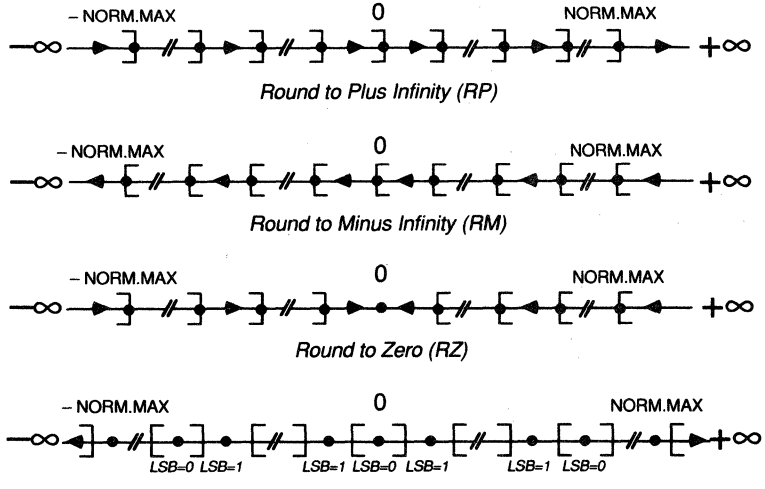


Figure 21. IEEE Rounding Modes

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.

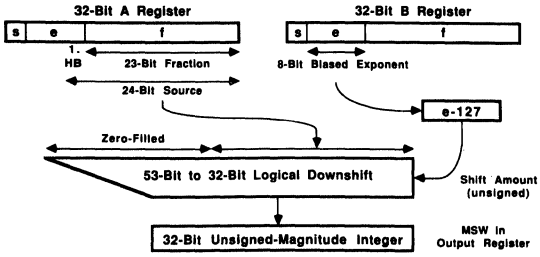


Figure 22. ADSP-3202 SITRN Instruction

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, $\overline{\text{HOLD}}$, that prevents the Output Register from being updated. $\overline{\text{HOLD}}$ 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, $\overline{\text{HOLD}}$ 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.

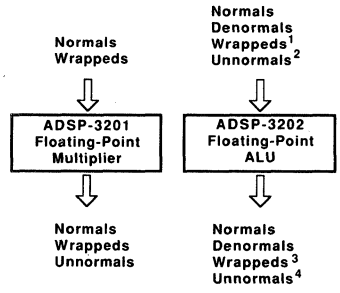
Note that for all operations (Figures T4 through T7) a new operation can begin the cycle before output results and status flags (other than DENORM) results from the previous operation are driven off chip. This feature leads to improved pipeline throughput.

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 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)

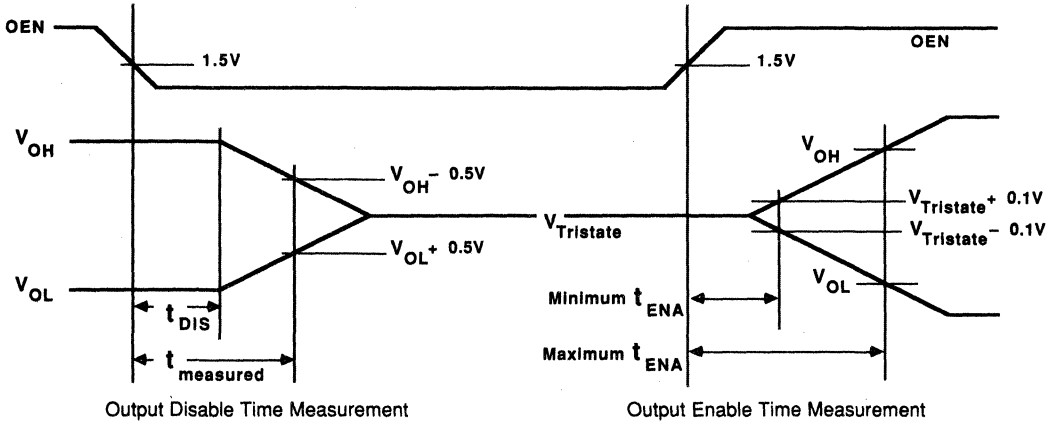
Mode \ B operand	B<ZERO		±ZERO		+DNRM		+WRAP		+NORM		+INF		±NAN	
	result	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*	*result*	*status*
IEEE	–NAN	INVALOP	±ZERO		+NAN	UNDFLO& INVALOP	NORM		NORM		+INF		±NAN	INVALOP
FAST	–NAN	INVALOP	±ZERO		+ZERO		NORM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

Table XII. ADSP-3202 Floating-Point Division Square Root √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



Output Disable Time Measurement

Output Enable Time Measurement

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

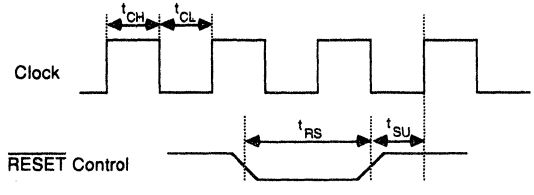


Figure T2. ADSP-3201/3202 Reset Timing

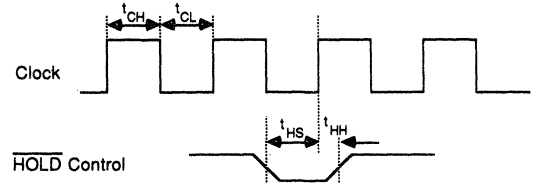
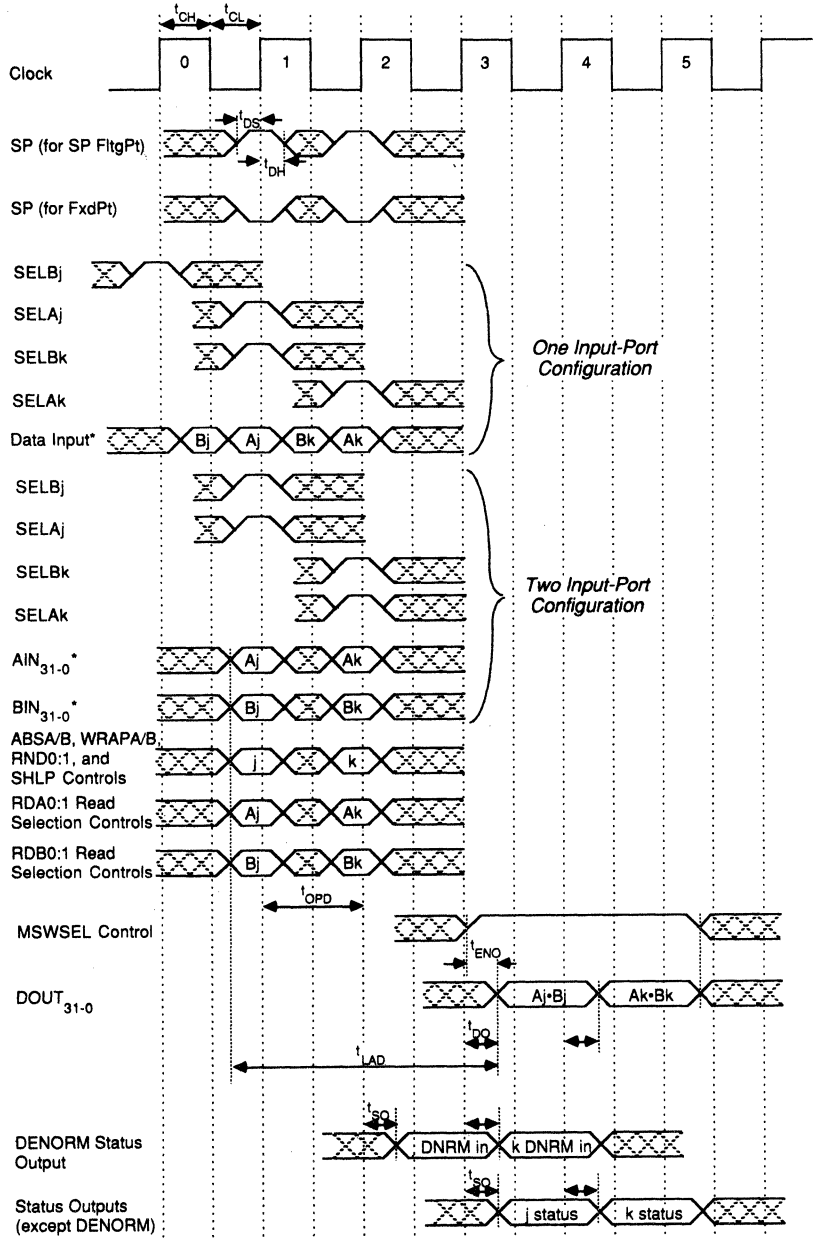
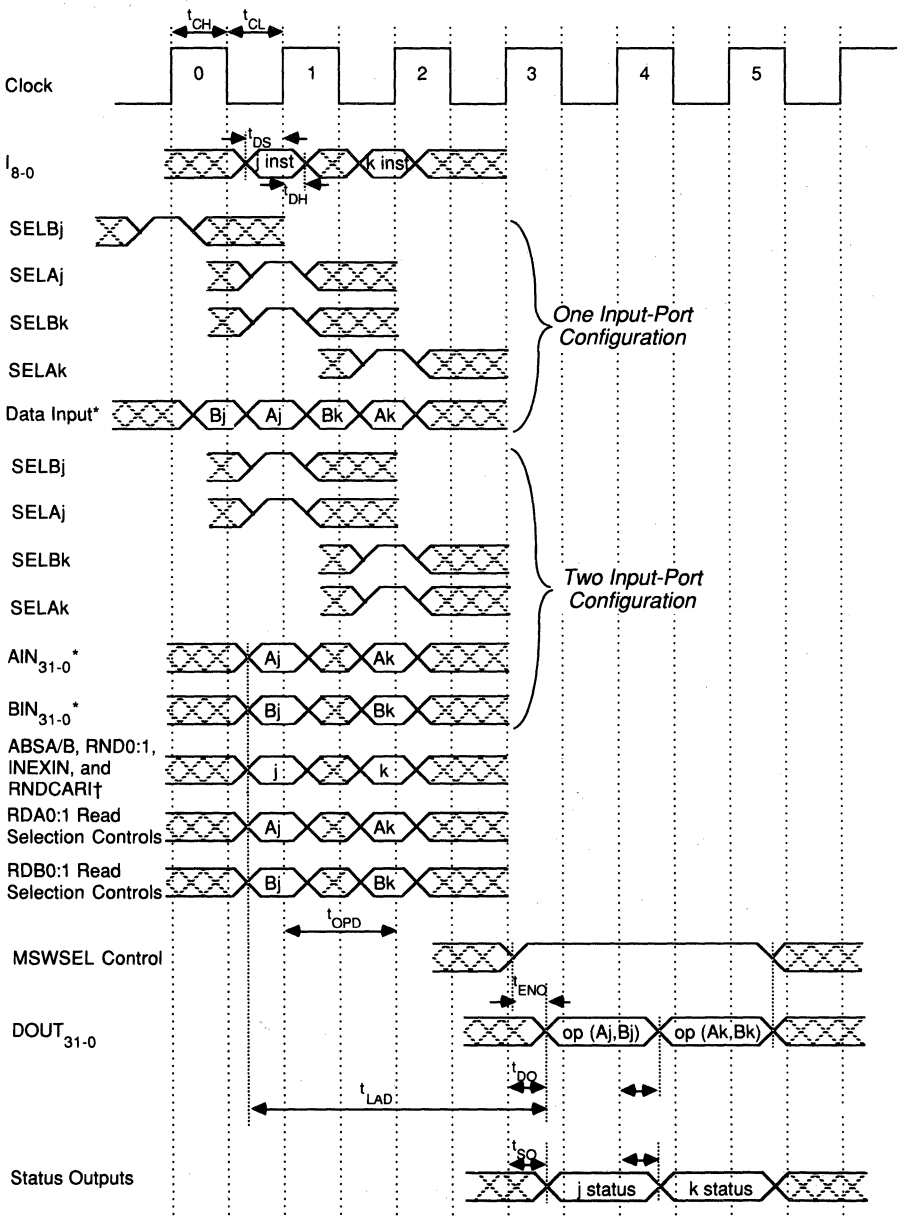


Figure T3. ADSP-3201 Multiplier Output Register Hold Timing



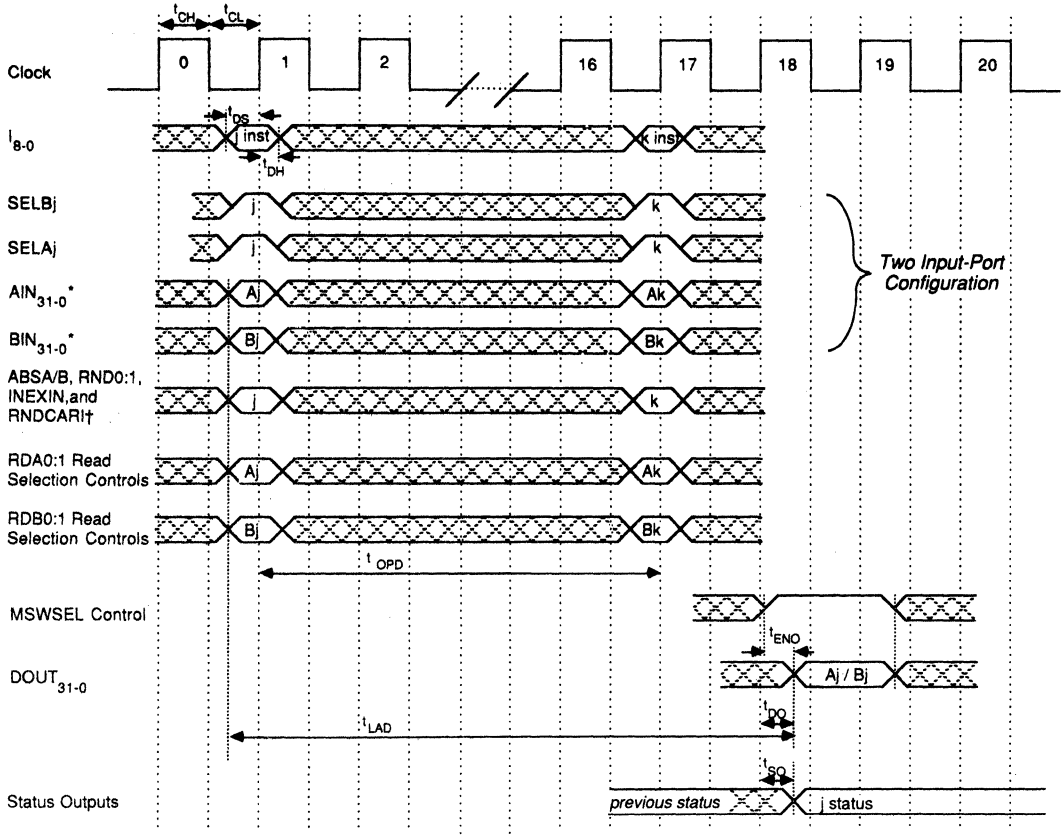
* See "Timing" section for additional sequencing options.

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



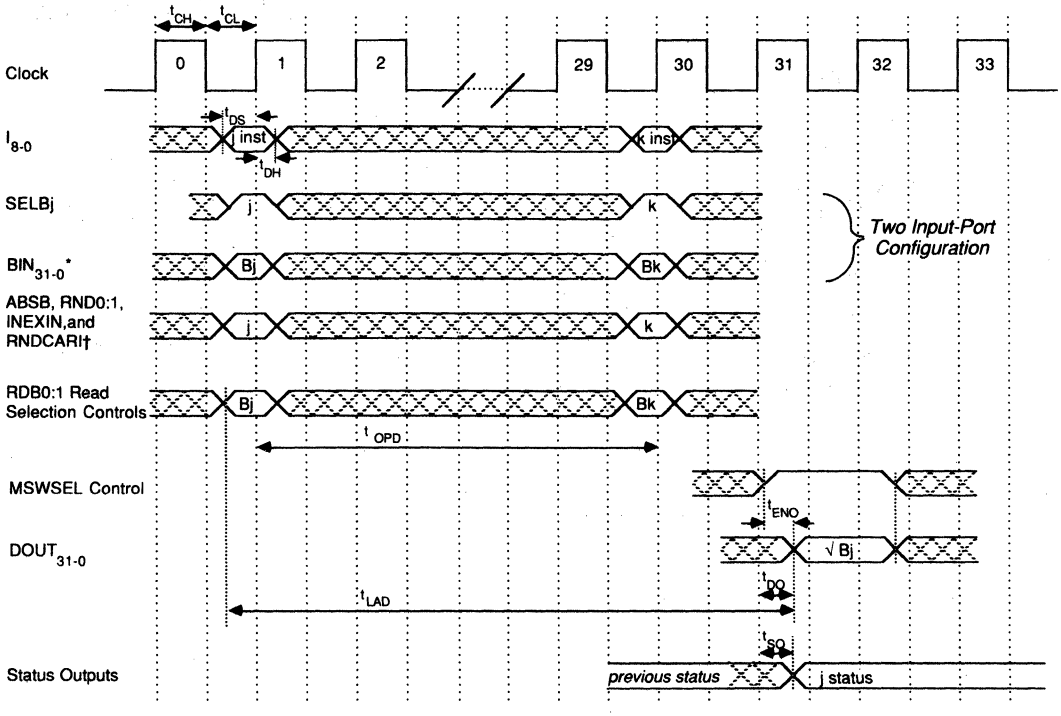
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

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

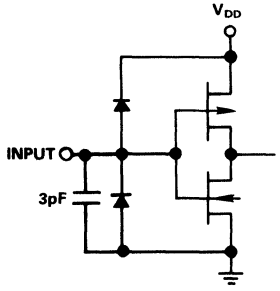


Figure 26. Equivalent Input Circuits

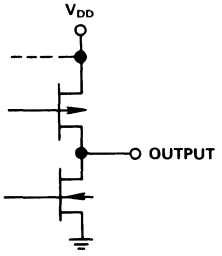


Figure 27. Equivalent Output Circuits

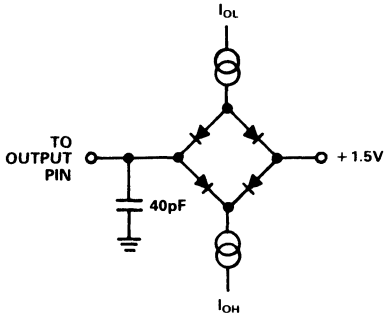


Figure 28. Normal Load for ac Measurements

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	Max	K Grade 0 to +70°C Min	Max	S Grade[2] −55°C to +125°C Min	Max	T Grade[2] −55°C to +125°C Min	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

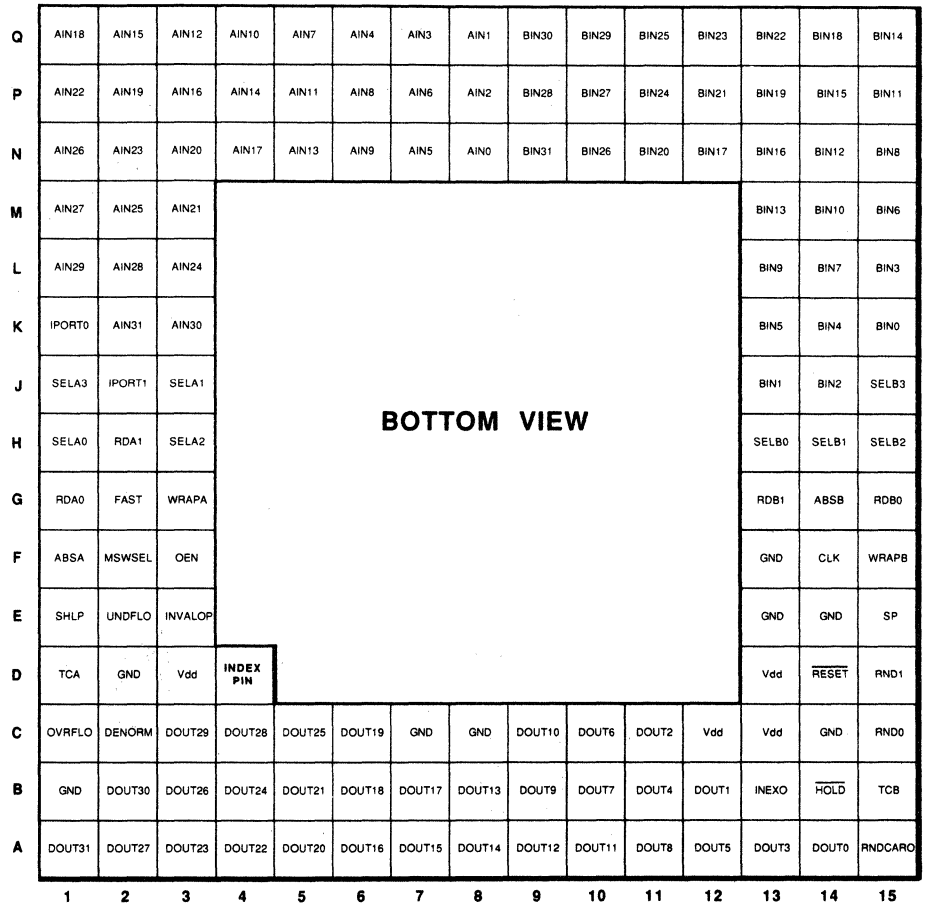
The ADSP-3201/3202 feature proprietary input protection to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-3201/3202 have been classified as Class 1 devices.

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

64-Bit IEEE Floating-Point Chipsets

ADSP-3210/3211/3220/3221

FEATURES

Complete Chipsets Implementing Floating-Point Arithmetic: Two Multiplier Options and Two ALU Options
Fully Compatible with IEEE Standard 754
Arithmetic Operations on Four Data Formats:
- **32-Bit Single-Precision Floating-Point**
- **64-Bit Double-Precision Floating-Point**
- **32-Bit Twos-Complement Fixed-Point**
- **32-Bit Unsigned Fixed-Point**

Only One Internal Pipeline Stage
High-Speed Pipelined Throughput
- **Single-Precision and Fixed-Point Multiplication Rates to 20 MFLOPS**
- **Double-Precision Multiplication Rates to 5 MFLOPS**
- **Single-, Double-, and Fixed-Point ALU Rates to 10 MFLOPS**

Low Latency for Scalar Operations
- **140ns for 32-Bit Multiplier Operations**
- **315ns for 64-Bit Multiplier Operations**
- **240ns for 32-Bit ALU Operations**
- **290ns for 64-Bit ALU Operations**

IEEE Divide and Square Root (ADSP-3221 ALU)
Flexible I/O Structures:
- **ADSP-3211/3220/3221: Either One or Two Input-Port Configuration Modes**
- **ADSP-3210: One Input Port**

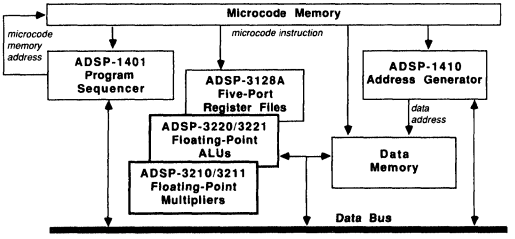
750mW Maximum Power Dissipation per Chip with 1.5μm CMOS Technology
100-Lead Pin Grid Array (ADSP-3210 Multiplier)
144-Lead Pin Grid Array (ADSP-3211/3220/3221)
Available Specified to MIL-STD-883, Class B

APPLICATIONS

High-Performance Digital Signal Processing
Engineering Workstations
Floating-Point Accelerators
Array Processors
Mini-Supercomputers
RISC Processors

GENERAL DESCRIPTION

The ADSP-3210/3211 Floating-Point Multipliers and the ADSP-3220/3221 Floating-Point ALUs are high-speed, low-power arithmetic processors conforming to IEEE Standard 754. A chipset consisting of either Multiplier used with either ALU contains the basic computational elements for implementing a high-speed numeric processor. Operations are supported on four data formats: 32-bit IEEE single-precision floating-point, 64-bit IEEE double-precision floating-point, 32-bit twos-complement fixed-point, and 32-bit unsigned-magnitude fixed-point.



Word-Slice® Microcoded System with ADSP-3210/3211/3220/3221

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-3210/3211/3220/3221 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).

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-3210/3211/3220/3221 are oriented to system-level implementations of function calculations. Specific instructions are included to facilitate such operations as floating-point division and square root, table lookup, quadrant normalization for trig functions, extended-precision integer operations, logical operations, and conversions between all data formats.

The ADSP-3210 Floating-Point Multiplier is a one input- and one output-port device with four input registers. The ADSP-3211

Word-Slice is a registered trademark of Analog Devices, Inc.

Floating-Point Multiplier adds a second input port and doubles the number of input registers to eight. It executes all ADSP-3210 operations. The ADSP-3210 supports 32-bit twos-complement fixed-point multiplications. The ADSP-3211 adds support for unsigned-magnitude and mixed-mode integer multiplications. Finally, the ADSP-3211 adds a $\overline{\text{HOLD}}$ control that prevents the updating of the output data and status registers.

The ADSP-3220 and ADSP-3221 Floating-Point ALUs differ only in that the ADSP-3221's instruction set is extended to include exact IEEE floating-point division and square root operations. The ADSP-3221 is pin-compatible with the ADSP-3220. Both ALUs are three-port, 144-lead devices with eight input registers.

The ADSP-3210/3211/3220/3221 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-3210/3211/3220/3221 can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-3210/3211/3220/3221 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-3210 Multiplier is packaged in a ceramic 100-lead pin grid array. The ADSP-3211, -3220, and -3221 are packaged in a ceramic 144-lead pin grid array.

TABLE OF CONTENTS **PAGE**

FUNCTIONAL DESCRIPTION OVERVIEW

The ADSP-3210/3211/3220/3221 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. (Note that the ADSP-3210, however, has a single input port.) 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, square root, and double-precision multiplication.) 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 double-precision result off chip through the 32-bit output port in one output cycle.

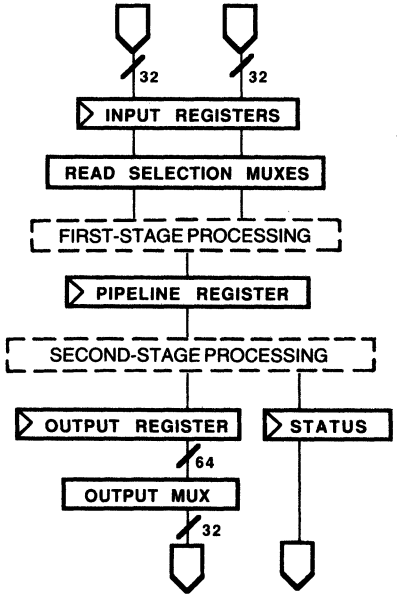


Figure 1. ADSP-3210/3211/3220/3221 Generic Architecture

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-3210/3211/3220/3221 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 clocks 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 clocks 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-3210/3211/3220/3221 fully supports the gradual underflow provisions of IEEE Standard 754 for floating-point arithmetic. The Floating-Point ALUs can operate directly on both normals and denormals, except in division and square root. The Floating-Point Multipliers operate 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 Floating-Point Multiplier. Information about rounding and inexact results generated by the Multipliers is needed by the ALUs to produce results in conformance to Standard 754. All ADSP-3210/3211/3220/3221 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 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-3210/3211/3220/3221

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 clocks rising edge (REG), other controls are latched in clock HI and transparent in clock LO (LAT), and others are asynchronous (ASYN).

ADSP-3210 Floating-Point Multiplier Pin List

PIN NAME	DESCRIPTION	TYPE
Data Pins		
DIN_{31-0}	32-Bit Data Input	
$DOUT_{31-0}$	32-Bit Data Output	
Control Pins		
$\overline{\text{RESET}}$	Reset	ASYN
SELA0	Load Selection for A0	LAT
SELA1	Load Selection for A1	LAT
SELB0	Load Selection for B0	LAT
SELB1	Load Selection for B1	LAT
RDA0	Register Ax Read Selection Control 0	REG
RDB0	Register Bx Read Selection Control 0	REG
WRAPA	Wrapped Contents in Register Ax	REG
WRAPB	Wrapped Contents in Register Bx	REG
ABSA	Read Absolute Value of Ax	REG
ABSB	Read Absolute Value of Bx	REG
SP	Single-Precision 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 (Three Lines)	
GND	Ground Supply (Three Lines)	

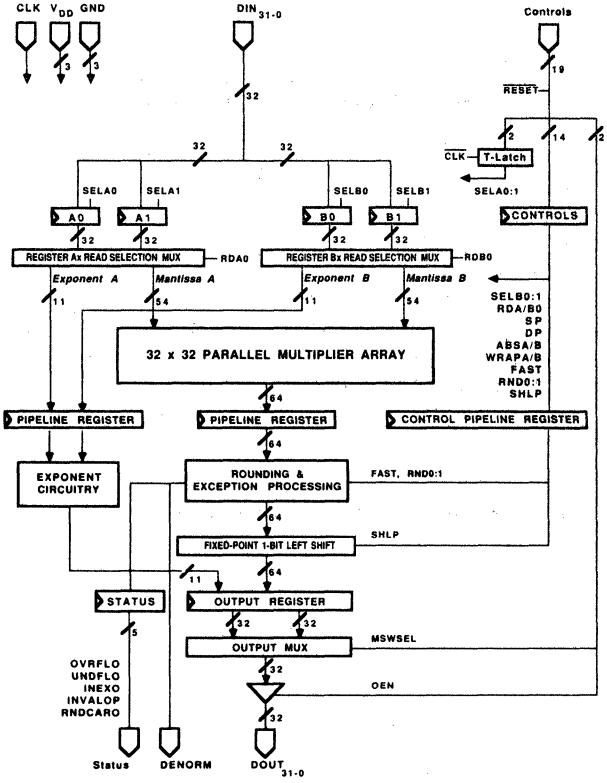


Figure 3. ADSP-3210 Functional Block Diagram

ADSP-3211 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
$\overline{HOLD}$	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 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 (Seven Lines)	

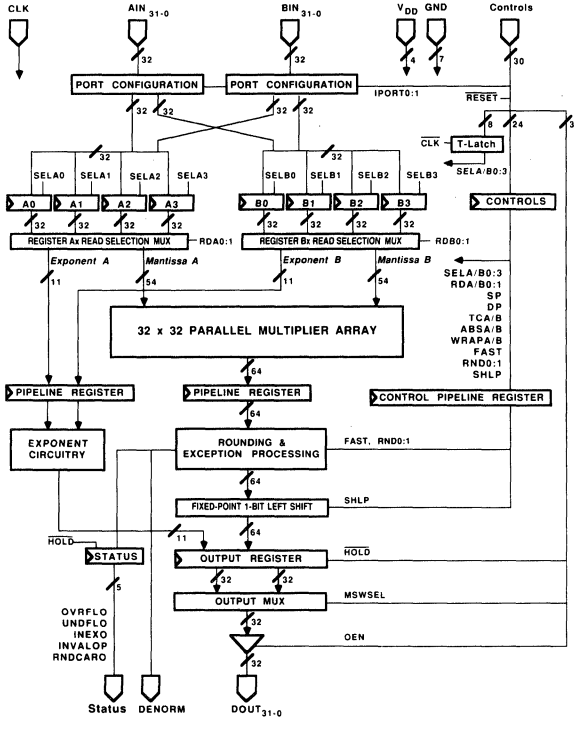


Figure 4. ADSP-3211 Functional Block Diagram

ADSP-3220 and -3221 Floating-Point ALUs 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
ABSA	Read Absolute Value of Ax	REG
ABSB	Read Absolute Value of Bx	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
RNDCARI	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)	

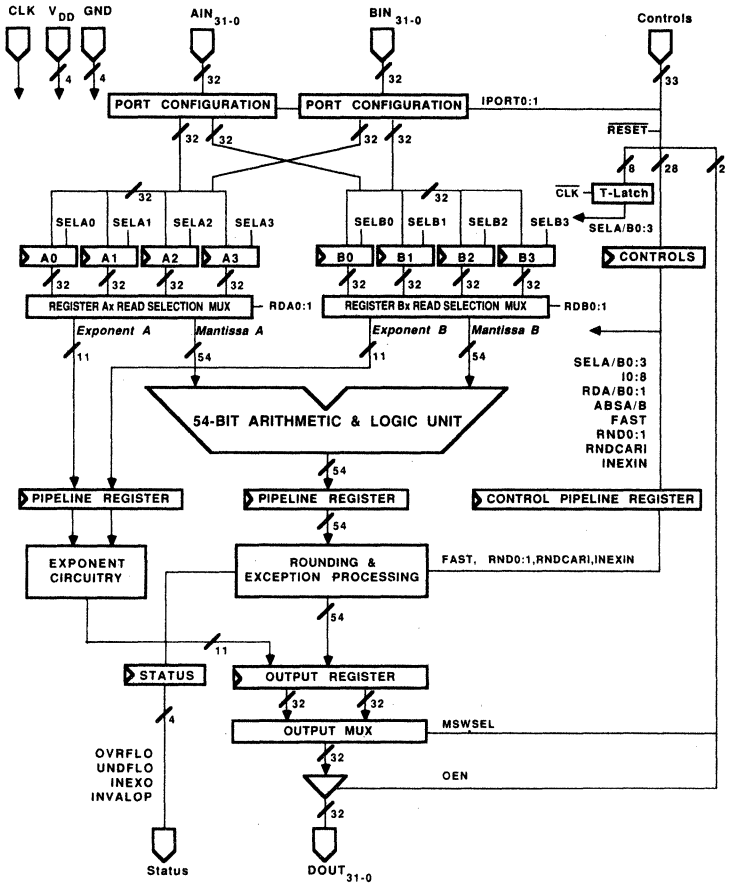


Figure 5. ADSP-3220/3221 Functional Block Diagram

METHOD OF OPERATION

DATA FORMATS

The ADSP-3210/3211/3220/3221 chipset supports both single- and double-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-3211 Multiplier and both ALUs. The ADSP-3210 Multipliers can perform fixed-point multiplication only on twos-complement numbers. All four chips operate 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

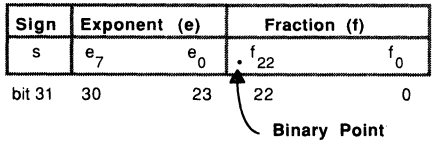


Figure 6. Single-Precision Floating-Point Format

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 $1 \leq e \leq 254$ 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 non-zero 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 non-zero 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 denormals 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-3210/3211/3220/3221 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-3210/3211 Multipliers and the ADSP-3221 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 an ADSP-3220/3221 ALU to wrapped numbers to be readable by a 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.")

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

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-3210/3211 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-3210/3211/3220/3221 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.

Double-Precision Floating-Point Data Format

IEEE Standard 754 specifies a 64-bit double-precision floating point format:

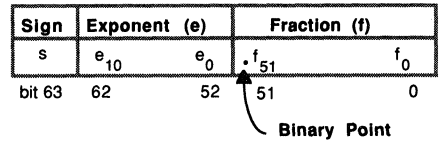


Figure 7. Double-Precision Floating-Point Format

The key differences with the single-precision format are that the exponent *e* is now 11 bits in length and the fraction *f* is now 52 bits in length, yielding a 53-bit significand for double-precision normals. Double-precision, like single-precision, has an implicit hidden bit, in this case the hidden bit precedes f_{51}. The binary point comes between the hidden bit and f_{51}. The exponent bias for double-precision floating-point normals is +1023 (2046÷2).

In other respects, IEEE double-precision floating-point is exactly analogous to single-precision, with the same data types whose values can be summarized in Table III.

The unbiased value of a denormal's exponent is −1022 for double-precision denormals, i.e. one minus the bias. Because of the extended width of the double-precision fraction, the exponent of double-precision wrapped numbers can range from $-51 \le e \le 0$. The exponent of unnormals can range from $-1125 \le e \le -52$. Again, the smallest unnormal is the result of multiplying the smallest wrapped number by itself.

Note that $e = -1024$ is the smallest double-precision exponent that is directly representable in the 11-bit IEEE twos-complement exponent field. The underflow flag should be thought of as a most-significant twelfth bit, the sign bit, as explained above for single-precision unnormals.

The ranges for the various double-precision data types are summarized in Table IV.

Mnemonic	Exponent	Fraction	Value	Name	IEEE Format?
NAN	2047	non-zero	undefined	not-a-number	yes
INF	2047	zero	$(-1)^s$(infinity)	infinity	yes
NORM	1 thru 2046	any	$(-1)^s(1.f)2^{e-1023}$	normal	yes
DNRM	0	non-zero	$(-1)^s(0.f)2^{-1022}$	denormal	yes
ZERO	0	zero	$(-1)^s 0.0$	zero	yes
WRAP	−51 thru 0	any	$(-1)^s(1.f)2^{e-1023}$	wrapped	no
UNRM	−1125 thru −52	any	$(-1)^s(1.f)2^{e-1023}$	unnormal	no

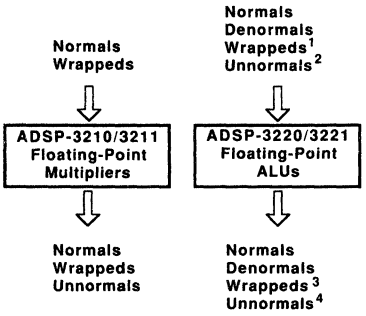
Table III. Double-Precision Floating-Point Data Types and Interpretations

Data name (positive)	Exponent	Exp. data type	Exponent bias	Hidden bit	Fraction (binary)	Unbiased absolute value
NORM.MAX	2046	unsigned	+1023	1	111.....11	$2^{+1023} \cdot (2-2^{-52})$
NORM.MIN	1	unsigned	+1023	1	000.....00	2^{-1022}
DNRM.MAX	0	unsigned	+1022	0	111.....11	$2^{-1022} \cdot (1-2^{-52})$
DNRM.MIN	0	unsigned	+1022	0	000.....01	$2^{-1022} \cdot 2^{-52}$
WRAP.MAX	0	2scmplmt	+1023	1	111.....11	$2^{-1023} \cdot (2-2^{-52})$
WRAP.MIN	−51	2scmplmt	+1023	1	000.....00	2^{-1074}
UNRM.MAX	−52	2scmplmt	+1023	1	111.....11	$2^{-1075} \cdot (2-2^{-52})$
UNRM.MIN	−1125	2scmplmt	+1023	1	000.....00	2^{-2148}

Table IV. Double-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 8. Data Types Directly Supported by the ADSP-3210/3211/3220/3221

Not every member of the ADSP-3210/3211/3220/3221 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-3211/3220/3221 chipset supports two 32-bit fixed-point formats: twos-complement and unsigned-magnitude. The ADSP-3210 Multiplier supports twos-complement only. With the ALUs, the output data format is identical with the input data format, i.e., 32-bits wide. In contrast, the Multipliers produce 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 9. 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-3210/3211 Multipliers produce a 64-bit product at their Output Registers. The ADSP-3210/3211 will produce results in the format of Figure 10 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 10. 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:

	Sign							
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 11. 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 9), 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-3211 Multiplier and inputs and outputs for both ALUs 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 12. 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-3211 Multiplier discriminates twos-complement from unsigned-magnitude inputs with TCA and TCB controls (see "Controls"). When TCA and TCB are both LO, the ADSP-3211 produces a 64-bit unsigned-magnitude product at its Output Register. The ADSP-3211 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 13. 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 14. 64-Bit Unsigned-Magnitude Fixed-Point Data Format at Multiplier Output Register with SHLP HI

The ADSP-3211 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 10.

CONTROLS

The controls for the ADSP-3210/3211/3220/3221 (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 clocks 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 an 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-3210/3211/3220/3221 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 Multipliers will treat denormal inputs as ZERO and produce a ZERO result. The ALUs will treat denormal inputs exactly as they do in IEEE mode but still flush post-rounded results less than NORM.MIN to ZERO.

Systems implementing gradual underflow with the ADSP-3210/3211/3220/3221 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-3210/3211/3220/3221. $\overline{\text{RESET}}$ 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)
The three-port members of this chipset (ADSP-3211/3220/3221) offer several options on their 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 16 below must be met.

The first and last configurations in Figure 15 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 Multipliers in the one-port configuration for all operations. The ADSP-3210 Multiplier has only one physical input port, so is always in a "one-port" configuration. The ALUs will, in contrast, become input-bandwidth-constrained at the input ports for double-precision operations in a one-port configuration. They are capable of operating on a pair of 64-bit

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 15. ADSP-3211/3220/3221 Input Port Configurations

operands at the clock rate, but a single input port could not accept operands at that rate.

The port configuration of the ADSP-3211/3220/3221 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 while IPORT bits change; see "Input Register Loading") during both the cycle in which IPORT bits are changed and the cycle following:

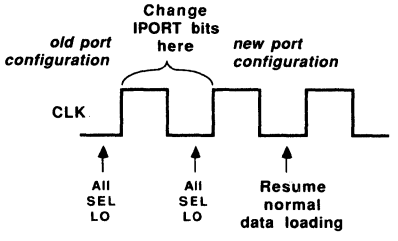


Figure 16. Timing Requirements for Changing the ADSP-3211/3220/3221 Input Port Configurations

Thus, data loading will be interrupted for two cycles whenever changing the ADSP-3211/3220/3221's port configuration. All other processing is unaffected.

INPUT REGISTER LOADING AND OPERAND STORAGE – SELA/B CONTROLS (LAT)
The chipsets' 32-bit input registers are selected for data loading with the latched Load Selection Controls, SELA/B0:3 (on the ADSP-3210, SELA/B0:1). 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 17. ADSP 3210/3211/3220/3221 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 (which minimizes double-precision latency). 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:

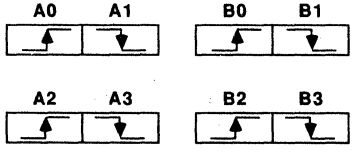


Figure 18. ADSP-3211/3220/3221 Clock Edge for Data Loading – Two-Port Configuration

Eight-register devices (ADSP-3211/3220/3221) in the one-port configuration load data to A registers on the rising edge and B registers on the falling edge:

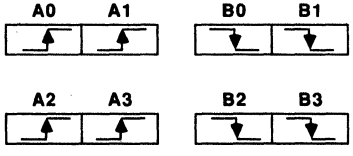


Figure 19. ADSP-3211/3220/3221 Clock Edge for Data Loading – One-Port Configuration

The ADSP-3210 Multiplier loads data like the two-input-port devices in a one-input port configuration. That is, the ADSP-3210 loads data to A registers on the rising edge and B registers on the falling edge:

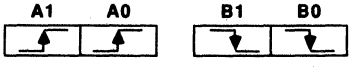


Figure 20. ADSP-3210 Clock Edge for Data Loading

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 chip's 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-3221 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, whether single-precision or double-precision. (See "Timing" and the timing diagrams below for a precision definition of "operation time".)

With 64-bit double-precision data, there are constraints on which registers hold which 32-bit halves of operands. 64-bit data must be loaded in adjacent pairs of 32-bit registers as shown in Figures 21 and 22. The 32-bit Most Significant Word (MSW) will be in one register and the 32-bit Least Significant Word (LSW) in its neighbor. The four-register ADSP-3210 has different double-precision operand storage requirements from the other members of this chipset. Double-precision operand storage for the ADSP-3211/3220/3221 is:

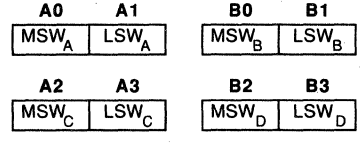


Figure 21. ADSP-3211/3220/3221 Operand Storage for Double-Precision Operations

For the four-register ADSP-3210, operands for double-precision operations should be stored as shown in Figure 22. Note that the MSWs are in A1 and B1, in contrast with 64-bit data storage with the other members of this chipset.

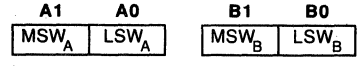


Figure 22. ADSP-3210 Operand Storage for Double-Precision Operations

Restrictions on Register Stability

With 64-bit data – as with 32-bit data – registers should not be loaded that are currently in use by the processing elements (i.e., selected by the RDA/B controls). Half the 32-bit registers in any pair of 64-bit operands will loaded on the falling edge (regardless of port configuration) with all members of this chipset.

To operate the ALUs at full throughput in single-cycle double-precision operations, 64-bit register sets should be alternated every cycle. For example, A0 & A1 and B2 & B3 could be loaded with new operands while A2 & A3 and B0 & B1 were feeding the computational circuits (and were not changing). In

this way, data loading will not disturb the contents of registers in use.

The ADSP-3221 ALU includes two double-precision multicycle operations in its instruction set: IEEE division and square root. For double-precision floating-point division, the 64-bit dividend can be stored in either pair of A registers consistent with Figure 21. The divisor can be stored in either pair of B registers, also consistent with Figure 21. Double-precision operands for IEEE square root can be stored in either pair of B registers consistent with Figure 21. Registers containing operands in use must remain unchanged until the end of the operation time.

The ADSP-3210/3211 Multipliers perform double-precision multiplications at a four-cycle throughput rate. This process requires computing four cross-products, and the only requirement on operand registers is that they remain stable, i.e., unchanged, for the cycles in which they are used. For this reason, the ADSP-3210 can maintain full four-cycle double-precision multiplication throughput even though it has only two pairs of 32-bit registers. The sequence of operations for double-precision multiplications and the requirements on register stability are as follows:

	ADSP-3210	A1	A0	B1	B0
	ADSP-3211	A0,A2	A1,A3	B0,B2	B1,B3
Cycle	Operation	MSW	LSW	MSW	LSW
1	$A_{LSW} \cdot B_{LSW}$		stable		stable
2	$A_{MSW} \cdot B_{LSW}$	stable	stable		stable
3	$A_{LSW} \cdot B_{MSW}$	stable	stable	stable	
4	$A_{MSW} \cdot B_{MSW}$	stable		stable	

Figure 23. ADSP-3210/3211 Double-Precision Multiplication Input Register Requirements

To achieve maximum throughput with the ADSP-3210 Multiplier, the two LSWs from the operands to multiplied should be loaded first (to B0 followed by A0). The actual double-precision multiplication can begin as soon as both are loaded to A0 and B0 (beginning of cycle 1 in Figure 23). At the midpoint and end of cycle 1, the MSWs can be loaded (though only the MSW in A1 is actually needed in cycle 2). At the end of cycle 2, the LSW in B0 can be overwritten with an LSW needed in the next multiplication. At the end of cycle 3, the LSW in A0 can be overwritten.

The ADSP-3211 Multiplier has additional registers and therefore fewer constraints on data loading and storage than the ADSP-3210 Multiplier. The only requirements that must be observed are those indicated in Figures 21 and 23.

DATA FORMAT SELECTION – SP & DP CONTROLS (REG)

The three data formats processed by the ADSP-3210/3211/3220/3221 chipset are *single-precision* floating-point, *double-precision* floating-point, and *fixed*. With the ADSP-3210/3211 Multipliers, the data format is indicated explicitly by the states of the DP and the SP registered controls:

SP	DP	Data Format Selection
0	0	fixed
0	1	double-precision
1	0	single-precision
1	1	illegal mode

Figure 24. ADSP-3210/3211 Multipliers Data Format Selection

The state of the SP and DP controls at the rising edge when data is read into the Multiplier Array determines whether the data is interpreted as single-precision floating-point, double-precision floating-point, or fixed-point. Double-precision multiplication is a multicycle operation; once initiated, the states of SP and DP don't matter until the next data is read to the processing circuitry.

For the ADSP-3220/3221 ALUs, 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 (on the ADSP-3210, RDA/B0, 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").

The data format selected affects the interpretation of the RDA/B controls. The four-register ADSP-3210 Multiplier needs only two Register Read Selection Controls, which are defined below separately.

For the ADSP-3211/3220/3221, register read selection is defined:

RDA1	RDA0	SP & Fixed: A register selected	DP: A registers selected
0	0	A2	illegal state
0	1	A3	A2, A3
1	0	A0	illegal state
1	1	A1	A0, A1

RDB1	RDB0	SP & Fixed: B register selected	DP: B registers selected
0	0	B2	illegal state
0	1	B3	B2, B3
1	0	B0	illegal state
1	1	B1	B0, B1

Figure 25. ADSP-3211/3220/3221 Input Register Read Selection

For the ADSP-3210, register read selection is defined:

RDA0	SP & Fixed: A register selected	DP: A registers selected
0	A1	illegal state
1	A0	A1, A0

RDB0	SP & Fixed: B register selected	DP: B registers selected
0	B1	illegal state
1	B0	B1, B0

Figure 26. ADSP-3210 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-3220/3221 ALUs, the ABSA/B controls are effective with most fixed-point and all single-precision and double-precision operations. If the ABSA/B controls are asserted in logical operations, the results will be undefined.

For the ADSP-3210/3211 Multipliers, the absolute value operation is available on single-precision and double-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-3221)
The ADSP-3210/3211 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 Multipliers 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; WRAPB, that the selected B register contains a wrapped number.

The ALUs in general operate directly on denormals and hence don't need a similar set of controls. However, for ADSP-3221 IEEE division and square root operations, the ALU cannot operate directly on denormals. Like the Multipliers, 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 a Multiplier. To indicated 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 CONTROL – TCA/B (REG)
The registered ADSP-3211'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-3210/3211/3220/3221 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-3210/3211/3220/3221 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 as shown in Figure 27.

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.

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 27. Round Controls

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.

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 10, 11, 13, and 14) as the destination format.

The four rounding modes are illustrated by number lines in Figure 28. 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.

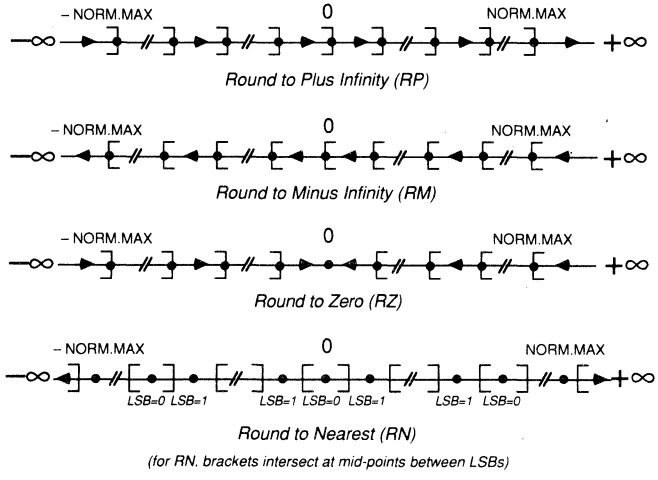


Figure 28. IEEE Rounding Modes

STATUS FLAGS

The ADSP-3210/3211/3220/3221 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.

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-3210/3211 Multipliers have 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. For double-precision multiplications, DENORM goes HI during the third cycle after a denormal was read into the multiplier array. See Figure T5. Both Multipliers produce 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-3210/3211 Multipliers 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 Multipliers to produce this correct IEEE result.

Note that the ALUs in general operate directly on denormals and therefore do not flag any exception. The ADSP-3221 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-3210/3211/3220/3221 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 non-zero dividend by a divisor of zero (Std 754 Section 7.2). The Division-by-Zero exception is indicated on the ADSP-3221 ALU by the simultaneous assertion of both OVRFLO and INVALOP. The ALU result is always a correctly signed INF.

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-3221 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-3210/3211 Multipliers are in fixed-point mode.

Underflow

Underflow is defined in four ways in Std 754 Section 7.4. The IEEE Standard allows the implementer to chose 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 non-zero, 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-3210/3211/3220/3221 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 the chipset is operating in IEEE or FAST mode, whether its result is generated by a Multiplier or an ALU, and whether the operation is division.

With the ADSP-3210/3211 Multipliers, UNDFLO is generated whenever the unbounded, post-rounded, non-zero 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-3220/3221 ALUs in the FAST mode, UNDFLO is also generated whenever the unbounded, post-rounded, non-zero 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-3220/3221 ALUs 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-3221 ALU. The data result will be in the wrapped format. Note that $\sqrt{(DNRM.MIN)} \geq NORM.MIN$. 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-3210/3211 Multipliers, whenever the unbounded, post-rounded, non-zero result is of lesser magnitude than NORM.MIN in the destination format
- With the ADSP-3220/3221 ALUs in the FAST mode, whenever the unbounded, post-rounded, non-zero result is of lesser magnitude than NORM.MIN in the destination format
- With the ADSP-3220/3221 ALUs in IEEE mode, whenever an inexact denormal is produced or whenever the unbounded, post-rounded, non-zero 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
- Conversions from DP floating-point to SP floating-point whenever the unbounded, post-rounded, non-zero result is less than SP DNRM.MIN or whenever an inexact denormal is produced.
- 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-3210/3211/3220/3221 chipset by INEXO.

For fixed-point operations, the ADSP-3210/3211 Multipliers will assert INEXO HI if and only if any of the least-significant 32 bits of prerounded 64-bit products are ones. They never assert INEXO for logical operations. The ADSP-3220/3221 ALUs never assert INEXO for fixed-point or logical operations.

In an ADSP-3221 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 ALUs, 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 X1 and X11.

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-3210/3211 generates a Round Carry Propagation Out flag, RNDCARO, that indicates whether or not a carry bit propagated into the destination format's fraction during the Multiplier's 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 format's 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 Multiplier's rounding and re-round to achieve the properly rounded denormal.

To do this, the ALU has to know if any carry bits in the Multipliers 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 Multipliers Inexact Result (INEXO) flag.

The ADSP-3220/3221 ALUs have 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-3221 ALU itself generates WNRMs in underflowed division operations. These WNRMs must be fed back to the ALU to be unwrapped to DNRMs. The ADSP-3221, unlike the Multipliers, 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-3221 ALU in division are rounded differently than they are on the Multipliers; 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-3221 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-3210/3211 Multipliers execute 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 IX and X. Note that double-precision floating-point multiplications are multicycle operations. Data must be available in the input registers as shown above in Figure 23.

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-3210/3211 Multipliers, 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-3220/3221 ALUs, in contrast to the Multipliers, are instruction driven with the operation specified by I_{8-0}. The ALU instructions fall into four categories: Fixed-Point, Logical, Single-Precision Floating-Point, and Double-Precision Floating-Point. Instructions are summarized in Tables V through VIII and described in this section below. The data format of results and status flags from the various ALU operations are shown in Tables XI and XII. Division is shown in Tables XIII and XIV; square root in Table XV. Conversions are illustrated in Tables XVI, XVII, and XVIII.

The ADSP-3220/3221 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 V. ADSP-3220/3221 Fixed-Point ALU Operations

The ADSP-3220/3221 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. Preserve Output Register contents with ADSP-3221 only.
CLR	100	000	000	Clear all status flags. Data register contents are unaffected.

Table VI. ADSP-3220/3221 ALU Logical Operations

The ADSP-3220/3221 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
DOUBLEA	011	101	101	Convert SP FltgPt A to DP FltgPt
DOUBLEB	011	101	110	Convert SP FltgPt B to DP 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.
ADSP-3221 ALU only:				
SDIV	011	110	111	SP FltgPt (A ÷ B)
SSQR	111	110	110	SP FltgPt $\sqrt{B}$

Table VII. ADSP-3220/3221 ALU Single-Precision Floating-Point Operations

The ADSP-3220/3221 Double-Precision Floating-Point Operations are:

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
DADD	110	000	011	DP FltgPt (A + B)
DSUBB	110	000	111	DP FltgPt (A – B)
DSUBA	110	001	011	DP FltgPt (B – A)
DCOMP	110	001	111	DP FltgPt comparison of A to B. Result is (A – B). Greater Than≡(OVRFLO HI & UNDFLO LO) Equal≡(OVRFLO LO & UNDFLO LO) Less Than≡(OVRFLO LO & UNDFLO HI) Unordered≡INVALOP HI
DADDAS	010	000	011	DP FltgPt | A + B|
DSUBBAS	010	000	111	DP FltgPt |A – B|
DSUBAAS	010	001	011	DP FltgPt |B – A|
DFIXA	010	011	101	Convert DP FltgPt A to twos-complement integer
DFIXB	010	011	110	Convert DP FltgPt B to twos-complement integer
DFLOATA	010	100	101	Convert twos-complement integer A (even A register sources only) to DP FltgPt
DFLOATB	010	100	110	Convert twos-complement integer B (even B register sources only) to DP FltgPt
SINGLEA	110	011	101	Convert DP FltgPt A to SP FltgPt
SINGLEB	110	011	110	Convert DP FltgPt B to SP FltgPt
DPASSA	010	110	001	Pass DP FltgPt A. NANs cause INVALOP.
DPASSB	010	110	010	Pass DP FltgPt B. NANs cause INVALOP.
DWRAPA	010	100	001	Wrap DP DNRM A to DP WNRM
DWRAPB	010	100	010	Wrap DP DNRM B to DP WNRM
DUNWRAPA	010	010	001	Unwrap DP WNRM A to DP DNRM
DUNWRAPB	010	010	010	Unwrap DP WNRM B to DP DNRM
DSIGN	010	111	101	Copy sign from DP FltgPt B to DP FltgPt A. Result is [sign B, exponent A, fraction A].
DXSUB	010	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 ≥ + 1024, INF results. If the unbiased exponent is ≤ – 1023, ZERO results.
DITRN	010	010	101	Downshift DP FltgPt A mantissa (with hidden bit) logically by the unbiased DP FltgPt B exponent to a 32-bit unsigned-magnitude integer. Use RZ only.
ADSP-3221 ALU only:				
DDIV	010	110	111	DP FltgPt (A + B)
DSQR	110	110	110	DP FltgPt $\sqrt{B}$

Table VIII. ADSP-3220/3221 ALU Double-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-3220/3221s 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-3220/3221. 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-3220/3221 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 will be preserved, but Output Register contents will not with the ADSP-3220. The ADSP-3221 preserves both flags and output register contents during NOP. 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 single-precision and double-precision floating-point operations are exactly analogous and both will be discussed here. The data types and flags resulting from additions, subtractions, comparisons, absolute sums, and absolute differences are shown in Tables XI and XII. 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 ALUs 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 and DFIXA/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 non-zero 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 Tables XVI and XVII for illustrations of fixing single- and double-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].

For conversion from twos-complement integer to double-precision floating-point (DFLOATA/B), the numerical result will always be an IEEE normal. No flags will be set. Only even-numbered registers (A0, A2, B0, or B2) can be sources for the DFLOAT operation.

Conversions: Floating to Floating

For conversion from single-precision to double-precision (DOUBLEA/B), all single-precision normals and denormals will convert without exceptions. A single-precision NAN will convert to a double-precision all-ones NAN; the INVALOP flag will be set. Single-precision INF converts to double-precision INF; no flags are set. Single-precision ZERO converts to double-precision ZERO; no flags are set.

Conversions from double-precision to single-precision floating-point (SINGLEA/B) can cause exceptions because overflow, underflow, and inexact status can result in mapping to the smaller destination format. See Table XVIII for illustrations. A double-precision NAN will convert to a single-precision all-ones NAN; the INVALOP flag will be set. DP INFs convert to SP INFs; no flags are set. Finite numbers greater in magnitude than single-precision NORM.MAX will result in SP INF or SP NORM.MAX, depending on the rounding mode. (See "Round Controls" above.) Non-zero, post-rounded operands whose magnitudes are between SP NORM.MAX and SP NORM.MIN inclusive will be SP NORMs. In IEEE mode, operands between SP DNRM.MAX and SP DNRM.MIN inclusive will be SP DNRMs; in FAST mode, ZERO will result with UNDFLO and INEXO set.

For both normals and denormals, INEXO will be asserted if the conversion from double-precision to single-precision floating-point is inexact. If the conversion to denormals is inexact, both INEXO and UNDFLO will be set, in accordance with the IEEE definition in terms of loss of accuracy when representing a denormal. (See "Underflow" in "Status Flags" above.) Post-rounded, non-zero numbers less than SP DNRM.MIN will convert to ZERO; UNDFLO and INEXO will be set. DP ZERO converts to SP ZERO without exception.

Pass

Pass instructions (SPASSA/B and DPASSA/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 and DWRAPA/B) convert a denormal to a wrapped number readable by a Multiplier or the ADSP-3221 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 (SUNWRAPA/B and DUNWRAPA/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.

The ADSP-3220 and ADSP-3221 differ slightly in how inexactness is defined for unwrapping. With the ADSP-3220, inexactness is determined solely by whether or not there was a loss of accuracy when unwrapping the operand supplied to the ALU. The ADSP-3221 goes beyond the ADSP-3220 in also considering 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 both ALUs. 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-3220 uses INEXIN only for this purpose. The ADSP-3221 also uses this information to determine if the operation creating the wrapped number was inexact. When the ADSP-3221 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 and DSIGN operations copy 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 and DSIGN simplify 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 Substraction

Exponent subtraction (SXSUB and DXSUB) 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/DXSUB operations; 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 and DITRN operations. (See Figures 29 and 30.) 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.

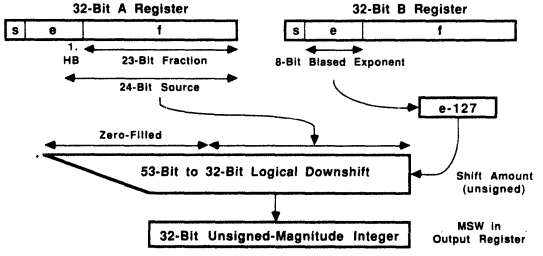


Figure 29. ADSP-3220/3221 SITRN Instruction

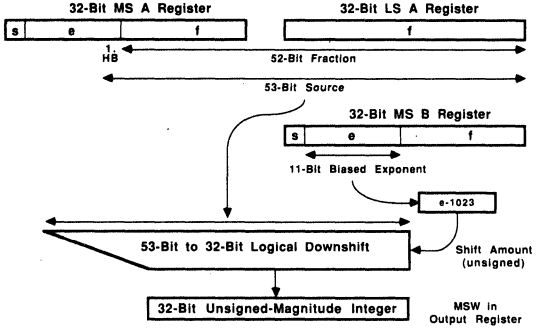


Figure 30. ADSP-3220/3221 DITRN Instruction

Logical downshift is only defined for NORMs. Results from operands than are not normals are undefined. A NAN A-operand input to SITRN/DITRN will cause INVALOP and produce all-ones NANs of the same sign. Round-toward-Zero (RZ) must be specified for SITRN and DITRN. 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.

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-3221 ALU supports multicycle division (SDIV and DDIV) and square root (SSQR and DSQR) operations. Tables XIII and XIV illustrate the resultant data types and status conditions for division. Table XV 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.

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-3221 division operations, the sign of the NAN will be the exclusive OR of the signs of the dividend and the divisor.

4

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

All members of the ADSP-3210/3211/3220/3221 chipset have a 64-bit Output Register. The Output Registers are clocked every cycle, except for multicycle operations (double-precision multiplication, division, and square root) when $\overline{\text{HOLD}}$ is LO on the ADSP-3211 and when the ADSP-3221 is executing NOP. Output Registers are clocked at the conclusion of multicycle operations and not before.

Results appear in the Multipliers Output Registers as follows:

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

Figure 31. ADSP-3210/3211 Multiplier Output Registers

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 Multipliers have 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 ALUs Output Registers as follows:

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

Figure 32. ADSP-3220/3221 ALU Output Registers

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-3211 Multiplier has an asynchronous, active LO control, $\overline{HOLD}$, that prevents the Output Register from being updated. $\overline{HOLD}$ must be setup 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 T12. 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 Figure 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 Figure 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 Figures T4 through T12 must be setup t_{DS} before the rising edge and held t_{DH}. Both input-port configurations are shown in most 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.

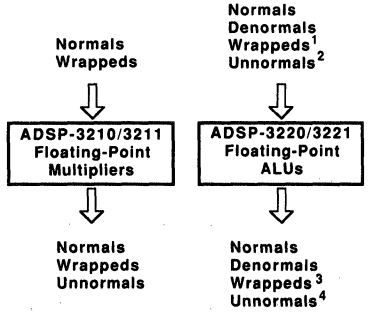
The asynchronous MSWSEL control's delay is t_{ENO}. The maximum specification for t_{ENO} is the delay which guarantees valid data. The minimum specification for t_{ENO} is the earliest time after the MSWSEL control is changed that data can change.

Status flags have a maximum output delay of t_{SO} referenced from the clock rising edge. All status flags except the Multipliers 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 Figures T4 and T5. DENORM should therefore not be used by itself to externally trigger a denormal input exception processing routine.

Note that for all operations (Figures T5 through T12) 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 33.



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

Figure 33. Data Types Directly Supported by the ADSP-3210/3220/3221

Denormals are detected by the Multipliers when read into their processing circuitry. The ADSP-3210/3211 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-3221 ALU will also detect denormals when read into internal circuitry for division or square root operations. The UNDFLO and INVALOP flags will both be asserted on the ADSP-3221 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-3221.)

The ALU wraps denormals with its SWRAP or DWRAP instructions. Note from Tables II and IV 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-3221 ALU 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 ALU's 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 IX, X, XIII and IV.) Square root on IEEE numbers only produces normals. (See Tables XI and XII.) 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 and DUNWRAP instructions) 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-3210/3211/3220/3221 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 IX. ADSP-3210/3211 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 X. ADSP-3210/3211 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 XI. ADSP-3220/3221 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	UNDFLO	NORM		INF		NAN	INVALOP
DNRM	ZERO	UNDFLO	NORM ZERO		INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	INF		NAN	INVALOP
NORM	NORM		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)+(∓ 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. Exact result.
5. In FAST mode, WRAP inputs are illegal.

Table XII. ADSP-3220/3221 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 XIII. ADSP-3221 Floating-Point Division (A÷B) (IEEE Mode)

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."
2. In FAST mode, WRAP inputs are illegal.

Table XIV. ADSP-3221 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 XV. ADSP-3221 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 XVI. Conversion of 32-Bit Single-Precision Floating-Point to 32-Bit Twos-Complement Integer

Sign	HB	f51 . . . f22	f21	f20	f19 . . . f1	f0		Unbiased Expnt	Source Name
0	1	X . . . X	X	X	X . . . X	X	2**	1024	+NAN
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	+INF
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	31	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	30	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	30	
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	30	
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	30	
0	1	1 . . . 1	0	1	1 . . . 1	1	2**	30	
0	1	1 . . . 1	0	1	1 . . . 1	1	2**	30	
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	30	
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	30	
0	1	1 . . . 1	0	0	0 . . . 0	0	2**	30	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	0	one
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	one − 1LSB
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	one − 1LSB
0	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	1/2 + 1LSB
0	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	1/2 + 1LSB
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	1/2
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	1/2
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DENORM.MIN
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DENORM.MIN
0	0	0 . . . 0	0	0	0 . . . 0	0		0	+ZERO
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DENORM.MIN
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DENORM.MIN
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	−1/2
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	−1/2
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	−1/2 − 1LSB
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	−1/2 − 1LSB
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	−one + 1LSB
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	−one + 1LSB
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	0	−one
1	1	1 . . . 1	0	0	0 . . . 0	0	2**	30	
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	30	
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	30	
1	1	1 . . . 1	0	1	1 . . . 1	1	2**	30	
1	1	1 . . . 1	0	1	1 . . . 1	1	2**	30	
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	30	
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	30	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	30	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	30	
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	31	
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	31	
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	31	
1	1	0 . . . 0	0	1	0 . . . 0	0	2**	31	
1	1	0 . . . 0	0	1	0 . . . 0	0	2**	31	
1	1	0 . . . 0	0	1	1 . . . 1	1	2**	31	
1	1	0 . . . 0	0	1	1 . . . 1	1	2**	31	
1	1	0 . . . 0	1	0	0 . . . 0	0	2**	31	
1	1	0 . . . 0	1	0	0 . . . 0	1	2**	31	
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	32	
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	−INF
1	1	X . . . X	X	X	X . . . X	X	2**	1024	−NAN

Sign	i30 . . . i1	i0	Rounding Modes	Status Flags
0	1 . . . 1	1	all	INVALOP
0	1 . . . 1	1	all	INVALOP
U*	U . . . U	U	all	OVRFLO
U	U . . . U	U	RP,RN	OVRFLO,INEXO
0	1 . . . 1	1	RZ,RM	INEXO
U	U . . . U	U	RP,RN	OVRFLO,INEXO
0	1 . . . 1	1	RZ,RM	INEXO
U	U . . . U	U	RP	OVRFLO,INEXO
0	1 . . . 1	1	RM,RN,RZ	INEXO
U	U . . . U	U	RP	OVRFLO,INEXO
0	1 . . . 1	1	RM,RN,RZ	INEXO
0	1 . . . 1	1	all	
0	0 . . . 0	1	all	
0	0 . . . 0	1	RN,RP	UNDFLO,INEXO
0	0 . . . 0	0	RZ,RM	UNDFLO,INEXO
0	0 . . . 0	1	RN,RP	UNDFLO,INEXO
0	0 . . . 0	0	RZ,RM	UNDFLO,INEXO
0	0 . . . 0	1	RP	UNDFLO,INEXO
0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	0 . . . 0	1	RP	UNDFLO,INEXO
0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	0 . . . 0	1	RP	UNDFLO,INEXO
0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	0 . . . 0	0	all	
1	1 . . . 1	1	RM	UNDFLO,INEXO
0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1 . . . 1	1	RM	UNDFLO,INEXO
0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1 . . . 1	1	RM	UNDFLO,INEXO
0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1 . . . 1	1	RM,RN	UNDFLO,INEXO
0	0 . . . 0	0	RP,RZ	UNDFLO,INEXO
1	1 . . . 1	1	RM,RN	UNDFLO,INEXO
0	0 . . . 0	0	RP,RZ	UNDFLO,INEXO
1	1 . . . 1	1	all	
1	0 . . . 0	1	all	
1	0 . . . 0	0	RM	INEXO
1	0 . . . 0	1	RP,RN,RZ	INEXO
1	0 . . . 0	0	RM	INEXO
1	0 . . . 0	1	RP,RN,RZ	INEXO
1	0 . . . 0	0	RM,RN	INEXO
1	0 . . . 0	1	RP,RZ	INEXO
1	0 . . . 0	0	RM,RN	INEXO
1	0 . . . 0	1	RP,RZ	INEXO
1	0 . . . 0	0	all	
1	0 . . . 0	0	RP,RN,RZ	INEXO
U	U . . . U	U	RM	OVRFLO,INEXO
1	0 . . . 0	0	RP,RZ	INEXO
U	U . . . U	U	RM,RN	OVRFLO,INEXO
1	0 . . . 0	0	RP,RZ	INEXO
U	U . . . U	U	RM,RN	OVRFLO,INEXO
U	U . . . U	U	all	OVRFLO
U	U . . . U	U	all	OVRFLO,INEXO
U	U . . . U	U	all	OVRFLO
1	1 . . . 1	1	all	INVALOP
1	1 . . . 1	1	all	INVALOP

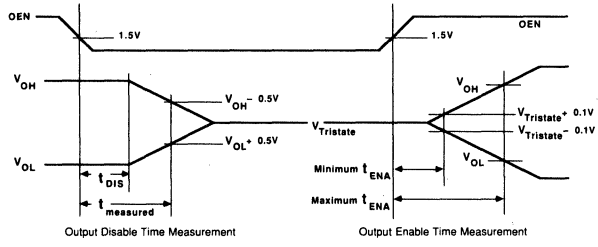
*"U" denotes an undefined result.
NOTE: Heavy line indicates rounding boundary in source.

Table XVII. Conversion of 64-Bit Double-Precision Floating-Point to 32-Bit Twos-Complement Integer

Sign	HB	f51 . . .f30	f29	f28	f27 . . .f1	f0		Unbiased Expnt	Source Name	Sign	HB	f22 . . .f1	f0		Unbiased Expnt	Result Name	Rounding Modes	Status Flags
0	1	X . . . X	X	X	X . . . X	X	2**	1024	+NAN	0	1	1 . . . 1	1	2**	128	+NAN	all	INVALOP
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	+INF	0	1	0 . . . 0	0	2**	128	+INF	all	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	+NORM.MAX	0	1	0 . . . 0	0	2**	128	+INF	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	+NORM.MAX	0	1	1 . . . 1	1	2**	127	+NORM.MAX	RZ,RM	OVRFLO,INEXO
0	1	1 . . . 1	1	1	0 . . . 0	0	2**	127		0	1	1 . . . 1	1	2**	127	+INF	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	1	0 . . . 0	0	2**	127		0	1	1 . . . 1	1	2**	127	+NORM.MAX	RZ,RM	INEXO
0	1	1 . . . 1	1	0	0 . . . 0	1	2**	127		0	1	1 . . . 1	1	2**	127	+INF	RP	OVRFLO,INEXO
0	1	1 . . . 1	1	0	0 . . . 0	1	2**	127		0	1	1 . . . 1	1	2**	127	+NORM.MAX	RM,RN,RZ	INEXO
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	127		0	1	1 . . . 1	1	2**	127	+NORM.MAX	all	
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	127		0	1	1 . . . 1	1	2**	127	+NORM.MAX	RP	INEXO
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	127		0	1	1 . . . 1	0	2**	127		RM,RN,RZ	INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−126		0	1	0 . . . 0	0	2**	−126	+NORM.MIN	all	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127		0	0	1 . . . 1	1	2**	−126	+NORM.MIN	RP,RN	INEXO
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127		0	0	1 . . . 1	1	2**	−126	+DNRM.MAX	RZ,RM	UNDFLO,INEXO
0	1	1 . . . 1	0	0	0 . . . 0	0	2**	−127		0	0	1 . . . 1	1	2**	−126	+DNRM.MAX	all	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−149		0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	all	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN	0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN	0	0	0 . . . 0	0	2**	−126	+ZERO	RM,RN,RZ	UNDFLO, INEXO
0	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	+DNRM.MAX	0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	+DNRM.MAX	0	0	0 . . . 0	0	2**	−126	+ZERO	RM,RN,RZ	UNDFLO, INEXO
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DNRM.MIN	0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DNRM.MIN	0	0	0 . . . 0	0	2**	−126	+ZERO	RM,RN,RZ	UNDFLO, INEXO
0	0	0 . . . 0	0	0	0 . . . 0	0		0	+ZERO	0	0	0 . . . 0	0		0	+ZERO	all	
1	0	0 . . . 0	0	0	0 . . . 0	0		0	−ZERO	1	0	0 . . . 0	0		0	−ZERO	all	
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DNRM.MIN	1	0	0 . . . 0	1	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DNRM.MIN	1	0	0 . . . 0	0	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	−DNRM.MAX	1	0	0 . . . 0	1	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	−DNRM.MAX	1	0	0 . . . 0	0	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN	1	0	0 . . . 0	1	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN	1	0	0 . . . 0	0	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−149		1	0	0 . . . 0	1	2**	−126	−DNRM.MIN	all	
1	1	1 . . . 1	0	0	0 . . . 0	0	2**	−127		1	0	1 . . . 1	1	2**	−126	−DNRM.MAX	all	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127		1	0	1 . . . 1	1	2**	−126	−NORM.MIN	RM,RN	INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127		1	0	1 . . . 1	1	2**	−126	−DNRM.MAX	RP,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−126		1	1	0 . . . 0	0	2**	−126	−NORM.MIN	all	
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	127		1	1	1 . . . 1	1	2**	127	−NORM.MAX	RM	INEXO
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	127		1	1	1 . . . 1	0	2**	127		RP,RN,RZ	INEXO
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	127		1	1	1 . . . 1	1	2**	127	−NORM.MAX	all	
1	1	1 . . . 1	1	0	0 . . . 0	1	2**	127		1	1	1 . . . 1	1	2**	127	−INF	RM	OVRFLO,INEXO
1	1	1 . . . 1	1	0	0 . . . 0	1	2**	127		1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RN,RZ	INEXO
1	1	1 . . . 1	1	1	0 . . . 0	0	2**	127		1	1	1 . . . 1	1	2**	127	−INF	RM,RN	OVRFLO,INEXO
1	1	1 . . . 1	1	1	0 . . . 0	0	2**	127		1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RZ	INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	−NORM.MAX	1	1	0 . . . 0	0	2**	128,	−INF	RM,RN	OVRFLO,INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	−NORM.MAX	1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RZ	OVRFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	−INF	1	1	0 . . . 0	0	2**	128	−INF	all	
1	1	X . . . X	X	X	X . . . X	X	2**	1024	−NAN	1	1	1 . . . 1	1	2**	128	−NAN	all	INVALOP

NOTE: Heavy line indicates rounding boundary in source.

Table XVIII. Conversion of 64-Bit Double-Precision Floating-Point to 32-Bit Single-Precision Floating-Point (IEEE Mode)



Refer to the discussion in the section "Timing" in the text of the data sheet for a description of this figure.

Figure T1. ADSP-3210/3211/3220/3221 Three-State Disable and Enable Timing

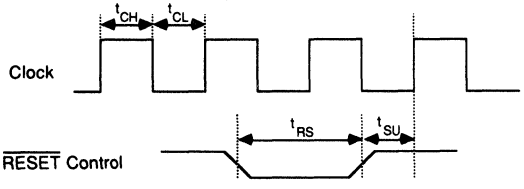


Figure T2. ADSP-3210/3211/3220/3221 Reset Timing

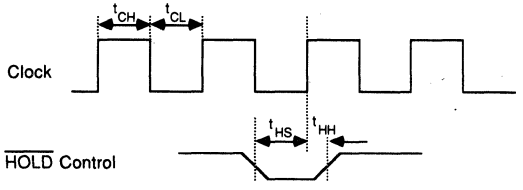
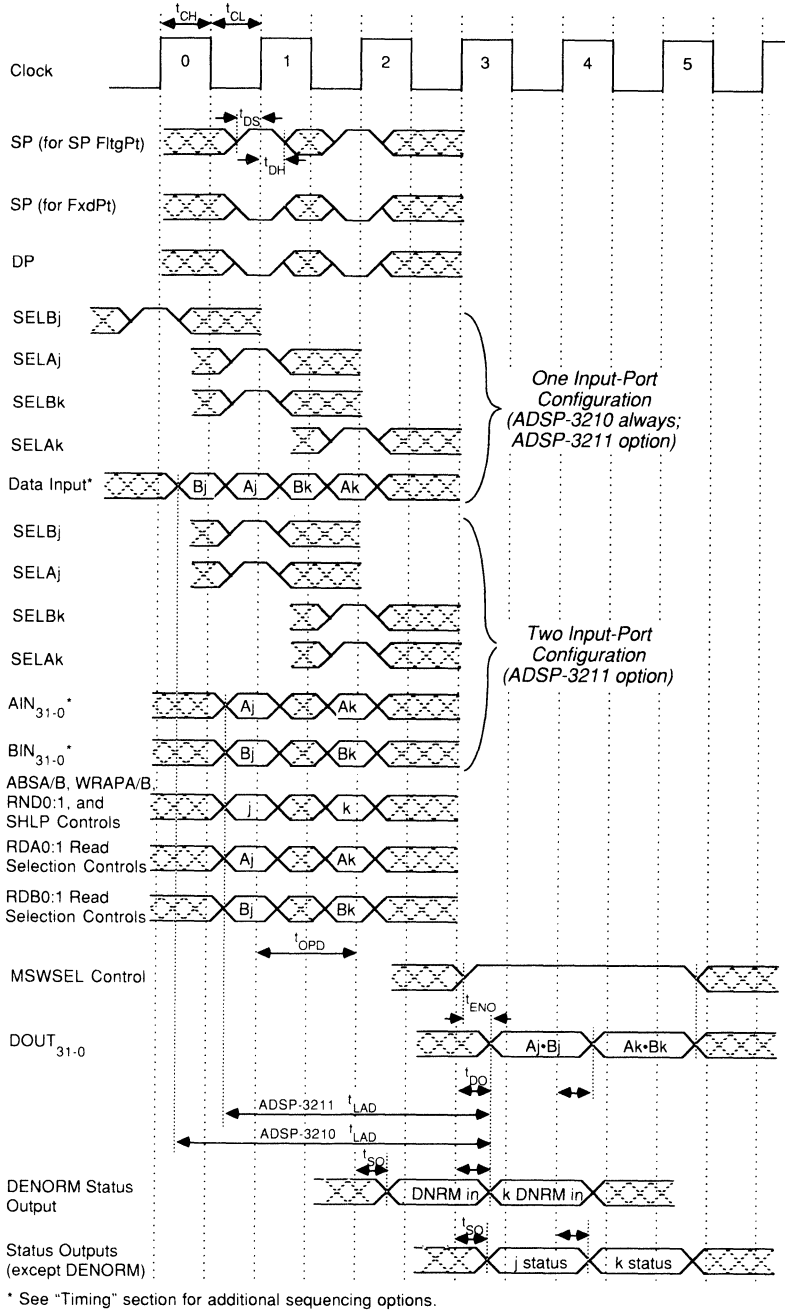
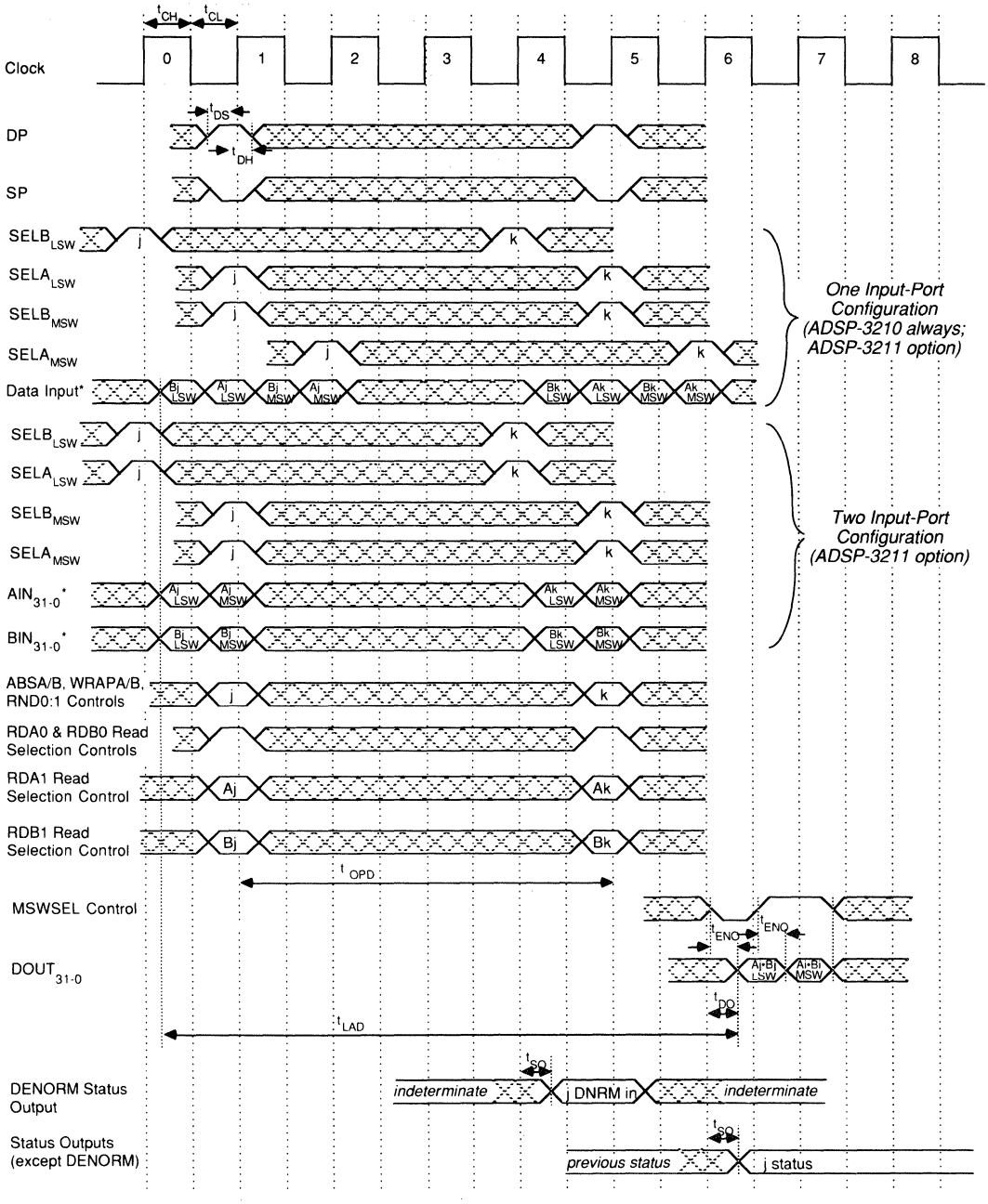


Figure T3. ADSP-3211 Multiplier Output Register Hold Timing



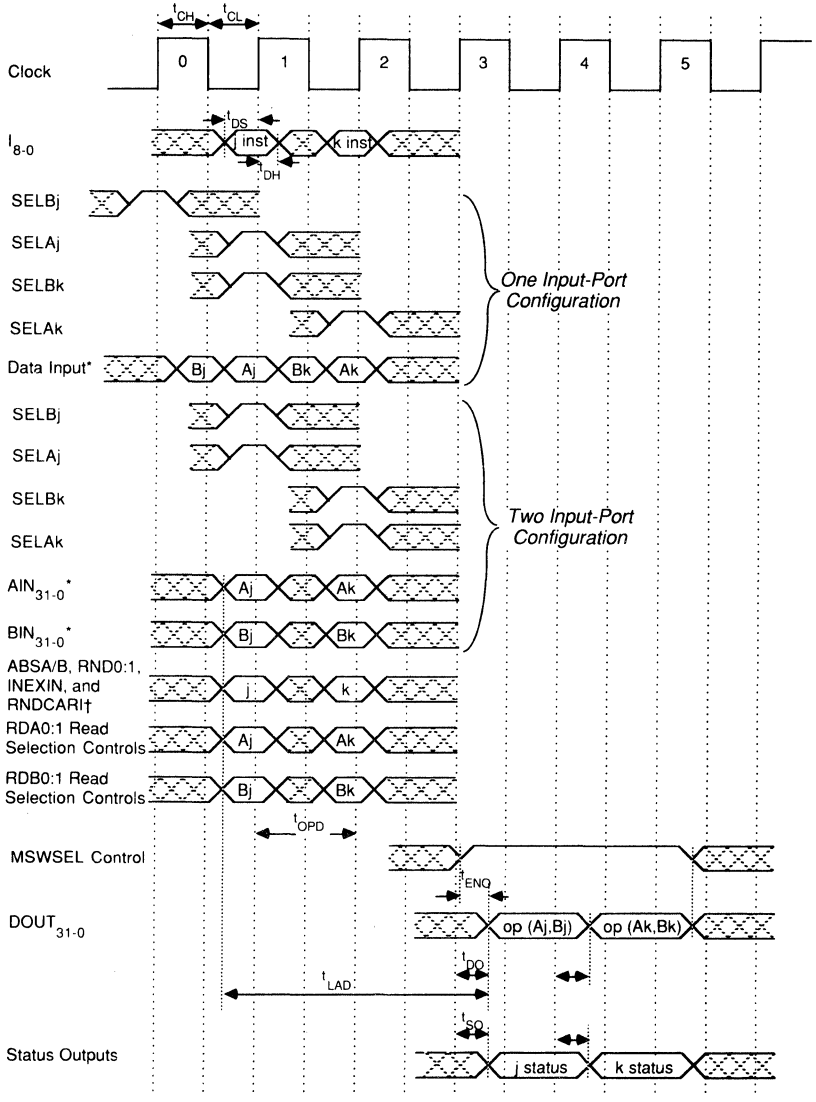
* See "Timing" section for additional sequencing options.

Figure T4. ADSP-3210/3211 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplications



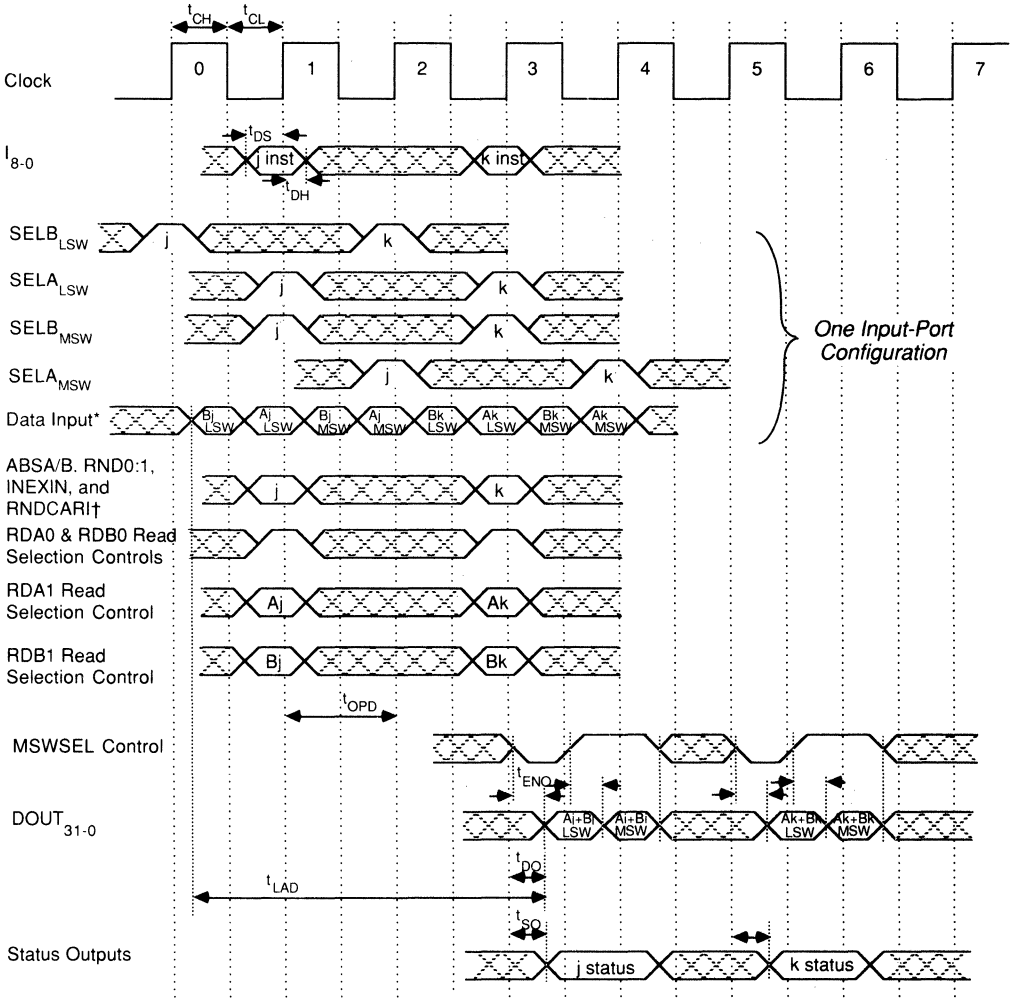
* See "Timing" section for additional sequencing options.

Figure T5. ADSP-3210/3211 64-Bit Double-Precision Floating-Point Multiplications



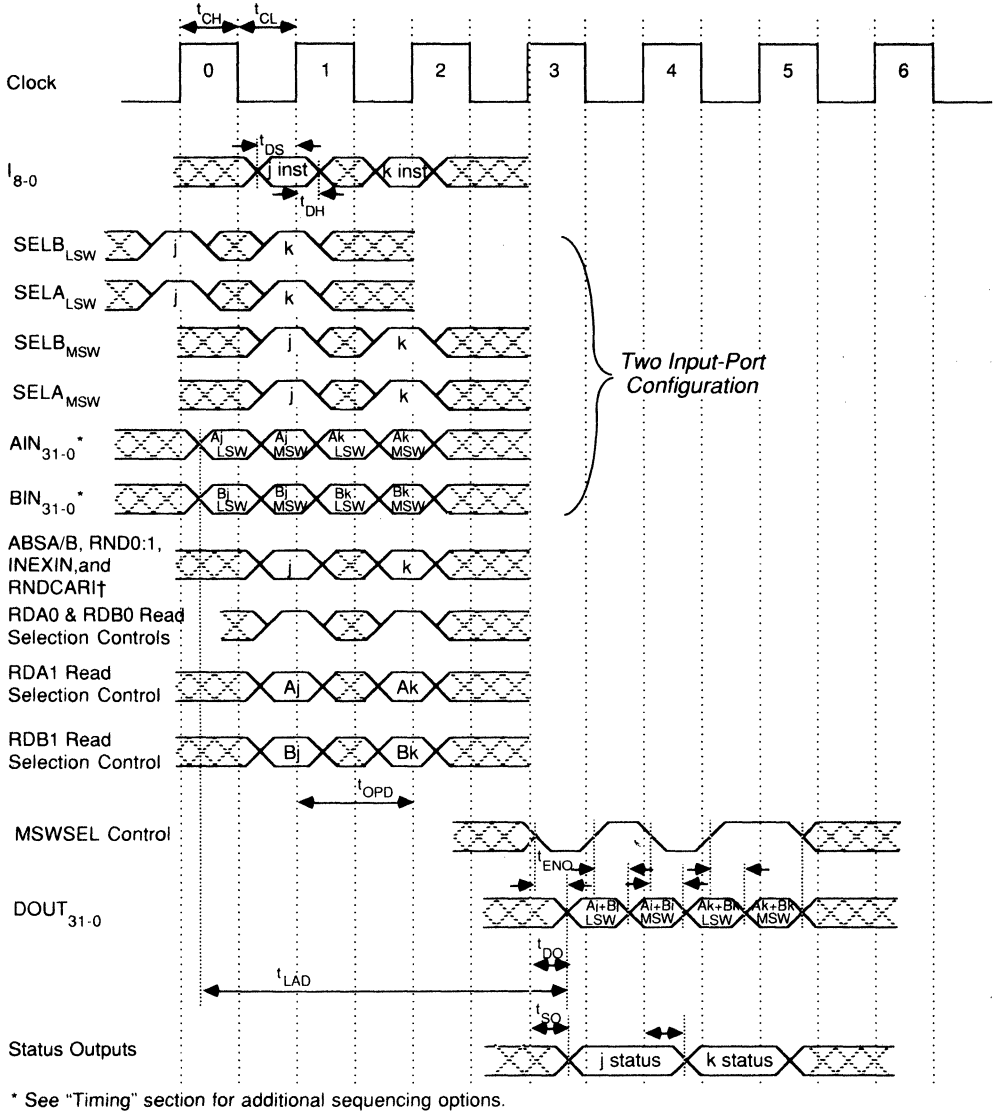
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T6. ADSP-3220/3221 32-Bit Single-Precision Floating-Point Logical, and Fixed-Point ALU Operations



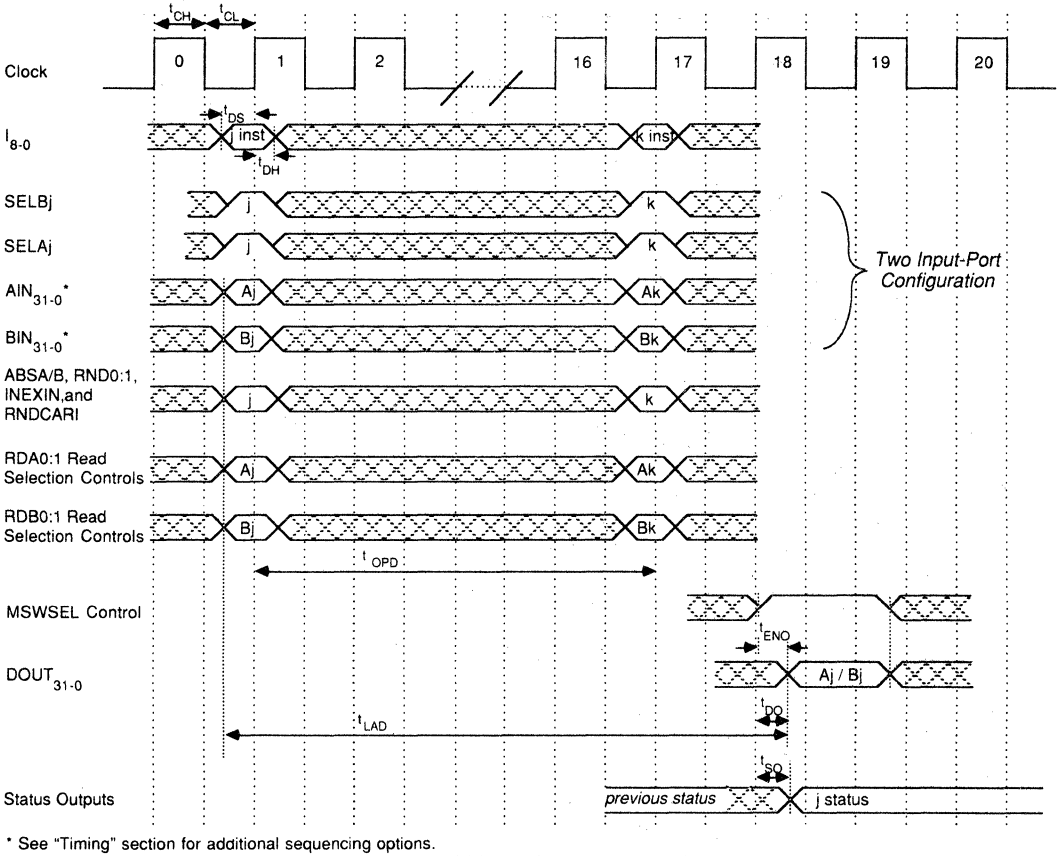
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T7. ADSP-3220/3221 64-Bit Double-Precision Floating-Point ALU Operations – One-Port Configuration



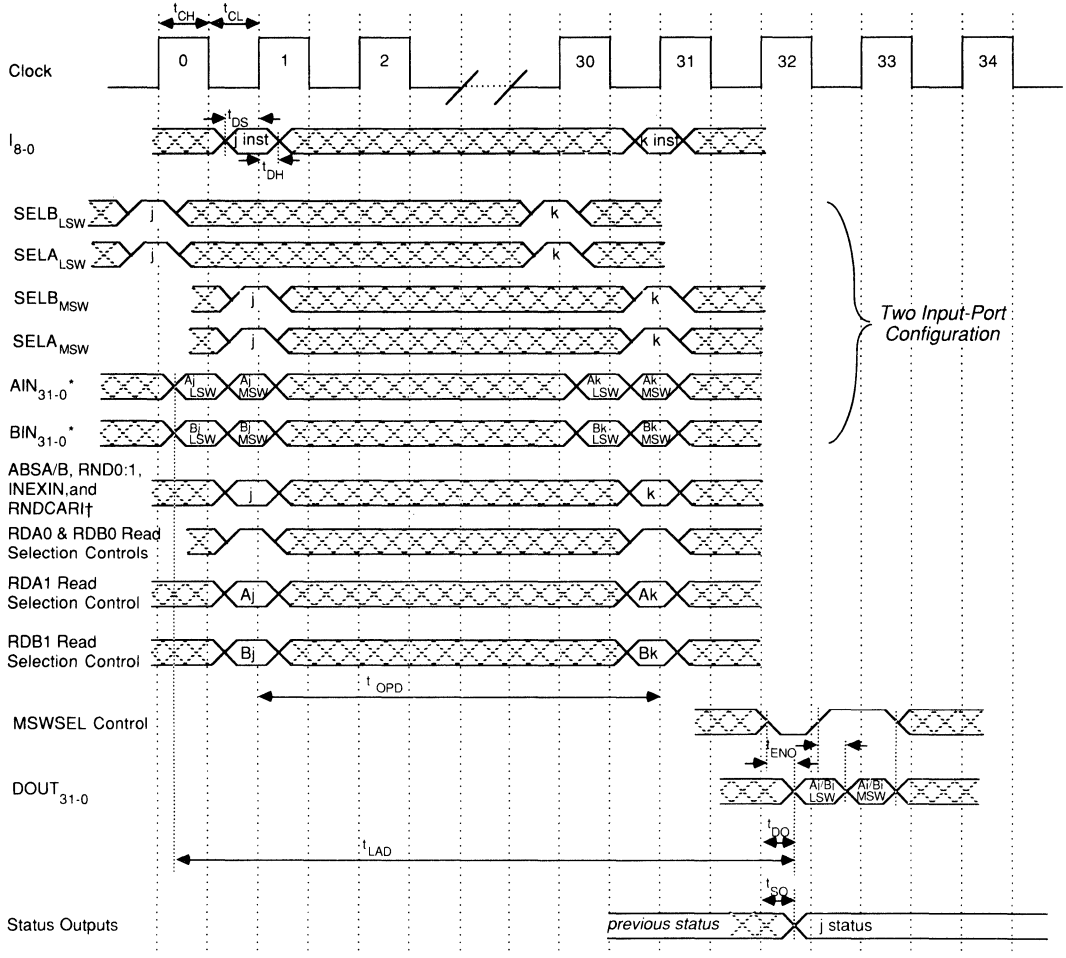
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T8. ADSP-3220/3221 64-Bit Double-Precision Floating-Point ALU Operations – Two-Port Configuration



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

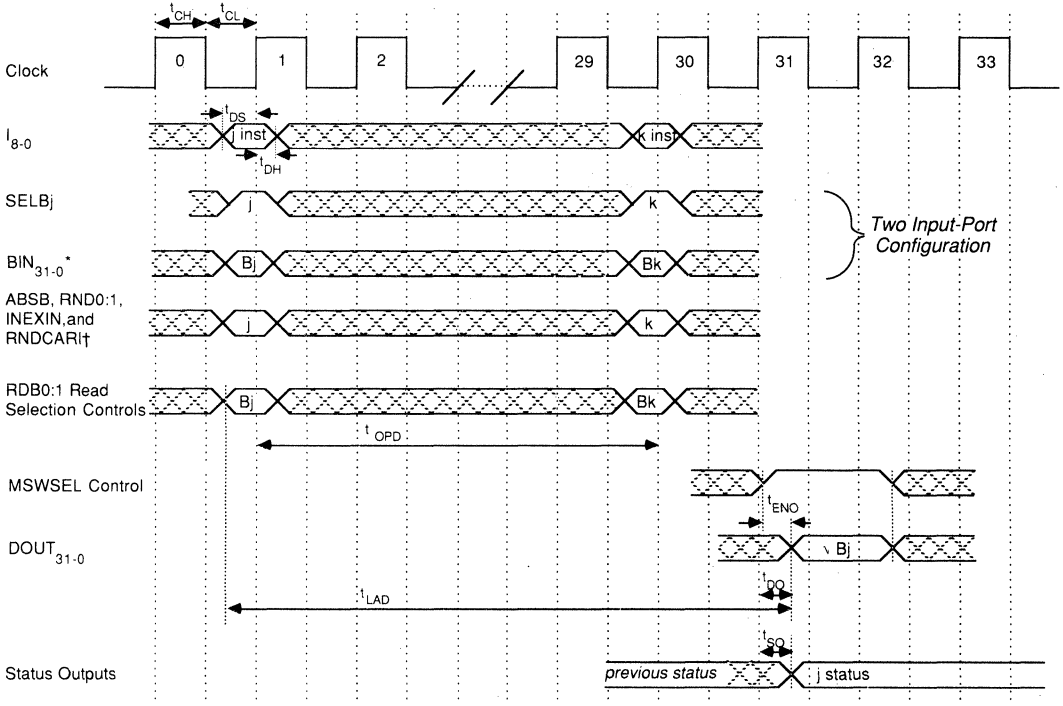
Figure T9. ADSP-3221 32-Bit Single-Precision Floating-Point Division – Two Input Port Configuration



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

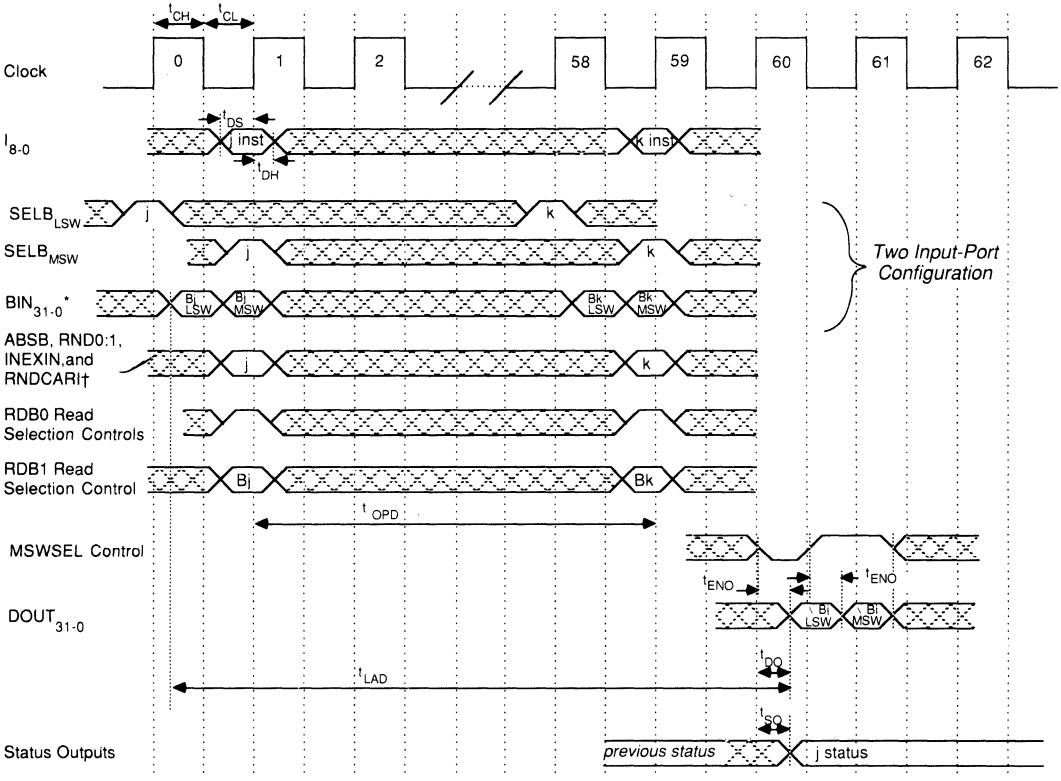
Figure T10. ADSP-3221 64-Bit Double-Precision Floating-Point Division – Two Input-Port Configuration

4



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

Figure T11. ADSP-3221 32 Bit Single-Precision Floating-Point Square Root – 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 T12. ADSP-3221 64-Bit Double-Precision Floating-Point Square Root – Two Input-Port Configuration

4

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3210/3211/3220/3221 J, K, and L Grades Min	J, K, and L Grades Max	S, T, and U Grades[2] Min	S, T, and U 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-3210/3211/3220/3221 J, K, and L Grades Min	J, K, and L Grades Max	S, T, and U Grades[2] Min	S, T, and U 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-3210/3211/3220/3221 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	3	30	3	25	ns
t_{SU}	RESET Setup	25		25		25		25		ns
t_{RS}	RESET Pulse Duration	75		75		75		75		ns
t_{HS}	HOLD Setup	20		15		22		18		ns
t_{HH}	HOLD Hold	3		3		3		3		ns

Parameter		ADSP-3210/3211/3220/3221 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_{OPD}	Operation Time									
	32-Bit Multiplication		125		100		150		125	ns
	64-Bit Multiplication		500		400		600		500	ns
	32-Bit ALU Operations		125		100		150		125	ns
	64-Bit ALU Operations		125		100		150		125	ns
	32-Bit Division (3221)		2.0		1.6		2.4		2.0	µs
	64-Bit Division (3221)		3.75		3.0		4.5		3.75	µs
	32-Bit Square Root (3221)		3.625		2.9		4.35		3.625	µs
	64-Bit Square Root (3221)		7.25		5.8		8.7		7.25	µs
t_{LAD}	Total Latency									
	32-Bit Multiplication (3210)		363		290		435		363	ns
	32-Bit Multiplication (3211)		300		240		360		300	ns
	64-Bit Multiplication		738		590		885		738	ns
	32-Bit ALU Operation		300		240		360		300	ns
	64-Bit ALU Operation		363		290		435		363	ns
	32-Bit Division (3221)		2.175		1.74		2.61		2.175	µs
	64-Bit Division (3221)		3.925		3.14		4.71		3.925	µs
	32-Bit Square Root (3221)		3.8		3.04		4.56		3.8	µs
	64-Bit Square Root (3221)		7.425		5.94		8.91		7.425	µs

Parameter		ADSP-3210 L Grade 0 to 70°C Min	ADSP-3210 L Grade 0 to 70°C Max	ADSP-3211 L Grade 0 to 70°C Min	ADSP-3211 L Grade 0 to 70°C Max	ADSP-3210 U Grade −55°C to +125°C Min	ADSP-3210 U Grade −55°C to +125°C Max	ADSP-3211 U Grade −55°C to +125°C Min	ADSP-3211 U Grade −55°C to +125°C Max	Unit
t_{CY}	Clock Cycle		60		50		75		70	ns
t_{CL}	Clock LO	20		20		30		30		ns
t_{CH}	Clock HI	20		20		30		30		ns
t_{DS}	Data & Control Setup	15		15		20		20		ns
t_{DH}	Data & Control Hold	3		3		3		3		ns
t_{DO}	Data Output Delay		25		25		30		30	ns
t_{SO}	Status Output Delay		25		25		30		30	ns
t_{ENO}	MSWSEL-to-Data Delay		20		20		25		25	ns
t_{DIS}	Three-State Disable Delay		15		15		20		20	ns
t_{ENA}	Three-State Enable Delay	3	20	3	20	3	25	3	25	ns
t_{SU}	RESET Setup	15		15		20		20		ns
t_{RS}	RESET Pulse Duration	50		50		50		50		ns
t_{HS}	HOLD Setup	15		15		20		20		ns
t_{HH}	HOLD Hold	3		3		3		3		ns
t_{OPD}	Operation Time									
	32-Bit Multiplication		60		50		75		70	ns
	64-Bit Multiplication		240		200		300		280	ns

Parameter	ADSP-3210 L Grade 0 to 70°C		ADSP-3211 L Grade 0 to 70°C		ADSP-3210 U Grade −55°C to +125°C		ADSP-3211 U Grade −55°C to +125°C		Unit
	Min	Max	Min	Max	Min	Max	Min	Max	
t_{LAD} Total Latency									
32-Bit Multiplication		190		140		238		190	ns
64-Bit Multiplication		370		315		463		400	ns

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-3210/3211/3220/3221 can be found in Analog Devices' Military Databook.
[3]Input levels are GND and +3.0V. Rise times are 5ns max. Input timing reference levels and output reference levels are 1.5V, except for 1) t_{ENA} and $_{DIS}$ which are as indicated in Figure T1 and 2) t_{DS} and t_{DH} which are measured from clock V_{IHA} to data input V_{IH} or V_{IL} crossing points.
Specifications subject to change without notice.

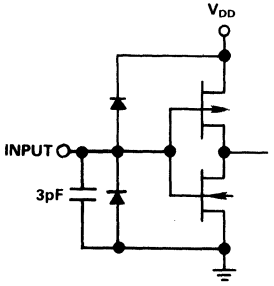


Figure 34. Equivalent Input Circuits

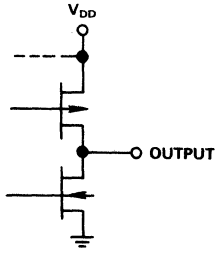


Figure 35. Equivalent Output Circuits

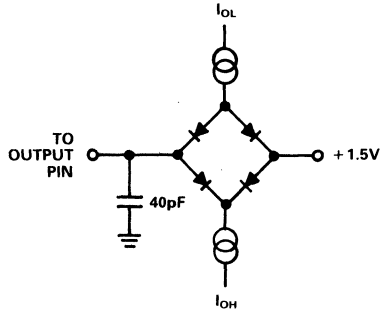


Figure 36. Normal Load for ac Measurements

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-3210JG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210KG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210LG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210SG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210TG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210UG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210SG/883B	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210TG/883B	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210UG/883B	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3211JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3211KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3211LG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3211SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3211TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3211UG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3211SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3211TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3211UG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3220JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3220KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3220SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3220TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3220SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3220TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3221JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3221KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3221SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3221TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3221SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3221TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A

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

4

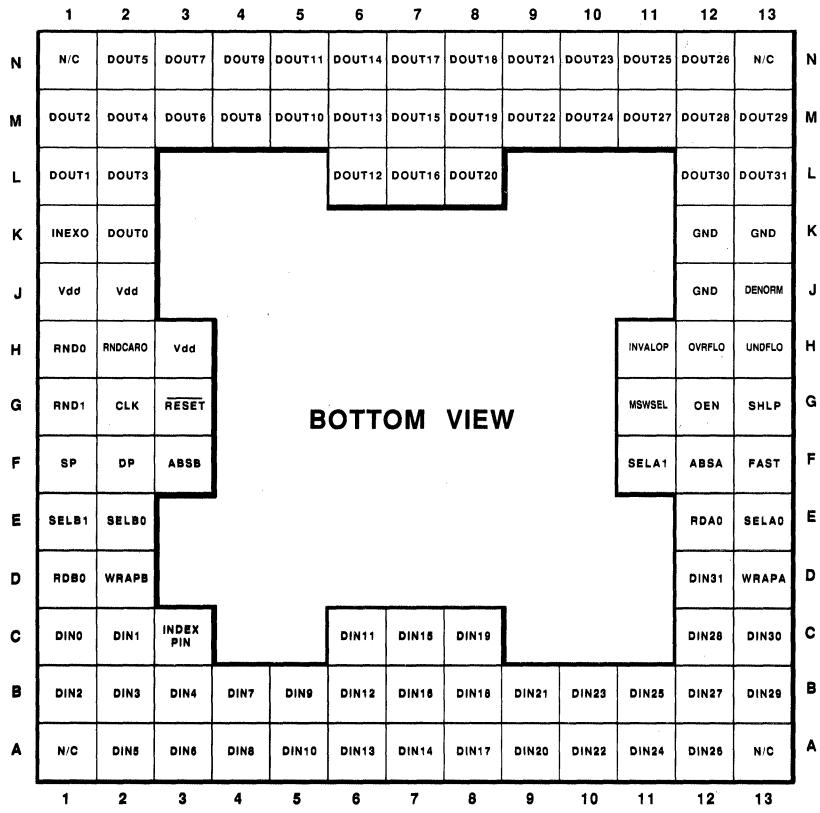
ABSOLUTE MAXIMUM RATINGS

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

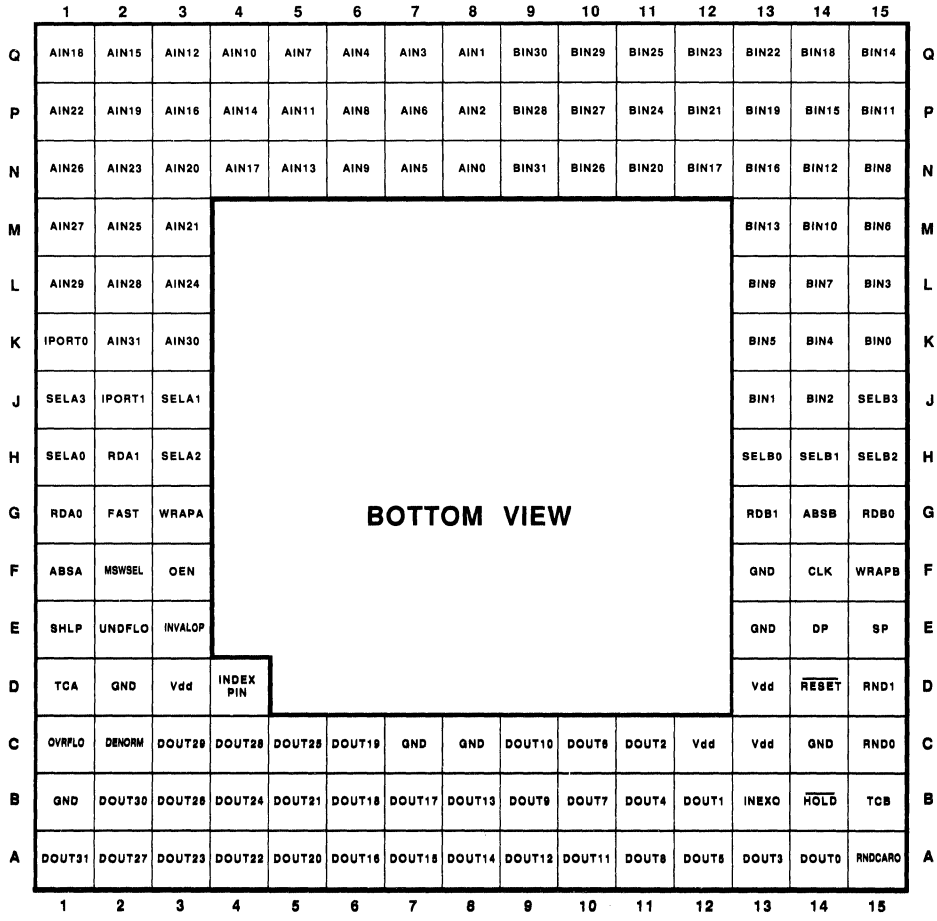
ESD SENSITIVITY

Each chip in the ADSP-3210/3211/3220/3221 chipset features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, these chips have been classified as Class 1 devices.

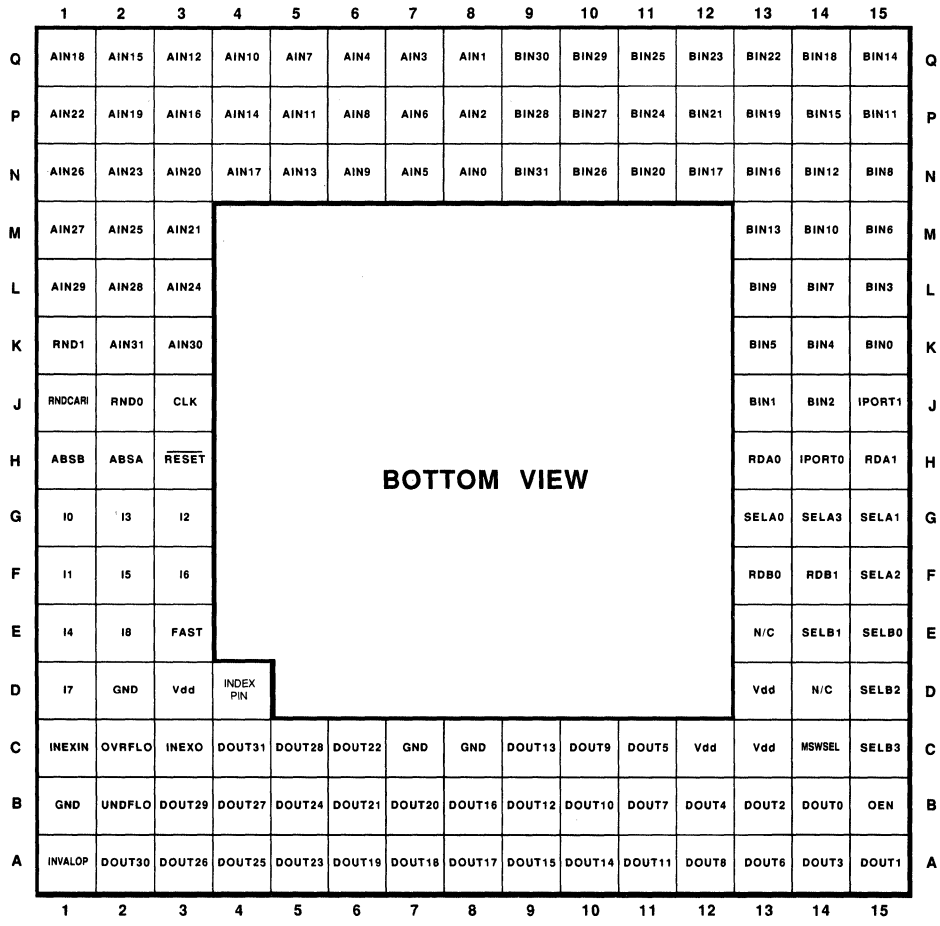
Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should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



ADSP-3210 Pinouts



ADSP-3211 Pinouts



ADSP-3220/3221 Pinouts

ANALOG DEVICES

64-Bit IEEE Floating-Point Chipset

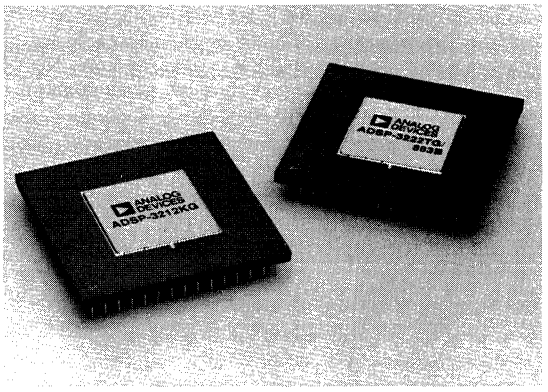
ADSP-3212/ADSP-3222

FEATURES

Complete 40 MFLOPS Floating-Point Chipset
Multiplier/Divider and ALU
Fully Compatible with IEEE Standard 754
Arithmetic Operations on Four Data Formats:
32-Bit Single-Precision Floating-Point
64-Bit Double-Precision Floating-Point
32-Bit Twos-Complement Fixed-Point
32-Bit Unsigned-Magnitude Fixed-Point
Only One Internal Pipeline Stage
20 MFLOPS Pipelined Throughput For Multiplication and Standard ALU Operations
Exact Division: 300ns Single Precision and 600ns Double Precision
Low Latency for Scalar Operations
130ns for 32-Bit Multiplication or Standard ALU Operations
155ns for 64-Bit Multiplication or Standard ALU Operations
Exact Square Root ALU Instruction
2.5W Maximum Power Dissipation per Chip with 1.0 μm CMOS Technology
144-Lead Pin Grid Array
Available Specified to MIL-STD-883, Class B
Pin-Compatible Upgrades From ADSP-3211/ADSP-3221

APPLICATIONS

High Performance Digital Signal Processing
Engineering Workstations
Floating-Point Accelerators
Array Processors
Mini-Supercomputers
RISC Processors



4

GENERAL DESCRIPTION

The ADSP-3212 Floating-Point Multiplier/Divider and the ADSP-3222 Floating-Point ALU are high speed, low power arithmetic processors conforming to IEEE Standard 754. The multiplier/divider and ALU comprise the basic computational elements for implementing a high speed numeric processor. Operations are supported on four data formats: 32-bit IEEE single-precision floating-point, 64-bit IEEE double-precision floating-point, 32-bit twos-complement fixed-point and 32-bit unsigned-magnitude fixed-point.

The high throughput of the ADSP-3212/ADSP-3222 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-3212/ADSP-3222 results in very low latency, important in scalar processing and in algorithms with data dependencies.

Both chips have internal feedback paths from the output to four of the eight input registers and feedforward paths from all input registers to the output register. Feedback to both banks of input registers facilitates interleaving partial sums and partial products for maximum throughput.

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 four 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.

IEEE floating-point division is supported by both the ADSP-3212 and the ADSP-3222. The ADSP-3212 is the faster of the two, performing single-precision division in six cycles and double-precision division in 12 cycles. The division operation is initiated by the assertion of the multiplier/divider's DIVMUL input. On the ADSP-3222 ALU two instructions, SDIV and DDIV, calculate single-precision division (16 cycles) and double-precision division (30 cycles), respectively. ADSP-3222 division instructions are supported for compatibility with the ADSP-3221.

The instruction set of the ADSP-3212/ADSP-3222, a superset of the ADSP-3210/3211/3220/3221 instruction set, is oriented to system-level implementations of function calculations. Specific instructions are included to facilitate such operations as floating-point division and square root, table lookup, quadrant normalization for trigonometric functions, extended-precision integer operations, logical operations and conversions between all data formats.

Both chips have two input ports and eight input registers (two banks of four registers) and are always in a two-input-port configuration; data can always be input on both ports simultaneously. In the 32-bit data loading mode, input can be directed to registers in either bank, although both ports may not input to the same bank at once. If 64-bit parallel data loading is enabled, 64-bit data from both ports may be directed to one of four register pairs.

In addition to double- and single-precision floating-point multiplication and division, the ADSP-3212 Floating-Point Multiplier/Divider supports 32-bit fixed-point multiplications: twos-complement, unsigned-magnitude and mixed-mode. The ADSP-3212 also has a $\overline{\text{HOLD}}$ control that prevents the updating of the output data and status registers.

TABLE OF CONTENTS

The instruction set of the ADSP-3222 Floating-Point ALU includes exact IEEE floating-point division and square-root operations. The ADSP-3222 is pin-compatible with the ADSP-3220/ADSP-3221. It also includes a $\overline{\text{HOLD}}$ control that is enabled through an overhead instruction.

The ADSP-3212/ADSP-3222 chipset is fabricated in double-metal 1.0μm CMOS. Each chip consumes 2.5W 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-3212/ADSP-3222 can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-3212/ADSP-3222 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-3212 and ADSP-3222 are packaged in a ceramic 144-lead pin grid array.

FUNCTIONAL DESCRIPTION OVERVIEW

The ADSP-3212/ADSP-3222 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. Two 32-bit operands can be loaded simultaneously; alternatively, the ADSP-3212/ADSP-3222 can operate in a mode in which both halves of a 64-bit operand are loaded in parallel. The input 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 double-precision result off-chip through the 32-bit output port in one output cycle.

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.

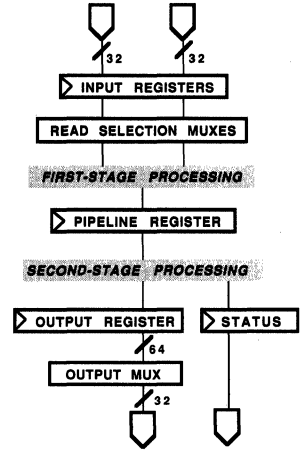


Figure 1. ADSP-32XX Generic Architecture

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.

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-3212/ADSP-3222

The ADSP-3212/ADSP-3222 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 ADSP-3212/ADSP-3222 can also operate in a mode in which all 64 bits of a double-precision word are loaded into an input register pair in parallel. This mode allows direct connection to a 64-bit input bus. 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 multiplexers can select data that was just loaded at the clock's rising edge, 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 port configuration, output multiplexer and tristate controls at a higher rate, however.)

The ADSP-3212/ADSP-3222 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 division and square root). The Floating-Point Multiplier/Divider operates on normals but cannot operate on denormals directly. Denormals must first be "wrapped" (converted to a format acceptable for multiplication, division or square root) by an ALU. Several flags are available for detecting and handling exceptions caused by loading a denormal into a Floating-Point Multiplier/Divider. Information about rounding and inexact results generated by the multiplier/divider is needed by the ALU to produce results in conformance to Standard 754. Both ADSP-3212/ADSP-3222 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/Divider, speeding exception processing.

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-3212 FLOATING-POINT MULTIPLIER/DIVIDER 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	LAT
IPORT	Input Port Configuration Control	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 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
FDBK0	Feedback Control 0	REG
FDBK1	Feedback Control 1	REG
LOAD64	Enable 64-Bit Parallel Input	REG
DIVMUL	Divide/Multiply	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN

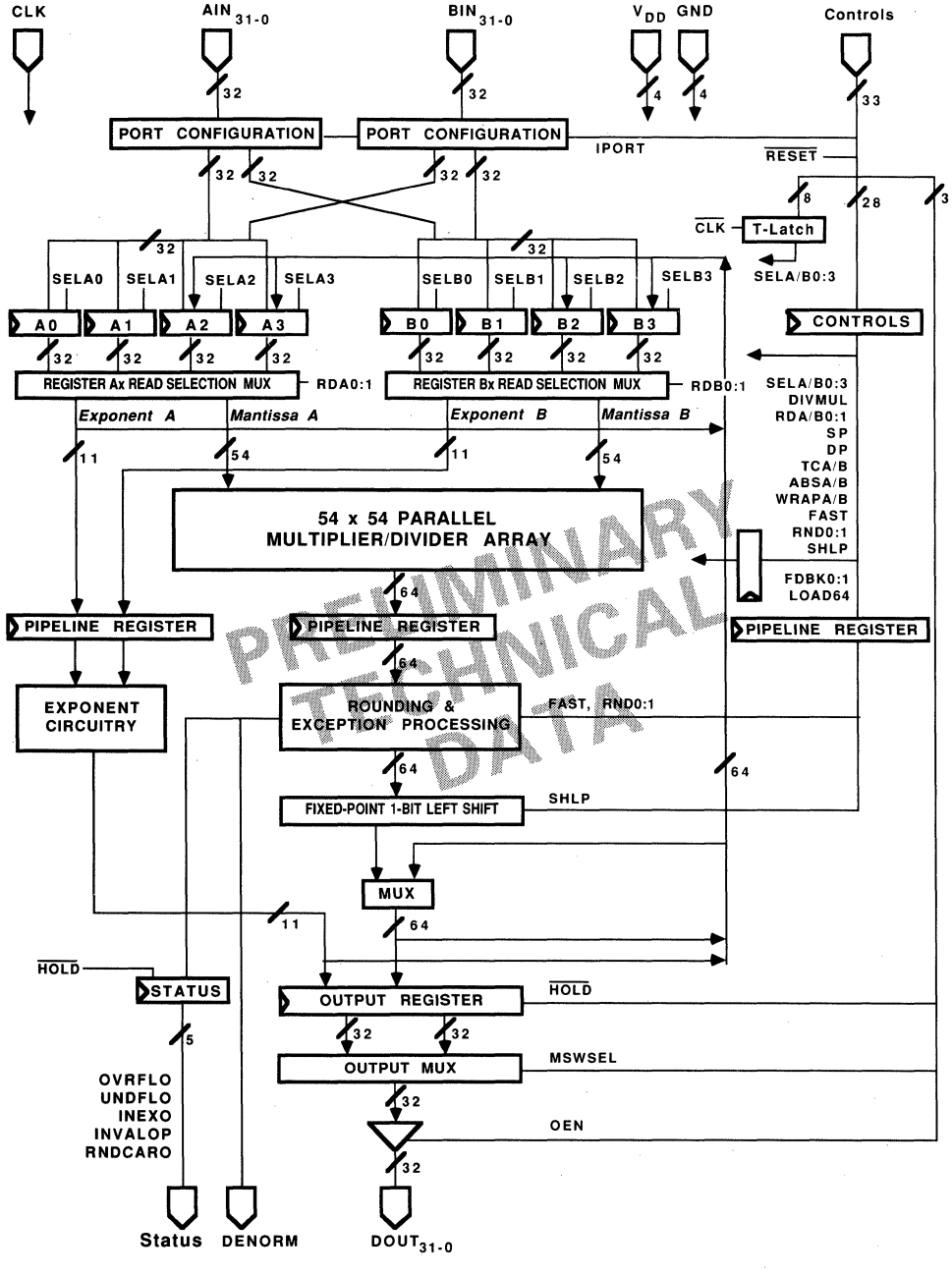


Figure 3. ADSP-3212 Block Diagram

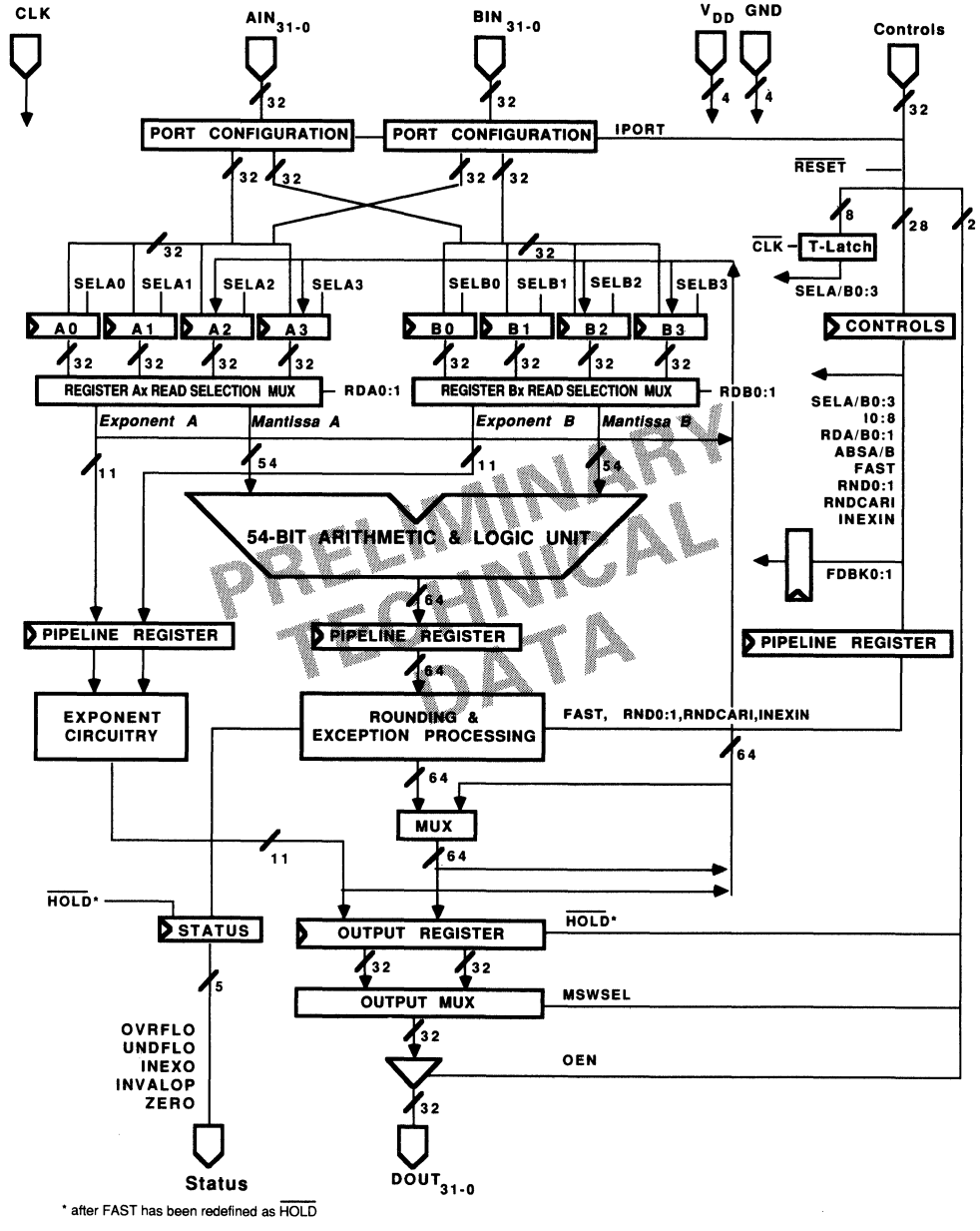


Figure 4. ADSP-3222 Block Diagram

Pin Name	Description	Type
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 (four lines)	

ADSP-3222 FLOATING-POINT ALU 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
IPORT	Input Port Configuration Control	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
ABSA	Read Absolute Value of Ax	REG
ABSB	Read Absolute Value of Bx	REG
I_{8-0}	ALU Instruction	REG
RND0	Rounding Mode Control 0	REG
RND1	Rounding Mode Control 1	REG
FAST	Fast Mode/$\overline{\text{HOLD}}$ Control	REG
FDBK0	Feedback Control 0	REG
FDBK1	Feedback Control 1	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN
STATUS IN		
INEXIN	Inexact Data In	REG
RNDCARI	Round Carry Propagation In	REG
STATUS OUT		
INEXO	Inexact Result	
OVRFLO	Overflowed Result	
UNDFLO	Underflowed Result	
INVALOP	Invalid Operation	
ZERO	Zero Result	
MISCELLANEOUS		
CLK	Clock Input	
V_{DD}	+5V Power Supply (four lines)	
GND	Ground Supply (four lines)	

METHOD OF OPERATION

Data Formats

The ADSP-3212/ADSP-3222 chipset supports both single- and double-precision floating-point data formats and operations as defined in IEEE Standard 754-1985. Both chips support 32-bit twos-complement fixed-point as well as 32-bit unsigned-magnitude data formats and operations (the ADSP-3212 supports fixed-point multiplication but not fixed-point division). Both chips operate directly on 32-bit fixed-point data. No time consuming conversions to and from floating-point formats are required.

IEEE Single-Precision Floating-Point Data Format

IEEE Standard 754 specifies a 32-bit single-precision floating-point format, which consists of a sign bit *s*, a 24-bit significand

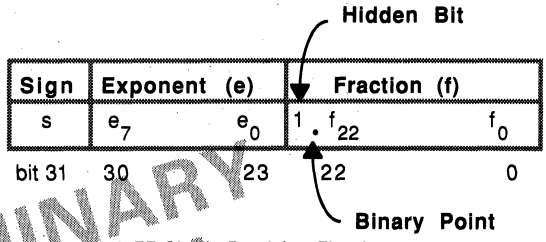


Figure 5. IEEE Single-Precision Floating-Point Format

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 (254÷2) 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 non-zero 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 non-zero 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:

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	$(-1)^s0.0$	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. IEEE Single-Precision Floating-Point Data Types and Interpretations

The ADSP-3212/ADSP-3222 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-3222 ALU (during division and square root) and the ADSP-3212 Multiplier/Divider 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-3222 ALU to wrapped numbers to be readable by the ADSP-3212 Multiplier/Divider. Wrapped and unnormal multiplier/divider results must also be unwrapped by the ADSP-3222 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 into wrapped numbers where the exponent e ranges between $-22 \le e \le 0$. WRAPA and WRAPB controls on the ADSP-3212 tell the multiplier/divider to interpret a data value as a wrapped number.

The ranges of the various single-precision IEEE floating-point data formats supported by the ADSP-3212/ADSP-3222 are summarized in Table II.

The multiplication of a wrapped number by a normal number or another wrapped number 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 is $-171 \le e \le -23$ 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. (Unnormals can unwrap to DRNM.MIN, depending on the 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.

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. IEEE Single-Precision Floating-Point Range Limits

IEEE Double-Precision Floating-Point Data Format

IEEE Standard 754 specifies a 64-bit double-precision floating-point format:

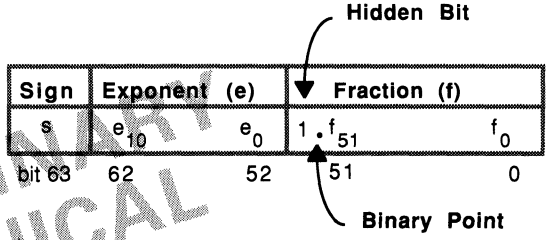


Figure 6. IEEE Double-Precision Floating-Point Format

The key differences with the single-precision format are that the exponent e is now 11 bits in length and the fraction f is now 52 bits in length, yielding a 53-bit significand for double-precision normals. Double-precision, like single-precision, has an implicit hidden bit; in this case the hidden bit precedes f_{51}. The binary point comes between the hidden bit and f_{51}. The exponent bias for double-precision floating-point normals is +1023 (2046÷2).

In other respects, IEEE double-precision floating-point is exactly analogous to single-precision, with the same data types whose values are summarized in Table III.

The unbiased value of a denormal's exponent is −1022 for double-precision denormals, i.e., one minus the bias. Because of the extended width of the double-precision fraction, the exponent of double-precision wrapped numbers can range from $-51 \le e \le 0$. The exponent of unnormals can range from $-1125 \le e \le -52$. Again, the smallest unnormal is the result of multiplying the smallest wrapped number by itself. Note that $e = -1024$ is the smallest double-precision exponent that is directly representable in the eleven-bit IEEE twos-complement exponent field. The underflow flag should be thought of as a most significant twelfth bit, the sign bit, as explained above for single-precision unnormals.

Mnemonic	Exponent	Fraction	Value	Name	IEEE Format?
NAN	2047	non-zero	undefined	not-a-number	yes
INF	2047	zero	$(-1)^s$(infinity)	infinity	yes
NORM	1 thru 2046	any	$(-1)^s(1.f)2^{e-1023}$	normal	yes
DNRM	0	non-zero	$(-1)^s(0.f)2^{-1022}$	denormal	yes
ZERO	0	zero	$(-1)^s0.0$	zero	yes
WRAP	−51 thru 0	any	$(-1)^s(1.f)2^{e-1023}$	wrapped	no
UNRM	−1125 thru −52	any	$(-1)^s(1.f)2^{e-1023}$	unnormal	no

Table III. IEEE Double-Precision Floating-Point Data Types and Interpretations

The ranges for the various double-precision data types are:

Data name (positive)	Exponent	Exp. data type	Exponent bias	Hidden bit	Fraction (binary)	Unbiased absolute value
NORM.MAX	2046	unsigned	+1023	1	111.....11	$2^{+1023}\cdot(2-2^{-52})$
NORM.MIN	1	unsigned	+1023	1	000.....00	2^{-1022}
DNRM.MAX	0	unsigned	+1022	0	111.....11	$2^{-1022}\cdot(1-2^{-52})$
DNRM.MIN	0	unsigned	+1022	0	000.....01	$2^{-1022}\cdot 2^{-52}$
WRAP.MAX	0	2scmplmt	+1023	1	111.....11	$2^{-1023}\cdot(2-2^{-52})$
WRAP.MIN	−51	2scmplmt	+1023	1	000.....00	2^{-1074}
UNRM.MAX	−52	2scmplmt	+1023	1	111.....11	$2^{-1075}\cdot(2-2^{-52})$
UNRM.MIN	−1125	2scmplmt	+1023	1	000.....00	2^{-2148}

Table IV. IEEE Double-Precision Floating-Point Range Limits

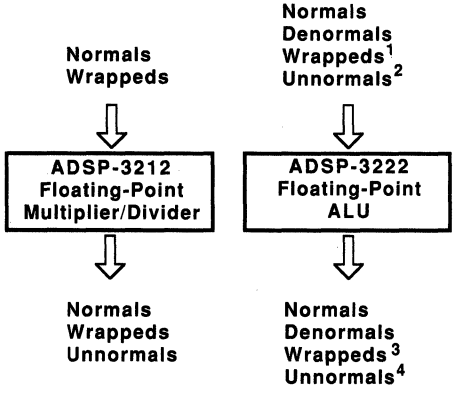
Supported Floating-Point Data Types

The floating-point data types supported directly by the ADSP-3212/ADSP-3222 chipset are summarized in Figure 7.

Not all the data types described above are supported 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-3212/ADSP-3222 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/divider produces a 64-bit product from two 32-bit inputs. Fixed-point division and fixed-point square root are not supported directly. However, they can be accomplished using the fixed-point/floating-point conversions.



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

Figure 7. Data Types Directly Supported by the ADSP-3212/ADSP-3222

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

WEIGHT	Sign -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 8. 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 to 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-3212 Multiplier/Divider can produce a 64-bit product at its output register. The ADSP-3212 will produce results in the format of Figure 9 at the DOUT port if the Shift Left Fixed-Point Product (SHLP) control (described below in "Output Control") is LO:

WEIGHT	Sign -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 9. 64-Bit Twos-Complement Fixed-Point Data Format at Multiplier/Divider 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 10. 64-Bit Twos-Complement Fixed-Point Data Format at Multiplier/Divider 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 8), then a single-bit left-shift will renormalize the MSP to a fractional format (because r = 2•k = 2•(−31) = −62).

For unsigned-magnitude data formats, inputs to the ADSP-3212 Multiplier/Divider and inputs and outputs for the ADSP-3222 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 11. 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 to the 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-3212 Multiplier/Divider discriminates twos-complement from unsigned-magnitude inputs with TCA and TCB controls (see "Controls"). When TCA and TCB are both LO, the ADSP-3212 produces a 64-bit unsigned-magnitude product at its output register. The ADSP-3212 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 12. 64-Bit Unsigned-Magnitude Fixed-Point Data Format at Multiplier/Divider 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 shown in Figure 13.

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 13. 64-Bit Unsigned-Magnitude Fixed-Point Data Format at Multiplier/Divider Output Register with SHLP HI

The ADSP-3212 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 9.

Controls

The controls for the ADSP-3212/ADSP-3222 (see Pin Definitions 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 an 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 multi-cycle 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 set up 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 and exception processing pipeline stage. FAST also affects the definition of some exception flags (see "Status 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-3212/ADSP-3222 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/divider will treat denormal inputs as ZERO. 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-3212/ADSP-3222 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.

On the ADSP-3222, the FAST input can be redefined after reset as a $\overline{\text{HOLD}}$ input through the HOLDEN instruction (see Table XIII). The mode (IEEE or FAST) that is active when the instruction is executed remains in effect. To restore the FAST function, you must reset the ADSP-3222.

$\overline{\text{RESET}}$ Control (ASYN)
The asynchronous, active LO $\overline{\text{RESET}}$ control clears all control functions in the ADSP-3212/ADSP-3222. $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$.

On reset, the ADSP-3222 is set for 32-bit input data loads and for the IEEE/FAST function on its FAST input.

Port Configuration – IPORT Control (ASYN)
This chipset offers two options on input port configuration. Both configurations in Figure 14 are "two-port" configurations. The options are controlled by the asynchronous input IPORT, which allows you to switch the port configuration dynamically—as often as every half cycle. Take care that IPORT is at valid logic levels at all clock edges at which data is loaded. IPORT must meet setup and hold times at every point at which data is loaded—a rising edge or falling edge of the clock. Proper data loading cannot be guaranteed unless this requirement is observed.

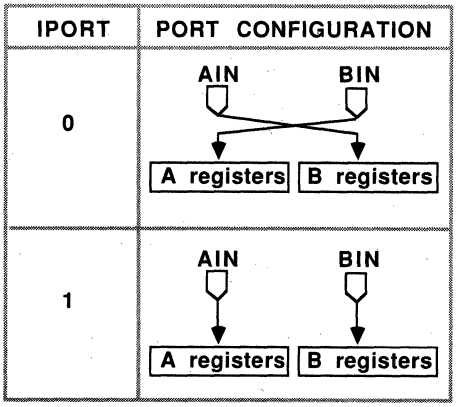


Figure 14. ADSP-3212/ADSP-3222 Input Port Configurations

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, SELA0:3 and SELB0:3. 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 in Figure 15.

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

Figure 15. ADSP-3212/ADSP-3222 Load Selection Controls

For 32-bit data loading, even-numbered registers load data on rising edges and odd-numbered registers on falling edges, as shown in Figure 16. However, you should not load a 32-bit register on the clock's falling edge in the same cycle that the register is read into the chip's processing circuits (see "Restrictions on Register Storage," below).

In the 64-bit parallel loading mode, inputs from both ports can be directed to appropriate register pairs (see "Restrictions on Register Storage," below). This mode is enabled by the LOAD64 pin on the ADSP-3212; on the ADSP-3222, the LOAD64 instruction sets 64-bit loading until the next reset. If

A registers are selected, they are loaded on the rising edge of the clock; B registers are loaded on the falling clock edge. LOAD64 is pipelined for one cycle on both parts; that is, the 64-bit load option will become effective at the *next* rising edge after the pin is asserted or the instruction is presented.

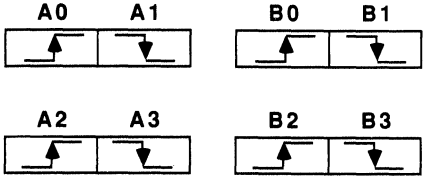


Figure 16. ADSP-3212/ADSP-3222 Clock Edge for 32-Bit Data Loading

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 multiplications 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 chip's processing circuits in that same cycle (with the RDA/B controls described in "Input Data Register Read Selection"). Pick a register not in use.

The ADSP-3222 ALU is capable of two multicycle operations: IEEE floating-point division (16 cycles for single precision and 30 cycles for double precision) and square root (29 cycles for single precision and 58 cycles for double precision). The ADSP-3212 Multiplier/Divider performs multicycle (6 and 12 cycles for single precision and double precision, respectively) floating-point division. 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, whether single-precision or double-precision. (See "Timing" and the timing diagrams below for a precise definition of "operation time.")

With 64-bit double-precision data, there are constraints on which registers hold which 32-bit halves of operands. You must load 64-bit data in adjacent pairs of 32-bit registers as shown in Figure 18. The 32-bit most significant word (MSW) will be in one even register and the 32-bit least significant word (LSW) in its odd neighbor.

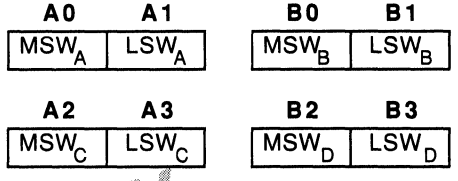


Figure 18. ADSP-3212/ADSP-3222 Operand Storage for Double-Precision Operations

Restrictions on Register Stability
With 64-bit data—as with 32-bit data—registers should not be loaded that are currently in use by the processing elements (i.e., selected by the RDA/B controls). Half the 32-bit registers in any pair of 64-bit operands will loaded on the falling edge with both members of this chipset.

To operate the ALU at full throughput in single-cycle double-precision operations, 64-bit register sets should be alternated every cycle. For example, A_0/A_1 and B_2/B_3 could be loaded with new operands while A_2/A_3 and B_0/B_1 were feeding the computational circuits (and were not changing). In this way, data loading will not disturb the contents of registers in use.

The ADSP-3222 ALU includes two double-precision multicycle operations in its instruction set: IEEE division (30 cycles) and square root (58 cycles). The ADSP-3212 performs a 12-cycle double-precision floating-point division. For double-precision floating-point division, the 64-bit dividend can be stored in either pair of A registers consistent with Figure 18. The divisor can be stored in either pair of B registers, also consistent with Figure 18. Double-precision operands for IEEE square root can be stored in either pair of B registers consistent with Figure 18. Registers containing operands in use must remain unchanged until the end of the operation time.

Data Format Selection – SP and DP Controls (REG)
The three data formats processed by the ADSP-3212/ADSP-3222 chipset are *single-precision* floating-point, *double-precision* floating-point, and *fixed-point*. With the ADSP-3212 Multiplier/Divider, the data format is indicated explicitly by the states of the DP and the SP registered controls:

SP	DP	Data Format Selection
0	0	fixed
0	1	double-precision
1	0	single-precision
1	1	illegal mode

Figure 19. ADSP-3212 Multiplier/Divider Data Format Selection

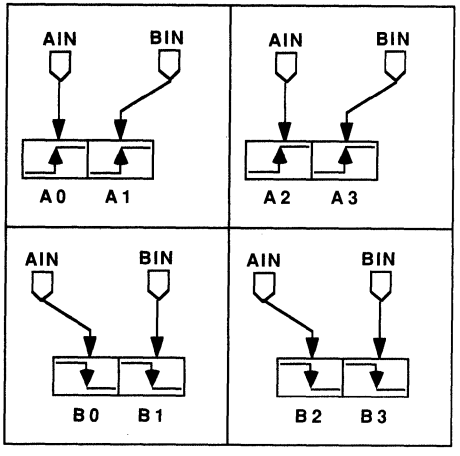


Figure 17. ADSP-3212/ADSP-3222 64-Bit Parallel Load

The state of the SP and DP controls at the rising edge when data is read into the multiplier array determines whether the data is interpreted as single-precision floating-point, double-precision floating-point, or fixed-point. Fixed-point division is not supported; fixed-point numbers should be converted to floating-point format for division and the result converted back to fixed-point format. Division is a multicycle operation (6 cycles for single precision and 12 cycles for double-precision); once initiated, the states of SP and DP don't matter until the next data is read to the processing circuitry.

For the ADSP-3222 ALU, data format selection is implicit in the ALU instruction, I_{8-0}. (See "Instructions and Operations" section below.)

Input Data Register Read Selection – RDA/B Controls (REG)
The register read selection controls, RDA0:1 and RDB0:1, are registered controls which 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.

The data format selected affects the interpretation of the RDA/B controls as follows:

RDA1	RDA0	SP & Fixed: A register selected	DP: A registers selected
0	0	A2	illegal state
0	1	A3	A2, A3
1	0	A0	illegal state
1	1	A1	A0, A1

RDB1	RDB0	SP & Fixed: B register selected	DP: B registers selected
0	0	B2	illegal state
0	1	B3	B2, B3
1	0	B0	illegal state
1	1	B1	B0, B1

Figure 20. ADSP-3212/ADSP-3222 Input Register Read Selection

Note that when feedforward is activated, the definitions of the RDA/B controls change. See "Feedback and Feedforward," below.

After the initiation of multicycle operations, the RDA/B controls are ignored.

Feedback and Feedforward – FDBK Controls (REG)
The ADSP-3212/ADSP-3222 have feedback paths to A_2, A_3, B_2, and B_3 and feedforward paths from all registers to the output register. The feedback controls FDBK0:1 determine whether the device is in normal operation, whether the feedback data goes to the A registers or the B registers, and whether feedforward is activated, as shown below:

FDBK1	FDBK0	Interpretation
0	0	normal operation
0	1	feedforward
1	0	feedback to A reg
1	1	feedback to B reg

Figure 21. Feedback and Feedforward Controls

These controls are pipelined for one cycle; that is, they take effect at the *next* rising clock edge from the one at which they are presented. For feedback operations from the 64-bit result (before the output register), each register to receive data must also be selected for loading in the usual way, by asserting the corresponding SELA or SELB; thus, you would assert the FDBK controls in one cycle and the SELA or SELB controls in the next cycle. For feedback, all input registers are loaded in parallel on the rising clock edge.

Both SP and DP feedback transfers are supported. In DP feedback transfers, the MSW is written to A_2 or B_2, and the LSW is written to A_3 or B_3. In SP feedback transfers, the 32-bit result is written to A_2 or B_2; A_3 or B_3 should not be selected in this case. The registers in the bank *not* selected by FDBK0:1 can be loaded concurrently in the normal manner. Also, the low-order (0 and 1) registers in the selected bank can be loaded concurrently in the normal manner. You should not, however, load registers intended to receive feedback data in the cycle before the feedback data is written (the same cycle in which you assert the FDBK controls). In theory, such an action would serve no purpose, because the newly loaded data would be overwritten by the fed-back data before it could be passed to the processing circuits; in practice, the data load will actually inhibit the feedback.

When feedforward is selected, a pair of input registers will be fed directly to the output port in the same cycle. RDA0 determines whether A registers (HI) or B registers (LO) are fed forward. The pair is selected by the register read selection controls, RDA1/B1, as shown in Figure 22.

RDA0	RDA1	RDB1	Feedforward Registers
0	X	0	B2/B3
0	X	1	B0/B1
1	0	X	A2/A3
1	1	X	A0/A1

Figure 22. Feedforward Register Selection

The pair fedforward must have been in input registers from a previous cycle but will reach the output register on the rising edge following the cycle in which the FDBK controls are set up for a feedforward operation (and the read selection controls are also set up). There is no cycle time or output delay penalty for feedforward operations.

For normal operation, FDBK0:1 must both be LO. Feedback and feedforward timing is shown in Figures T5 and T6 in the Timing section.

Absolute Value – ABSA/B Controls (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-3222 ALU, the ABSA/B controls are effective with most fixed-point and all single-precision and double-precision operations. If the ABSA/B controls are asserted in logical operations, the results will be undefined.

For the ADSP-3212 Multiplier/Divider, the absolute value operation is available on single-precision and double-precision

floating-point operands only. If the ABSA/B controls are asserted with a multiplier/divider for a fixed-point operation, the results will be undefined.

Wrapped Input – WRAPA/B Controls (REG) (and INEXIN and RNDCARI on the ADSP-3222)
The ADSP-3212 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/divider 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 the multiplier/divider that a wrapped number has been selected as an operand (RDA/B controls) to the multiplier/divider array. WRAPA indicates (HI) that the selected A register contains a wrapped number; WRAPB, that the selected B register contains a wrapped number.

The ALU in general operates directly on denormals and hence doesn't need a similar set of controls. However, for IEEE division and square root operations, the ALU cannot operate directly on denormals. Like the multiplier/divider, 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 a multiplier/divider. To indicated that either the divisor in a B register or a square root operand in a B register is 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 – TCA/B Controls (REG)
The registered ADSP-3212's Twos-Complement Input Controls inform the multiplier/divider 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-3212/ADSP-3222 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-3212/ADSP-3222 chipset is controlled by a pair of pipelined, registered round controls, RND0:1. They should be set up 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 as:

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 23. Round Controls

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." (Standard 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.

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." (Standard 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." (Standard 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." (Standard 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 + 1/2 LSB 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 9, 10, 12, and 13) as the destination format.

The four rounding modes are illustrated by number lines in Figure 24. 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.

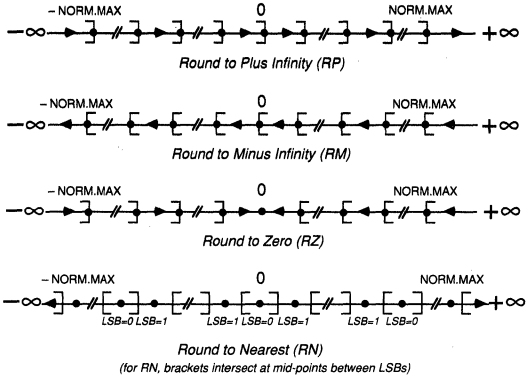


Figure 24. IEEE Rounding Modes

Status Flags

The ADSP-3212/ADSP-3222 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 Standard 754 of non-trapping exceptions. The ADSP-3222 also generates a ZERO flag to indicate that the result of the ALU operation is ZERO.

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

In addition to the IEEE status flags, the ADSP-3212 Multiplier/Divider 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/divider 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. For both multiplication and division, DENORM goes HI during the cycle after a denormal was read into the array (with the RDA/B controls). See Figures T7 through T10. In the cycle following the assertion of DENORM, the INEXO status flag (see "Inexact," below) indicates which operand was denormal; INEXO is HI if the B operand or both operands were denormal and LO if only the A operand was denormal. 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/divider will produce anyway—so there is no need to signal an exception. Any finite nonzero number multiplied by INF should yield INF, and in the IEEE mode, the ADSP-3212 Multiplier/Divider will produce this result with a DNRM operand, hence no wrapping is required. In FAST mode, DNRM is treated as ZERO, so a NAN results, and no wrapping is needed. And multiplication of any number by a NAN produces a NAN (and the INVALOP flag); no wrapping is necessary for the multiplier/divider to produce this correct IEEE result.

Similarly, divisions that have a denormal operand and ZERO, INF, or NAN as the other operand do not require wrapping and do not cause the assertion of the DENORM flag on the multiplier/divider (or the INVALOP and UNDFLO combination on the ALU, which flags a denormal operand for division or square root). Zero divided by a finite number always yields zero, so in IEEE mode, ZERO ÷ DNRM yields ZERO without signalling an exception. DNRM ÷ ZERO results in INF, because any finite nonzero number divided by zero should yield INF. In FAST mode, DNRM is treated as ZERO, so ZERO ÷ DNRM and DNRM ÷ ZERO both yield a NAN (and INVALOP). Any finite number divided by INF should yield ZERO, and INF divided by any finite number should yield INF. In both IEEE and FAST modes, INF ÷ DNRM results in INF and DNRM ÷ INF results in ZERO, without generating any flags. Division of any number by a NAN or division of a NAN by any number produces a NAN (and the INVALOP flag); therefore, the multiplier/divider and the ALU generate this result without flagging a denormal and without wrapping.

Note that the ALU in general operates directly on denormals and therefore does not flag any exception. However, it 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 LO indicates that the dividend is a DNRM; INEXO HI 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.

Invalid Operation and NAN results

INVALOP is generated whenever attempting to execute an invalid operation, as defined in Standard 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 (on the ADSP-3222). The default non-trapping result is required to be a quiet NAN. Except when passing a NAN with PASS or copying a sign bit to a NAN in the ALU, a NAN output will always have an exponent and fraction of all ones.

Conditions that cause the assertion of INVALOP are:

- NAN input read to computational circuitry (except for logical PASS); INEXO indicates the B operand is a NAN
- Multiplication of either ±INF by either ±ZERO
- In FAST mode, multiplication of either ±INF by either ±DNRM or ±ZERO
- 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. Division by ±ZERO yields INF rather than a NAN (see "Division-by-Zero," below).
- Attempting the square root of a negative number
- In conjunction with OVRFLO, the Division-by-Zero exception
- On the ADSP-3222 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 (Standard 754, Section 7.2). The Division-by-Zero exception is indicated by the simultaneous assertion of both OVRFLO and INVALOP. The result is a correctly signed INF.

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 Standard 754, Section 7.3. Note that the overflow condition can occur both during computations and during data format conversions. The result in IEEE or FAST mode 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
- 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.

Underflow

Underflow is defined in four ways in Standard 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-3212/ADSP-3222 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 the chipset is operating in IEEE or FAST mode, whether the result is generated by a multiplier/divider or an ALU and whether the operation is division.

With the ADSP-3212 Multiplier/Divider, UNDFLO is generated whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format. 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-3222 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, nonzero 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-3222 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. The data result will be in the wrapped format. Note that $\sqrt{(DNRM.MIN)} \geq NORM.MIN$. 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-3212 Multiplier/Divider, whenever the unbounded, post-rounded, nonzero result is of lesser magnitude than NORM.MIN in the destination format
- With the ADSP-3222 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-3222 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
- Conversions from DP floating-point to SP floating-point whenever the unbounded, post-rounded, nonzero result is less than SP DNRM.MIN or whenever an inexact denormal is produced
- 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
- With the ADSP-3222 ALU in IEEE mode, 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 Standard 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-3212/ADSP-3222 chipset by INEXO.

For fixed-point multiplications, the ADSP-3212 Multiplier/Divider will assert INEXO HI if and only if any of the least significant 32 bits of the pre-rounded 64-bit product are ones.

INEXO is also used for signaling various other conditions. If the ADSP-3212 or the ADSP-3222 detects a NAN input to a floating-point operation, it asserts INVALOP. At the same time, it asserts INEXO if the B operand or both operands are NANs. If the ADSP-3222 detects a denormal input to division, it asserts UNDFLO and INVALOP to flag the denormal and asserts INEXO if the B operand or both operands are denormals. Similarly, the ADSP-3212 asserts DENORM to flag a denormal input to floating-point multiplication or division. INEXO, in that context, signals which of the two was the denormal: INEXO HI indicates that the B operand or both operands are denormals; INEXO LO indicates that only the A operand is a denormal.

If you convert a fixed-point number in the B input to floating-point format and the fixed-point number has the same bit pattern as a floating-point NAN (i.e., bits 23-30 are all ones and at least one of bits 0-22 is a one), INEXO is asserted. This occurs only in the ALU's DFLOATB and SFLOATB instructions, not DFLOATA or SFLOATA. These instructions are described in *Floating-Point ALU Operations*.

Conditions that cause the assertion of INEXO are:

- Loss of accuracy when fitting result to destination format
- For fixed-point multiplications, the pre-rounded 64-bit product contains ones in the least-significant 32-bits
- For floating-point operations, in conjunction with INVALOP, the B operand is a NAN
- In IEEE mode on the ADSP-3222, in conjunction with UNDFLO and INVALOP, dividend is a denormal (LO) or divisor is a denormal or both are denormals (HI)
- In IEEE mode on the ADSP-3212, in conjunction with DENORM, A operand is a denormal (LO) or B operand is a denormal or both are denormals (HI).

Zero

The ADSP-3222 has a dedicated ZERO flag. ZERO goes HI whenever the post-rounded result in the output register is ±ZERO, except for a few floating-point instructions. Thus, inexact results that round to ±ZERO are flagged, as well as ±ZERO results from format conversions and logical operations. The instructions for which ZERO is not valid are: SXSUB, DXSUB (exponent subtract), SITRN, DITRN (logical downshift), SFIXA, DFIXA, SFIXB and DFIXB (floating-point to fixed-point conversion). For these instructions, the state of ZERO is undefined.

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 Standard 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 XIV and XV.

In IEEE comparisons, the data types are always ordered in ascending sequence: −INF, −NORM, −DNRM, 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-3212 generates a Round Carry Propagation Out flag, RNDCARO, that indicates whether or not a carry bit propagated into the destination format's fraction during the multiplier/divider's floating-point rounding operation. The rounding

that the multiplier/divider does in creating the wrapped or unnormal result may cause a carry bit into the LSB in the destination's format's fraction. This rounding position will not in general be correct for a properly rounded denormal. Thus, when the underflowed multiplier/divider result is unwrapped to a denormal, the ALU has to undo the multiplier/divider's 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/divider's rounding operation propagated into the fraction of the result. This information is provided in the multiplier/divider's RNDCARO flag. The ALU also needs to know if the multiplier/divider's rounded result caused a loss of accuracy when expressed in its destination wrapped format, indicated by the multiplier/divider's Inexact Result (INEXO) flag.

The ADSP-3222 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 WRAP 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/divider and ALU must be using the same rounding mode.

The ADSP-3222 ALU itself generates WRAPs in underflowed division operations. These WRAPs must be fed back to the ALU to be unwrapped to DNRMs. The ADSP-3222, unlike the multiplier/divider, does not have a RNDCARO pin to signal whether or not a carry bit propagated into the destination format on rounding. For this reason, WRAPs produced by the ADSP-3222 ALU in division are rounded differently than they are on the multiplier/divider; underflowed (only) quotients are always truncated (Round-toward-Zero) to the destination wrapped format. Hence there *is* no carry bit propagation. When unwrapping a WRAP quotient produced by the ALU, RNDCARI should always be held LO. INEXIN should reflect the status of INEXO when the ALU produced the underflowed wrapped quotient.

The ADSP-3222 ALU also uses the RNDCARI and INEXIN pins to indicated wrapped A and B operands, respectively, to ALU division and square root operations. Both RNDCARI and INEXIN should be held LO except for unwrap, ALU division, and square root operations.

Instructions and Operations

The ADSP-3212 multiplier/divider executes one of two instructions: multiply or divide. If the DIVMUL input is LO, the ADSP-3212 multiplies; if DIVMUL is HI, the ADSP-3212 divides. The instruction need not be specified explicitly in microcode. The data format of results and status flags from multiplication are shown in Tables V and VI. Format and status for division are shown in Tables VII and VIII.

A denormal input operand will generally cause the DENORM exception (see "Status Flags" above) unless the other operand is ZERO, INF or a NAN. (See Tables V through VIII.) 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-3212 Multiplier/Divider, the sign of a NAN result will be the exclusive OR of the signs of the input operands.

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 INEXO
DNRM	ZERO		ZERO	DENORM	ZERO	DENORM	ZERO	DENORM	INF		NAN	INVALOP INEXO
WRAP	ZERO		ZERO	DENORM	UNRM	UNDFLO	NORM WRAP UNRM	UNDFLO UNDFLO	INF		NAN	INVALOP INEXO
NORM	ZERO		ZERO	DENORM	NORM WRAP UNRM	UNDFLO UNDFLO	INF,NORM,MAX[1] NORM WRAP	OVRFLO UNDFLO	INF		NAN	INVALOP INEXO
INF	NAN	INVALOP	INF		INF		INF		INF		NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

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

Table V. ADSP-3212 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 INEXO
DNRM	ZERO		ZERO	DENORM	ZERO	DENORM	NAN	INVALOP	NAN	INVALOP INEXO
NORM	ZERO		ZERO	DENORM	INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	INF		NAN	INVALOP INEXO
INF	NAN	INVALOP	NAN	INVALOP	INF		INF		NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

In FAST mode, WRAP inputs are illegal.

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

Table VI. ADSP-3212 Floating-Point Multiplication (FAST Mode)

The product of INF with anything except ZERO or NAN is a correctly signed INF. INF•ZERO will cause INVALOP and yield a NAN. A NAN operand in either multiplication or division will also cause INVALOP and yield a NAN. If the NAN is the B operand, INEXO is also asserted.

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 INEXO
DNRM	INF	INVALOP OVRFLO	NAN	DENORM	NAN	DENORM	NAN	DENORM	ZERO		NAN	INVALOP INEXO
WRAP	INF	INVALOP OVRFLO	NAN	DENORM	NORM		NORM WRAP	UNDFLO	ZERO		NAN	INVALOP INEXO
NORM	INF	INVALOP OVRFLO	NAN	DENORM	INF, NORM.MAX[1] NORM	OVRFLO	INF,NORM.MAX[1] NORM WRAP	OVRFLO UNDFLO	ZERO		NAN	INVALOP INEXO
INF	INF		INF		INF		INF		NAN	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

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

Table VII. ADSP-3212 Floating-Point Division (A÷B) (IEEE Mode)

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 INEXO
DNRM	NAN	INVALOP	NAN	INVALOP DENORM	ZERO	DENORM	ZERO		NAN	INVALOP INEXO
NORM	INF	INVALOP OVRFLO	INF	OVRFLO INVALOP DENORM	INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	ZERO		NAN	INVALOP INEXO
INF	INF		INF		INF		NAN	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

In FAST mode, WRAP inputs are illegal.

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

Table VIII. ADSP-3212 Floating-Point Division (A÷B) (FAST Mode)

The ADSP-3222 ALU, in contrast to the multiplier/divider, is instruction driven with the operation specified by I_{8-0}. The ALU instructions fall into five categories: Fixed-Point, Logical, Single-Precision Floating-Point, Double-Precision Floating-Point and Miscellaneous. Instructions are summarized in Tables IX through XIII and described in this section below. The data format of results and status flags from the various ALU operations are shown in Tables XIV and XV. Division is shown in Tables XIX and XX; square root in Table XXI. Conversions are illustrated in Tables XVI, XVII, and XVIII.

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|
ISUBAAS	001	100	111	Fixed-point |B – A|

Table IX. ADSP-3222 Fixed-Point ALU Operations

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 Register contents.
CLR	100	000	000	Clear all status flags. Data register contents are unaffected.

Table X. ADSP-3222 ALU Logical Operations

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
DOUBLEA	011	101	101	Convert SP FltgPt A to DP FltgPt
DOUBLEB	011	101	110	Convert SP FltgPt B to DP 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 WRAP
SWRAPB	011	100	010	Wrap SP DNRM B to SP WRAP
SUNWRAPA	011	010	001	Unwrap SP WRAP A to SP DNRM
SUNWRAPB	011	010	010	Unwrap SP WRAP 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 is ≥ +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.
SDIV	011	110	111	SP FltgPt (A÷B)
SSQR	111	110	110	SP FltgPt $\sqrt{B}$

Table XI. ADSP-3222 ALU Single-Precision Floating-Point Operations

4

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
DADD	110	000	011	DP FltgPt (A + B)
DSUBB	110	000	111	DP FltgPt (A – B)
DSUBA	110	001	011	DP FltgPt (B – A)
DCOMP	110	001	111	DP 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
DADDAS	010	000	011	DP FltgPt |A + B|
DSUBBAS	010	000	111	DP FltgPt |A – B|
DSUBAAS	010	001	011	DP FltgPt |B – A|
DFIXA	010	011	101	Convert DP FltgPt A to twos-complement integer DFIXB
	010	011	110	Convert DP FltgPt B to twos-complement integer
DFLOATA	010	100	101	Convert twos-complement integer A (even A register sources only) to DP FltgPt
DFLOATB	010	100	110	Convert twos-complement integer B (even B register sources only) to DP FltgPt
SINGLEA	110	011	101	Convert DP FltgPt A to SP FltgPt
SINGLEB	110	011	110	Convert DP FltgPt B to SP FltgPt
DPASSA	010	110	001	Pass DP FltgPt A. NANs cause INVALOP.
DPASSB	010	110	010	Pass DP FltgPt B. NANs cause INVALOP.
DWRAPA	010	100	001	Wrap DP DNRM A to DP WRAP
DWRAPB	010	100	010	Wrap DP DNRM B to DP WRAP
DUNWRAPA	010	010	001	Unwrap DP WRAP A to DP DNRM
DUNWRAPB	010	010	010	Unwrap DP WRAP B to DP DNRM
DSIGN	010	111	101	Copy sign from DP FltgPt B to DP FltgPt A. Result is [sign B, exponent A, fraction A].
DXSUB	010	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 is ≥ +1024, INF results. If the unbiased exponent is ≤ −1023, ZERO results.
DITRN	010	010	101	Downshift DP FltgPt A mantissa (with hidden bit) logically by the unbiased DP FltgPt B exponent to a 32-bit unsigned-magnitude integer. Use RZ only.
DDIV	010	110	111	DP FltgPt (A÷B)
DSQR	110	110	110	DP FltgPt √B

Table XII. ADSP-3222 ALU Double-Precision Floating-Point Operations

Mnemonic	Instruction (I_{8-0})			Description
	I_{8-6}	I_{5-3}	I_{2-0}	
HOLDEN	100	000	100	Redefine FAST to $\overline{\text{HOLD}}$ control
LOAD64	100	001	100	Enable 64-Bit parallel data loading

Table XIII. ADSP-3222 ALU Miscellaneous 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) 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-3222's fixed-point 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-3222. 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.

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. Both the status flags and the output register will be preserved during NOP. 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 single-precision and double-precision floating-point operations are exactly analogous and both will be discussed here. The data types and flags resulting from additions, subtractions, comparisons, absolute sums, and absolute differences are shown in Tables XIV and XV. 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.

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 INEXO
DNRM	DNRM		NORM DNRM ZERO		INF,NORM,MAX[1] NORM DNRM	OVRFLO	INF		NAN	INVALOP INEXO
NORM	NORM		INF,NORM,MAX[1] NORM DNRM	OVRFLO	INF,NORM,MAX[1] NORM DNRM ZERO	OVRFLO	INF		NAN	INVALOP INEXO
INF	INF		INF		INF		INF[3] NAN[3]	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

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. If DNRM result is inexact, UNDFLO will be set.
5. The ADSP-3222 does not accept wrapped numbers as inputs for standard arithmetic operations.

Table XIV. ADSP-3222 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	UNDFLO	NORM		INF		NAN	INVALOP INEXO
DNRM	ZERO	UNDFLO	NORM ZERO		INF,NORM,MAX[1] NORM ZERO	OVRFLO UNDFLO	INF		NAN	INVALOP INEXO
NORM	NORM		INF,NORM,MAX[1] NORM ZERO	OVRFLO UNDFLO	INF,NORM,MAX[1] NORM ZERO ZERO[4]	OVRFLO UNDFLO	INF		NAN	INVALOP INEXO
INF	INF		INF		INF		INF[3] NAN[3]	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

In FAST mode, WRAP inputs are illegal.

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 XV. ADSP-3222 Floating-Point Addition/Subtraction (FAST Mode)

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:

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

In this notation, the expression to the left of the equals sign in the first line refers to (+INF) + (+INF) or to (−INF) + (−INF). The leading expressions on the second and third lines refer to (+INF) + (−INF) or to (−INF) + (+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 ALUs 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 and DFIXA/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 both convert to an all-ones twos-complement integer with the sign bit preserved, either full-scale positive (for +NAN or +INF) or −1 (for −NAN or −INF). INVALOP will be asserted. See Tables XVI and XVII for illustrations of fixing single- and double-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 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].

For conversion from twos-complement integer to double-precision floating-point (DFLOATA/B), the numerical result will always be an IEEE normal. No flags will be set. Only even-numbered registers (A_0, A_2, B_0 or B_2) can be sources for the DFLOAT operation.

If you convert a fixed-point number in the B input to floating-point format and the fixed-point number has the same bit pattern as a floating-point NAN (i.e., Bits 23-30 are all ones and at least one of Bits 0-22 is a one), INEXO will be asserted. This occurs only in the ALU's DFLOATB and SFLOATB instructions, not DFLOATA or SFLOATA.

Conversions: Floating to Floating
For conversion from single-precision to double-precision (DOUBLEA/B), all single-precision normals and denormals will convert without exceptions. A single-precision NAN will convert to a double-precision all-ones NAN; the INVALOP flag will be set. Single-precision INF converts to double-precision INF; no flags are set. Single-precision ZERO converts to double-precision ZERO; no flags are set.

Conversions from double-precision to single-precision floating-point (SINGLEA/B) can cause exceptions because overflow, underflow and inexact status can result in mapping to the smaller destination format. See Table XVIII for illustrations. A double-precision NAN will convert to a single-precision all-ones NAN; the INVALOP flag will be set. DP INFs convert to SP INFs; no flags are set. Finite numbers greater in magnitude than single-precision NORM.MAX will result in an OVRFLO flag and SP INF or SP NORM.MAX , depending on the rounding mode, (see "Rounding – RND Controls" above). Nonzero, post-rounded operands whose magnitudes are between SP NORM.MAX and SP NORM.MIN inclusive will be SP NORMs. In IEEE mode, operands between SP DNRM.MAX and SP DNRM.MIN inclusive will be SP DNRMs, whereas in FAST mode, the result is ZERO with UNDFLO and INEXO flags set.

For both normals and denormals, INEXO will be asserted if the conversion from double-precision to single-precision floating-point is inexact. If the conversion to denormals is inexact, both INEXO and UNDFLO will be set, in accordance with the IEEE definition in terms of loss of accuracy when representing a denormal (see "Underflow" in "Status Flags" above). Post-rounded, nonzero numbers less than SP DNRM.MIN will convert to ZERO; UNDFLO and INEXO will be set. DP ZERO converts to SP ZERO without exception.

Pass
Pass instructions (SPASSA/B and DPASSA/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-

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

Sign	HB	f51 . . . f22	f21	f20	f19 . . . f1	f0		Unbiased Expnt	Source Name	Sign	i30 . . . i1	i0	Rounding Modes	Status Flags
0	1	X . . . X	X	X	X . . . X	X	2**	1024	+NAN	0	1 . . . 1	1	all	INVALOP
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	+INF	0	1 . . . 1	1	all	INVALOP
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	31		U*	U . . . U	U	all	OVRFLO
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	30		U	U . . . U	U	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	30		0	1 . . . 1	1	RZ,RM	INEXO
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	30		U	U . . . U	U	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	30		0	1 . . . 1	1	RZ,RM	INEXO
0	1	1 . . . 1	0	1	1 . . . 1	1	2**	30		U	U . . . U	U	RP	OVRFLO,INEXO
0	1	1 . . . 1	0	1	1 . . . 1	1	2**	30		0	1 . . . 1	1	RM,RN,RZ	INEXO
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	30		U	U . . . U	U	RP	OVRFLO,INEXO
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	30		0	1 . . . 1	1	RM,RN,RZ	INEXO
0	1	1 . . . 1	0	0	0 . . . 0	0	2**	30		0	1 . . . 1	1	all	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	0	one	0	0 . . . 0	1	all	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	one − 1LSB	0	0 . . . 0	1	RN,RP	UNDFLO,INEXO
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	one − 1LSB	0	0 . . . 0	0	RZ,RM	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	1/2 + 1LSB	0	0 . . . 0	1	RN,RP	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	1/2 + 1LSB	0	0 . . . 0	0	RZ,RM	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	1/2	0	0 . . . 0	1	RP	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	1/2	0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN	0	0 . . . 0	1	RP	UNDFLO,INEXO
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN	0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DENORM.MIN	0	0 . . . 0	1	RP	UNDFLO,INEXO
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DENORM.MIN	0	0 . . . 0	0	RM,RN,RZ	UNDFLO,INEXO
0	0	0 . . . 0	0	0	0 . . . 0	0		0	+ZERO	0	0 . . . 0	0	all	
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DENORM.MIN	1	1 . . . 1	1	RM	UNDFLO,INEXO
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DENORM.MIN	0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN	1	1 . . . 1	1	RM	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN	0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	−1/2	1	1 . . . 1	1	RM	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1	−1/2	0	0 . . . 0	0	RP,RN,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	−1/2 − 1LSB	1	1 . . . 1	1	RM,RN	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	−1	−1/2 − 1LSB	0	0 . . . 0	0	RP,RZ	UNDFLO,INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	−one + 1LSB	1	1 . . . 1	1	RM,RN	UNDFLO,INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−1	−one + 1LSB	0	0 . . . 0	0	RP,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	0	−one	1	1 . . . 1	1	all	
1	1	1 . . . 1	0	0	0 . . . 0	0	2**	30		1	0 . . . 0	1	all	
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	30		1	0 . . . 0	0	RM	INEXO
1	1	1 . . . 1	0	0	0 . . . 0	1	2**	30		1	0 . . . 0	1	RP,RN,RZ	INEXO
1	1	1 . . . 1	0	1	1 . . . 1	1	2**	30		1	0 . . . 0	0	RM	INEXO
1	1	1 . . . 1	0	1	1 . . . 1	1	2**	30		1	0 . . . 0	1	RP,RN,RZ	INEXO
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	30		1	0 . . . 0	0	RM,RN	INEXO
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	30		1	0 . . . 0	1	RP,RZ	INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	30		1	0 . . . 0	0	RM,RN	INEXO
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	30		1	0 . . . 0	1	RP,RZ	INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	31		1	0 . . . 0	0	all	
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	31		1	0 . . . 0	0	RP,RN,RZ	INEXO
1	1	0 . . . 0	0	0	0 . . . 0	1	2**	31		U	U . . . U	U	RM	OVRFLO,INEXO
1	1	0 . . . 0	0	0	1 . . . 0	0	2**	31		1	0 . . . 0	0	RP,RZ	INEXO
1	1	0 . . . 0	0	1	0 . . . 0	0	2**	31		U	U . . . U	U	RM,RN	OVRFLO,INEXO
1	1	0 . . . 0	0	1	1 . . . 1	1	2**	31		1	0 . . . 0	0	RP,RZ	INEXO
1	1	0 . . . 0	0	1	1 . . . 1	1	2**	31		U	U . . . U	U	RM,RN	OVRFLO,INEXO
1	1	0 . . . 0	1	0	0 . . . 0	0	2**	31		U	U . . . U	U	all	OVRFLO
1	1	0 . . . 0	1	0	0 . . . 0	1	2**	31		U	U . . . U	U	all	OVRFLO,INEXO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	32		U	U . . . U	U	all	OVRFLO
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	−INF	1	1 . . . 1	1	all	INVALOP
1	1	X . . . X	X	X	X . . . X	X	2**	1024	−NAN	1	1 . . . 1	1	all	INVALOP

*"U" denotes an undefined result.

NOTE: Heavy line indicates rounding boundary in source.

Table XVII. Conversion of 64-Bit Double-Precision Floating-Point to 32-Bit Twos-Complement Integer

Sign	HB	f51 . . . f30	f29	f28	f27 . . . f1	f0		Unbiased Expnt	Source Name
0	1	X . . . X	X	X	X . . . X	X	2**	1024	+NAN
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	+INF
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	+NORM.MAX
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	+NORM.MAX
0	1	1 . . . 1	1	1	0 . . . 0	0	2**	127	
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	127	
0	1	1 . . . 1	1	0	0 . . . 0	1	2**	127	
0	1	1 . . . 1	1	0	0 . . . 0	1	2**	127	
0	1	1 . . . 1	1	0	0 . . . 0	0	2**	127	
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	127	
0	1	1 . . . 1	0	0	0 . . . 0	1	2**	127	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−126	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127	
0	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127	
0	1	1 . . . 1	0	0	0 . . . 0	0	2**	−127	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−149	
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN
0	1	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	+NORM.MIN
0	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	+DNRM.MAX
0	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	+DNRM.MAX
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DNRM.MIN
0	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	+DNRM.MIN
0	0	0 . . . 0	0	0	0 . . . 0	0		0	+ZERO
1	0	0 . . . 0	0	0	0 . . . 0	0		0	−ZERO
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DNRM.MIN
1	0	0 . . . 0	0	0	0 . . . 0	1	2**	−1022	−DNRM.MIN
1	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	−DNRM.MAX
1	0	1 . . . 1	1	1	1 . . . 1	1	2**	−1022	−DNRM.MAX
1	0	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN
1	0	0 . . . 0	0	0	0 . . . 0	0	2**	−1022	−NORM.MIN
1	0	0 . . . 0	0	0	0 . . . 0	0	2**	−149	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	−127	
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	−126	
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	127	
1	1	1 . . . 1	1	0	0 . . . 0	1	2**	127	
1	1	1 . . . 1	1	0	0 . . . 0	0	2**	127	
1	1	1 . . . 1	1	0	0 . . . 0	1	2**	127	
1	1	1 . . . 1	1	0	0 . . . 0	1	2**	127	
1	1	1 . . . 1	1	1	0 . . . 0	0	2**	127	
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	−NORM.MAX
1	1	1 . . . 1	1	1	1 . . . 1	1	2**	1023	−NORM.MAX
1	1	0 . . . 0	0	0	0 . . . 0	0	2**	1024	−INF
1	1	X . . . X	X	X	X . . . X	X	2**	1024	−NAN

Sign	HB	f22 . . . f1	f0		Unbiased Expnt	Result Name	Rounding Modes	Status Flags
0	1	1 . . . 1	1	2**	128	+NAN	all	INVALOP
0	1	0 . . . 0	0	2**	128	+INF	all	
0	1	0 . . . 0	0	2**	128	+INF	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	2**	127	+NORM.MAX	RZ,RM	OVRFLO,INEXO
0	1	0 . . . 0	0	2**	128	+INF	RP,RN	OVRFLO,INEXO
0	1	1 . . . 1	1	2**	127	+NORM.MAX	RZ,RM	INEXO
0	1	0 . . . 0	0	2**	128	+INF	RP	OVRFLO,INEXO
0	1	1 . . . 1	1	2**	127	+NORM.MAX	RM,RN,RZ	INEXO
0	1	1 . . . 1	1	2**	127	+NORM.MAX	all	
0	1	1 . . . 1	1	2**	127	+NORM.MAX	RP	INEXO
0	1	1 . . . 1	0	2**	127		RM,RN,RZ	INEXO
0	1	0 . . . 0	0	2**	−126	+NORM.MIN	all	
0	0	1 . . . 1	1	2**	−126	+NORM.MIN	RP,RN	INEXO
0	0	1 . . . 1	1	2**	−126	+DNRM.MAX	RZ,RM	UNDFLO,INEXO
0	0	1 . . . 1	1	2**	−126	+DNRM.MAX	all	
0	0	0 . . . 0	0	2**	−126	+DNRM.MIN	all	
0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	0	0 . . . 0	0	2**	−126	+ZERO	RM,RN,RZ	UNDFLO, INEXO
0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	0	0 . . . 0	1	2**	−126	+DNRM.MIN	RP	UNDFLO, INEXO
0	0	0 . . . 0	0	2**	−126	+ZERO	RM,RN,RZ	UNDFLO, INEXO
0	0	0 . . . 0	0		0	+ZERO	all	
1	0	0 . . . 0	0		0	−ZERO	all	
1	0	0 . . . 0	1	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	0	0 . . . 0	0	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	0	1 . . . 1	1	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	0	1 . . . 1	1	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	0	0 . . . 0	0	2**	−126	−DNRM.MIN	RM	UNDFLO, INEXO
1	0	0 . . . 0	0	2**	−126	−ZERO	RP,RN,RZ	UNDFLO, INEXO
1	1	0 . . . 0	1	2**	−126	−DNRM.MIN	all	
1	1	1 . . . 1	1	2**	−126	−DNRM.MAX	all	
1	1	1 . . . 1	1	2**	−126	−NORM.MIN	RM,RN	INEXO
1	1	1 . . . 1	1	2**	−126	−DNRM.MAX	RP,RZ	UNDFLO,INEXO
1	1	0 . . . 0	0	2**	−126	−NORM.MIN	all	
1	1	1 . . . 1	1	2**	127	−NORM.MAX	RM	INEXO
1	1	1 . . . 1	0	2**	127		RP,RN,RZ	INEXO
1	1	1 . . . 1	1	2**	127	−NORM.MAX	all	
1	1	0 . . . 0	0	2**	128	−INF	RM	OVRFLO,INEXO
1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RN,RZ	INEXO
1	1	0 . . . 0	0	2**	128	−INF	RM,RN	OVRFLO,INEXO
1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RZ	INEXO
1	1	0 . . . 0	0	2**	128	−INF	RM,RN	OVRFLO,INEXO
1	1	1 . . . 1	1	2**	127	−NORM.MAX	RP,RZ	OVRFLO,INEXO
1	1	0 . . . 0	0	2**	128	−INF	all	
1	1	1 . . . 1	1	2**	128	−NAN	all	INVALOP

NOTE: Heavy line indicates rounding boundary in source.

Table XVIII. Conversion of 64-Bit Double-Precision Floating-Point to 32-Bit Single-Precision Floating-Point (IEEE Mode)

point pass instructions to reset the unmodified NAN's sign bit to zero.

Wrap

Wrap instructions (SWRAPA/B and DWRAPA/B) convert a denormal to a wrapped number readable by a multiplier/divider or the ADSP-3222 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 (SUNWRAPA/B and DUNWRAPA/B) convert a wrapped number to the IEEE denormal format. After rounding, the result can 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. In unwrapping numbers, UNDFLO and INEXO are set if 1) unwrapping shifts ones out of the DNRM destination format or 2) INEXIN is asserted and no ones are shifted out.

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 in the ALU 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.

The ADSP-3222 determines inexactness by whether or not there was a loss of accuracy when unwrapping the operand supplied to the ALU and also 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. 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-3222 uses this information to determine if the operation creating the wrapped number was inexact. When the ADSP-3222 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 and DSIGN operations copy 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 and DSIGN simplify 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 and DXSUB) 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/DXSUB operations; 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.

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 and DITRN operations. (See Figures 25 and 26.) 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 be a 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/DITRN will cause INVALOP and produce all-ones NANs of the same sign. Round-toward-Zero (RZ) must be specified for SITRN and DITRN. 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.

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-3222 ALU supports multicycle division (SDIV, 16 cycles, and DDIV, 30 cycles) and square root (SSQR, 29 cycles, and DSQR, 58 cycles) operations. (The ADSP-3212 performs faster division; the ADSP-3222 supports division for code-compatibility with the earlier ADSP-3221.) Tables XIX and XX illustrate the resultant data types and status conditions for division. Table XXI serves a similar role for square root. Neither operation can accept denormal inputs directly. The ADSP-3222 will process DNRMs if they are first wrapped to the wrapped format. Otherwise, DNRM 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.

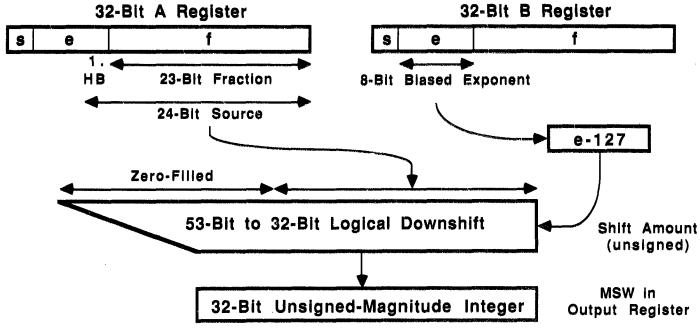


Figure 25. ADSP-3222 SITRN Instruction

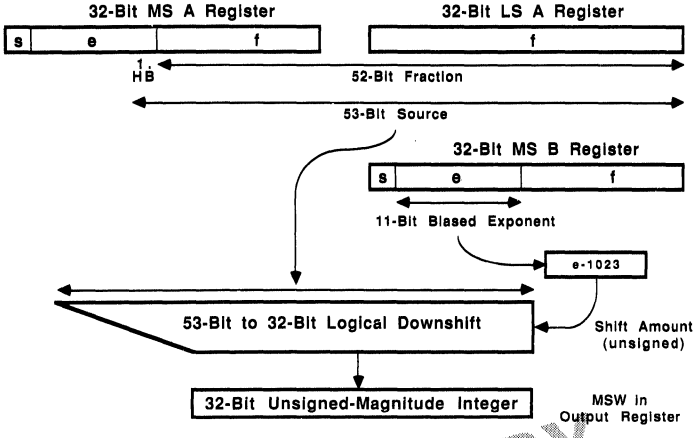


Figure 26. ADSP-3222 DITRN Instruction

4

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 INEXO
DNRM	INF	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	NAN	UNDFLO INVALOP	NAN	UNDFLO INVALOP	ZERO		NAN	INVALOP INEXO
WRAP	INF	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	NORM		NORM WRAP UNRM	UNDFLO UNDFLO	ZERO		NAN	INVALOP INEXO
NORM	INF	OVRFLO& INVALOP	NAN	UNDFLO& INVALOP	INF,[1] NORM.MAX NORM	OVRFLO	INF,[1] NORM.MAX NORM WRAP UNRM	OVRFLO UNDFLO UNDFLO	ZERO		NAN	INVALOP INEXO
INF	INF		INF		INF		INF		NAN	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

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

Table XIX. ADSP-3222 Floating-Point Division (A÷B) (IEEE Mode)

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 INEXO
DNRM	NAN	INVALOP	NAN	INVALOP	ZERO		ZERO		NAN	INVALOP INEXO
NORM	INF	OVRFLO& INVALOP	INF,[1] NORM.MAX	OVRFLO& INVALOP	INF,NORM.MAX[1] NORM ZERO	OVRFLO UNDFLO	ZERO		NAN	INVALOP INEXO
INF	INF		INF		INF		NAN	INVALOP	NAN	INVALOP INEXO
NAN	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP	NAN	INVALOP INEXO

In FAST mode, WRAP inputs are illegal.

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

Table XX. ADSP-3222 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 XXI. ADSP-3222 Floating-Point Square Root ($\sqrt{B}$)

The square root of any nonnegative 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. Note that division using the SDIV and DDIV instructions of the ALU can produce wrapped results that are slightly different than those produced by the multiplier/divider; this difference is explained in the *Special Flags for Unwrapping* section. When the results are unwrapped with the correct flag inputs, the same denormal number is produced.

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 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 $\overline{\text{HOLD}}$ (ASYN)

All members of the ADSP-3212/ADSP-3222 chipset have a 64-bit output register. The output registers are clocked every cycle, except for multicycle operations (division and square root), when $\overline{\text{HOLD}}$ is LO, and when the ADSP-3222 is executing NOP. Output registers are clocked at the conclusion of multicycle operations and not before.

Results appear in the multiplier/divider's output register as follows:

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

Figure 27. ADSP-3212 Multiplier/Divider Output Registers

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/divider 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/divider'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 or divisions. 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 register as follows:

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

Figure 28. ADSP-3222 ALU Output Registers

Both chips 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.

Both chips 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-3212 Multiplier/Divider has a synchronous, active LO control, $\overline{\text{HOLD}}$, that prevents the output register from being updated. The IEEE/FAST pin on the ADSP-3222 ALU can be redefined to a $\overline{\text{HOLD}}$ pin by executing the HOLDEN instruction. (To change the pin back to IEEE/FAST, you must reset the ADSP-3222.) $\overline{\text{HOLD}}$ 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, $\overline{\text{HOLD}}$ must be held HI.

TIMING

Timing diagrams are numbered T1 through T16. Three-state timing for DOUT is shown in 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:

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

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. $\overline{RESET}$ must be LO for at least t_{RS}. In addition, $\overline{RESET}$ 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.

Figure T4 shows IPORT timing. IPORT must be set up at least t_{PS} before each data load (on a rising or a falling clock edge) and be held at least t_{PH} after the data load.

All data, registered and latched controls, and instructions shown in T5 through T16 must be set up t_{DS} before the rising edge and held t_{DH}. 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.

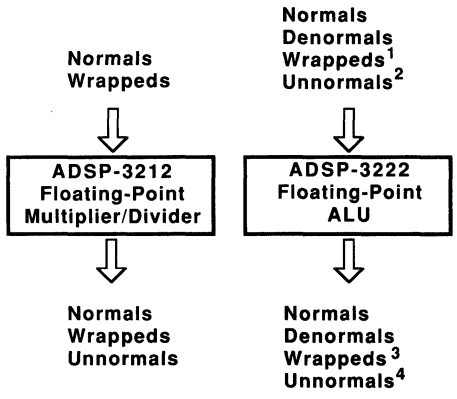
Status flags have a maximum output delay of t_{SO} referenced from the clock rising edge. All status flags except the multiplier/divider's DENORM are available in parallel with their associated output results. DENORM is available earlier to speed up recovery from a denormal input exception. Note that DENORM is LO except in the cycles indicated in Figures T7 through T10.

For all operations (Figures T7 through T16), 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 are shown in Figure 29.

Denormals are detected by the multiplier/divider when read into its processing circuitry. The ADSP-3212 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/divider and present it to an ALU for wrapping.



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

Figure 29. Data Types Directly Supported by the ADSP-3212/3222

The ADSP-3222 ALU will also detect denormals when read into internal circuitry for division or square root operations. The UNDFLO and INVALOP flags will both be asserted on the ADSP-3222 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-3222.)

The ALU wraps denormals with its SWRAP or DWRAP instructions. Note from Tables II and IV 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/divider for multiplication or division or the ADSP-3222 ALU for division or square root. The controlling system must tell the multiplier/divider to interpret the wrapped input as wrapped by asserting WRAPA/B when it is read into the multiplier/divider's processing circuitry. For ALU 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 ALU's 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 V through VIII, XIX, and XX.) Square root on IEEE numbers only produces normals. (See Table XXI.) An underflowed result (wrapped or unnormal) from either multiplier/divider or ALU will be indicated by the UNDFLO flag and must be passed to the ALU for unwrapping. Note that the ALU and the multiplier/divider may produce slightly different wrapped results from the same division operation. When these results are unwrapped with the correct flag inputs, however, they produce the same number. See *Special Flags for Unwrapping* for an explanation of this difference.

For full conformance to the IEEE Standard, all wrapped and unnormal results must be unwrapped in an ALU (with the

SUNWRAP and DUNWRAP instructions) 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 WRAPs 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 WRAPs 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 Standard 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-3212/ADSP-3222 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 Standard 754-1985 for the details of these operations.

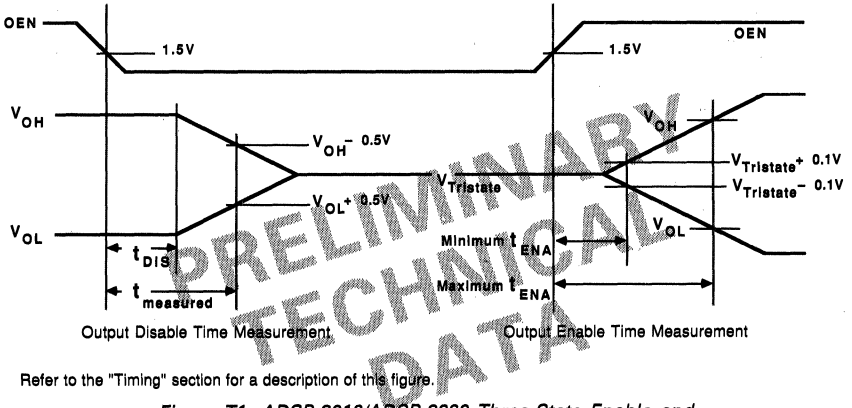


Figure T1. ADSP-3212/ADSP-3222 Three-State Enable and Disable Timing

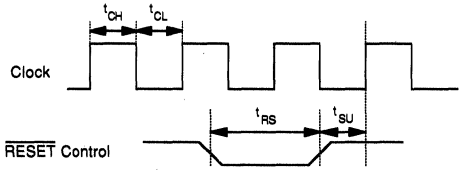


Figure T2. ADSP-3212/ADSP-3222 Reset Timing

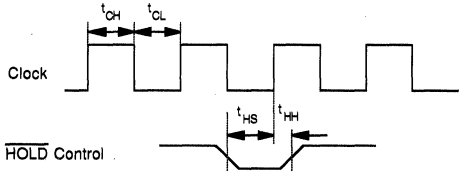


Figure T3. ADSP-3212/ADSP-3222 Output Register Hold Timing

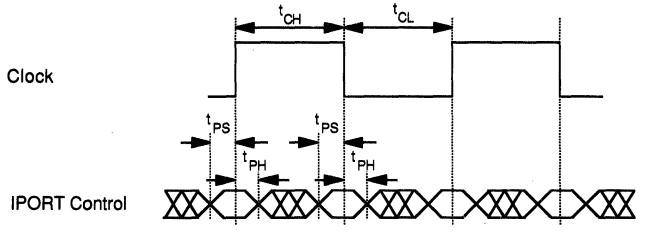


Figure T4. ADSP-3212/ADSP-3222 IPORT Timing

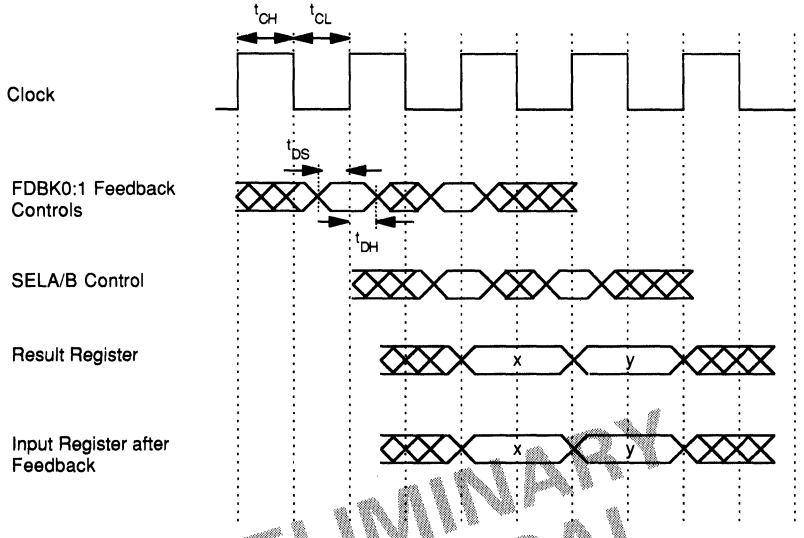


Figure T5. ADSP-3212/ADSP-3222 Feedback Timing

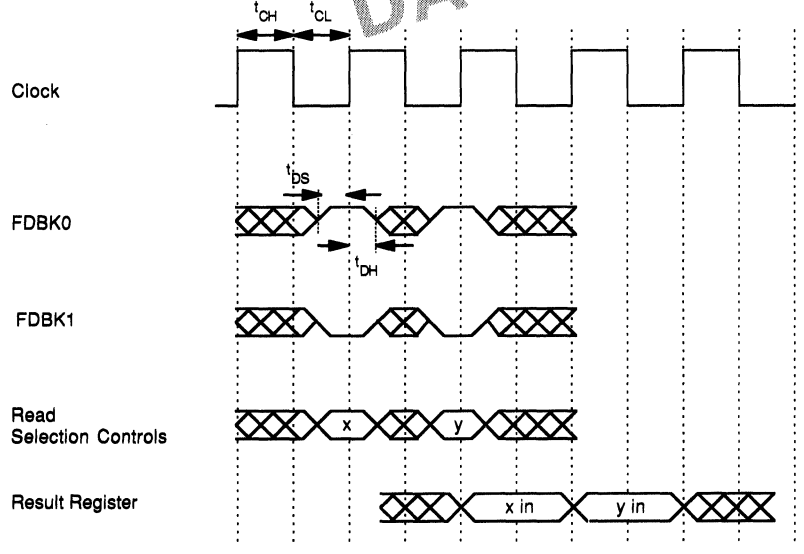
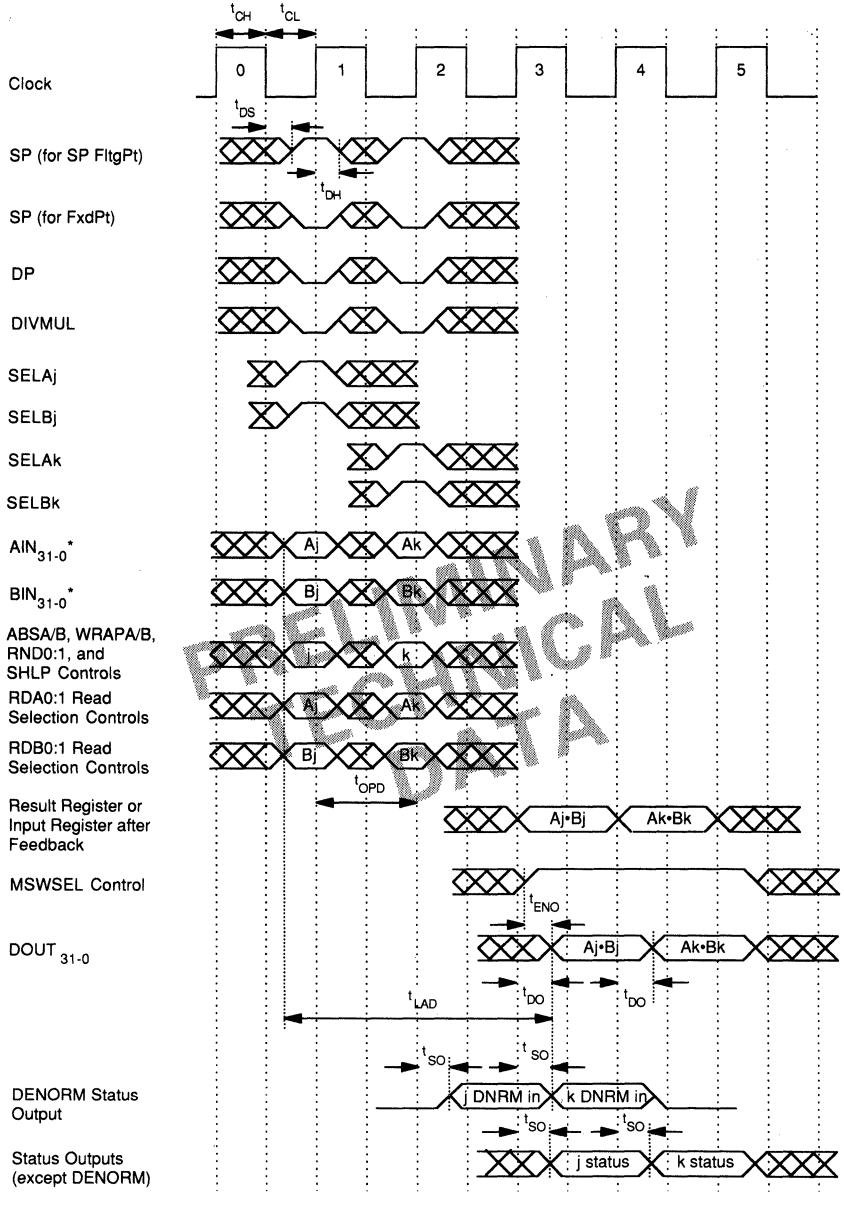
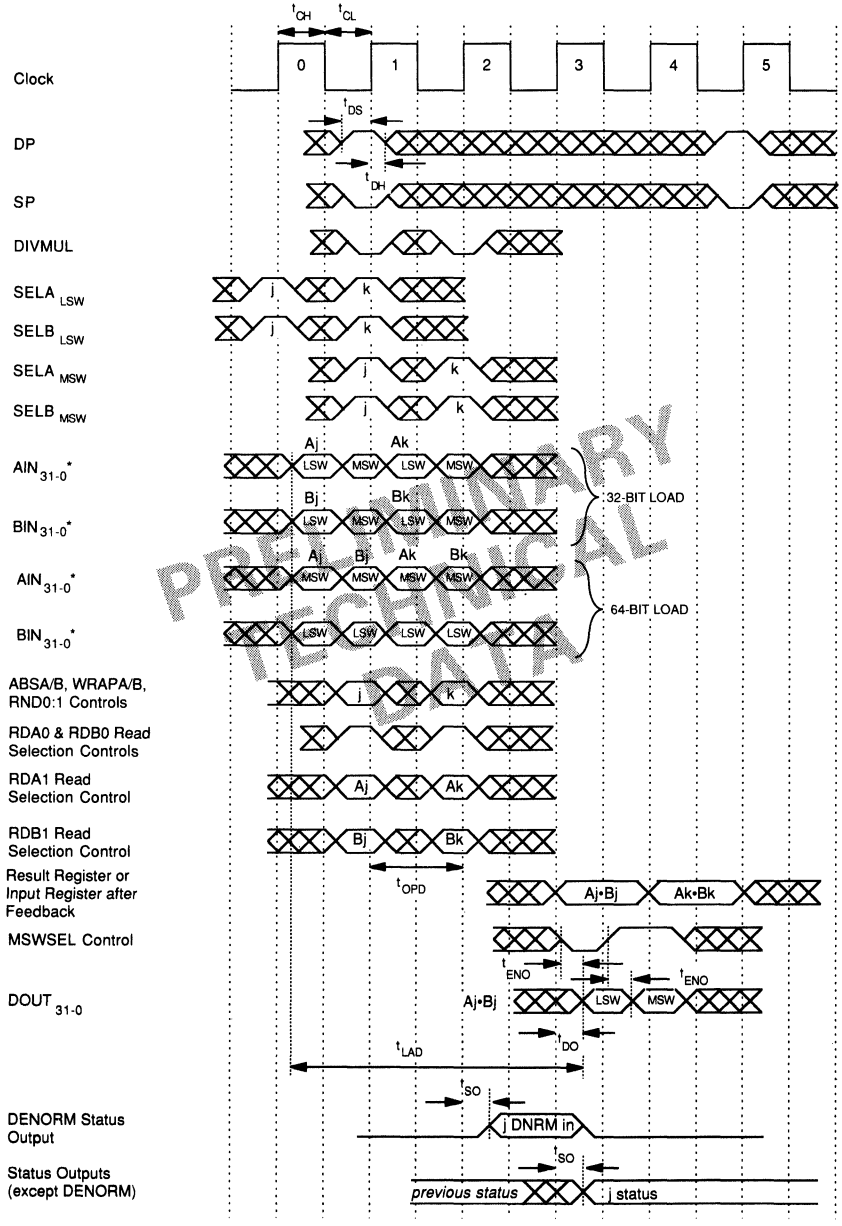


Figure T6. ADSP-3212/ADSP-3222 Feedforward Timing



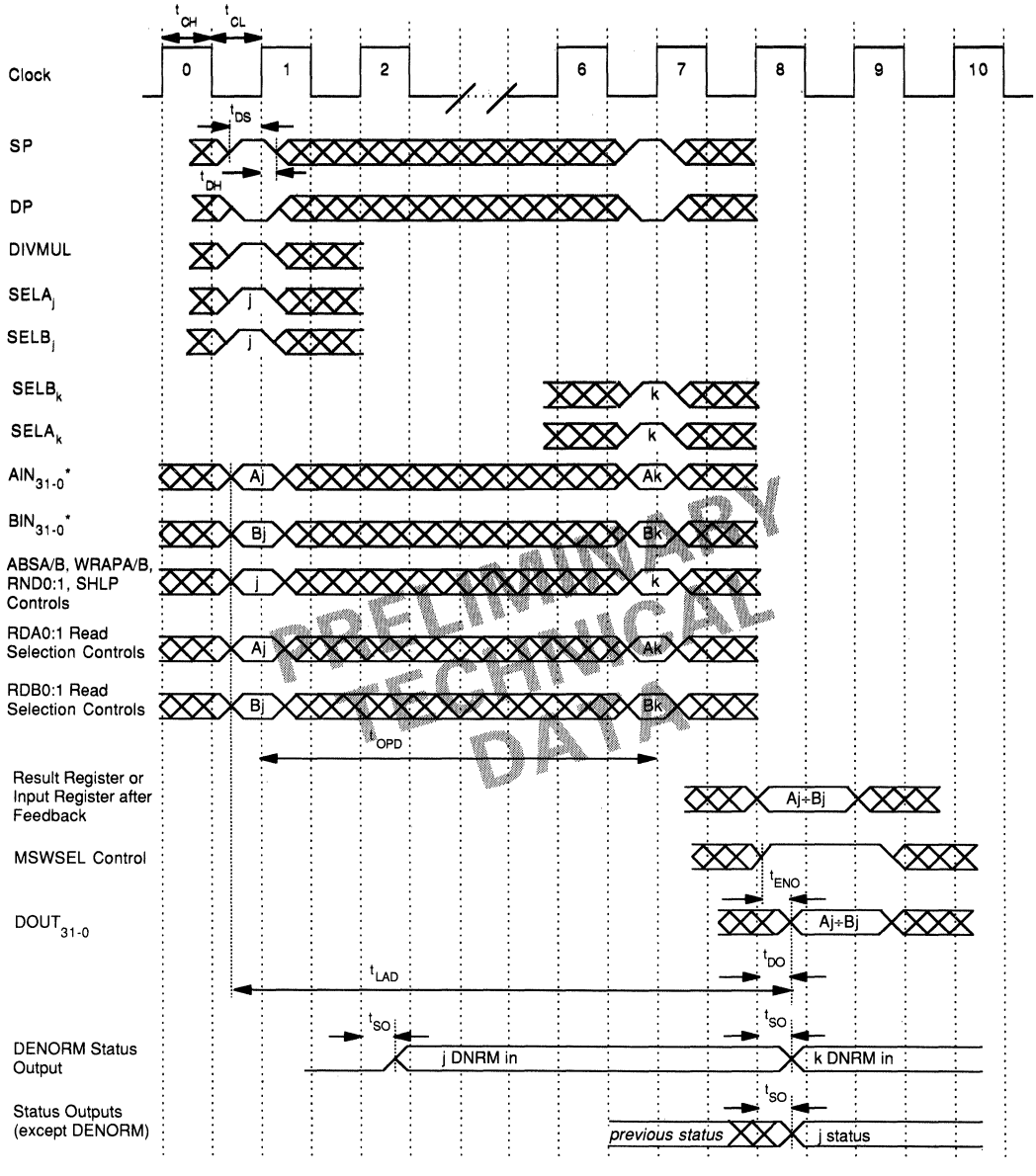
* See "Timing" section for additional sequencing options.

Figure T7. ADSP-3212 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplication



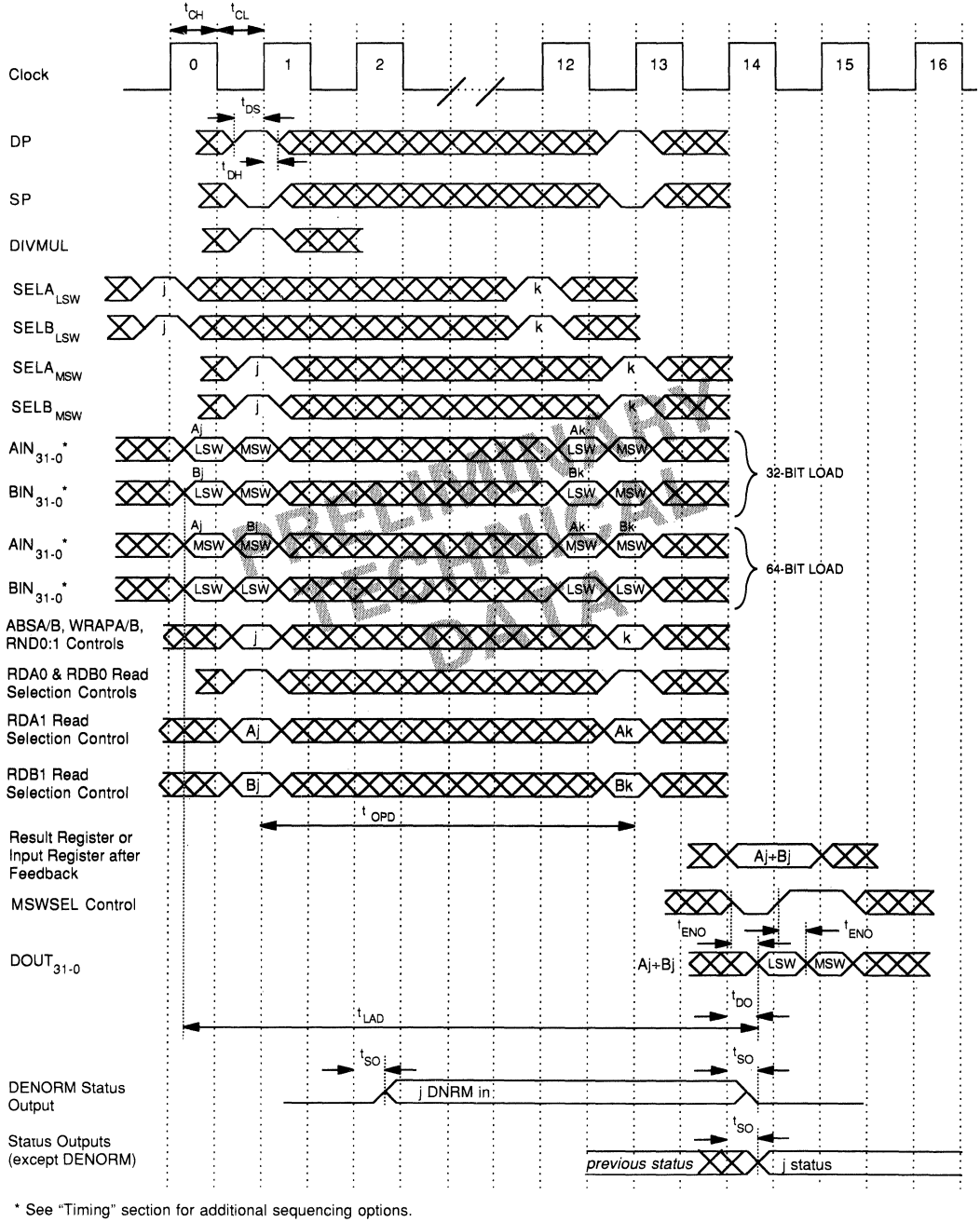
* See "Timing" section for additional sequencing options.

Figure T8. ADSP-3212 64-Bit Double-Precision Floating-Point Multiplication



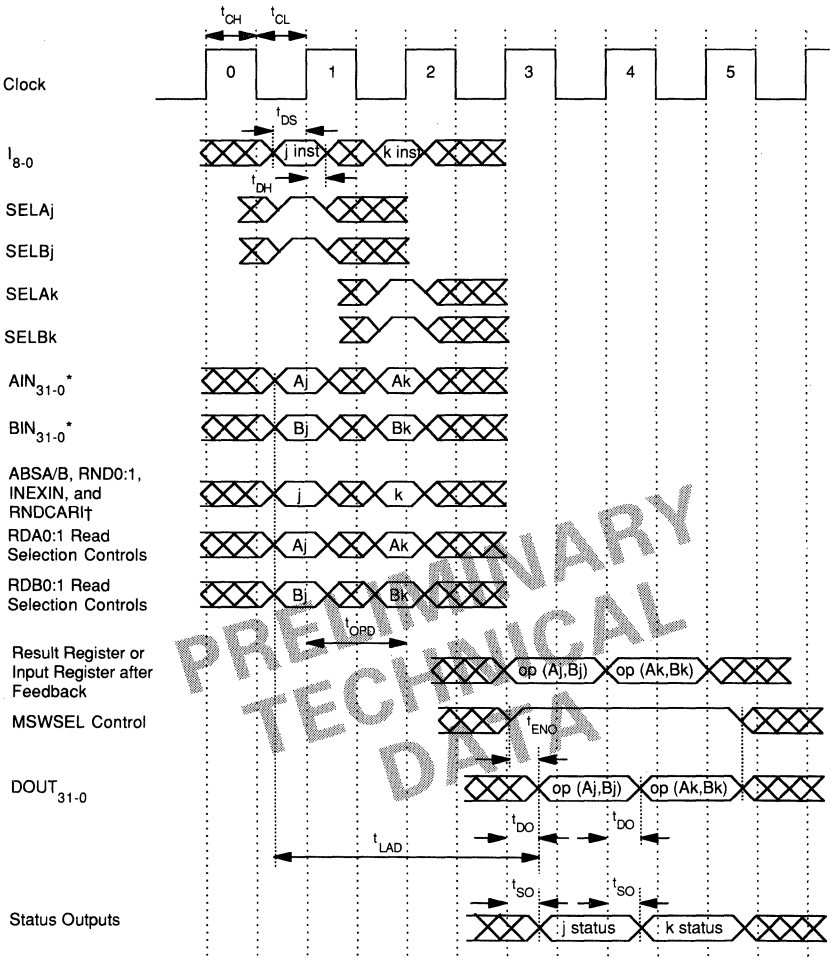
* See "Timing" section for additional sequencing options.

Figure T9. ADSP-3212 32-Bit Single-Precision Floating-Point Division



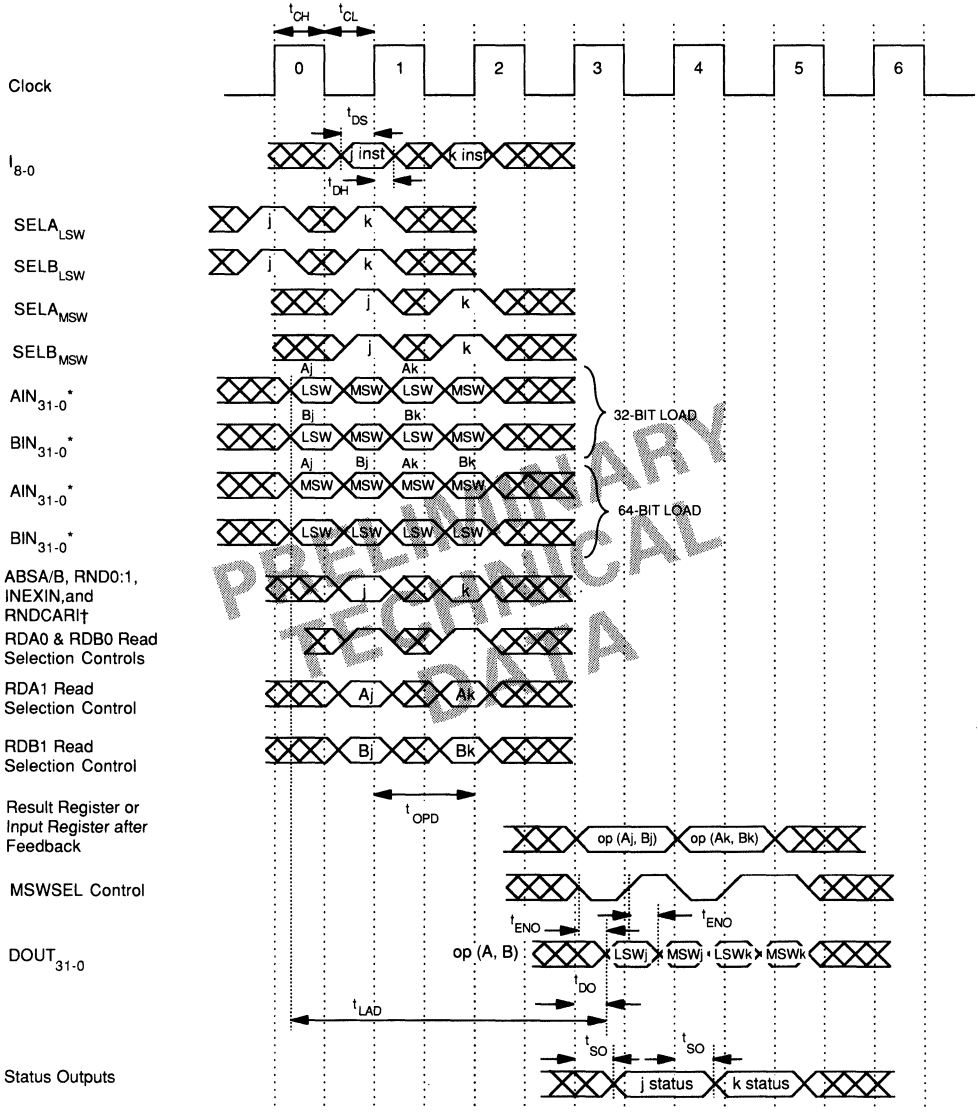
* See "Timing" section for additional sequencing options.

Figure T10. ADSP-3212 64-Bit Double-Precision Floating-Point Division



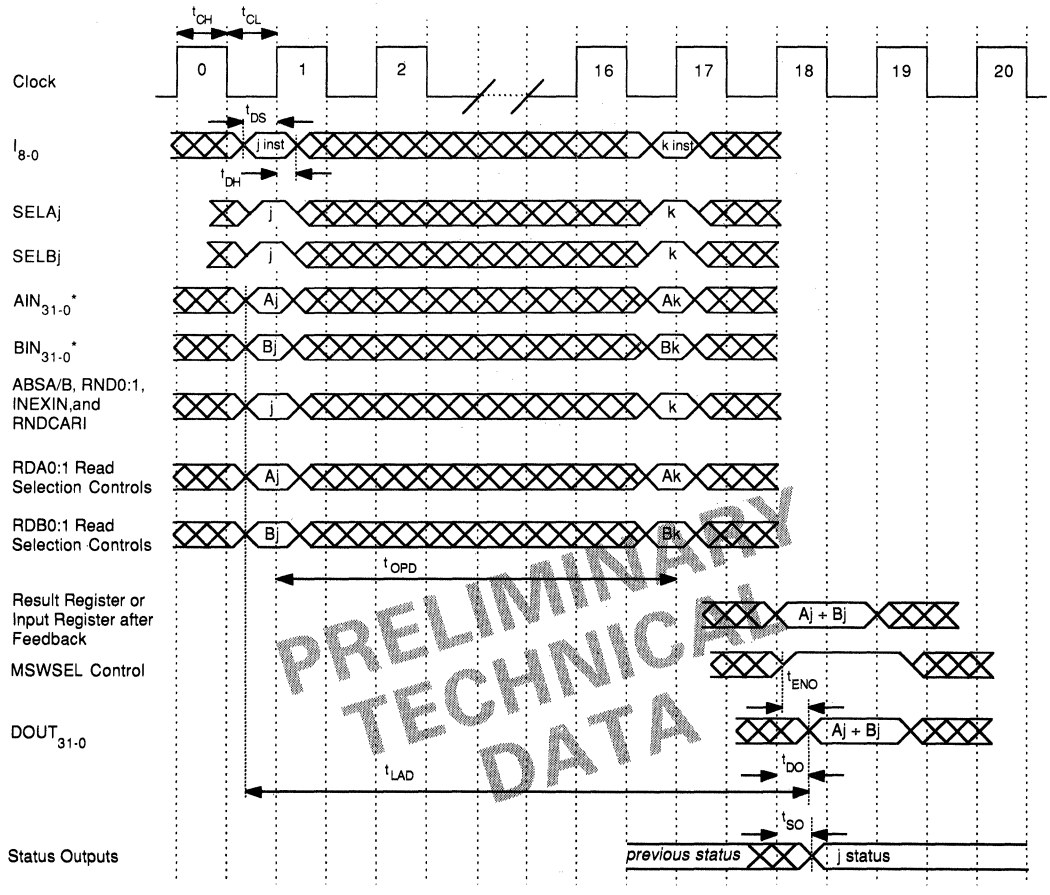
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T11. ADSP-3222 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.

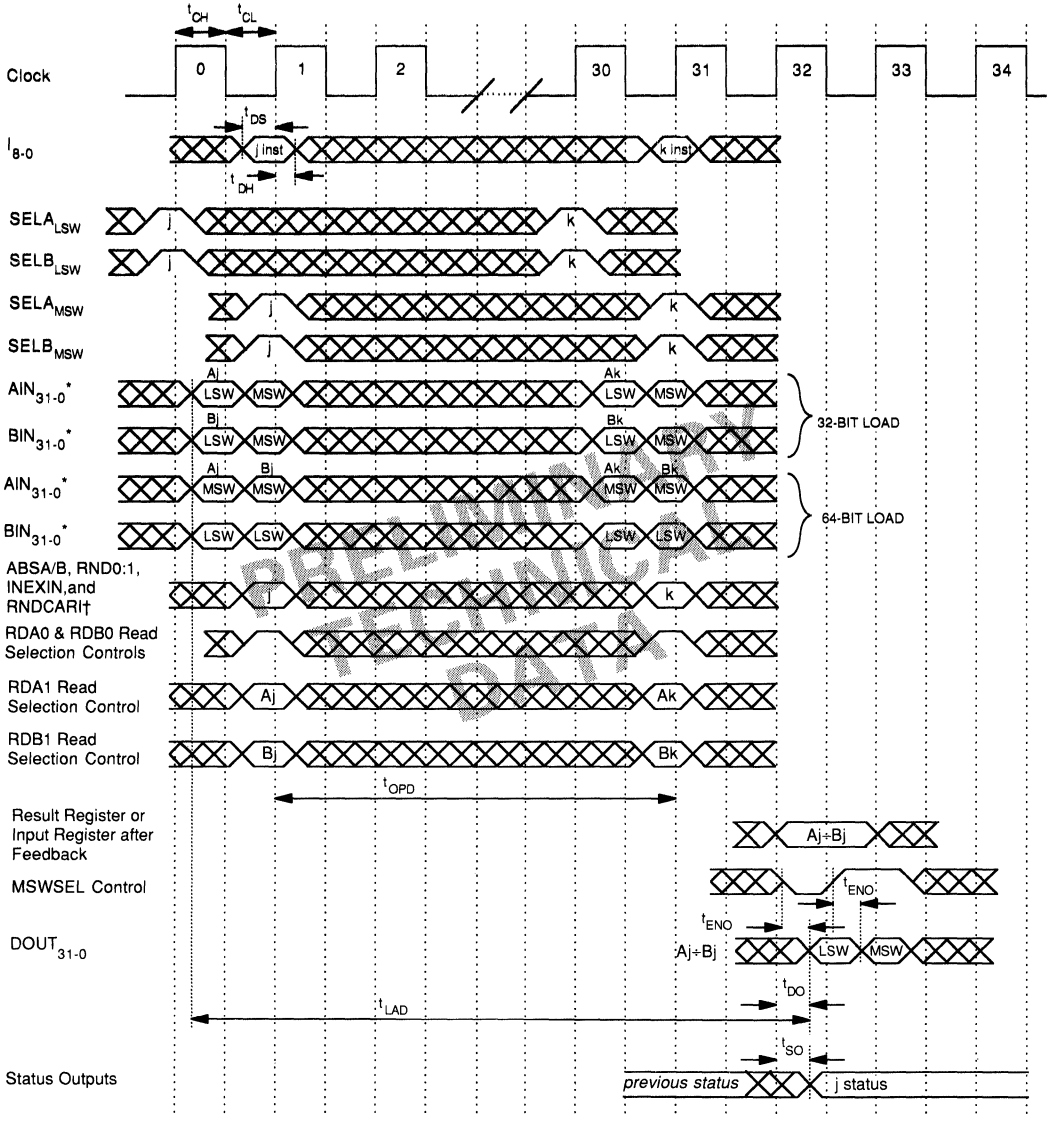
Figure T12. ADSP-3222 64-Bit Double-Precision Floating-Point ALU Operations



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

Note: The ADSP-3212 performs faster division. See Figure T9.

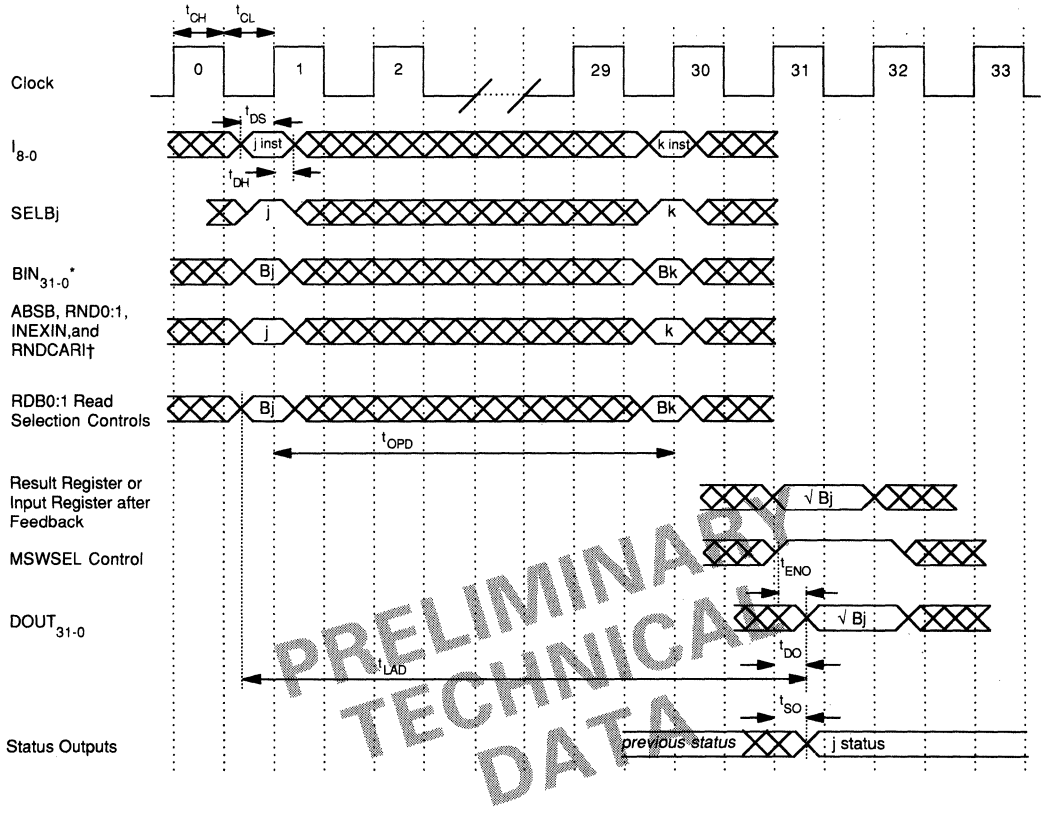
Figure T13. ADSP-3222 32-Bit Single-Precision Floating-Point Division



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

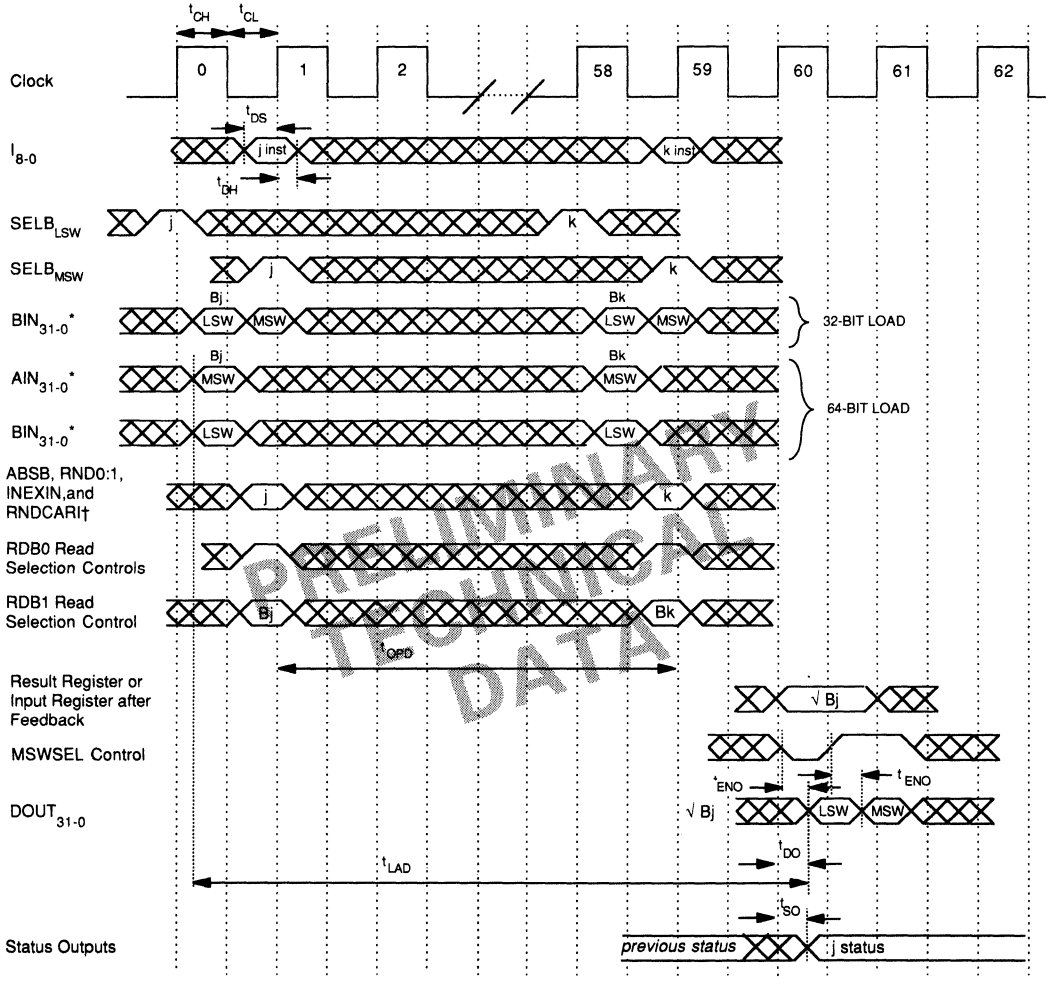
Note: The ADSP-3212 performs faster division. See Figure T10.

Figure T14. ADSP-3222 64-Bit Double-Precision Floating-Point Division



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

Figure T15. ADSP-3222 32-Bit Single-Precision Floating-Point Square Root



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

Figure T16. ADSP-3222 64-Bit Double-Precision Floating-Point Square Root

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3212/ADSP-3222 J, K Grades Min	J, K Grades Max	S, T Grades[2] Min	S, 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		ADSP-3212/ADSP-3222 J, K Grades Min	J, K Grades Max	S, T Grades Min	S, T Grades 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 and I_{OH}=−1.0mA	2.4		2.4		V
V_{OL}	Low Level Output Voltage @V_{DD}=min and I_{OL}=4.0mA		0.5		0.6	V
I_{IH}	High Level Input Current, All Inputs @V_{DD}=max and V_{IN}=5.0V		10		10	μA
I_{IL}	Low Level Input Current, All Inputs @V_{DD}=max and V_{IN}=0.0V		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		200		250	mA
I_{DD}	Supply Current-Quiescent All V_{IN}=2.4V		50		60	mA

NOTES

[1]All min and max specifications are over power supply and temperature ranges indicated.

[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-3212/ADSP-3222 can be found in Analog Devices' *Military Products Databook*. Regular S and T grade parts are tested at +125°C.

[3]Input levels are GND and +3.0V. Rise times are 5ns max. 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_{IH} or V_{IL} crossing points.

Specifications subject to change without notice.

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-3212/ADSP-3222 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 −55°C to +125°C Min	S Grade −55°C to +125°C Max	T Grade −55°C to +125°C Min	T Grade −55°C to +125°C Max	Unit
t_{CY}	Clock Period	60		50				58		ns
t_{CL}	Clock LO			20				23		ns
t_{CH}	Clock HI			20				23		ns
t_{DS}	Data Setup	12		7				8		ns
t_{CS}	Control Setup			10				12		ns
t_{DH}	Data and Control Hold	3		3				3		ns
t_{DO}	Data Output Delay		25		18				21	ns
t_{SO}	Status Output Delay		25		18				21	ns
t_{ENO}	MSWSEL-to-Data Delay		25		18				21	ns
t_{DIS}	Three-State Disable Delay				12				14	ns
t_{ENA}	Three-State Enable Delay			1	18			1	21	ns
t_{SU}	$\overline{\text{RESET}}$ Setup			5				6		ns
t_{RS}	$\overline{\text{RESET}}$ Pulse Duration			50				58		ns
t_{HS}	$\overline{\text{HOLD}}$ Setup	12		10				12		ns
t_{HH}	$\overline{\text{HOLD}}$ Hold	3		3				3		ns
t_{PS}	IPORT Setup	12		10				12		ns
t_{PH}	IPORT Hold	3		3				3		ns
t_{OPD}	Operation Time									
	32-Bit Multiplication		60		50				58	ns
	64-Bit Multiplication		60		50				58	ns
	32-Bit Division (3212)		360		300				345	ns
	64-Bit Division (3212)		720		600				690	ns
	32-Bit ALU Operations		60		50				58	ns
	64-Bit ALU Operations		60		50				58	ns
	32-Bit Division (3222)		960		800				920	ns
	64-Bit Division (3222)		1800		1500				1725	ns
	32-Bit Square Root		1740		1450				1668	ns
	64-Bit Square Root		3480		2900				3335	ns
t_{LAD}	Total Latency									
	32-Bit Multiplication		157		130				150	ns
	64-Bit Multiplication		187		155				179	ns
	32-Bit Division (3212)		457		380				437	ns
	64-Bit Division (3212)		847		705				811	ns
	32-Bit ALU Operations		157		130				150	ns
	64-Bit ALU Operations		187		155				179	ns
	32-Bit Division (3222)		1057		880				1012	ns
	64-Bit Division (3222)		1927		1580				1817	ns
	32-Bit Square Root		1897		1580				1817	ns
	64-Bit Square Root		3577		2980				3427	ns

NOTES
[1]All min and max specifications are over power supply and temperature ranges indicated.
[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-3212/ADSP-3222 can be found in Analog Devices' *Military Products Databook*. Regular S and T grade parts are tested at +125°C.
[3]Input levels are GND and +3.0V. Rise times are 5ns max. 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_{IH} or V_{IL} crossing points.

Specifications subject to change without notice.

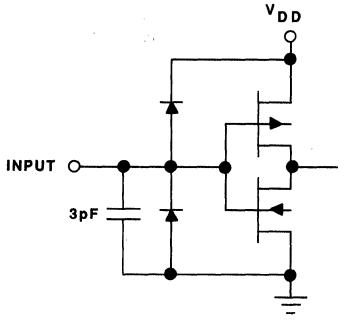


Figure 30. Equivalent Input Circuits

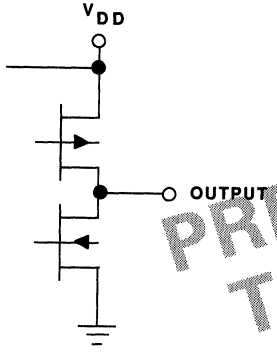


Figure 31. Equivalent Output Circuits

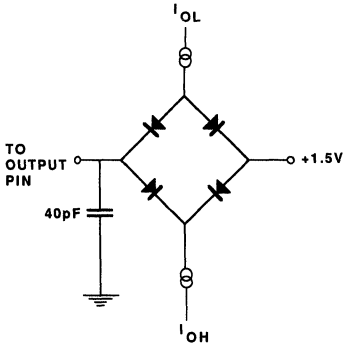


Figure 32. Normal Load for ac Measurements

ABSOLUTE MAXIMUM RATINGS

Supply Voltage . . . . . −0.3V to +7V
Input Voltage . . . . . −0.3V to V_{DD}
Output Voltage Swing . . . . . −0.3V to V_{DD}
Load Capacitance 200pF
Operating Temperature Range (Ambient) . . . . −55°C to +125°C
Storage Temperature Range . . . . . −65°C to +150°C
Lead Temperature (10sec) . . . . . +300°C

ESD SENSITIVITY

The ADSP-3212 and ADSP-3222 feature proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-3212 and ADSP-3222 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

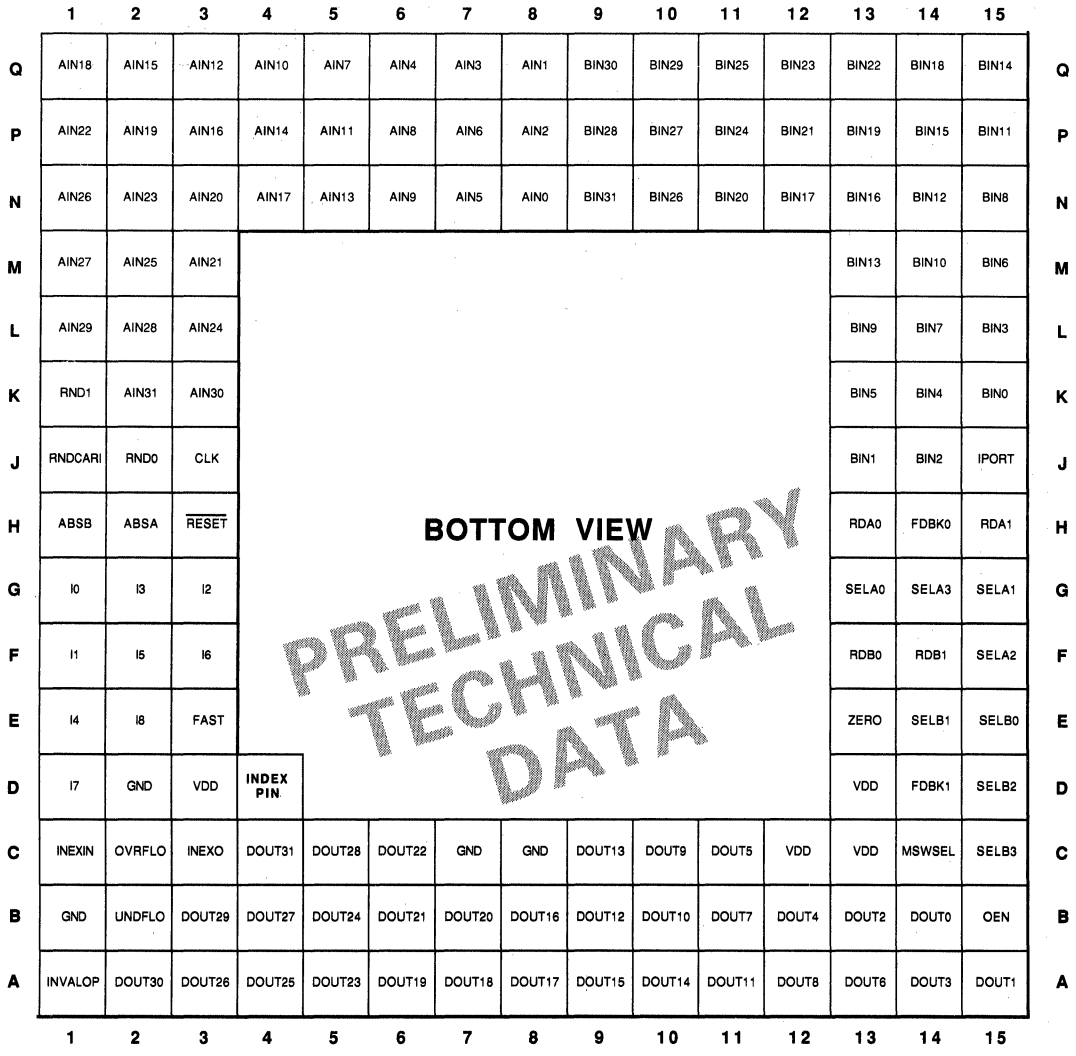
Part Number	Temperature Range	Package	Package Outline
ADSP-3212JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3212KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3212SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3212TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3212SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3212TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3222JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3222KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3222SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3222TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3222SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3222TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A

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

	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	Q
P	AIN22	AIN19	AIN16	AIN14	AIN11	AIN8	AIN6	AIN2	BIN28	BIN27	BIN24	BIN21	BIN19	BIN15	BIN11	P
N	AIN26	AIN23	AIN20	AIN17	AIN13	AIN9	AIN5	AIN0	BIN31	BIN26	BIN20	BIN17	BIN16	BIN12	BIN8	N
M	AIN27	AIN25	AIN21										BIN13	BIN10	BIN6	M
L	AIN29	AIN28	AIN24										BIN9	BIN7	BIN3	L
K	LOAD64	AIN31	AIN30										BIN5	BIN4	BIN0	K
J	SELA3	IPORT	SELA1										BIN1	BIN2	SELB3	J
H	SELA0	RDA1	SELA2										SELB0	SELB1	SELB2	H
G	RDA0	FAST	WRAPA										RDB1	ABSB	RDB0	G
F	ABSA	MSWSEL	OEN										DIVMUL	CLK	WRAPB	F
E	SHLP	UNDFLO	INVALOP										FDBK1	DP	SP	E
D	TCA	GND	VDD	INDEX PIN									VDD	$\overline{\text{RESET}}$	RND1	D
C	OVRFLO	DENORM	DOUT29	DOUT28	DOUT25	DOUT19	GND	GND	DOUT10	DOUT6	DOUT2	VDD	VDD	FDBK0	RND0	C
B	GND	DOUT30	DOUT26	DOUT24	DOUT21	DOUT18	DOUT17	DOUT13	DOUT9	DOUT7	DOUT4	DOUT1	INEXO	$\overline{\text{HOLD}}$	TCB	B
A	DOUT31	DOUT27	DOUT23	DOUT22	DOUT20	DOUT16	DOUT15	DOUT14	DOUT12	DOUT11	DOUT8	DOUT5	DOUT3	DOUT0	RNDCARC	A
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

BOTTOM VIEW

ADSP-3212 Pinout



ADSP-3222 Pinout

Fixed-Point Components
Contents

5

GENERAL INFORMATION

In 1983, Analog Devices was the first company to offer CMOS versions of the industry standard multipliers and multiplier/accumulators. Our initial offerings were fabricated using a 5 micron CMOS process.

Since that time, we have upgraded the process from 5 micron to 1.5 micron and now to 1 micron. These process improvements have significantly increased the speed of our industry-standard multipliers and multiplier/accumulators beyond earlier bipolar designs. In addition, these process improvements have allowed us to offer innovative new products such as the ADSP-1024A 24×24-bit multiplier and the ADSP-1110A single port multiplier/accumulator.

Currently both the 1.5 micron and 1 micron process are in production. All multipliers and multiplier/accumulators are identified by the "A" suffix if they are manufactured in 1.5 micron CMOS or by the "B" suffix if they are manufactured in 1 micron CMOS. For example, the ADSP-1016A is manufactured in 1.5 micron CMOS while the ADSP-1010B is manufactured in 1 micron CMOS.

The specifications in this section of this databook supersede the specifications in all previous publications including individual data sheets and the *1987 DSP Products Databook*. In the event of conflicts, this publication takes precedence.

Contact your local sales office for information about new, faster versions of these fixed-point components.

Selection Guide

FIXED-POINT MULTIPLIERS

Word Size	Model Number	Multiplication Time, ns[1] Clocked Comm	Clocked MIL	Unclocked Comm	Unclocked MIL	I_{DD}[2] Comm	I_{DD}[2] MIL	Data Formats Mixed Twos Comp	Unsign Mag.	Mixed Mode	No. of Pins	Package Options[3]
8×8	ADSP-1080A	J=45 K=33	S=55 T=45	N/A	N/A	45	55	Yes			40	D, N
8×8	ADSP-1081A	J=45 K=33	S=55 T=45	N/A	N/A	45	55		Yes		40	D, N
12×12	ADSP-1012A	J=75 K=50	S=90 T=60	J=105 K=80	S=125 T=95	60	70	Yes	Yes	Yes	64 68	D, N G, E
16×16	ADSP-1016A	J=85 K=70	S=95 T=80	J=105 K=90	S=120 T=105	65	55	Yes	Yes	Yes	64 68	D, N G, E
24×24	ADSP-1024A	J=120 K=95	S=150 T=120	N/A	N/A	70	90	Yes			84	G

NOTES
[1]ns max @ T_A=+70°C commercial, +125°C MIL.
[2]mA max, f_{CLK}=max, V_{DD}= +5V @ T_A=+70°C commercial, +125°C MIL.
[3]D=ceramic DIP, N=plastic DIP, E=leadless chip carrier, G=pin grid array, P=PLCC.

FIXED-POINT MULTIPLIER/ACCUMULATORS

Word Size	Model Number	MAC Time, ns[1] Comm	MAC Time, ns[1] MIL	Accumulators Size	Accumulators Number	I_{DD}[2] Comm	I_{DD}[2] MIL	No. of Pins	Package Options[3]
8×8	ADSP-1008A	J=60 K=50	S=75 T=60	19	1	40	45	48	D, N
12×12	ADSP-1009A	J=85 K=70	S=100 T=85	27	1	70	75	64 68	D, N G, E
16×16	ADSP-1010A	J=85 K=75	S=100 T=90	35	1	80	100	64 68	D, N G, E
16×16	ADSP-1010B	J=55 K=45	S=65 T=55	35	1	110	125	64 68	D, N[4] G, E[4], P
16×16	ADSP-1101	J=90 K=80	S=105 T=95	40	2	75	75	100	G
16×16	ADSP-1110A	J=100 K=85	S=120 T=100	40	1	70	80	28	D, N, P

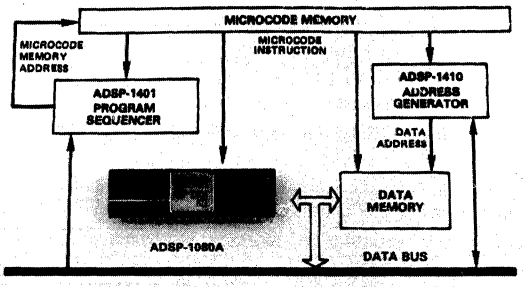
NOTES
[1]ns max @ T_A=+70°C commercial, +125°C MIL.
[2]mA max, f_{CLK}=max, V_{DD}=+5V @ T_A=+70°C commercial, +125°C MIL.
[3]D=ceramic DIP, N=plastic DIP, E=leadless chip carrier, G=pin grid array, P=PLCC.
[4]Contact factory.

8 × 8-Bit Twos Complement CMOS Multiplier

ADSP-1080A

FEATURES
- **8 × 8-Bit Parallel Multiplication**
- **30MHz Multiplication Rate**
- **275mW Power Dissipation with TTL-Compatible 1.5 Micron CMOS Technology**
- **Twos-Complement Data Format**
- **Available in Hermetically Sealed 40-Pin DIP or Plastic 40-Pin DIP**
- **Available Specified from −55°C to +125°C Ambient**
- **Pin-Compatible with ADSP-1080 and MPY008HJ5**

APPLICATIONS
- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1080A

GENERAL DESCRIPTION

The ADSP-1080A is a high-speed, low-power 8 × 8-bit parallel multiplier fabricated in 1.5 micron CMOS.

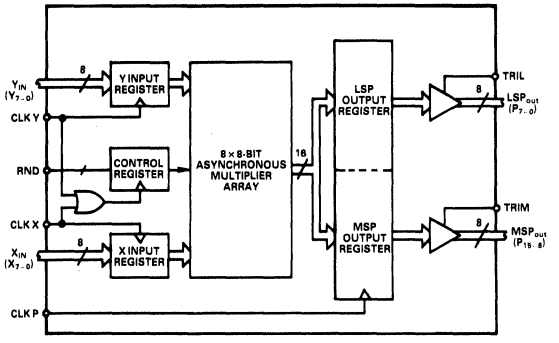
The ADSP-1080A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, and an 8-bit Least Significant Product (LSP) port. Input data is interpreted in twos-complement format. The ADSP-1080A produces a 16-bit result whose twos-complement MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Both of the product registers have their own three-state output controls. Three-state outputs and independently clocked inputs allow the ADSP-1080A to be connected directly to a single 8-bit bus.

The ADSP-1080A is a pin-for-pin replacement for Analog Devices' ADSP-1080 and is also pin-for-pin compatible in a DIP package with TRW's MPY008HJ5 and MPY008HJ5-1. The ADSP-1080A's multiply time is faster than either TRW device.

The power consumption of the ADSP-1080A is 275mW maximum, 5% of the power required by equivalent bipolar devices. The differential between the ADSP-1080A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-1080A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1080A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1080A is available in either a 40-pin hermetically sealed ceramic DIP or a plastic 40-pin DIP.



ADSP-1080A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1080A J and K Grades Min	ADSP-1080A J and K Grades Max	ADSP-1080A S and T Grades[2] Min	ADSP-1080A 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-1080A J and K Grades Min	ADSP-1080A J and K Grades Max	ADSP-1080A S and T Grades[2] Min	ADSP-1080A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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} = 4mA		0.4		0.6	V
I_{IH}	High-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 5V		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	@ 20MHz, TTL Inputs		45		55	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		30		35	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1080A 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_D	Output Delay		25		25		30		30	ns
t_{ENA}	Three State Enable Delay		20		20		25		25	ns
t_{DIS}	Three State Disable Delay		20		20		25		25	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	20		20		20		20		ns
t_H	Input Hold Time	2		2		2		2		ns
t_{MC}	Clocked Multiply Time		45		33		55		45	ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1080A 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 t_{ENA} and t_{DIS} which are as indicated in Figure 2.

Specifications subject to change without notice.

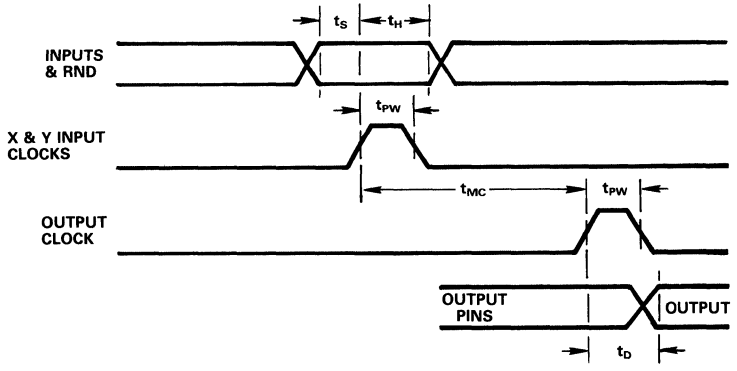


Figure 1. Timing Diagram

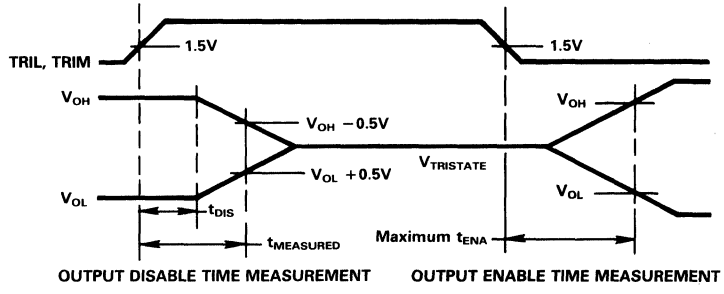


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

METHOD OF OPERATION

The X and Y input registers are positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers by the rising edges of CLK X and CLK Y, respectively.

The X and Y input data is interpreted in twos-complement

INPUT DATA FORMAT (X & Y)								OUTPUT DATA FORMATS (P)															
								MSP (P_{15-8})								LSP (P_{7-0})							
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACTIONAL TWOS COMPLEMENT																							
sign								sign								sign							
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	-2^0	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
INTEGER TWOS COMPLEMENT																							
sign								sign								sign							
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	-2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	-2^{14}	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table I. ADSP-1080A Data Formats

notation. (See Table I for the ADSP-1080A's data formats. Unsigned-magnitude and mixed-mode data formats are not supported.)

RND is a registered input control latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in RND. When RND is HI (logic 1), the MSP will be rounded by adding a binary 1 to the MSB of the LSP, consistently rounding toward positive infinity at LSP mid-scale. Truncating the MSP (RND LO) introduces a large-sample statistical bias of −127/2 LSBs of the LSP, while rounding (RND HI) reduces the bias to only +1/2LSB of the LSP.

The ADSP-1080A's output is fielded into an 8-bit twos-complement MSP and an 8-bit LSP (see Table I). The LSP consists of the 7LSBs of the product and the sign bit from the MSB of the MSP mapped to the MSB of the LSP. (Note that the LSP is not in proper twos-complement form.)

To increase the number of significant bits in the MSP, the ADSP-1080A left-shifts bits 14 through 7 from the multiplier array to product bits P15 through P8, eliminating one of the two normally redundant MSBs. This automatic left-shift doubles the dynamic range of the MSP. However, an overflow will occur when full-scale negative is multiplied by itself, yielding full-scale negative instead of the correct positive product. To avoid this overflow, disallow X and Y inputs that are both full-scale negative.

The rising edge of CLK P latches the LSP and MSP into the output registers. Each of these registers has its own three-state control. A HI on the asynchronous TRIL or TRIM line disables the corresponding LSP or MSP output driver to a high-impedance state. Conversely, a LO on TRIL or TRIM enables the corresponding output driver, driving the output bus.

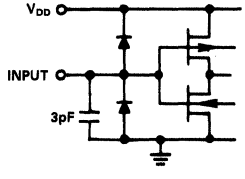


Figure 3. Equivalent Input Circuit

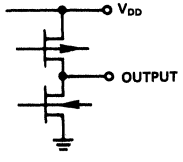


Figure 4. Equivalent Output Circuit

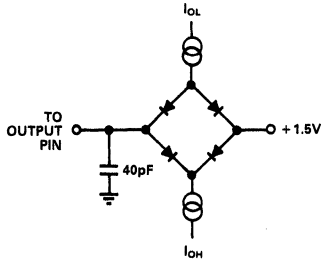


Figure 5. Normal Load for ac Measurements

ABSOLUTE MAXIMUM RATINGS

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

ESD SENSITIVITY

The ADSP-1080A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1080A has been classified as a Class 1 device.

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*.

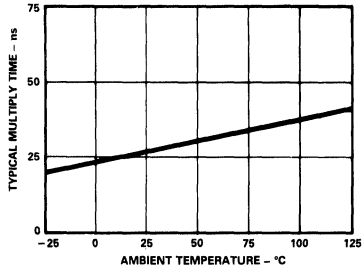


Figure 6. Approximate Clocked Multiply Time vs. Temperature

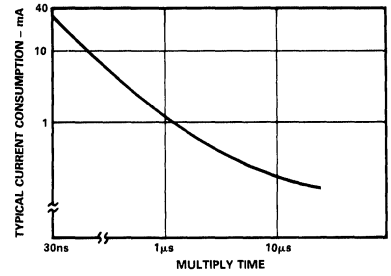


Figure 7. Typical I_{DD} vs. Frequency of Operation

ADSP-1080A PIN CONFIGURATION
DIP
D-40A
N-40A

PIN	FUNCTION	PIN	FUNCTION
1	P10	21	X6
2	P9	22	X7 (MSB)
3	P8	23	CLK X
4	CLK P	24	CLK Y
5	TRIM	25	RND
6	TRIL	26	Y0
7	P7	27	Y1
8	P6	28	Y2
9	P5	29	Y3
10	P4	30	V_{DD}
11	P3	31	Y4
12	P2	32	GND
13	P1	33	Y5
14	P0	34	Y6
15	X0	35	Y7 (MSB)
16	X1	36	P15 (MSB)
17	X2	37	P14
18	X3	38	P13
19	X4	39	P12
20	X5	40	P11

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1080AKD	0 to +70°C	40-Pin Ceramic DIP	D-40A
ADSP-1080AKN	0 to +70°C	40-Pin Plastic DIP	N-40A
ADSP-1080AJD	0 to +70°C	40-Pin Ceramic DIP	D-40A
ADSP-1080AJN	0 to +70°C	40-Pin Plastic DIP	N-40A
ADSP-1080ATD	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1080ASD	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1080ATD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1080ASD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40A

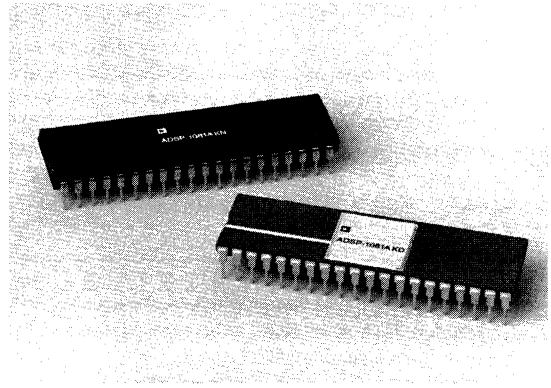
Contact DSP Marketing in Norwood concerning the availability of other package types.

8 × 8-Bit Unsigned-Magnitude CMOS Multiplier

ADSP-1081A

FEATURES
- **8 × 8-Bit Parallel Multiplication**
- **30MHz Multiplication Rate**
- **275mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Unsigned-Magnitude Data Format**
- **Available in Hermetically-Sealed 40-Pin DIP or Plastic 40-Pin DIP**
- **Available Specified from −55°C to +125°C Ambient**
- **Pin-Compatible with ADSP-1081 and MPY08HUJ5**

APPLICATIONS
- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**



GENERAL DESCRIPTION

The ADSP-1081A is a high-speed, low-power 8 × 8-bit parallel multiplier fabricated in 1.5 micron CMOS.

The ADSP-1081A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, and an 8-bit Least Significant Product (LSP) port. Input data is interpreted in unsigned-magnitude format. The ADSP-1081A produces a 16-bit result whose unsigned-magnitude MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

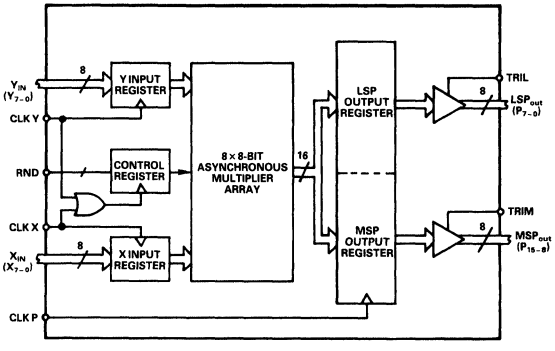
All input pins are ESD protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Both of the product registers have their own three-state output controls. Three-state outputs and independently-clocked inputs allow the ADSP-1081A to be connected directly to a single 8-bit bus.

The ADSP-1081A is a pin-for-pin replacement for Analog Devices' ADSP-1081 and is also pin-for-pin compatible in a DIP package with TRW's MPY08HUJ5 and MPY08HUJ5-1. The ADSP-1081A's multiply time is faster than either TRW device.

The power consumption of the ADSP-1081A is 275mW maximum, less than 15% of the power required by equivalent bipolar devices. The differential between the ADSP-1081A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-1081A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1081A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1081A is available in either a 40-pin hermetically-sealed ceramic DIP or a plastic 40-pin DIP.

5



ADSP-1081A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1081A J and K Grades Min	ADSP-1081A J and K Grades Max	ADSP-1081A S and T Grades[2] Min	ADSP-1081A 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-1081A J and K Grades Min	ADSP-1081A J and K Grades Max	ADSP-1081A S and T Grades[2] Min	ADSP-1081A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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} = 4mA		0.4		0.6	V
I_{IH}	High-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 5V		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	@ min of (20MHz, max clock rate); TTL Inputs		45		55	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		30		35	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1081A J Grade Min	ADSP-1081A J Grade Max	ADSP-1081A K Grade Min	ADSP-1081A K Grade Max	ADSP-1081A S Grade[2] Min	ADSP-1081A S Grade[2] Max	ADSP-1081A T Grade[2] Min	ADSP-1081A T Grade[2] Max	Unit
t_D	Output Delay		25		25		30		30	ns
t_{ENA}	Three State Enable Delay		20		20		25		25	ns
t_{DIS}	Three State Disable Delay		20		20		25		25	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	20		20		20		20		ns
t_H	Input Hold Time	2		2		2		2		ns
t_{MC}	Clocked Multiply Time		45		33		55		45	ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1081A 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 t_{ENA} and t_{DIS} which are as indicated in Figure 2.
Specifications subject to change without notice.

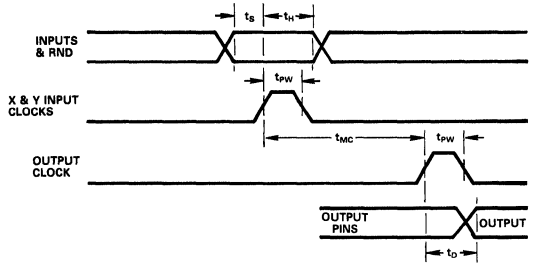


Figure 1. Timing Diagram

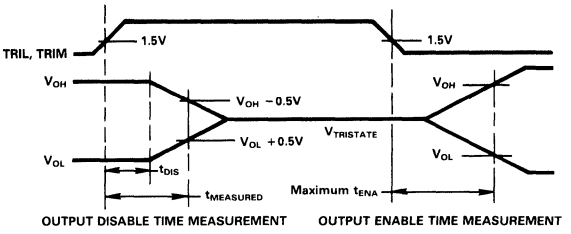


Figure 2. Three-State Disable and Enable Timing

METHOD OF OPERATION

The X and Y input registers are positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers by the rising edges of CLK X and CLK Y, respectively.

The X and Y input data is interpreted in unsigned-magnitude notation. (See Table I for the ADSP-1081A's data formats. Twos-complement and mixed-mode data formats are not supported. For twos-complement 8 × 8 multiplication, use the ADSP-1080A.)

RND is a registered input control latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in RND. When RND is HI (logic 1), the MSP will be rounded by adding a binary 1 to the MSB of the LSP, consistently rounding toward positive infinity. Truncating the MSP (RND LO) introduces a large-sample statistical bias of −127/2LSBs of the LSP, while rounding (RND HI) reduces the bias to only +1/2LSB of the LSP.

The ADSP-1081A's output is fielded into an 8-bit MSP and an 8-bit LSP (see Table I). The rising edge of CLK P latches the LSP and MSP into the output registers. Each of these registers has its own three-state control. A HI on the asynchronous TRIL or TRIM line disables the corresponding LSP or MSP output driver to a high-impedance state. Conversely, a LO on TRIL or TRIM enables the corresponding output driver, driving the output bus.

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

X & Y INPUT DATA FORMAT								OUTPUT DATA FORMATS																	
								MSP (P_{15-8})								LSP (P_{7-0})									
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
UNSIGNED FRACTIONAL																									
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}		
UNSIGNED INTEGER																									
2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}		

Table I. Data Formats for the ADSP-1081A

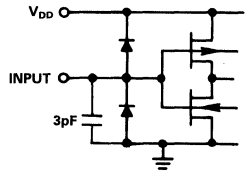


Figure 3. Equivalent Input Circuit

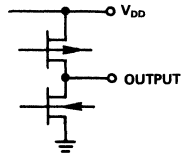


Figure 4. Equivalent Output Circuit

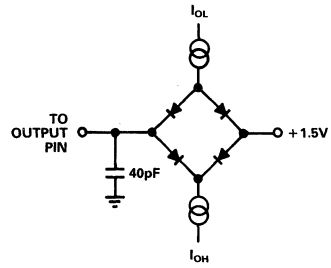


Figure 5. Normal Load for ac Measurements

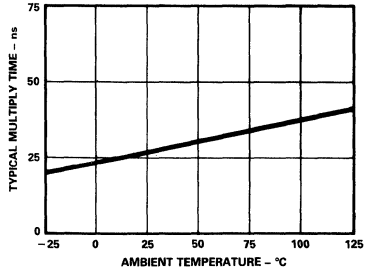


Figure 6. Approximate Clocked Multiply Time vs. Temperature

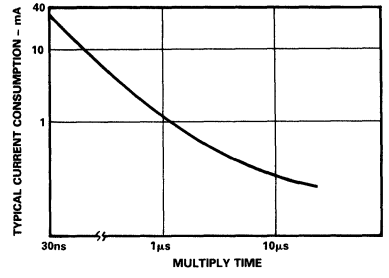


Figure 7. Typical I_{DD} vs. Frequency of Operation

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	−0.3V to 7.0V
Input Voltage	−0.3V to V_{DD}
Output Voltage	−0.3V to V_{DD}
Operating Temperature Range ($T_{AMBIENT}$)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10sec)	+300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1081AKD	0 to +70°C	40-Pin Ceramic DIP	D-40A
ADSP-1081AKN	0 to +70°C	40-Pin Plastic DIP	N-40A
ADSP-1081AJD	0 to +70°C	40-Pin Ceramic DIP	D-40A
ADSP-1081AJN	0 to +70°C	40-Pin Plastic DIP	N-40A
ADSP-1081ATD	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1081ASD	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1081ATD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40A
ADSP-1081ASD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40A

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

PIN CONFIGURATION

PIN	FUNCTION	PIN	FUNCTION
1	P10	21	X6
2	P9	22	X7 (MSB)
3	P8	23	CLK X
4	CLK P	24	CLK Y
5	TRIM	25	RND
6	TRIL	26	Y0
7	P7	27	Y1
8	P6	28	Y2
9	P5	29	Y3
10	P4	30	V_{DD}
11	P3	31	Y4
12	P2	32	GND
13	P1	33	Y5
14	P0	34	Y6
15	X0	35	Y7 (MSB)
16	X1	36	P15 (MSB)
17	X2	37	P14
18	X3	38	P13
19	X4	39	P12
20	X5	40	P11

ESD SENSITIVITY

The ADSP-1081A features proprietary protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1081A has been classified as a Class 1 device.

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*.

ANALOG DEVICES

12 × 12-Bit CMOS Multiplier

ADSP-1012A

FEATURES

- **12 × 12-Bit Parallel Multiplication**
- **20MHz Multiplication Rate (Worst Case)**
- **300mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Twos-Complement, Unsigned-Magnitude, and Mixed-Mode Data Formats**
- **Available in Hermetically-Sealed 64-Pin DIP, Hermetically-Sealed 68-Pin PGA, Plastic 64-Pin DIP, or 68-Contact LCC**
- **Available Specified to MIL-STD-883, Class B**
- **Pin-Compatible with ADSP-1012 and MPY012HJ1**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**

GENERAL DESCRIPTION

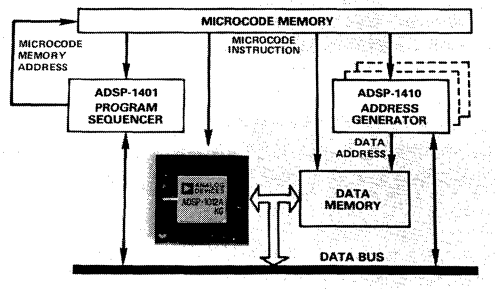
The ADSP-1012A is a high-speed, low-power 12 × 12-bit parallel multiplier fabricated in 1.5 micron CMOS.

The ADSP-1012A has two 12-bit input ports, a 12-bit Most Significant Product (MSP) port, and a 12-bit Least Significant Product (LSP) port. Input data is interpreted in twos-complement, unsigned-magnitude, or mixed-mode formats. The ADSP-1012A produces a 24-bit result whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. Both of the product registers have their own independent clock lines and their own independent three-state output controls. Three-state outputs and independently clocked inputs allow the ADSP-1012A to be connected directly to a single 12-bit bus.

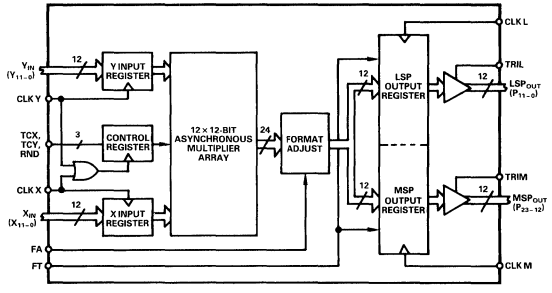
The ADSP-1012A is a pin-for-pin replacement for Analog Devices' ADSP-1012 and is also pin-for-pin compatible in a DIP package with TRW's MPY012HJ1. The ADSP-1012A's multiply time is over twice as fast as the TRW device.

The power consumption of the ADSP-1012A is 300mW maximum, 10% of the power required by equivalent bipolar devices. The differential between the ADSP-1012A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-1012A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1012A

The ADSP-1012A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1012A is available in either a 64-pin hermetically sealed ceramic DIP, a hermetically sealed ceramic 68-pin grid array, a plastic 64-pin DIP, or a 68-contact LCC.



Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1012A J and K Grades Min	ADSP-1012A J and K Grades Max	ADSP-1012A S and T Grades[2] Min	ADSP-1012A 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-1012A J and K Grades Min	ADSP-1012A J and K Grades Max	ADSP-1012A S and T Grades[2] Min	ADSP-1012A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.2		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		0.4		0.5	V
I_{IH}	High-Level Input Current, All Inputs	@ V_{DD} = max & V_{IN} = 5V		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		60		70	mA
I_{DD}	Supply Current – Quiescent	All V_{IN} = 2.4V		30		35	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1012A J Grade 0 to +70° Min	J Grade Max	K Grade Min	K Grade Max	S Grade[2] −55°C to +125°C Min	S Grade[2] Max	T Grade[2] Min	T Grade[2] Max	Unit
t_D	Output Delay		30		30		35		35	ns
t_{ENA}	Three-State Enable Delay		30		30		35		35	ns
t_{DIS}	Three-State Disable Delay		30		30		35		35	ns
t_{PW}	Clock Pulse Width	20		20		20		20		ns
t_S	Input Setup Time	20		20		20		20		ns
t_H	Input Hold Time	2		2		2		2		ns
t_{MC}	Clocked Multiply Time		75		50		90		60	ns
t_{MUC}	Unclocked Multiply Time		105		80		125		95	ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1012A 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 t_{ENA} and t_{DIS} which are indicated in Figure 2.
Specifications subject to change without notice.

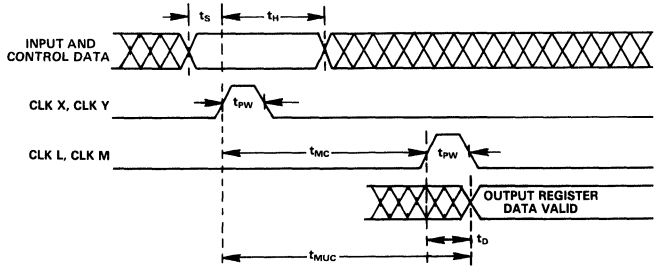


Figure 1. ADSP-1012A Timing Diagram

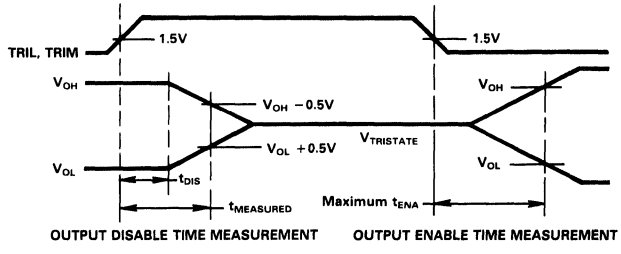


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

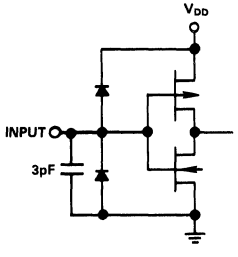


Figure 3. Equivalent Input Circuit

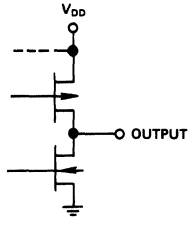


Figure 4. Equivalent Output Circuit

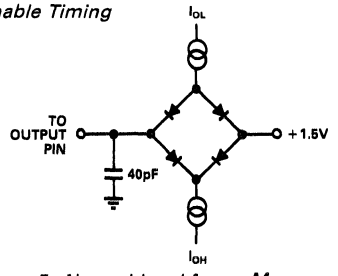


Figure 5. Normal Load for ac Measurements

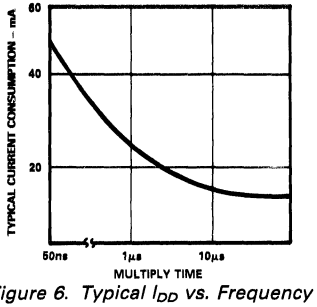


Figure 6. Typical I_{DD} vs. Frequency

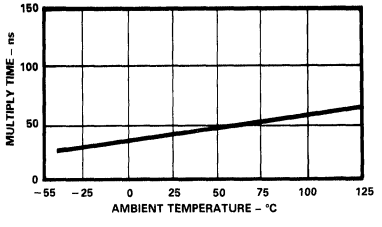


Figure 7. Approximate Worst Case Multiply Time vs. Temperature

METHOD OF OPERATION

The X and Y input registers are positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers by the rising edges of CLK X and CLK Y, respectively.

The X and Y input data can be either in twos-complement, unsigned-magnitude, or mixed-mode formats (Table I.) Twos-complement input data is indicated by HI (logic 1) levels on the TCX line for X input data and by HI levels on the TCY line for Y input data. Unsigned-magnitude X and Y inputs are indicated by LO (logic 0) levels on the TCX and TCY lines, respectively. Outputs will be in the same format as inputs unless the input formats are mixed, in which case the outputs will be in twos-complement representation.

The ADSP-1012A's output is fielded into an 12-bit MSP and an 12-bit LSP. When RND is HI, the MSP will be rounded by adding a binary 1 with carry to the MSB of the LSP, consistently rounding toward positive infinity. Truncating the MSP (RND LO) introduces a large-sample statistical bias $-(2^{12}-1)/2$ LSBs of the LSP, while rounding (RND HI) reduces the bias to only $+1/2$ LSBs of the LSP.

TCX, TCY, and RND are registered input controls. TCX and TCY are latched by the rising edges of CLK X and CLK Y, respectively. RND is latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO before attempting to clock in RND.

The asynchronous FA control format-adjusts the output from the multiplier array (Table II). FA must be HI to get a product for unsigned-magnitude or mixed-mode multiplications in a standard format. In a mixed-mode product, the sign bit will be product Bit 23 (P23). For twos-complement multiplications, FA can be LO. If FA is at a LO level, the MSP and the MSB of the LSP are left-shifted one bit and the sign bit is duplicated in the MSB of the LSP.

Format-adjusting a twos-complement product increases the number of significant bits in the MSP by eliminating one of the two normally redundant MSBs in the MSP. However, an overflow on format-adjust will occur when full-scale negative is multiplied by itself, yielding full-scale negative instead of the correct positive product (which is not representable in format-adjusted twos-complement format). To avoid this overflow, disallow X and Y inputs that are both full-scale negative.

The output latches can be bypassed for asynchronous operation by setting the feed-through (FT) line HI. Data previously latched in the output registers is unaffected by FT going HI. If FT is later restored to LO, the output registers will drive the three-state outputs with the product most recently clocked to those registers (even if clocked while FT was HI).

Products are clocked into the MSP and LSP output registers with the rising edges of CLK M and CLK L, respectively. Each of these registers has its own three-state control. A HI on the asynchronous TRIL or TRIM line disables the corresponding LSP or MSP output driver to a high-impedance state. Conversely, a LO on TRIL or TRIM enables the corresponding output driver, driving the output bus.

X & Y INPUT DATA FORMATS	OUTPUT DATA FORMATS	
	MOST SIGNIFICANT PRODUCT	LEAST SIGNIFICANT PRODUCT
bit 11 | 10 | ... | 0	P23 | P22 | ... | P12 | P11 | P10 | ... | P0	
TWOS-COMPLEMENT INTEGER (TCX, TCY = 1)	UNSHIFTED (FA = 1)	
sign -2^{11} | 2^{10} | ... | 2^{0}	sign -2^{23} | 2^{22} | ... | 2^{12} | 2^{11} | 2^{10} | ... | 2^{0}	
	SHIFTED (FA = 0)	
	sign -2^{22} | 2^{21} | ... | 2^{11} | sign -2^{22} | 2^{10} | ... | 2^{0}	
TWOS-COMPLEMENT FRACTIONAL (TCX, TCY = 1)	UNSHIFTED (FA = 1)	
sign -2^{0} | 2^{-1} | ... | 2^{-11}	sign -2^{1} | 2^{0} | ... | 2^{-10} | 2^{-11} | 2^{-12} | ... | 2^{-22}	
	SHIFTED (FA = 0)	
	sign -2^{0} | 2^{-1} | ... | 2^{-11} | sign -2^{0} | 2^{-12} | ... | 2^{-22}	
UNSIGNED-MAGNITUDE INTEGER (TCX, TCY = 0)	UNSHIFTED (FA = 1)	
2^{11} | 2^{10} | ... | 2^{0}	2^{23} | 2^{22} | ... | 2^{12} | 2^{11} | 2^{10} | ... | 2^{0}	
UNSIGNED-MAGNITUDE FRACTIONAL (TCX, TCY = 0)	UNSHIFTED (FA = 1)	
2^{-1} | 2^{-2} | ... | 2^{-12}	2^{-1} | 2^{-2} | ... | 2^{-12} | 2^{-13} | 2^{-14} | ... | 2^{-24}	
MIXED-MODE INTEGER (TCX, TCY mixed)	UNSHIFTED (FA = 1)	
-2^{11} | 2^{10} | ... | 2^{0} & 2^{11} | 2^{10} | ... | 2^{0}	sign -2^{23} | 2^{22} | ... | 2^{12} | 2^{11} | 2^{10} | ... | 2^{0}	
MIXED-MODE FRACTIONAL (TCX, TCY mixed)	UNSHIFTED (FA = 1)	
-2^{0} | 2^{-1} | ... | 2^{-11} & 2^{-1} | 2^{-2} | ... | 2^{-12}	sign -2^{0} | 2^{-1} | ... | 2^{-11} | 2^{-12} | 2^{-13} | ... | 2^{-23}	

Table I. ADSP 1012A Data Formats

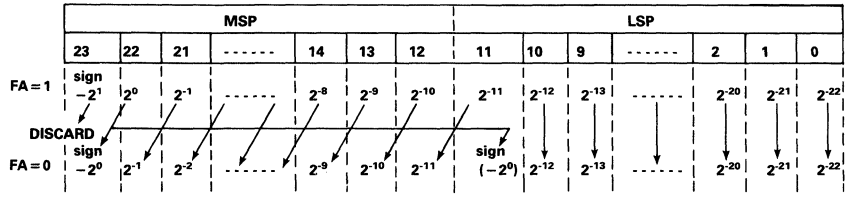


Table II. ADSP-1012A Format Adjust

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 Seconds)	300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1012AJN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1012AKN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1012AJD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1012AKD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1012AJG	0 to +70°C	68-Lead Pin Grid Array	G-68A
ADSP-1012AKG	0 to +70°C	68-Lead Pin Grid Array	G-68A
ADSP-1012ASD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1012ATD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1012ASD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1012ATD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1012ASG	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1012ATG	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1012ASG/883B	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1012ATG/883B	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1012ASE/883B	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1012ATE/883B	−55°C to +125°C	68-Contact LCC	E-68A

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

5

ESD SENSITIVITY

The ADSP-1012A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1012A has been classified as a Class 1 device.

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*.

ADSP-1012A PIN CONFIGURATIONS

DIP
D-64A
N-64A

PIN	FUNCTION	PIN	FUNCTION
1	X7	33	P16
2	X6	34	P17
3	X5	35	P18
4	X4	36	P19
5	X3	37	P20
6	X2	38	P21
7	X1	39	P22
8	X0	40	P23
9	P0	41	TCY
10	P1	42	Y11
11	P2	43	Y10
12	P3	44	Y9
13	P4	45	Y8
14	P5	46	Y7
15	P6	47	Y6
16	P7	48	$+V_{DD}$
17	P8	49	$+V_{DD}$
18	P9	50	$+V_{DD}$
19	P10	51	Y5
20	P11	52	Y4
21	TRIL	53	Y3
22	TRIM	54	Y2
23	GND	55	Y1
24	GND	56	Y0
25	FT	57	TCX
26	FA	58	RND
27	CLK L	59	CLK Y
28	CLK M	60	CLK X
29	P12	61	X11
30	P13	62	X10
31	P14	63	X9
32	P15	64	X8

PIN GRID ARRAY
G-68A

PIN	FUNCTION	PIN	FUNCTION
1	P0	35	TCY
2	P1	36	Y11
3	P2	37	Y10
4	P3	38	Y9
5	P4	39	Y8
6	P5	40	Y7
7	P6	41	Y6
8	P7	42	V_{DD}
9	P8	43	V_{DD}
10	P9	44	V_{DD}
11	P10	45	Y5
12	P11	46	Y4
13	TRIL	47	Y3
14	TRIM	48	Y2
15	GND	49	Y1
16	GND	50	Y0
17	N/C	51	N/C
18	FT	52	TCX
19	FA	53	RND
20	CLK L	54	CLK Y
21	CLK M	55	CLK X
22	P12	56	X11
23	P13	57	X10
24	P14	58	X9
25	P15	59	X8
26	P16	60	X7
27	P17	61	X6
28	P18	62	X5
29	P19	63	X4
30	P20	64	X3
31	P21	65	X2
32	P22	66	X1
33	P23	67	X0
34	N/C	68	N/C

LCC
E-68A

PIN	FUNCTION	PIN	FUNCTION
1	X7	35	P16
2	X6	36	P17
3	X5	37	P18
4	X4	38	P19
5	X3	39	P20
6	X2	40	P21
7	X1	41	P22
8	X0	42	P23
9	N/C	43	N/C
10	P0	44	TCY
11	P1	45	Y11
12	P2	46	Y10
13	P3	47	Y9
14	P4	48	Y8
15	P5	49	Y7
16	P6	50	Y6
17	P7	51	$+V_{DD}$
18	P8	52	$+V_{DD}$
19	P9	53	$+V_{DD}$
20	P10	54	Y5
21	P11	55	Y4
22	TRIL	56	Y3
23	TRIM	57	Y2
24	GND	58	Y1
25	GND	59	Y0
26	N/C	60	N/C
27	FT	61	TCX
28	FA	62	RND
29	CLK L	63	CLK Y
30	CLK M	64	CLK X
31	P12	65	X11
32	P13	66	X10
33	P14	67	X9
34	P15	68	X8

ANALOG DEVICES

16 × 16-Bit CMOS Multiplier

ADSP-1016A

FEATURES

- **16 × 16-Bit Parallel Multiplication**
- **70ns Multiplication Time**
- **225mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Twos-Complement, Unsigned-Magnitude and Mixed-Mode Data Formats**
- **Available in Hermetically Sealed 64-Pin DIP, Hermetically Sealed 68-Pin PGA, Plastic 64-Pin DIP, or 68-Contact LCC**
- **Available Specified to MIL-STD-883, Class B**
- **Pin Compatible with ADSP-1016 and MPY016HJ1**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**
- **General Purpose Computing**

GENERAL DESCRIPTION

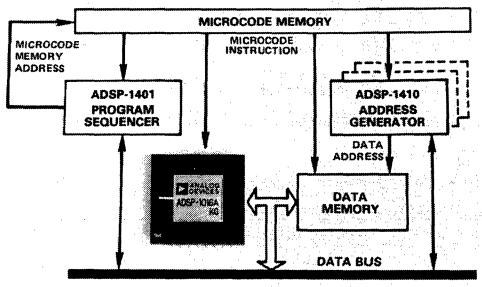
The ADSP-1016A is a high-speed low-power 16 × 16-bit parallel multiplier fabricated in 1.5 micron CMOS.

The ADSP-1016A has two 16-bit input ports, a 16-bit Most Significant Product (MSP) port, and a 16-bit Least Significant Product (LSP) port. Input data is interpreted in twos-complement, unsigned-magnitude, or mixed-mode formats. The ADSP-1016A produces a 32-bit result whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. Both of the product registers have their own independent clock lines and their own independent three-state output controls. Three-state outputs and independently clocked inputs allow the ADSP-1016A to be connected directly to a single 16-bit bus.

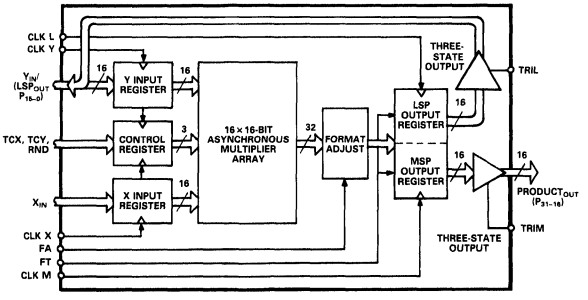
The ADSP-1016A is a pin-for-pin replacement for Analog Devices' ADSP-1016 and is also pin-for-pin compatible in a DIP package with TRW's MPY016HJ1. The ADSP-1016A's multiply time is more than twice as fast as the TRW device.

The power consumption of the ADSP-1016A is 225mW maximum, less than 10% of the power required by equivalent bipolar devices. The differential between the ADSP-1016A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-1016A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1016A

The ADSP-1016A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1016A is available in either a 64-pin hermetically sealed ceramic DIP, a hermetically sealed ceramic 68-pin grid array, a plastic 64-pin DIP, or a 68-contact LCC.



Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1016A J and K Grades Min	ADSP-1016A J and K Grades Max	ADSP-1016A S and T Grades[2] Min	ADSP-1016A 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-1016A J and K Grades Min	ADSP-1016A J and K Grades Max	ADSP-1016A S and T Grades[2] Min	ADSP-1016A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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} = −0.4mA	2.4		2.4		V
V_{OL}	Low-Level Output Voltage	@ V_{DD} = min & I_{OL} = 4.0mA		0.4		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		65		55	mA
I_{DD}	Supply Current–Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1016A J Grade 0 to +70°C Min	J Grade Max	K Grade Min	K Grade Max	S Grade[2] −55°C to +125°C Min	S Grade[2] Max	T Grade[2] Min	T Grade[2] Max	Unit
t_D	Output Delay		20		20		25		25	ns
t_{ENA}	Three-State Enable Delay		20		20		25		25	ns
t_{DIS}	Three-State Disable Delay		20		20		25		25	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_{DS}	Input Data Register Setup Time	25		25		25		25		ns
t_{CS}	Input Controls Setup Time	30		30		30		30		ns
t_H	Input Register Hold Time	2		2		2		2		ns
t_{MC}	Clocked Multiply Time		85		70		95		80	ns
t_{MUC}	Unclocked Multiply Time		105		90		120		105	ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1016A 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 t_{ENA} and t_{DIS} which are indicated in Figure 2.
Specifications subject to change without notice.

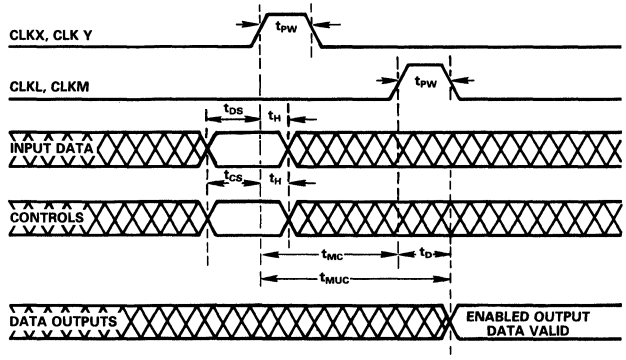


Figure 1. ADSP-1016A Timing Diagram

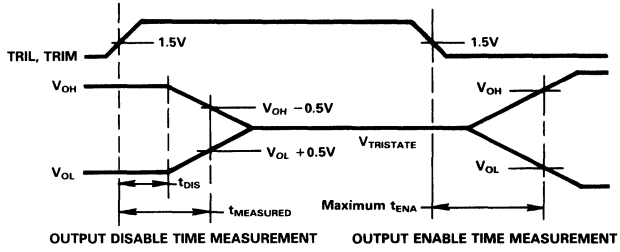


Figure 2. Three-State Disable and Enable Timing

The ADSP-1016A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1016A is available in either a 64-pin hermetically sealed ceramic DIP, a hermetically sealed ceramic 68-pin grid array, a plastic 64-pin DIP, or a 68-contact LCC.

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

METHOD OF OPERATION

The X and Y input registers are positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers by the rising edges of CLKX and CLKY, respectively.

The X and Y input data can be either in twos-complement, unsigned-magnitude, or mixed-mode formats (Table I). Twos-complement input data is indicated by HI (logic 1) levels on the TCX line for X input data and HI levels on the TCY line for Y input data. Unsigned-magnitude X and Y inputs are indicated by LO (logic 0) levels on the TCX and TCY lines, respectively. Outputs will be in the same format as inputs unless the input formats are mixed, in which case the outputs will be in twos-complement representation.

The ADSP-1016A's output is fielded into a 16-bit MSP and a 16-bit LSP. When RND is HI, the MSP will be rounded by adding a binary 1 (with carry) to the MSB of the LSP, consistently rounding toward positive infinity at mid-scale. Truncating the MSP (RND LO) introduces a large-sample statistical bias $-(2^{16}-1)/2$ LSBs of the LSP, while rounding (RND HI) reduces the bias to +1/2LSBs of the LSP.

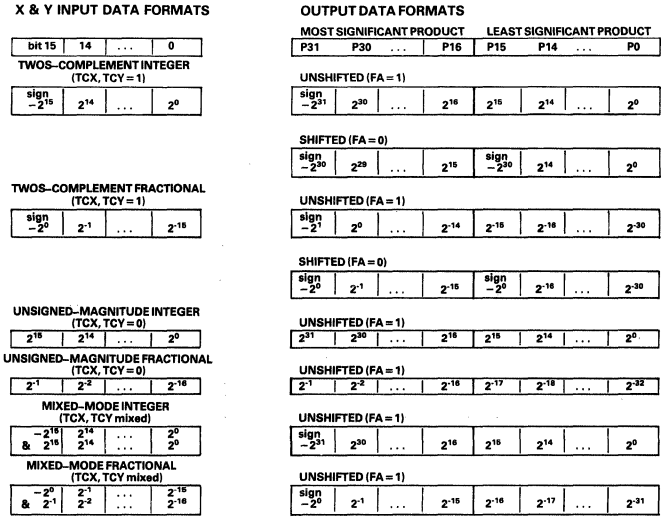


Table I. ADSP-1016A Data Formats

TCX, TCY and RND are registered input controls. TCX and TCY are latched by the rising edges of CLKX and CLKY, respectively. RND is latched by the rising edge of the logical OR of CLKX and CLKY. Be sure that CLKX and CLKY are both LO before attempting to clock in RND.

The asynchronous FA control format-adjusts the output from the multiplier array (Table II). FA must be HI to get a product for unsigned-magnitude or mixed-mode multiplication in a standard format. In a mixed-mode product, the sign bit will be product Bit 31 (P31). For twos-complement multiplications, FA can be LO. If FA is at a LO level, the MSP and MSB of the LSP are left-shifted one bit and the sign bit is duplicated in the MSB of the LSP.

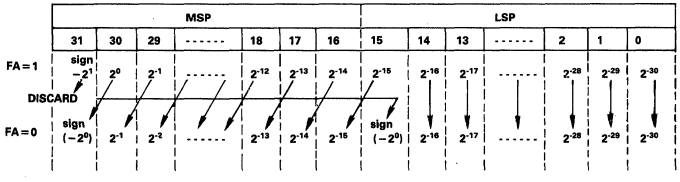
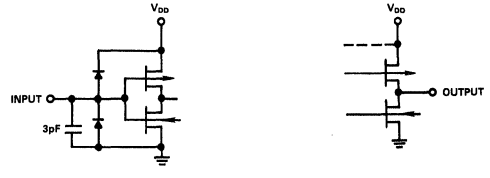


Table II. Format Adjust

Format-adjusting a twos-complement product increases the number of significant bits in the MSP by eliminating one of the two normally redundant MSBs in the MSP. However, an overflow on format-adjust will occur when full-scale negative is multiplied by itself, yielding full-scale negative instead of the correct positive product (which is not representable in format-adjusted twos-complement format). To avoid this overflow, disallow X and Y inputs that are both full-scale negative.

The output latches can be bypassed for asynchronous operation by setting the feedthrough (FT) line HI. Data previously latched in the output registers is unaffected by FT going HI. If FT is later restored to LO, the output registers will drive the three-state outputs with the product most recently clocked to those registers (even if clocked while FT was HI).

Products are clocked into the MSP and LSP output registers with the rising edges of CLKM and CLKL, respectively. Each of these registers has its own three-state control. A HI on the asynchronous TRIL or TRIM lines disables the corresponding LSP or MSP output driver to a high-impedance state. Conversely, a LO on TRIL or TRIM enables the corresponding output driver, driving the output bus.



a. Equivalent Input Circuit *b. Equivalent Output Circuit*

Figure 3.

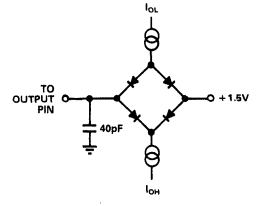


Figure 4. Normal Load for ac Measurements

ESD SENSITIVITY

The ADSP-1016A features proprietary protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1016A has been classified as a Class 1 device.

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.*

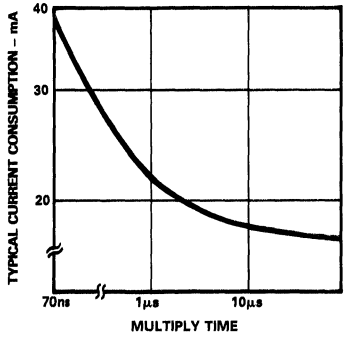


Figure 5. Typical Power Dissipation vs. Frequency

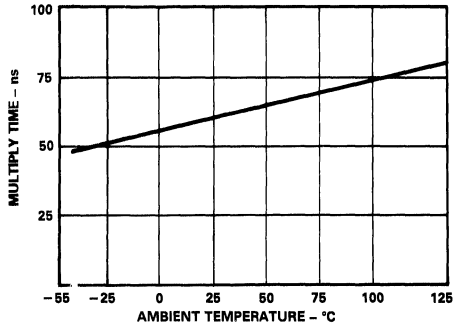


Figure 6. Approx. Multiply Time vs. Temperature

5

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 Seconds)	300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1016AJN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1016AKN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1016AJD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1016AKD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1016AJG	0 to +70°C	68-Lead Pin Grid Array	G-68A
ADSP-1016AKG	0 to +70°C	68-Lead Pin Grid Array	G-68A
ADSP-1016ASD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1016ATD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1016ASD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1016ATD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1016ASG	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1016ATG	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1016ASG/883B	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1016ATG/883B	−55°C to +125°C	68-Lead Pin Grid Array	G-68A
ADSP-1016ASE	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1016ATE	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1016ASE/883B	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1016ATE/883B	−55°C to +125°C	68-Contact LCC	E-68A

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

8×8-Bit CMOS Multiplier/Accumulator

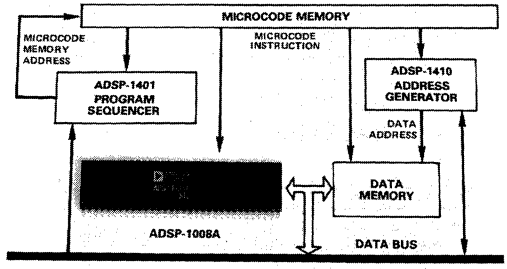
ADSP-1008A

FEATURES

- **8 × 8-Bit Parallel Multiplication/Accumulation**
- **50ns Multiply/Accumulate Time**
- **200mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Twos-Complement or Unsigned-Magnitude**
- **Preloadable Accumulation Registers**
- **Available in Hermetically-Sealed 48-Pin DIP or Plastic 48-Pin DIP**
- **Available Specified to MIL-STD-883, Class B**
- **Pin-Compatible with TDC1008J4**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**
- **Telecommunications**



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1008A

GENERAL DESCRIPTION

The ADSP-1008A is a high-speed, low-power 8 × 8-bit parallel multiplier/accumulator fabricated in 1.5 micron CMOS.

The ADSP-1008A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, an 8-bit Least Significant Product (LSP) port and a 3-bit Extended Product (XTP) port. Inputs can be represented in either twos-complement or unsigned-magnitude formats. The ADSP-1008A produces a 16-bit product whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP. After multiplying, the ADSP-1008A can latch its product directly into the output register or update the output registers with its previous contents added to or subtracted from the product. The output registers can also be initialized prior to multiplication/accumulation with data preloaded from the output ports.

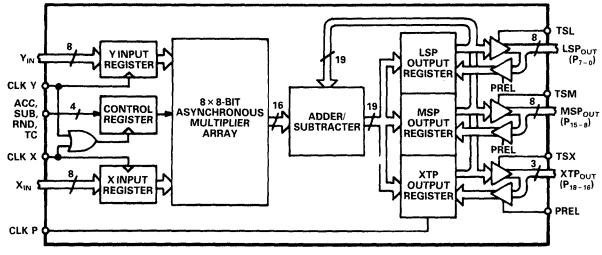
All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Each of the three product registers has its own three-state output control. Three-state outputs and independently clocked inputs allow the ADSP-1008A to be connected directly to a single 8-bit bus.

The ADSP-1008A is a pin-for-pin replacement for Analog Devices' ADSP-1008 and is also pin-for-pin compatible with TRW's TDC1008J4.

The power consumption of the ADSP-1008A is 200mW maximum, less than 10% of the power required by equivalent bipolar devices. The differential between the ADSP-1008A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-1008A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1008A is available for both commercial and MIL temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1008A is available in either a 48-pin hermetically sealed ceramic DIP or a plastic 48-pin DIP.

Word-Slice is a registered trademark of Analog Devices, Inc.



Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1008A J and K Grades Min	ADSP-1008A J and K Grades Max	ADSP-1008A S and T Grades[2] Min	ADSP-1008A 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-1008A J and K Grades Min	ADSP-1008A J and K Grades Max	ADSP-1008A S and T Grades[2] Min	ADSP-1008A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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		0.4		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		40		45	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		25		30	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1008A J Grade 0 to +70°C Min	J Grade Max	K Grade 0 to +70°C Min	K Grade Max	S Grade −55°C to +125°C Min	S Grade Max	T Grade −55°C to +125°C Min	T Grade Max	Unit
t_D	Output Delay		25		25		30		30	ns
t_{ENA}	Three-State Enable Delay		25		20		35		35	ns
t_{DIS}	Three-State Disable Delay		25		20		35		35	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	15		15		15		15		ns
t_H	Input Hold Time	3		3		3		3		ns
t_{MAC}	Multiply/Accumulate Time		60		50		75		60	ns

NOTES

[1]All min and max specifications are over power supply and temperature range indicated.

[2]S and T grades 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-1008A 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 t_{ENA} and t_{DIS} which are as indicated in Figure 2.

Specifications subject to change without notice.

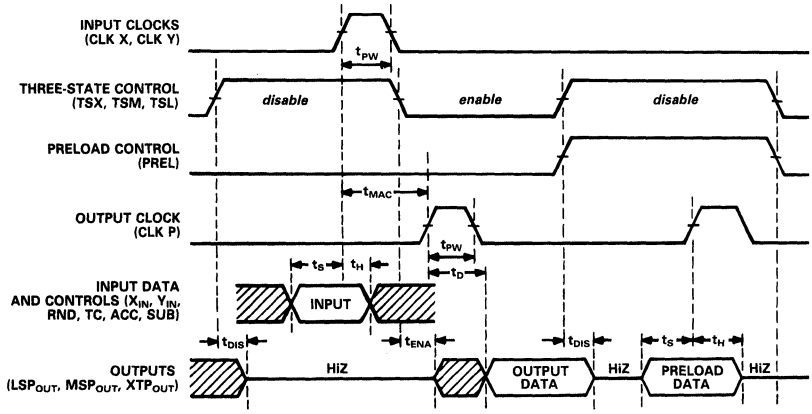


Figure 1. ADSP-1008A Timing Diagram

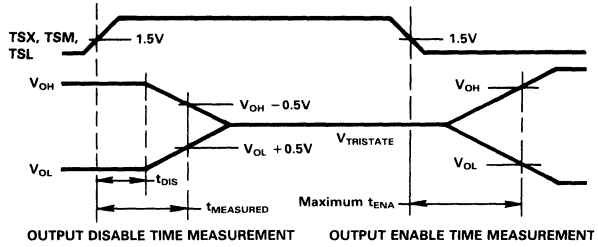


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

METHOD OF OPERATION

The X and Y input registers are positive-edge triggered D-type flip-flops. Input data is loaded to the X and Y registers with the rising edges of CLK X and CLK Y, respectively. The X and Y input data can be represented in either twos-complement or unsigned-magnitude formats. (Mixed-mode is not supported.)

TC, RND, ACC and SUB are registered input controls. Note that these four controls are latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in these controls.

When the registered twos-complement control, TC, is HI (logic 1), the inputs are interpreted as twos-complement numbers. See Table I for the ADSP-1008A's data formats. When TC is LO, the inputs are interpreted as unsigned-magnitude numbers. In both cases, outputs will be in the same format as inputs. No shifting is performed in the ADSP-1008A, so all multiplications, including (twos-complement) negative full scale multiplied by negative full scale, yield valid results.

When the registered round control, RND, is HI, the product will be rounded to the 8 most significant bits by adding a 1 to the MSB of the LSP (which introduces a large-sample statistical bias of +1/2LSB of the LSP).

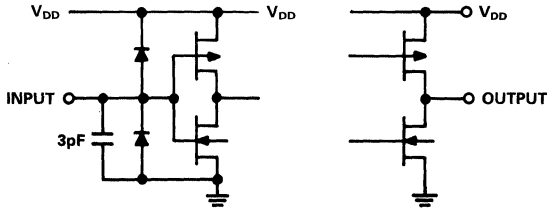
Registered ACC and SUB controls determine whether the product will be latched directly into the output registers or whether they will be updated with the previous contents of the output registers added to or subtracted from the product. If ACC is LO, the product will overwrite the previous contents of the output registers. Holding ACC low at the beginning of a summation avoids the need for a separate operation to clear the output registers. If ACC is HI and SUB is LO, the previous contents of the output registers will be added to the product and stored in the output registers. If ACC is HI and SUB is HI, the previous contents of the output registers will be subtracted from the product and stored in the output registers. Table II displays these conditions in a truth table.

X & Y INPUT DATA								OUTPUT DATA FORMATS																			
BIT							LSB	XTP			MSP								LSP								
7	6	5	4	3	2	1	0	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACTIONAL TWOS COMPLEMENT																											
sign								sign																			
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	-2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
INTEGER TWOS COMPLEMENT																											
sign								sign																			
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	-2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
UNSIGNED MAGNITUDE																											
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table I. Data Formats

ACC	SUB	Function
1	1	$\text{Accumulator}_t = X_t \cdot Y_t - \text{Accumulator}_{t-1}$
1	0	$\text{Accumulator}_t = X_t \cdot Y_t + \text{Accumulator}_{t-1}$
0	X	$\text{Accumulator}_t = X_t \cdot Y_t$

Table II. Function Truth Table



a. Equivalent Input Circuit *b. Equivalent Output Circuit*

Figure 3.

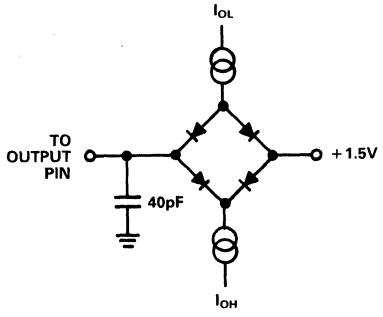


Figure 4. Normal Load for ac Measurements

The accumulation register is partitioned into three words: an 8-bit LSP, an 8-bit MSP and a 3-bit XTP. The 3-bit extension register makes possible summing at least eight large products without overflow. In twos-complement mode, the MSB of the XTP will be the product sign bit. Sign bits, or zeros in the case of unsigned-magnitude, are extended from the MSB of the product to the MSB of the XTP in the adder/subtracter. (Data preloaded to the accumulation registers will not be sign-extended until it is added to or subtracted from a product.)

The rising edge of CLK P latches the LSP, MSP, and XTP into the accumulation registers. Each of these registers has its own three-state control. A HI on the asynchronous TSL, TSM, or TSX line disables the corresponding LSP, MSP or XTP output driver to a high-impedance state. Conversely, a LO on TSL, TSM or TSX enables the corresponding output driver, driving the output bus.

The asynchronous preload control, PREL, can be used to initialize the output registers. In conjunction with TSL, TSM, and TSX, PREL can be used to preload either one, two or all three of the output registers simultaneously. If PREL is HI while either TSL, TSM or TSX is also HI, then the data at the output ports is loaded into the respective output registers on the rising edge of CLK P. See Table III for a truth table of these conditions.

PREL	TSX	TSM	TSL	XTP	MSP	LSP
0	0	0	0	Q	Q	Q
0	0	0	1	Q	Q	Z
0	0	1	0	Q	Z	Q
0	0	1	1	Q	Z	Z
0	1	0	0	Z	Q	Q
0	1	0	1	Z	Q	Z
0	1	1	0	Z	Z	Q
0	1	1	1	Z	Z	Z
1	0	0	0	Z	Z	Z
1	0	0	1	Z	Z	Preload
1	0	1	0	Z	Preload	Z
1	0	1	1	Z	Preload	Preload
1	1	0	0	Preload	Z	Z
1	1	0	1	Preload	Z	Preload
1	1	1	0	Preload	Preload	Z
1	1	1	1	Preload	Preload	Preload

NOTE:
Z = Output buffers at high impedance (output disabled)
Q = Output buffers at low impedance. Contents of output register will be transferred to output pins.
Preload = Output buffers at high impedance. Preload data (PD) supplied externally at output pins will be loaded into the output register at the rising edge of CLK P.

Table III. Preload Truth Table

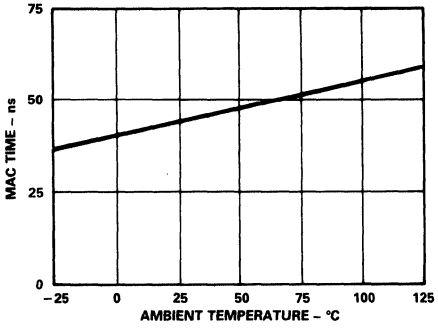


Figure 5. Typical MAC Time vs. Temperature

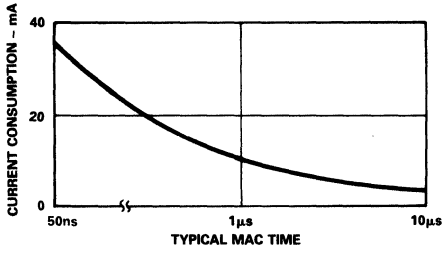


Figure 6. Typical I_{DD} vs. Frequency of Operation

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	−0.3V to 7V
Input Voltage	−0.3 to V_{DD}
Output Voltage	−0.3 to V_{DD}
Operating Temperature Range ($T_{AMBIENT}$)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10 Seconds)	300°C
Junction Temperature	175°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1008AJN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1008AKN	0 to +70°C	48-Pin Plastic DIP	N-48A
ADSP-1008AJD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1008AKD	0 to +70°C	48-Pin Ceramic DIP	D-48A
ADSP-1008ASD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1008ATD	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1008ASD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A
ADSP-1008ATD/883B	−55°C to +125°C	48-Pin Ceramic DIP	D-48A

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

PIN CONFIGURATIONS

PIN	FUNCTION	PIN	FUNCTION
1	P12	25	X3
2	P11	26	X4
3	P10	27	X5
4	P9	28	X6
5	P8	29	X7
6	TSM	30	CLKX
7	CLKP	31	CLKY
8	PREL	32	Y0
9	P7	33	Y1
10	P6	34	Y2
11	P5	35	Y3
12	GND	36	Y4
13	P4	37	V_{CC}
14	P3	38	Y5
15	P2	39	Y6
16	P1	40	Y7
17	P0	41	TC
18	TSL	42	TSX
19	SUB	43	P18
20	ACC	44	P17
21	RND	45	P16
22	X0	46	P15
23	X1	47	P14
24	X2	48	P13

5

ESD SENSITIVITY

The ADSP-1008A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1008A has been classified as a Class 1 device.

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*.

12 × 12-Bit CMOS Multiplier/Accumulator

ADSP-1009A

FEATURES

- **12 × 12-Bit Parallel Multiplication/Accumulation**
- **70ns Multiply/Accumulate Time**
- **375mW Power Dissipation with TTL-Compatible 1.5 Micron CMOS Technology**
- **Pin-Compatible with ADSP-1009, TDC1009J1, and TMC2009J3**
- **Twos Complement or Unsigned Magnitude**
- **Preloadable Accumulation Registers**
- **Available in Hermetically-Sealed 64-Pin DIP, Hermetically-Sealed 68-Pin Grid Array, Plastic 64-Pin DIP, or 68-Contact LCC**
- **Available Specified from −55°C to +125°C Ambient**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**
- **Telecommunications**

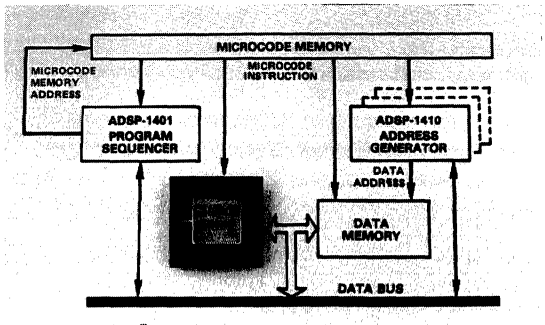
GENERAL DESCRIPTION

The ADSP-1009A is a high-speed, low-power 12 × 12-bit parallel multiplier/accumulator fabricated in 1.5 micron CMOS.

The ADSP-1009A has two 12-bit input ports, a 12-bit Most Significant Product (MSP) port, a 12-bit Least Significant Product (LSP) port, and a 3-bit Extended Product (XTP) port. Inputs can be represented in either twos-complement or unsigned-magnitude formats. The ADSP-1009A produces a 24-bit product whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP. After multiplying, the ADSP-1009A can latch its product directly into the output registers or update the output registers with their previous contents added to or subtracted from the product. The output registers can also be initialized prior to multiplication/accumulation with data preloaded from the output ports.

All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Each of the three product registers has its own three-state output control. Three-state outputs and independently clocked inputs allow the ADSP-1009A to be connected directly to a single 12-bit bus.

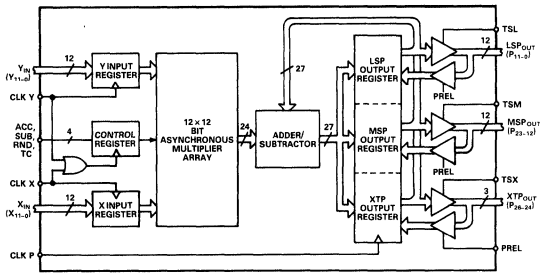
The ADSP-1009A is a pin-for-pin replacement for Analog Devices' ADSP-1009 and is also pin-for-pin compatible in a DIP package with TRW's TDC1009J1 and TMC2009J3. The ADSP-1009A's multiply/accumulate time is over twice as fast as either TRW device.



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1009A

The power consumption of the ADSP-1009A is 375mW maximum, less than 10% of the power required by equivalent bipolar devices. The differential between the ADSP-1009A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, unlike equivalent bipolar devices, the ADSP-1009A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1009A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1009A is available in either a 64-pin hermetically sealed ceramic DIP, a space-saving, hermetically sealed 68-pin grid array, a plastic 64-pin DIP, or a 68-contact LCC.



ADSP-1009A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1009A J and K Grades Min	ADSP-1009A J and K Grades Max	ADSP-1009A S and T Grades[2] Min	ADSP-1009A 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-1009A J and K Grades Min	ADSP-1009A J and K Grades Max	ADSP-1009A S and T Grades[2] Min	ADSP-1009A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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} = −0.4mA	2.4		2.4		V
V_{OL}	Low-Level Output Voltage	@ V_{DD} = min & I_{OL} = 4.0mA		0.4		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		70		75	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1009A 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_D	Output Delay		35		35		35		35	ns
t_{ENA}	Three-State Enable Delay		25		25		30		30	ns
t_{DIS}	Three-State Disable Delay		20		20		25		25	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	20		20		20		20		ns
t_H	Input Hold Time	2		2		2		2		ns
t_{MAC}	Multiply/Accumulate Time		85		70		100		85	ns

NOTES
[1]All min and max specifications are over power supply and temperature range indicated.
[2]S and T grades 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-1009A 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 t_{ENA} and t_{DIS} which are as indicated in Figure 2.
Specifications subject to change without notice.

ESD SENSITIVITY

The ADSP-1009A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1009A has been classified as a Class 1 device.

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*.

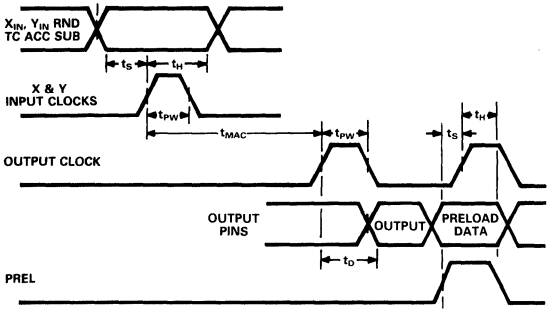


Figure 1. ADSP-1009A Timing Diagram

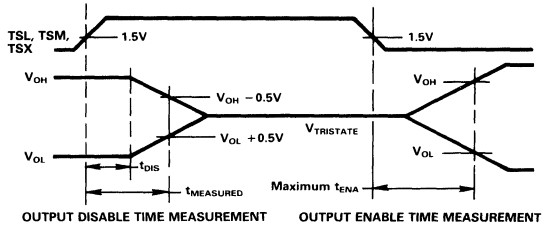


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

METHOD OF OPERATION

The X and Y input registers are positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers with the rising edges of CLK X and CLK Y, respectively. The X and Y input data can be represented in either twos-complement or unsigned-magnitude formats. (Mixed-mode is not supported.)

TC, RND, ACC, and SUB are registered input controls. Note that these four controls are latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in these controls.

When the registered twos-complement control, TC, is HI (logic 1), the inputs are interpreted as twos-complement numbers. (See Table I for the ADSP-1009A's data formats.) When TC is LO, the inputs are interpreted as unsigned-magnitude numbers. In both cases, outputs will be in the same format as inputs. No shifting is performed in the ADSP-1009A, so all multiplications, including (twos-complement) negative full scale multiplied by negative full scale, yield valid results.

When the registered round control, RND, is HI, the product will be rounded to the 12 most significant bits by adding a binary 1 to the MSB of the LSP, consistently rounding toward positive infinity. Truncating the MSP (RND LO) introduces a large-sample statistical bias of $-(2^{12}-1)/2$ LSBs of the LSP, while rounding (RND HI) reduces the bias to +1/2LSBs of the LSP.

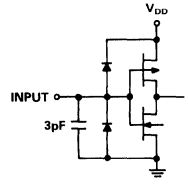


Figure 4. Equivalent Output Circuits

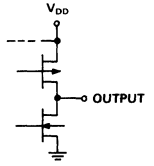


Figure 3. Equivalent Input Circuits

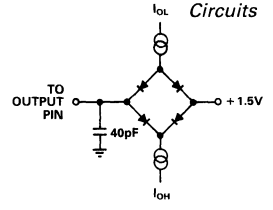


Figure 5. Normal Load for ac Measurements

X & Y INPUT DATA FORMATS	OUTPUT DATA FORMATS
	XTP: 26 25 24 | MSP: 23 22 21 --- 14 13 12 | LSP: 11 10 9 --- 2 1 0
11 10 9 --- 2 1 0	
INTEGER TWOS COMPLEMENT (TC = 1)	
sign (-2^{11}) 2^{10} 2^{9} --- 2^{2} 2^{1} 2^{0}	sign -2^{26} 2^{25} 2^{24} 2^{23} 2^{22} 2^{21} --- 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} --- 2^{2} 2^{1} 2^{0}
FRACTIONAL TWOS COMPLEMENT (TC = 1)	
sign (-2^{0}) 2^{-1} 2^{-2} --- 2^{-9} 2^{-10} 2^{-11}	sign -2^{4} 2^{3} 2^{2} 2^{1} 2^{0} 2^{-1} --- 2^{-8} 2^{-9} 2^{-10} 2^{-11} 2^{-12} 2^{-13} --- 2^{-20} 2^{-21} 2^{-22}
UNSIGNED MAGNITUDE INTEGER (TC = 0)	
2^{11} 2^{10} 2^{9} --- 2^{2} 2^{1} 2^{0}	2^{26} 2^{25} 2^{24} 2^{23} 2^{22} 2^{21} --- 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} --- 2^{2} 2^{1} 2^{0}

Table I. Data Formats

Registered ACC and SUB controls determine whether the product will be latched directly into the output registers or whether they will be updated with the previous contents of the output registers added to or subtracted from the product. If ACC is LO, the product will overwrite the previous contents of the output registers. Holding ACC low at the beginning of a summation avoids the need for a separate operation to clear the output registers. If ACC is HI and SUB is LO, the previous contents of the output registers will be added to the product and stored in the output registers. If ACC is HI and SUB is HI, the previous contents of the output registers will be subtracted from the product and stored in the output registers. (Table II displays these conditions in a truth table.)

ACC	SUB	Function
1	1	$Accumulator_t = X_t \cdot Y_t - Accumulator_{t-1}$
1	0	$Accumulator_t = X_t \cdot Y_t + Accumulator_{t-1}$
0	X	$Accumulator_t = X_t \cdot Y_t$

Table II. Function Truth Table

The accumulation register is partitioned into three words: a 12-bit LSP, a 12-bit MSP, and a 3-bit XTP. The 3-bit extension register makes possible summing at least eight large products without overflow. In twos-complement mode, the MSB of the XTP will be the product sign bit. Sign bits, or zeros in the case of unsigned-magnitude, are extended from the MSB of the product to the MSB of the XTP in the adder/subtracter. (Data preloaded to the accumulation registers will not be sign-extended until it is added to or subtracted from a product.)

The rising edge of CLK P latches the LSP, MSP, and XTP into the accumulation registers. Each of these registers has its own three-state control. A HI on the asynchronous TSL, TSM, or TSX line disables the corresponding LSP, MSP, or XTP output driver to a high-impedance state. Conversely, a LO on TSL, TSM, or TSX enables the corresponding output driver, driving the output bus.

The asynchronous preload control, PREL, can be used to initialize the output registers. In conjuction with TSL, TSM, and TSX, PREL can be used to preload either one, two or all three of the output registers simultaneously. If PREL is HI while either TSL, TSM, or TSX is also HI, then the data at the output ports is loaded into the respective output registers on the rising edge of CLK P. (See Table III for a truth table of these conditions.)

ABSOLUTE MAXIMUM RATINGS

Supply Voltage . . . . . −0.3V to 7V
Input Voltage . . . . . −0.3 to V_{DD}
Output Voltage . . . . . −0.3 to V_{DD}
Operating Temperature Range ($T_{AMBIENT}$) . −55°C to +125°C
Storage Temperature Range . . . . . −65°C to +150°C
Lead Temperature (10sec) . . . . . 300°C

PREL	TSX	TSM	TSL	XTP	MSP	LSP
0	0	0	0	Q	Q	Q
0	0	0	1	Q	Q	Z
0	0	1	0	Q	Z	Q
0	0	1	1	Q	Z	Z
0	1	0	0	Z	Q	Q
0	1	0	1	Z	Q	Z
0	1	1	0	Z	Z	Q
0	1	1	1	Z	Z	Z
1	0	0	0	Z	Z	Z
1	0	0	1	Z	Z	Preload
1	0	1	0	Z	Preload	Z
1	0	1	1	Z	Preload	Preload
1	1	0	0	Preload	Z	Z
1	1	0	1	Preload	Z	Preload
1	1	1	0	Preload	Preload	Z
1	1	1	1	Preload	Preload	Preload

NOTE:
Z = Output buffers at high impedance (output disabled)
Q = Output buffers at low impedance. Contents of output register will be transferred to output pins.
Preload = Output buffers at high impedance. Preload data supplied externally at output pins will be loaded into the output register at the rising edge of CLK P.

Table III. Preload Truth Table

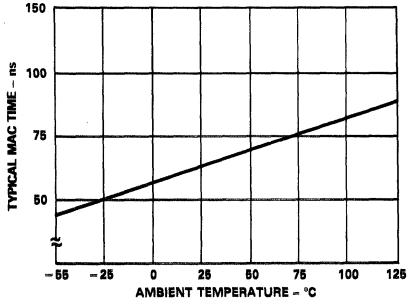


Figure 6. Approx. Multiply Time vs. Temperature

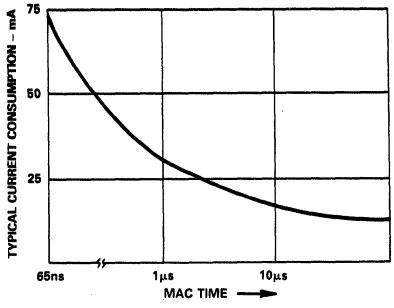


Figure 7. Typical I_{DD} vs. Frequency of Operation

PIN CONFIGURATIONS

DIP
D-64A
N-64A

PIN	FUNCTION	PIN	FUNCTION
1	X4	33	P19
2	X3	34	P20
3	X2	35	P21
4	X1	36	P22
5	X0	37	P23
6	ACC	38	P24
7	SUB	39	P25
8	RND	40	P26
9	TSL	41	TSX
10	P0	42	TC
11	P1	43	Y11
12	P2	44	Y10
13	P3	45	Y9
14	P4	46	Y8
15	P5	47	Y7
16	GND	48	Y6
17	P6	49	V_{DD}
18	P7	50	Y5
19	P8	51	Y4
20	P9	52	Y3
21	P10	53	Y2
22	P11	54	Y1
23	CLKP	55	Y0
24	PREL	56	CLKY
25	TSM	57	CLKX
26	P12	58	X11
27	P13	59	X10
28	P14	60	X9
29	P15	61	X8
30	P16	62	X7
31	P17	63	X6
32	P18	64	X5

PIN GRID ARRAY
G-68A

PIN	FUNCTION	PIN	FUNCTION
1	TSL	35	TSX
2	P0	36	TC
3	P1	37	Y11
4	P2	38	Y10
5	P3	39	Y9
6	P4	40	Y8
7	P5	41	Y7
8	GND	42	Y6
9	P6	43	V_{DD}
10	P7	44	Y5
11	P8	45	Y4
12	P9	46	Y3
13	P10	47	Y2
14	P11	48	Y1
15	CLKP	49	Y0
16	PREL	50	CLKY
17	N/C	51	N/C
18	TSM	52	CLKX
19	P12	53	X11
20	P13	54	X10
21	P14	55	X9
22	P15	56	X8
23	P16	57	X7
24	P17	58	X6
25	P18	59	X5
26	P19	60	X4
27	P20	61	X3
28	P21	62	X2
29	P22	63	X1
30	P23	64	X0
31	P24	65	ACC
32	P25	66	SUB
33	P26	67	RND
34	N/C	68	N/C

LCC
E-68A

PIN	FUNCTION	PIN	FUNCTION
1	X4	35	P19
2	X3	36	P20
3	X2	37	P21
4	X1	38	P22
5	X0	39	P23
6	ACC	40	P24
7	SUB	41	P25
8	RND	42	P26
9	N/C	43	N/C
10	TSL	44	TSX
11	P0	45	TC
12	P1	46	Y11
13	P2	47	Y10
14	P3	48	Y9
15	P4	49	Y8
16	P5	50	Y7
17	GND	51	Y6
18	P6	52	V_{DD}
19	P7	53	Y5
20	P8	54	Y4
21	P9	55	Y3
22	P10	56	Y2
23	P11	57	Y1
24	CLKP	58	Y0
25	PREL	59	CLKY
26	N/C	60	N/C
27	TSM	61	CLKX
28	P12	62	X11
29	P13	63	X10
30	P14	64	X9
31	P15	65	X8
32	P16	66	X7
33	P17	67	X6
34	P18	68	X5

5

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1009AKD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1009AKG	0 to +70°C	68-Pin Grid Array	G-68A
ADSP-1009AKN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1009AJD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1009AJG	0 to +70°C	68-Pin Grid Array	G-68A
ADSP-1009AJN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1009ATD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1009ATG	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1009ASD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1009ASG	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1009ATD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1009ATG/883B	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1009ASD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1009ASG/883B	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1009ASE	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1009ASE/883B	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1009ATE	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1009ATE/883B	−55°C to +125°C	68-Contact LCC	E-68A

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

ANALOG DEVICES

16 × 16-Bit CMOS Multiplier/Accumulator

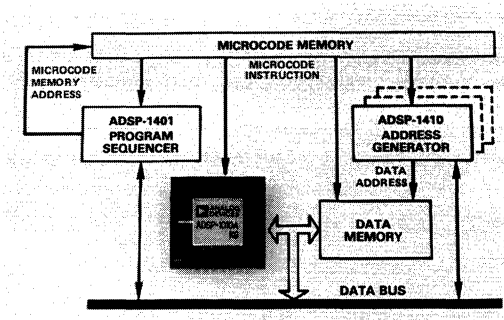
ADSP-1010A

FEATURES

- **16 × 16-Bit Parallel Multiplication/Accumulation**
- **75ns Multiply/Accumulate Time**
- **400mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Twos Complement or Unsigned Magnitude**
- **Preloadable Accumulation Registers**
- **Available in Hermetically Sealed 64-Pin DIP, Hermetically Sealed 68-Pin Grid Array, Plastic 64-Pin DIP, or 68-Contact LCC**
- **Available Specified to MIL-STD-883, Class B**
- **Pin-Compatible with ADSP-1010, TDC1010J1, TMC2010J3, and TMC2110J3**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**
- **Telecommunications**



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1010A

GENERAL DESCRIPTION

The ADSP-1010A is a high-speed, low-power 16 × 16-bit parallel multiplier/accumulator fabricated in 1.5 micron CMOS. The faster ADSP-1010B, fabricated in 1.0 micron CMOS, is described elsewhere in this Databook.

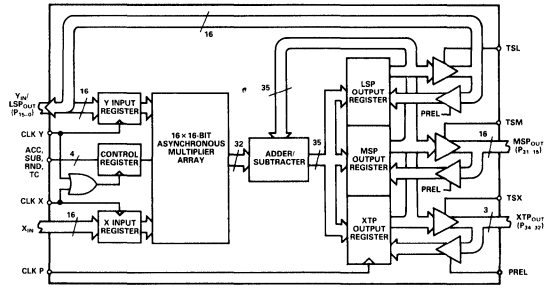
The ADSP-1010A has two 16-bit input ports, a 16-bit Most Significant Product (MSP) port, a 16-bit Least Significant Product (LSP) port, and a 3-bit Extended Product (XTP) port. The LSP output port is a bidirectional port shared with the Y input port. Inputs can be represented in either twos-complement or unsigned-magnitude formats. The ADSP-1010A produces a 32-bit product whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP. After multiplying, the ADSP-1010A can latch its product directly into the output register or update the output registers with its previous contents added to or subtracted from the product. The output registers can also be initialized prior to multiplication/accumulation with data preloaded from the output ports.

All input pins are ESD-protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Each of the three product registers has its own three-state output control. Three-state outputs and independently clocked inputs allow the ADSP-1010A to be connected directly to a single 16-bit bus.

The ADSP-1010A is a pin-for-pin replacement for Analog Devices' ADSP-1010 and is also pin-for-pin compatible in a DIP package with TRW's TDC1010J1, TMC2010J3 and TMC2110J3.

The power consumption of the ADSP-1010A is 400mW maximum, less than 10% of the power required by equivalent bipolar devices. The differential between the ADSP-1010A's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, unlike equivalent bipolar devices, the ADSP-1010A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1010A is available for both commercial and MIL temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1010A is available in either a 64-pin hermetically-sealed ceramic DIP, a space-saving, hermetically-sealed 68-pin grid array, a plastic 64-pin DIP, or a 68-contact LCC.



Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1010A J and K Grades Min	ADSP-1010A J and K Grades Max	ADSP-1010A S and T Grades[2] Min	ADSP-1010A 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-1010A J and K Grades Min	ADSP-1010A J and K Grades Max	ADSP-1010A S and T Grades[2] Min	ADSP-1010A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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		0.4		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		80		100	mA
I_{DD}	Supply Current–Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1010A 0 to +70°C J Grade Min	J Grade Max	K Grade Min	K Grade Max	−55°C to +125°C S Grade Min	S Grade Max	T Grade Min	T Grade Max	Unit
t_D	Output Delay		30		30		40		40	ns
t_{ENA}	Three State Enable Delay		25		25		35		35	ns
t_{DIS}	Three State Disable Delay		25		25		35		35	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	15		15		20		20		ns
t_H	Input Hold Time	3		3		3		3		ns
t_{MAC}	Multiply/Accumulate Time		85		75		100		90	ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1010A 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 t_{ENA} and t_{DIS} which are as indicated in Figure 2.

Specifications subject to change without notice.

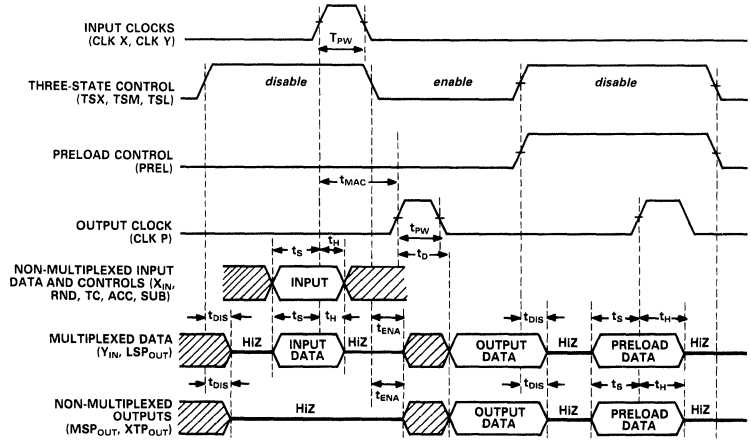


Figure 1. ADSP-1010A Timing Diagram

METHOD OF OPERATION

The X and Y input registers are positive-edge triggered D-type flip-flops. Input data is loaded to the X and Y registers with the rising edges of CLK X and CLK Y, respectively. The X and Y input data can be represented in either twos-complement or unsigned-magnitude formats. (Mixed-mode is not supported.)

TC, RND, ACC, and SUB are registered input controls. Note that these four controls are latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in these controls.

When the registered twos-complement control, TC, is HI (logic 1), the inputs are interpreted as twos-complement numbers. (See Table I for the ADSP-1010A's data formats.) When TC is LO, the inputs are interpreted as unsigned magnitude numbers. In both cases, outputs will be in the same format as inputs. No shifting is performed in the ADSP-1010A, so all multiplications, including (twos-complement) negative full scale multiplied by negative full scale, yield valid results.

When the registered RND control is HI, the MSP will be rounded by adding a binary 1 (with carry) to the most significant bit (MSB) of the LSP, consistently rounding toward positive infinity at mid-scale. Truncating the MSP (RND LO) introduces a large-sample statistical bias into the MSP of $-(2^{16}-1)/2$ times the LSB of the LSP, while rounding (RND HI) reduces the bias to $+1/2$ times the LSB of the LSP.

Registered ACC and SUB controls determine whether the product will be latched directly into the output registers or whether they will be updated with the previous contents of the output registers added to or subtracted from the product. If ACC is LO, the product will overwrite the previous contents of the output registers. Holding ACC low at the beginning of a summation avoids the need for a separate operation to clear the output registers. If ACC is HI and SUB is LO, the previous contents of the output registers will be added to the product and stored in the output registers. If ACC is HI and SUB is HI, the previous contents of the output registers will be subtracted from the product and stored in the output registers. (Table II displays these conditions in a truth table.)

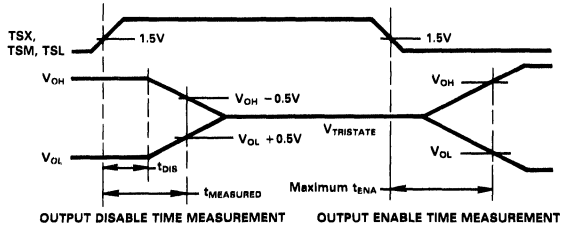


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

5

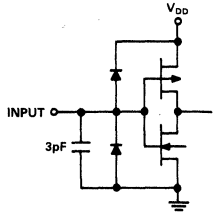


Figure 3. Equivalent Input Circuits

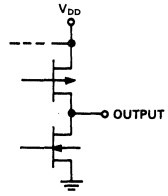


Figure 4. Equivalent Output Circuits

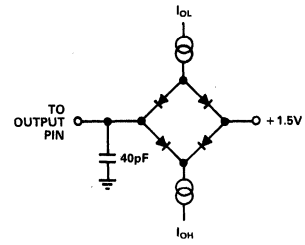


Figure 5. Normal Load for ac Measurements

X & Y INPUT DATA FORMATS							OUTPUT DATA FORMATS																	
							XTP			MSP							LSP							
15	14	13	------	2	1	0	34	33	32	31	30	29	------	18	17	16	15	14	13	------	2	1	0
INTEGER TWOS COMPLEMENT (TC = 1)																								
sign																								
(-2^{15})	2^{14}	2^{13}	------	2^2	2^1	2^0	-2^{34}	2^{33}	2^{32}	2^{31}	2^{30}	2^{29}	------	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0
FRACTIONAL TWOS COMPLEMENT (TC = 1)																								
sign																								
(-2^0)	2^{-1}	2^{-2}	------	2^{-13}	2^{-14}	2^{-15}	-2^4	2^3	2^2	2^1	2^0	2^{-1}	------	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	------	2^{-28}	2^{-29}	2^{-30}
UNSIGNED MAGNITUDE (INTEGER) (TC = 0)																								
2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0	2^{34}	2^{33}	2^{32}	2^{31}	2^{30}	2^{29}	------	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0

Table I. Data Formats

The accumulation register is partitioned into three words: a 16-bit LSP, a 16-bit MSP, and a 3-bit XTP. The 3-bit extension register makes possible summing at least eight large products without overflow. In twos-complement mode, the MSB of the XTP will be the product sign bit. Sign bits, or zeros in the case of unsigned-magnitude, are extended from the MSB of the product to the MSB of the XTP in the adder/subtracter. (Data preloaded to the accumulation registers will not be sign-extended until it is added to or subtracted from a product.)

The rising edge of CLK P latches the LSP, MSP, and XTP into the accumulation registers. Each of these registers has its own three-state control. A HI on the asynchronous TSL, TSM, or TSX line disables the corresponding LSP, MSP, or XTP output driver to a high-impedance state. Conversely, a LO on TSL, TSM, or TSX enables the corresponding output driver, driving the output bus.

The asynchronous preload control, PREL, can be used to initialize the output registers. In conjuction with TSL, TSM, and TSX, PREL can be used to preload either one, two or all three of the output registers simultaneously. If PREL is HI while either TSL, TSM, or TSX is also HI, then the data at the output ports is loaded into the respective output registers on the rising edge of CLK P. (See Table III for a truth table of these conditions.)

ABSOLUTE MAXIMUM RATINGS

Supply Voltage . . . . . −0.3V to 7V
Input Voltage . . . . . −0.3 to V_{DD}
Output Voltage . . . . . −0.3 to V_{DD}
Operating Temperature Range ($T_{AMBIENT}$) . −55°C to +125°C
Storage Temperature Range . . . . . −65°C to +150°C
Lead Temperature (10sec) . . . . . 300°C

ACC	SUB	Function
1	1	$Accumulator_t = X_t \cdot Y_t - Accumulator_{t-1}$
1	0	$Accumulator_t = X_t \cdot Y_t + Accumulator_{t-1}$
0	X	$Accumulator_t = X_t \cdot Y_t$

Table II. Function Truth Table

PREL	TSX	TSM	TSL	XTP	MSP	LSP
0	0	0	0	Q	Q	Q
0	0	0	1	Q	Q	Z
0	0	1	0	Q	Z	Q
0	0	1	1	Q	Z	Z
0	1	0	0	Z	Q	Q
0	1	0	1	Z	Q	Z
0	1	1	0	Z	Z	Q
0	1	1	1	Z	Z	Z
1	0	0	0	Z	Z	Z
1	0	0	1	Z	Z	Preload
1	0	1	0	Z	Preload	Z
1	0	1	1	Z	Preload	Preload
1	1	0	0	Preload	Z	Z
1	1	0	1	Preload	Z	Preload
1	1	1	0	Preload	Preload	Z
1	1	1	1	Preload	Preload	Preload

NOTE:
Z = Output buffers at high impedance (output disabled)
Q = Output buffers at low impedance. Contents of output register will be transferred to output pins.
Preload = Output buffers at high impedance. Preload data supplied externally at output pins will be loaded into the output register at the rising edge of CLK P.

Table III. Preload Truth Table

ESD SENSITIVITY

The ADSP-1010A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1010A has been classified as a Class 1 device.

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*.

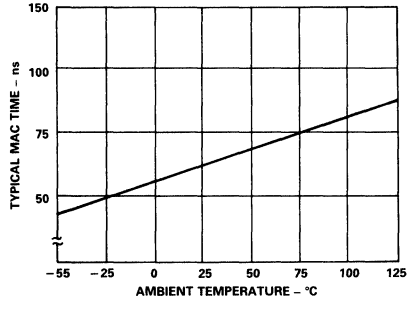


Figure 6. Approx. Multiply Time vs. Temperature

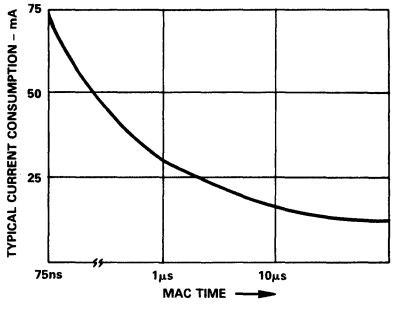


Figure 7. Typical I_{DD} vs. Frequency of Operation

5

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1010AJN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1010AKN	0 to +70°C	64-Pin Plastic DIP	N-64A
ADSP-1010AJD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1010AKD	0 to +70°C	64-Pin Ceramic DIP	D-64A
ADSP-1010AJG	0 to +70°C	68-Pin Grid Array	G-68A
ADSP-1010AKG	0 to +70°C	68-Pin Grid Array	G-68A
ADSP-1010ASD	–55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ATD	–55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ASD/883B	–55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ATD/883B	–55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ASG	–55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ATG	–55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ASG/883B	–55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ATG/883B	–55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ASE	–55°C to +125°C	68-Contact LCC	E-68A
ADSP-1010ATE	–55°C to +125°C	68-Contact LCC	E-68A
ADSP-1010ASE/883B	–55°C to +125°C	68-Contact LCC	E-68A
ADSP-1010ATE/883B	–55°C to +125°C	68-Contact LCC	E-68A

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

ADSP-1010A PIN CONFIGURATIONS

DIP
D-64A
N-64A

PIN	FUNCTION	PIN	FUNCTION
1	X6	33	P24
2	X5	34	P25
3	X4	35	P26
4	X3	36	P27
5	X2	37	P28
6	X1	38	P29
7	X0	39	P30
8	Y0, P0	40	P31
9	Y1, P1	41	P32
10	Y2, P2	42	P33
11	Y3, P3	43	P34
12	Y4, P4	44	CLK P
13	Y5, P5	45	TSM
14	Y6, P6	46	PREL
15	Y7, P7	47	TSX
16	GND	48	TC
17	Y8, P8	49	V_{DD}
18	Y9, P9	50	CLK Y
19	Y10, P10	51	CLK X
20	Y11, P11	52	ACC
21	Y12, P12	53	SUB
22	Y13, P13	54	RND
23	Y14, P14	55	TSL
24	Y15, P15	56	X15
25	P16	57	X14
26	P17	58	X13
27	P18	59	X12
28	P19	60	X11
29	P20	61	X10
30	P21	62	X9
31	P22	63	X8
32	P23	64	X7

PIN GRID ARRAY
G-68A

PIN	FUNCTION	PIN	FUNCTION
1	Y1, P1	35	P32
2	Y2, P2	36	P33
3	Y3, P3	37	P34
4	Y4, P4	38	CLK P
5	Y5, P5	39	TSM
6	Y6, P6	40	PREL
7	Y7, P7	41	TSX
8	GND	42	TC
9	Y8, P8	43	V_{DD}
10	Y9, P9	44	CLK Y
11	Y10, P10	45	CLK X
12	Y11, P11	46	ACC
13	Y12, P12	47	SUB
14	Y13, P13	48	RND
15	Y14, P14	49	TSL
16	Y15, P15	50	X15
17	N/C	51	N/C
18	P16	52	X14
19	P17	53	X13
20	P18	54	X12
21	P19	55	X11
22	P20	56	X10
23	P21	57	X9
24	P22	58	X8
25	P23	59	X7
26	P24	60	X6
27	P25	61	X5
28	P26	62	X4
29	P27	63	X3
30	P28	64	X2
31	P29	65	X1
32	P30	66	X0
33	P31	67	Y0, P0
34	N/C	68	N/C

LCC
E-68A

PIN	FUNCTION	PIN	FUNCTION
1	X6	35	P24
2	X5	36	P25
3	X4	37	P26
4	X3	38	P27
5	X2	39	P28
6	X1	40	P29
7	X0	41	P30
8	Y0, P0	42	P31
9	N/C	43	N/C
10	Y1, P1	44	P32
11	Y2, P2	45	P33
12	Y3, P3	46	P34
13	Y4, P4	47	CLK P
14	Y5, P5	48	TSM
15	Y6, P6	49	PREL
16	Y7, P7	50	TSX
17	GND	51	TC
18	Y8, P8	52	V_{DD}
19	Y9, P9	53	CLK Y
20	Y10, P10	54	CLK X
21	Y11, P11	55	ACC
22	Y12, P12	56	SUB
23	Y13, P13	57	RND
24	Y14, P14	58	TSL
25	Y15, P15	59	X15
26	N/C	60	N/C
27	P16	61	X14
28	P17	62	X13
29	P18	63	X12
30	P19	64	X11
31	P20	65	X10
32	P21	66	X9
33	P22	67	X8
34	P23	68	X7

16×16-Bit CMOS Multiplier/Accumulator

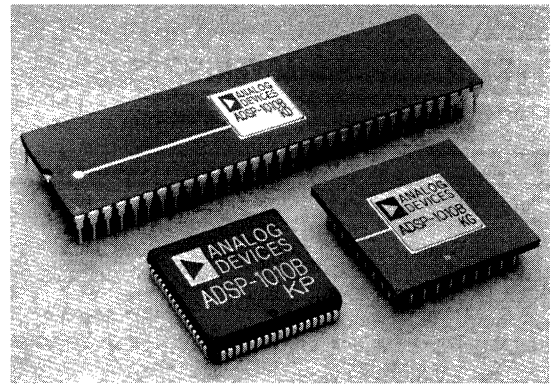
ADSP-1010B

FEATURES

- **Higher-Speed Version of ADSP-1010A**
- **16×16-Bit Parallel Multiplication/Accumulation**
- **45ns Multiply/Accumulate Time**
- **170mW Power Dissipation with 10MHz Clock**
- **Twos Complement or Unsigned Magnitude**
- **Preloadable Accumulation Registers**
- **Available in Hermetically Sealed 64-Pin Ceramic DIP, Hermetically Sealed 68-Pin Grid Array or 68-Lead PLCC**
- **Available Specified to MIL-STD-883, Class B**
- **Pin-Compatible with ADSP-1010, ADSP-1010A, TDC1010J1, TMC2010J3 and TMC2110J3**

APPLICATIONS

- **Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Image Processing**
- **Telecommunications**



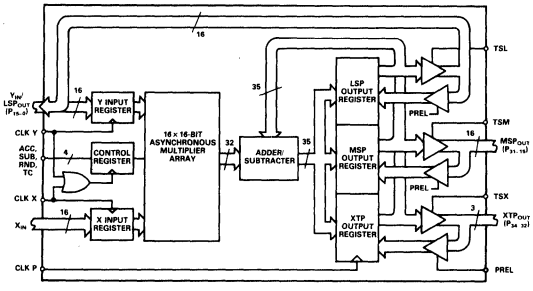
GENERAL DESCRIPTION

The ADSP-1010B is a high speed, low power 16×16-bit parallel multiplier/accumulator fabricated in 1.0 micron CMOS. The ADSP-1010B is a pin-for-pin replacement for Analog Devices' ADSP-1010 and ADSP-1010A and is also pin-for-pin compatible with TRW's TDC1010J1, TMC2010J3, and TMC2110J3. The ADSP-1010B consumes the same power as the ADSP-1010A at the rated maximum clock rate of the ADSP-1010A.

The ADSP-1010B has two 16-bit input ports, a 16-bit most significant product (MSP) port, a 16-bit least significant product (LSP) port and a 3-bit extended product (XTP) port. The LSP output port is a bidirectional port shared with the Y input port. Inputs can be represented in either twos complement or unsigned magnitude formats. The ADSP-1010B produces a 32-bit product whose MSP can be rounded with a control which causes a 1 to be added to the most significant bit (MSB) of the LSP. After multiplying, the ADSP-1010B can latch its product directly into the output register or update the output registers with its previous contents added to or subtracted from the product. The output registers can also be initialized prior to multiplication/accumulation with data preloaded from the output ports.

All input pins are ESD protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Each of the three product registers has its own three-state output control. Three-state outputs and independently clocked inputs allow the ADSP-1010B to be connected directly to a single 16-bit bus.

The ADSP-1010B is available for both commercial and MIL temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. Additionally, the ADSP-1010B is available in either a 64-pin hermetically sealed ceramic DIP, a space-saving, hermetically sealed 68-pin grid array or a 68-lead PLCC.



Functional Block Diagram

5

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

		ADSP-1010B J and K Grades		ADSP-1010B S and T Grades[2]		
Parameter		**Min**	**Max**	**Min**	**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

			ADSP-1010B J and K Grades		ADSP-1010B S and T Grades[2]		
Parameter		**Test Conditions**	**Min**	**Max**	**Min**	**Max**	**Unit**
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		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		0.4		0.4	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; V_{IN} = 0 to 3V		110[3]		125	mA
I_{DD}	Supply Current Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[4]

		ADSP-1010B J Grade 0 to +70°C		ADSP-1010B K Grade 0 to +70°C		ADSP-1010B S Grade[2] −55°C to +125°C		ADSP-1010B T Grade[2] −55°C to +125°C		
Parameter		**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
t_{MAC}	Multiply/Accumulate Time		55		45		65		55	ns
t_D	Output Delay		25		25		30		30	ns
t_{ENA}	Three-State Enable Delay		25		25		30		30	ns
t_{DIS}	Three-State Disable Delay		25		25		30		30	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	15		15		20		20		ns
t_H	Input Hold Time	3		3		3		3		ns

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-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1010B can be found in Analog Devices' Military Databook.

[3]Guaranteed but not tested.

[4]Input levels are GND and 3.0V. Rise times are 5ns. Input timing reference levels and output reference levels are 1.5V, except for t_{ENA} and t_{DIS} which are indicated in Figure 2.

Specifications subject to change without notice.

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline	Part Number	Temperature Range	Package	Package Outline
ADSP-1010BJP	0 to +70°C	68-Lead PLCC	P-68	ADSP-1010BTD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010BKP	0 to +70°C	68-Lead PLCC	P-68	ADSP-1010BSD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010BJD	0 to +70°C	64-Pin Ceramic DIP	D-64A	ADSP-1010BTD/883B	−55°C to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010BKD	0 to +70°C	64-Pin Ceramic DIP	D-64A	ADSP-1010BSG	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010BJG	0 to +70°C	68-Pin Grid Array	G-68A	ADSP-1010BTG	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010BKG	0 to +70°C	68-Pin Grid Array	G-68A	ADSP-1010BSG/883B	−55°C to +125°C	68-Pin Grid Array	G-68A
ADSP-1010BSD	−55°C to +125°C	64-Pin Ceramic DIP	D-64A	ADSP-1010BTG/883B	−55°C to +125°C	68-Pin Grid Array	G-68-A

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

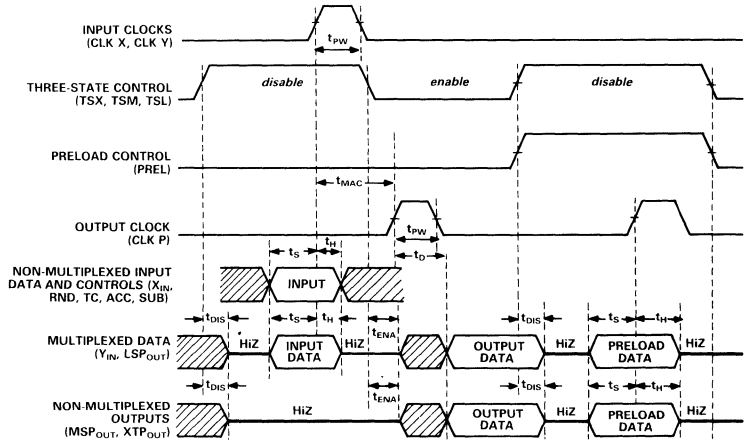


Figure 1. ADSP-1010B Timing Diagram

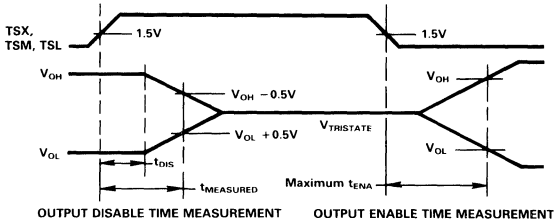


Figure 2. Three-State Disable and Enable

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

METHOD OF OPERATION

The X and Y input registers are positive-edge triggered D-type flip-flops. Input data is loaded to the X and Y registers with the rising edges of CLK X and CLK Y, respectively. The X and Y input data can be represented in either twos complement or unsigned magnitude formats. (Mixed-mode is not supported.)

TC, RND, ACC and SUB are registered input controls. Note that these four controls are latched by the rising edge of the logical OR of CLK X and CLK Y. Be sure that CLK X and CLK Y are both LO (logic 0) before attempting to clock in these controls.

When the registered twos complement control, TC, is HI (logic 1), the inputs are interpreted as twos complement numbers. (See Table I for the ADSP-1010B's data formats.) When TC is LO, the inputs are interpreted as unsigned magnitude numbers. In both cases, outputs will be in the same format as inputs. No shifting is performed in the ADSP-1010B, so all multiplications, including (twos complement) negative full scale multiplied by negative full scale, yield valid results.

When the registered RND control is HI, the MSP will be rounded by adding a binary 1 (with carry) to the most significant bit (MSB) of the LSP, consistently rounding toward positive infinity at midscale. Truncating the MSP (RND LO) introduces a large-sample statistical bias into the MSP of $-(2^{16}-1)/2$ times the LSB of the LSP, while rounding (RND HI) reduces the bias to +1/2 times the LSB of the LSP.

Registered ACC and SUB controls determine whether the product will be latched directly into the output registers or whether they will be updated with the previous contents of the output registers added to or subtracted from the product. If ACC is LO, the product will overwrite the previous contents of the output registers. Holding ACC low at the beginning of a summation avoids the need for a separate operation to clear the output registers. If ACC is HI and SUB is LO, the previous contents of the output registers will be added to the product and stored in the output registers. If ACC is HI and SUB is HI, the previous contents of the output registers will be subtracted from the product and stored in the output registers. (Table II displays these conditions in a truth table.)

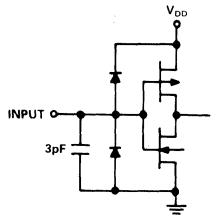


Figure 3. Equivalent Input Circuits

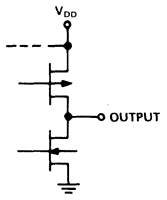


Figure 4. Equivalent Output Circuits

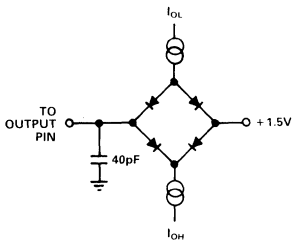


Figure 5. Normal Load for ac Measurements

The accumulation register is partitioned into three words: a 16-bit LSP, a 16-bit MSP and a 3-bit XTP. The 3-bit extension register makes possible summing at least eight large products without overflow. In twos complement mode, the MSB of the XTP will be the product sign bit. Sign bits, or zeros in the case of unsigned magnitude, are extended from the MSB of the product to the MSB of the XTP in the adder/subtracter. (Data preloaded to the accumulation registers will not be sign-extended until it is added to or subtracted from a product.)

The rising edge of CLK P latches the LSP, MSP and XTP into the accumulation registers. Each of these registers has its own three-state control. A HI on the asynchronous TSL, TSM or TSX line disables the corresponding LSP, MSP or XTP output driver to a high impedance state. Conversely, a LO on TSL, TSM or TSX enables the corresponding output driver, driving the output bus.

The asynchronous preload control, PREL, can be used to initialize the output registers. In conjunction with TSL, TSM and TSX, PREL can be used to preload either one, two or all three of the output registers simultaneously. If PREL is HI while either TSL, TSM or TSX is also HI, then the data at the output ports is loaded into the respective output registers on the rising edge of CLK P. (See Table III for a truth table of these conditions.)

X & Y INPUT DATA FORMATS							OUTPUT DATA FORMATS																
							XTP			MSP						LSP							
15	14	13	------	2	1	0	34	33	32	31	30	29	------	18	17	16	15	14	13	------	2	1	0
INTEGER TWOS COMPLEMENT (TC = 1)																							
sign																							
(-2^{15})	2^{14}	2^{13}	------	2^2	2^1	2^0	-2^{34}	2^{33}	2^{32}	2^{31}	2^{30}	2^{29}	------	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0
FRACTIONAL TWOS COMPLEMENT (TC = 1)																							
sign																							
(-2^0)	2^{-1}	2^{-2}	------	2^{-13}	2^{-14}	2^{-15}	-2^4	2^3	2^2	2^1	2^0	2^{-1}	------	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	------	2^{-28}	2^{-29}	2^{-30}
UNSIGNED MAGNITUDE (INTEGER) (TC = 0)																							
2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0	2^{34}	2^{33}	2^{32}	2^{31}	2^{30}	2^{29}	------	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	------	2^2	2^1	2^0

Table I. Data Formats

ABSOLUTE MAXIMUM RATINGS

Supply Voltage . . . . . −0.3V to 7V
Input Voltage . . . . . −0.3 to V_{DD}
Output Voltage . . . . . −0.3 to V_{DD}
Operating Temperature Range ($T_{AMBIENT}$) . . −55°C to +125°C
Storage Temperature Range . . . . . −65°C to +150°C
Lead Temperature (10sec) . . . . . +300°C

ESD SENSITIVITY

THE ADSP-1010B features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1010B has been classified as a Class 1 device.

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.*

ACC	SUB	Function
1	1	$Accumulator_t = X_t \bullet Y_t - Accumulator_{t-1}$
1	0	$Accumulator_t = X_t \bullet Y_t + Accumulator_{t-1}$
0	X	$Accumulator_t = X_t \bullet Y_t$

Table II. Function Truth Table

PREL	TSX	TSM	TSL	XTP	MSP	LSP
0	0	0	0	Q	Q	Q
0	0	0	1	Q	Q	Z
0	0	1	0	Q	Z	Q
0	0	1	1	Q	Z	Z
0	1	0	0	Z	Q	Q
0	1	0	1	Z	Q	Z
0	1	1	0	Z	Z	Q
0	1	1	1	Z	Z	Z
1	0	0	0	Z	Z	Z
1	0	0	1	Z	Z	Preload
1	0	1	0	Z	Preload	Z
1	0	1	1	Z	Preload	Preload
1	1	0	0	Preload	Z	Z
1	1	0	1	Preload	Z	Preload
1	1	1	0	Preload	Preload	Z
1	1	1	1	Preload	Preload	Preload

NOTE:
Z = Output buffers at high impedance (output disabled)
Q = Output buffers at low impedance. Contents of output register will be transferred to output pins.
Preload = Output buffers at high impedance. Preload data supplied externally at output pins will be loaded into the output register at the rising edge of CLK P.

Table III. Preload Truth Table

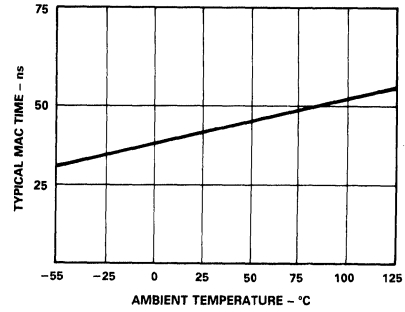


Figure 6. Approx. Multiply Time vs. Temperature

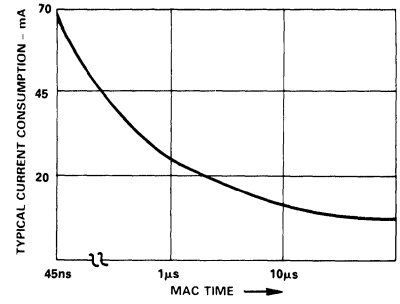


Figure 7. Typical I_{DD} vs. Frequency of Operation

PIN CONFIGURATION

PIN NO.	PACKAGE DIP	PIN-GRID	PLCC	PIN NO.	PACKAGE DIP	PIN-GRID	PLCC
1	X6	Y1, P1	X6	35	P26	P32	P25
2	X5	Y2, P2	X7	36	P27	P33	P24
3	X4	Y3, P3	X8	37	P28	P34	P23
4	X3	Y4, P4	X9	38	P29	CLK P	P22
5	X2	Y5, P5	X10	39	P30	TSM	P21
6	X1	Y6, P6	X11	40	P31	PREL	P20
7	X0	Y7, P7	X12	41	P32	TSX	P19
8	Y0, P0	GND	X13	42	P33	TC	P18
9	Y1, P1	Y8, P8	X14	43	P34	V_{DD}	P17
10	Y2, P2	Y9, P9	X15	44	CLK P	CLK Y	P16
11	Y3, P3	Y10, P10	TSL	45	TSM	CLK X	Y15, P15
12	Y4, P4	Y11, P11	RND	46	PREL	ACC	Y14, P14
13	Y5, P5	Y12, P12	SUB	47	TSX	SUB	Y13, P13
14	Y6, P6	Y13, P13	ACC	48	TC	RND	Y12, P12
15	Y7, P7	Y14, P14	CLK X	49	V_{DD}	TSL	Y11, P11
16	GND	Y15, P15	CLK Y	50	CLK Y	X15	Y10, P10
17	Y8, P8	N/C	V_{DD}	51	CLK X	N/C	Y9, P9
18	Y9, P9	P16	V_{DD}	52	ACC	X14	Y8, P8
19	Y10, P10	P17	V_{DD}	53	SUB	X13	GND
20	Y11, P11	P18	V_{DD}	54	RND	X12	GND
21	Y12, P12	P19	TC	55	TSL	X11	Y7, P7
22	Y13, P13	P20	TSX	56	X15	X10	Y6, P6
23	Y14, P14	P21	PREL	57	X14	X9	Y5, P5
24	Y15, P15	P22	TSM	58	X13	X8	Y4, P4
25	P16	P23	CLK P	59	X12	X7	Y3, P3
26	P17	P24	P34	60	X11	X6	Y2, P2
27	P18	P25	P33	61	X10	X5	Y1, P1
28	P19	P26	P32	62	X9	X4	Y0, P0
29	P20	P27	P31	63	X8	X3	X0
30	P21	P28	P30	64	X7	X2	X1
31	P22	P29	P29	65		X1	X2
32	P23	P30	P28	66		X0	X3
33	P24	P31	P27	67		Y0, P0	X4
34	P25	N/C	P26	68		N/C	X5

ANALOG DEVICES

24 × 24-Bit CMOS Multiplier

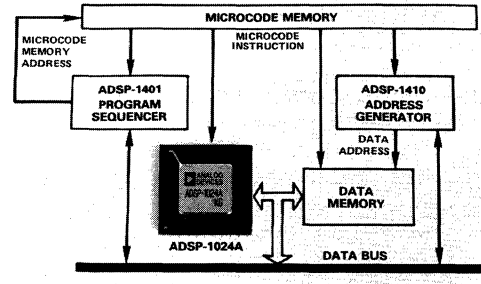
ADSP-1024A

FEATURES

- **24 × 24-Bit Parallel Multiplication**
- **95ns Multiply Time**
- **450mW Power Dissipation with TTL-Compatible CMOS Technology**
- **Twos-Complement Data Format**
- **Rounding Options at Three Positions**
- **Left-Shifts of 0, 1, or 2 Bits on Output**
- **Overflow and Normalization Status Flags**
- **Single-Cycle Output of Both 24-Bit Output Words**
- **Available in Hermetically-Sealed 84-Pin Grid Array**
- **Available Specified from −55°C to +125°C Ambient**
- **Pin-Compatible with ADSP-1024**

APPLICATIONS

- **High-Resolution Digital Signal Processing**
  - **Digital Filtering**
  - **Fourier Transformations**
  - **Correlations**
- **Voice Recognition**
- **Mantissa Multiplication for Floating-Point Operations**



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1024A

GENERAL DESCRIPTION

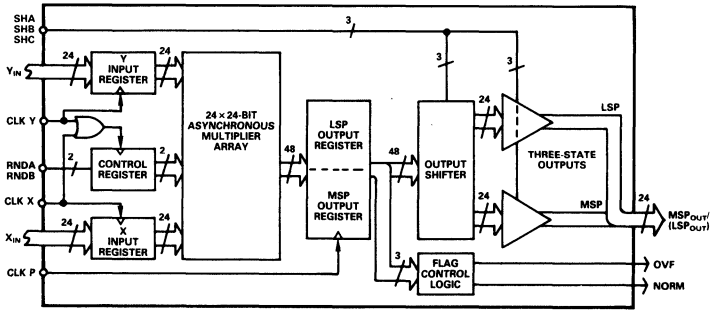
The ADSP-1024A is a high-speed, low-power 24 × 24-bit parallel multiplier fabricated in 1.5 micron CMOS. The ADSP-1024A is a pin-for-pin replacement for Analog Devices' ADSP-1024.

The ADSP-1024A is a three-port device which has two 24-bit input buses and two 24-bit product buses. The Most Significant Product (MSP) bus and the Least Significant Product (LSP) bus share the output port. In a single cycle, both MSP and LSP can be output. Input data must be in twos-complement format. The ADSP-1024A produces a 48-bit result whose twos-complement MSP can be rounded with controls which cause a 1 to be added to either bit 23, 22, or 21 of the LSP.

All input pins are ESD protected. The input and output registers are all D-type positive-edge-triggered flip-flops. The input registers are controlled by independent clock lines. A third clock line controls the product registers. Both of the product registers have their own three-state output controls. Three-state outputs and independently clocked inputs allow the ADSP-1024A to be connected directly to a single 24-bit bus.

The power consumption of the ADSP-1024A is 450mW maximum. The differential between the ADSP-1024A's junction temperature and the ambient temperature stays small because of this low-power dissipation. Thus, the ADSP-1024A can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-1024A is available for both commercial and military temperature ranges. MIL-grade parts are available processed fully to MIL-STD-883, Class B. The ADSP-1024A is available in a hermetically-sealed ceramic 84-pin grid array.



ADSP-1024A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1024A 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-1024A 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	2.2		2.2		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		0.6		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		75		90	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS

Parameter		ADSP-1024A 0 to +70°C J Grade Min	J Grade Max	K Grade Min	K Grade Max	−55°C to +125°C S Grade[2] Min	S Grade[2] Max	T Grade[2] Min	T Grade[2] Max	Unit
t_D	Output Delay		35		30		40		35	ns
t_{ENA}	Three-State Enable Delay		35		30		40		35	ns
t_{DIS}	Three-State Disable Delay		35		30		40		35	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_S	Input Setup Time	30		25		35		30		ns
t_H	Input Hold Time	2		2		3		3		ns
t_{DOVF}	Clock to OVF Valid		30		25		40		35	ns
t_{DNRM}	Clock to NORM Valid		30		25		40		35	ns
t_{MC}	Clocked Multiply Time		120		95		150		120	ns

NOTES

[1]All min and max specifications are over power supply and temperature range indicated. Input levels are GND and 3.0V. Rise times are 5ns. Input timing reference levels and output reference levels are 1.5V, except for t_{ENA} and t_{DIS} which are as indicated in Figure 2.

[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1024A can be found in Analog Devices' Military Databook.

Specifications subject to change without notice.

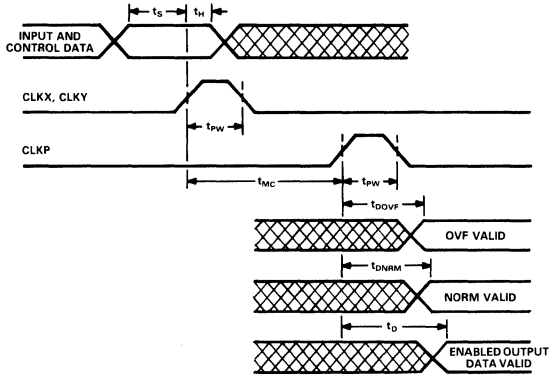


Figure 1. ADSP-1024A Timing Diagram

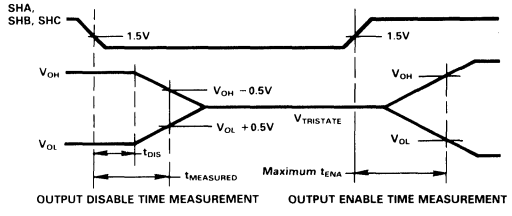


Figure 2. Three-State Disable and Enable Timing

Output disable time, t_{DIS}, is measured from the time the output enable control signal 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 maximum output enable time, maximum t_{ENA}, is also measured from output enable control signal 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.

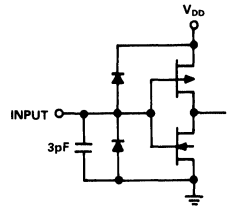


Figure 3. Equivalent Input Circuit

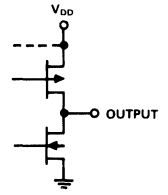


Figure 4. Equivalent Output Circuit

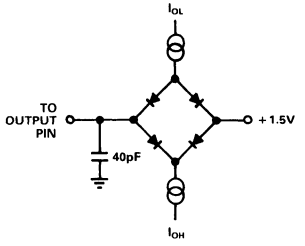


Figure 5. Normal Load for ac Measurements

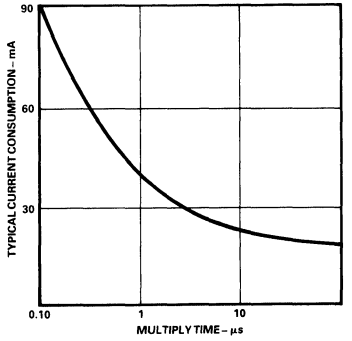


Figure 6. Typical Power Dissipation vs. Frequency

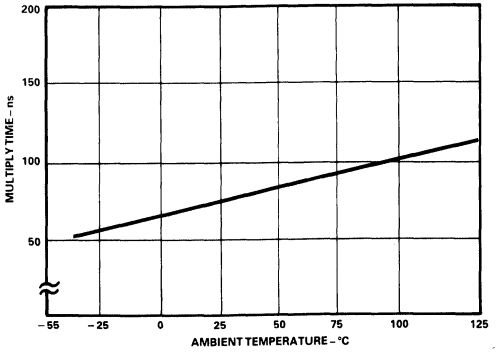


Figure 7. Approx. Worst Case Multiply Time vs. Temperature

METHOD OF OPERATION

The X and Y input registers are independently controlled, positive-edge-triggered D-type flip-flops. Input data is loaded to the X and Y registers by the rising edges of CLKX and CLKY, respectively. The X and Y input data is interpreted in twos-complement notation. (See Table I for the ADSP-1024A's data formats. Unsigned-magnitude and mixed-mode data formats are not supported.)

RNDA and RNDB are registered input controls latched by the rising edge of the logical OR of CLKX and CLKY. Be sure that CLKX and CLKY are both LO (logic 0) before attempting to clock in RNDA and RNDB. When either RNDA or RNDB in the control register are HI (logic 1), the product of the input data will be rounded by adding a binary 1 to one of three places. Normally, the position chosen for rounding is determined by the number of shifts that will be performed on output (so that rounding occurs at the bit position in the LSP that is output on line P23). RNDA and RNDB round the product as follows:

RNDA	RNDB	Effect on Product
0	0	no rounding
0	1	adds a one to LSP bit 22
1	0	adds a one to LSP bit 21
1	1	adds a one to LSP bit 23

The result from the ADSP-1024A's multiplier array is fielded into a 24-bit MSP and a 24-bit LSP. The rising edge of CLKP latches the LSP and MSP into the two corresponding 24-bit output registers. Each of these registers has its own set of three-state drivers, controlled by the asynchronous SHA, SHB, and SHC lines. A LO on all three lines disables all output drivers to a high-impedance state at the output port. Conversely, other combinations of SHA, SHB, and SHC can enable one set of output drivers, driving the output bus.

The three-state and shift control lines, SHA, SHB, SHC, control the shifter and output drivers as follows:

SHC	SHB	SHA	Effect at Product Port (P lines)
0	0	0	output port at high impedance
0	0	1	enable LSP, unshifted
0	1	0	enable LSP, left-shifted by one bit
0	1	1	enable LSP, left-shifted by two bits
1	0	0	undefined
1	0	1	enable MSP, unshifted
1	1	0	enable MSP, left-shifted by one bit
1	1	1	enable MSP, left-shifted by two bits

See Table I for resultant data formats. Note that the shifter is situated after the output register. So the MSP and LSP can be shifted independently to the P lines and read out in either order without affecting any bits in either word of the output register.

X & Y INPUT DATA FORMATS	OUTPUT DATA FORMATS	
23 | 22 | 21 | ------ | 2 | 1 | 0	P47 |P46 | P45 | ------ | P26 | P25 | P24	P23 | P22 | P21 | ------ | P2 | P1 | P0
INTEGER TWOS COMPLEMENT	NO SHIFT (SHC, B, A = 101 FOR MSP; = 001 FOR LSP)	
sign $(-2^{23})\|2^{22}$ | 2^{21} | ------ | 2^{2} | 2^{1} | 2^{0}	sign $(-2^{47})\|2^{46}$ | 2^{45} | ------ | 2^{26} | 2^{25} | 2^{24}	2^{23} | 2^{22} | 2^{21} | ------ | 2^{2} | 2^{1} | 2^{0}
	SHIFTED 1 BIT (SHC, B, A = 110 FOR MSP; = 010 FOR LSP)	
	sign $(-2^{46})\|2^{45}$ | 2^{44} | ------ | 2^{25} | 2^{24} | 2^{23}	2^{22} | 2^{21} | 2^{20} | ------ | 2^{1} | 2^{0} | 0
	SHIFTED 2 BITS (SHC, B, A = 111 FOR MSP; = 011 FOR LSP)	
	sign $(-2^{45})\|2^{44}$ | 2^{43} | ------ | 2^{24} | 2^{23} | 2^{22}	2^{21} | 2^{20} | 2^{19} | ------ | 2^{0} | 0 | 0
FRACTIONAL TWOS COMPLEMENT	NO SHIFT (SHC, B, A = 101 FOR MSP; = 001 FOR LSP)	
sign $(-2^{0})\|2^{-1}$ | 2^{-2} | ------ | 2^{-21} | 2^{-22} | 2^{-23}	sign $(-2^{1})\|2^{0}$ | 2^{-1} | ------ | 2^{-20} | 2^{-21} | 2^{-22}	2^{-23} | 2^{-24} | 2^{-25} | ------ | 2^{-44} | 2^{-45} | 2^{-46}
	SHIFTED 1 BIT (SHC, B, A = 110 FOR MSP; = 010 FOR LSP)	
	sign $(-2^{0})\|2^{-1}$ | 2^{-2} | ------ | 2^{-21} | 2^{-22} | 2^{-23}	2^{-24} | 2^{-25} | 2^{-26} | ------ | 2^{-45} | 2^{-46} | 0
	SHIFTED 2 BITS (SHC, B, A = 111 FOR MSP; = 011 FOR LSP)	
	sign $(-2^{-1})\|2^{-2}$ | 2^{-3} | ------ | 2^{-22} | 2^{-23} | 2^{-24}	2^{-25} | 2^{-26} | 2^{-27} | ------ | 2^{-46} | 0 | 0

Table I. Data Formats

When shifting the MSP left on output, the Most Significant Bits (MSBs) of the LSP are read out with the MSP (see Table I). For example, shifting the MSP left by two bits will bring LSP bit 23 out on product line P25 and LSP bit 22 out on product line P24. When shifting the LSP left on output, the least significant product lines will be zero-filled from the right. For example, shifting the LSP left by two bits will bring zeros out on product lines P1 and P0.

Except when multiplying full-scale negative by full-scale negative (a condition flagged by OVF), the two MSBs of the MSP are identical, hence redundant. Another bit of magnitude in the MSP can be gained by shifting left one bit on output and treating MSP bit 46 coming out on line P47 as the sign bit. Often products are not close to full scale and MSP bits 46 and 45 will also be identical, hence redundant (non-normalized condition). Yet another bit of magnitude in the MSP can be gained by shifting left two bits on output and treating MSP bit 45 coming out on line P47 as the sign bit. Left shifting by two bits can be useful when the 1024A is normalizing the mantissas of floating-point products and, more generally, for scaling.

OVF and NORM are flags that are generated from the output register (see Figure 7). They become valid t_{DOVF} and t_{DNRM}

after the rising edge of CLKP. When true (HI), OVF indicates that full-scale negative has been multiplied by full-scale negative. The product register has not overflowed; however, this condition warns the user that any left shift will cause erroneous outputs, since in this single case the two MSBs of the MSP are *not* redundant.

When true (HI), NORM indicates that the product is normalized – OVF is false (MSB bits 47 and 46 are identical) but all other bits are significant (MSB bits 46 and 45 differ). When NORM is true (HI), a one-bit left shift is safe but a two-bit left shift will cause the loss of a sign bit. When NORM is false (LO), the product can be safely shifted by two bits. Figure 8 shows the range of values that result in different values for OVF and NORM. Note that OVF and NORM are independent of the three-state and shift control lines (SHA, SHB, SHC) since they are generated at the output register, not at the product output port. In fact, OVF and NORM can be used to determine what shift option should be used.

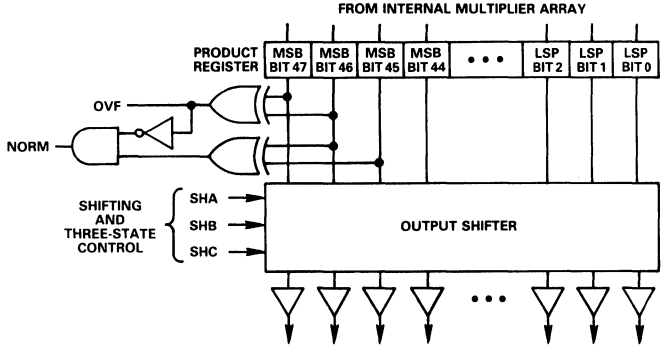


Figure 8. Flag and Shift Logic

OVF	NORM	MSB BIT 47	MSB BIT 46	MSB BIT 45	Range of Product Register (P) Fractional TC Format
0	0	0	0	0	$0 \leq P < +\frac{1}{2}$
		1	1	1	$-\frac{1}{2} \leq P < 0$
0	1	0	0	1	$+\frac{1}{2} \leq P < +1$
		1	1	0	$-1 \leq P < -\frac{1}{2}$
1	0	0	1	1	$+1\frac{1}{2} \leq P < +2$
		1	0	0	$-2 \leq P < -1\frac{1}{2}$
		0	1	0	$+1 \leq P < +1\frac{1}{2}$
		1	0	1	$-1\frac{1}{2} \leq P < -1$

OVF	NORM	MSB BIT 47	MSB BIT 46	MSB BIT 45	Range of Product Register (P) Integer TC Format
0	0	0	0	0	$0 \leq P < +2^{45}$
		1	1	1	$-2^{45} \leq P \leq -1$
0	1	0	0	1	$+2^{45} \leq P < +2^{46}$
		1	1	0	$-2^{46} \leq P < -2^{45}$
1	0	0	1	1	$(+2^{46} + 2^{45}) \leq P < +2^{47}$
		1	0	0	$-2^{47} \leq P < (-2^{47} + 2^{45})$
		0	1	0	$+2^{46} \leq P < (+2^{46} + 2^{45})$
		1	0	1	$(-2^{47} + 2^{45}) \leq P < -2^{46}$

Figure 9. Range of Values in Product Register and Their Effects on OVF and NORM

ABSOLUTE MAXIMUM RATINGS

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

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1024AKG	0 to +70°C	84-Pin Ceramic Pin Grid Array	G-84A
ADSP-1024AJG	0 to +70°C	84-Pin Ceramic Pin Grid Array	G-84A
ADSP-1024ASG	−55°C to +125°C	84-Pin Ceramic Pin Grid Array	G-84A
ADSP-1024ATG	−55°C to +125°C	84-Pin Ceramic Pin Grid Array	G-84A
ADSP-1024ATG/883B	−55°C to +125°C	84-Pin Ceramic Pin Grid Array	G-84A
ADSP-1024ASG/883B	−55°C to +125°C	84-Pin Ceramic Pin Grid Array	G-84A

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

ESD SENSITIVITY

The ADSP-1024A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1024A has been classified as a Class 1 device.

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*.

32-BIT FLOATING-POINT MULTIPLICATION TWOS COMPLEMENT (MIL-STD 1750A)

The ADSP-1024A is a useful building block for high-speed floating-point multipliers. The implementation described here accepts normalized 24-bit twos-complement mantissas and 8-bit twos-complement exponents as inputs. The product will be normalized to the same format. Pipelined throughput will be at the clocked multiply rate of the ADSP-1024A, e.g. 95ns for the ADSP-1024AK. This design exhibits very low latency as well.

The ADSP-1024A performs the mantissa multiplication. It also normalizes the mantissa product with its output shifter. The NORM and OVF flags determine the number of bits to be shifted on output and also provide the control lines to the external adders to denormalize the exponent as the mantissa is normalized.

In this implementation, a single clock drives the ADSP-1024A's CLKX, CLKY, and CLKP as well as the exponent circuitry. On the clock's rising edge, the pair of mantissas is loaded into the ADSP-1024A's input registers. At the same time, the two exponents are clocked into their respective 'LS273 octal D flip-flops.

During the clock cycle, the ADSP-1024A will compute the product of the mantissas. In parallel, the exponents will be added in the 'LS283 4-bit full adders. Their sum will be valid well before the clock goes high again, when it will be latched into a 'F273 octal D flip-flop. At this same rising edge, the mantissa product is clocked into the output register within the ADSP-1024A. New floating-point inputs can also be clocked into the circuit at the same time, making possible floating-point throughput at the ADSP-1024A's clocked multiply time.

NORM and OVF from the ADSP-1024A will be valid t_{DNRM} and t_{DOVF} after this second rising clock edge, respectively. When valid, these (decoded) flags normalize the mantissa using the 1024A's output shifter and denormalize the exponent using a pair of 'F382 4-bit ALUs. Output can be enabled as soon as NORM and OVF are valid. The ADSP-1024A already offers three-state control; an octal 'F244 buffers the ALUs to the output bus.

If OVF is LOW and NORM is HI, then we shift the mantissa product left one bit on output to eliminate the redundant sign bit. Since we are simply formattting the mantissa, the value at the 'F273 flip-flop is already the correct exponent, and we leave it alone. If OVF is HI (and NORM is LO), then we shift the mantissa product zero bits (because the product register's MSP is already normalized in this singular case of full-scale negative times full-scale negative). The exponent is incremented by one. If OVF and NORM are both LO, we shift the mantissa product left two bits on output to eliminate two redundant sign bits and produce a normalized result. The exponent is decremented by one.

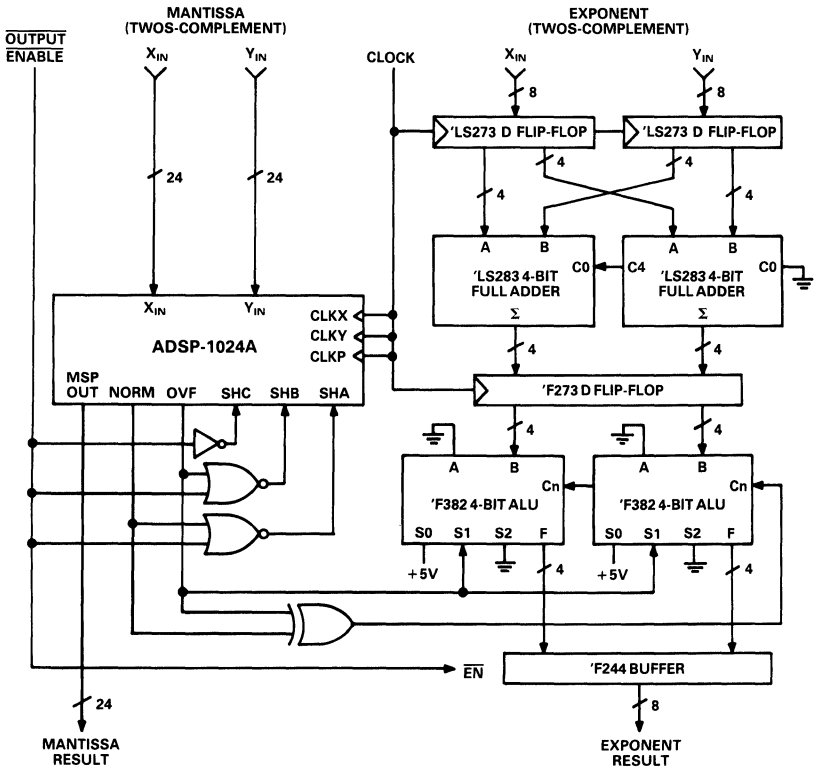


Figure 10. 32-Bit Floating-Point Multiplier Circuit

ANALOG DEVICES

16 × 16-Bit CMOS Single Port Multiplier/Accumulator

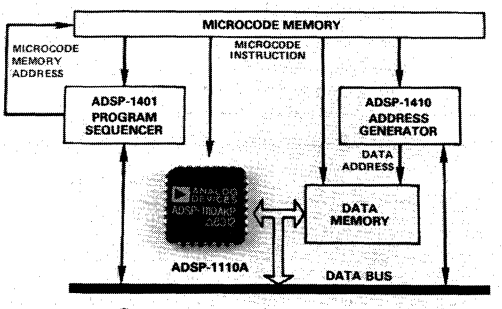
ADSP-1110A

FEATURES

16 × 16-Bit Parallel Multiplication/Accumulation
40-Bit Wide Accumulator with Overflow Flag, Saturation Arithmetic, and Shift-Left Control
Twos Complement or Unsigned Magnitude Inputs
85ns Multiply/Accumulate Time
28-Lead Ceramic DIP, Plastic DIP Package or Plastic Leaded Chip Carrier
350mW Power Dissipation with CMOS Technology
Specified Over the Extended Temperature Range
Pin-Compatible with ADSP-1110

APPLICATIONS

Digital Filtering
Fast Fourier Transforms
Matrix Multiplication
Microprocessor Acceleration



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1110A

GENERAL INFORMATION

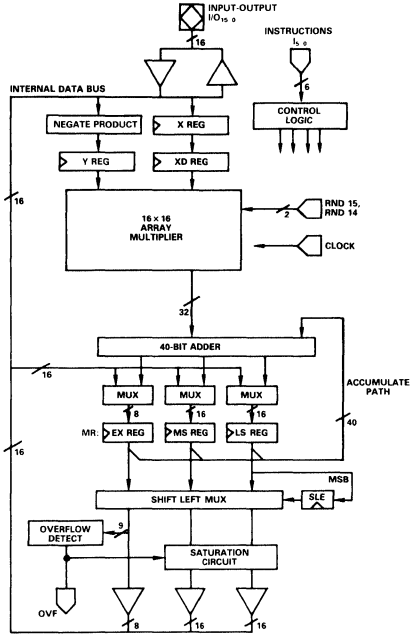
The ADSP-1110A is a high-speed, low-power single-port 16 × 16-bit multiplier/accumulator (MAC), with processing throughput comparable to existing three-port MACs. Its single-bus structure offers unique advantages: more compact packaging in a 28-pin package, simpler system interface to single-bus peripherals, and significantly reduced cost. In addition, innovative on-chip features extend the ADSP-1110A's capabilities and eliminate external hardware.

All inputs to and outputs from the ADSP-1110A pass through its single 16-bit I/O port. All I/O operations are single cycle. A multiplication or MAC operation requires two cycles to complete—consistent with the two cycles required to load input pairs to the multiplier. An internal pipeline register enables a new input to be loaded as the previous multiplication/accumulation is computed—allowing the device's full 11.7MHz computational bandwidth to be utilized.

A six-bit microcode instruction word governs the ADSP-1110A's operation. The instruction set centers around I/O and multiplication/accumulation operations. Additional instructions allow extra precision in single- and double-precision operations to be obtained efficiently.

Multiplier products are accumulated in a 40-bit wide Multiplier Result (MR) register, which consists of a 16-bit MS (Most Significant) and LS (Least Significant) register, and an 8-bit EX (Extension) register. Either multiplier input can be a twos complement or unsigned magnitude number. Overflow from the lower 32 bits of the MR into the upper eight guard bits is detected and can be monitored externally. Outputs can, conditional upon overflow status, be saturated to full scale. An MR register can be shifted left by one bit upon output; two independent controls allow rounding consistent with output formatting.

The ADSP-1110A is optimal for applications where board space is limited but the performance of a DSP processor is required. In addition, a microprocessor-based system can realize greater throughput by utilizing the ADSP-1110A in an accelerator.



ADSP-1110A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1110A J and K Grades Min	ADSP-1110A J and K Grades Max	ADSP-1110A S and T Grades[2] Min	ADSP-1110A S and T Grades[2] Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_{AMB}	Operating Temperature ($T_{AMBIENT}$)	0	70	−55	125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	ADSP-1110A J and K Grades Min	ADSP-1110A J and K Grades Max	ADSP-1110A S and T Grades[2] Min	ADSP-1110A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.2		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		0.4		0.6	V
I_{IH}	High-Level Input Current	@ V_{DD} = max & V_{IN} = 5.0V		10		10	μA
I_{IL}	Low-Level Input Current	@ V_{DD} = max & V_{IN} = 0V		10		10	μA
I_{OZH}	Three State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = max		50		50	μA
I_{OZL}	Three State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = 0		50		50	μA
I_{DD}	Supply Current	@ max clock rate; TTL-inputs		70		80	mA
I_{DD}	Supply Current – Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS

Parameter		ADSP-1110A 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_{CLK}	Clock Period	50		42.5		60		50		ns
t_{MAC}	Multiply/Accumulate Time		100		85		120		100	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_{DS}	Input Data Setup Time	15		15		15		15		ns
t_{CS}	Input Control Setup Time	25		20		25		20		ns
t_{DH}	Input Data Hold Time	3		3		4		4		ns
t_{CH}	Input Control Hold Time	5		5		6		6		ns
t_{D}	Control to Valid Output		30		25		30		30	ns
t_{DSAT}	Control to Valid Output with Saturation		35		32		40		35	ns
t_{DIS}	Output Driver Disable Time		25		25		25		25	ns
t_{O}	Control to Overflow Flag		30		25		35		30	ns
t_{LO}	Control to Overflow Flag w/sl		40		35		45		40	ns

NOTES

[1]All min & max specifications are over power supply and temperature range indicated. Rise times are 5ns. Input levels are GND and 3.0V. Input timing reference levels and output reference levels are 1.5V.

[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1110A can be found in Analog Devices' Military Databook.

Specifications subject to change without notice.

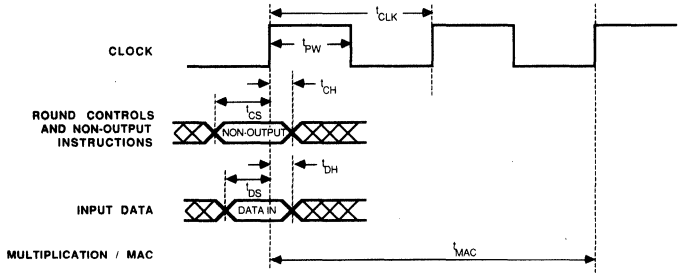


Figure 1a. ADSP-1110A Timing: Clocked (Synchronous) Operations All Non-Output Instructions

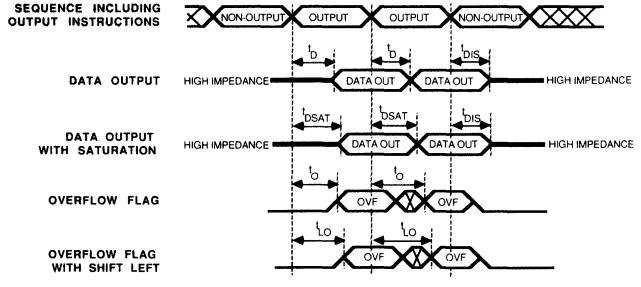


Figure 1b. ADSP-1110A Timing: Unclocked (Asynchronous) Operations All Output Instructions

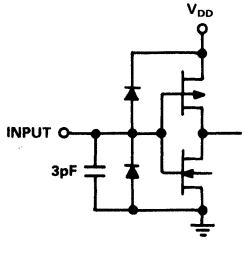


Figure 2a. Equivalent Input Circuits

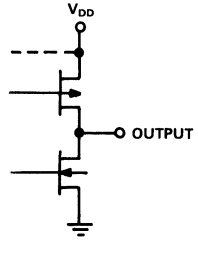


Figure 2b. Equivalent Output Circuits

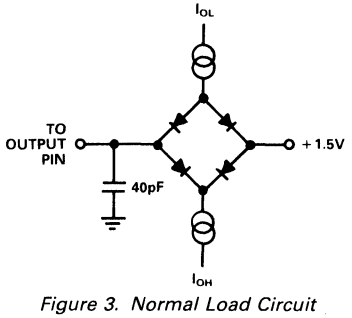


Figure 3. Normal Load Circuit for ac Measurements

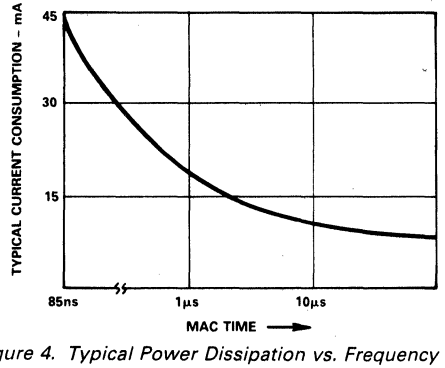


Figure 4. Typical Power Dissipation vs. Frequency

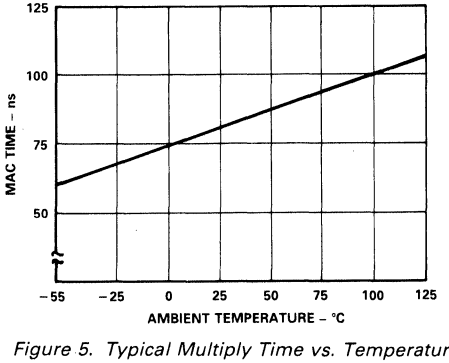


Figure 5. Typical Multiply Time vs. Temperature

METHOD OF OPERATION

The ADSP-1110A's operation is controlled by a six-bit microcode instruction and two rounding control pins. Table III presents instructions that are executed by the ADSP-1110A, along with the corresponding six-bit microcode instruction. The sections below further describe the instruction groups presented in Table III.

Input and Multi-Operation Instructions

A dedicated *input instruction* ("X = BUS") loads the X input at the rising edge of the clock. The X input is loaded with the data that is set up on the device's 16-bit I/O port.

A set of *multi-operation instructions* ("Y = BUS; CKMR; X*Y") are used to load the Y input and otherwise control the ADSP-1110A's multiplier/accumulator. Specifically, at the next rising clock edge, a multi-operation instruction i) loads Y input ii); clocks the result of the previous multiplication/MAC operation into the MR; and, iii) initiates the next multiplication/MAC operation. The multiplication/MAC operation is initiated at the rising edge of the clock and requires two cycles to complete. The instruction controls needed to govern the device's multiplier array and 40-bit adder during these two cycles are registered internally.

During the first cycle of a multi-operation instruction, the X input is transferred to an internal pipeline register (XD), and is latched there on the next rising clock edge. Consequently, a new X value can be loaded onto the chip during the second cycle of the multi-operation instruction. XD will not be overwritten until a new X value is loaded.

The ADSP-1110A supports the following multiplication and multiplication/accumulation operations:

$$\pm X*Y$$

and,

$$\pm X*Y \pm MR$$

The ADSP-1110A allows either input to be specified as a twos complement or unsigned magnitude number. Table II describes, for all combinations of inputs, the proper interpretation of the MR register if it is output with or without the left-shift option. Note that if the Y input is negative full scale and a negative product is specified, an invalid result is obtained. This happens because the ADSP-1110A will attempt to produce the unrepresentable twos complement of full-scale negative.

The result of a multiplication or MAC operation is latched into the MR register in either of two ways. A dedicated "CKMR" instruction performs this clocking. In addition, all multi-operation instructions clock the MR, eliminating overhead when computing MAC's (see *Instruction Sequences*). It is important to note that whenever "CKMR" is executed, it clocks the result of the *previous* operation into the MR. Also, in all cases, the clocking of the MR occurs at the rising edge of the clock.

MR Register Instructions

A number of the ADSP-1110A's instructions affect the contents of the MR register—including *preload instructions*, *transfer instructions*, and *sign extend instructions*. In addition, special output instructions allow for format adjusting the MR upon output.

The 40-bit accumulator of the ADSP-1110A is segmented into three registers: a 16-bit most significant product register (MS); a 16-bit least significant product register (LS); and, an 8-bit extended product register (EX) (see Table II). The eight guard bits of the EX allow at least 256 multiplication/accumulations without risk of overflow.

Dedicated instructions allow any of the MR's registers to be preloaded with data set up on the device's 16-bit I/O port. This preloading occurs at the rising edge of the clock.

The proper sequence for preloading a value Z into MR and adding it to the product $X_1 * Y_1$ is:

Instruction	Comment
1. X = BUS	Load X_1
2. Y = BUS; CKMR; X*Y + MR	Load Y_1; clock garbage into MR; initiate MAC
3. LS = BUS	Preload MR with Z
4. MS = BUS	Preload MR with Z
5. EX = BUS	Preload MR with Z
6. X = BUS	Load X_2
7. Y = BUS; CKMR; X*Y + MR	Load Y_2; MR = $X_1*Y_1 + Z$; initiate next multiplication.

This sequence ensures that the value Z preloaded by instructions 3, 4, and 5 is added to the product X_1*Y_1 and clocked into MR by instruction 7. If Z were preloaded prior to instruction 2, then instruction 2's "CKMR" operation would overwrite the Z value with the product of whatever values were last placed in the multiplier array.

Transfer operations allow one MR register to be moved down to an adjacent one—useful in double-precision operations. The ADSP-1110A can, in one cycle, shift the EX to the MS or the MS to the LS register. The shift left extend register (SLE) is a one-bit latch that is loaded with the value of the MSB of the LS register whenever the MS is transferred to LS. The SLE register retains its value until the next downshift of MS into LS overwrites its contents.

Anytime the result of a multiplication or multiplication/accumulation operation is clocked into the accumulator, the result is automatically sign extended into the upper MSBs of the accumulator. In addition, explicit instructions allow the MSB of the LS to be sign extended to the MS ("MS = SIGN EXT LS") or the MSB of the MS to be sign extended to the EX register ("EX = SIGN EXT MS"). Such sign extend capability may be needed to properly initialize the MR after the MS or LS is preloaded, or after an MR register transfer.

Output Instructions

Output instructions allow any MR register to be read. When written onto the ADSP-1110A's 16-bit bus, the 8-bit EX register is automatically sign-extended into the upper 8 MSBs of the bus. Standard output instructions of the ADSP-1110A are supplemented with two important options: a shift-left capability and conditional saturation.

The ADSP-1110A's output instructions include the ability to shift any MR register (EX, MS, or LS) left by one bit upon output. This shift does not affect the contents of MR, but does affect what appears on the ADSP-1110A's 16-bit I/O port. Figure 6 shows which bits of the 40-bit wide MR register are output if the shift-left option is invoked.

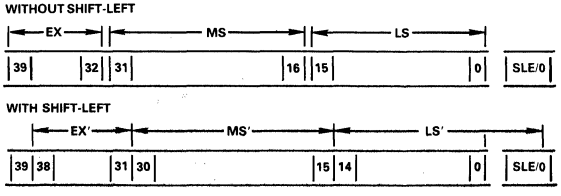


Figure 6. Effect of Left Shift on MR Outputs

The shift left-on-output control, which scales up the MR outputs by a factor of two, is useful under many circumstances. Twos complement multiplication—for all but one case (negative full-scale times negative full-scale)—results in redundancy in the two MSBs of the 32-bit product. This redundancy means that the 16-bit MS register contains two identical sign bits (bits 31 and 30 of MR) and just 14 bits of magnitude. The ADSP-1110A's shift-left control allows full precision in twos complement operations to be attained. Left shift control also provides a means for maximizing resolution when using block floating point, when downscaling twos complement results, and when upscaling mixed and unsigned magnitude results.

Whenever the RND14 pin is asserted during a "BUS = LS (sl)" or "BUS = LS (sl, sat)" instruction, the SLE bit will be appended to the upper 15 bits of the shifted LS. If the RND14 is low, however, a zero will be inserted into the LSB of LS. Appending the SLE bit to the shifted LS provides an extra bit of precision in applications such as double-precision multiplication/accumulations.

Round Controls

The RND14 and RND15 pins are two independent controls that allow rounding consistent with shifted or unshifted outputs, respectively. The round control signals are latched at the rising clock edge whenever the device receives a multiplication or MAC instruction. Asserting the RND 15 (RND14) pin will cause a 1 to be added to bit 15 (bit 14) of the LS. The rounding will not occur until the subsequent cycle in which the result of the multiplication or MAC operation is clocked into the MR.

Overflow and Saturation

The ADSP-1110A's overflow flag monitors 9 bits (8 in the EX register and the MSB of the MS register). If any bits in EX differ from the MSB of MS, then an overflow has occurred from the MS into the EX, and the overflow flag is asserted (HI) following an output instruction. Generally, the status of the overflow flag reflects the current contents of the MR and is updated each time a new result is clocked into the MR. However, if the MS register is output with a left-shift, the overflow logic determines whether the shifted MS overflows into the EX (when bits 38 through 30 are not identical) and is set accordingly. On the cycle following any left-shifted output, the overflow flag status reverts to reflect the contents of the MR. During cycles when a non-output instruction is executed, the overflow flag is always LO.

Serious data glitches can result from wraparound effects due to overflow in long multiply/accumulate chains. For example, if a positive number is added to positive full-scale, the 32 MSBs of the MR register will overflow into the 8-bit EX register. Simply reading the MS register will yield a negative twos complement number. To prevent this wraparound, the ADSP-1110A can—conditioned on overflow status—saturate an output to twos-complement full scale.

The ADSP-1110A's saturation logic operates only on output values; it has no effect on the contents of the MR register. This logic examines the sign of the MR (bit 39, the MSB of the EX register) and the overflow status. As Table IV indicates, the low 32 bits of the MR are saturated to full-scale positive (negative) if overflow has occurred in a positive (negative) MR.

Either the MS or LS registers can be left-shifted on output with conditional saturation. If the shifted value overflows the lower 32 bits, the outputted result will be saturated to full scale.

While the saturation control protects against overflow from the MS to the EX register, the user is not protected in the event the accumulated result overflows the entire 40-bit MR register.

5

OVF	MR BIT 39 (SIGN BIT)	Output Value with Saturation MS	LS
0	0	← No Change →	
0	1	← No Change →	
1	0	0111111111111111	1111111111111111
1	1	1000000000000000	0000000000000000

Table I. Overflow and Saturation Circuitry Conditions

	MR															
INPUTS (X & Y)	EX (8 BITS)				MS (16 BITS)				LS (16 BITS)							
b_{15} b_{14} ------ b_0	b_{39}	b_{38}	------	b_{32}	b_{31}	b_{30}	------	b_{16}	b_{15}	------	b_1	b_0				
TWOS COMPLEMENT INTEGER	(w/o sl)															
-2^{15} 2^{14} ------ 2^0	-2^{39}	2^{38}	------	2^{32}	2^{31}		------	2^{16}	2^{15}	------	2^1	2^0				
	(w/ sl)															
	-2^{38}	2^{37}	------	2^{31}	2^{30}		------	2^{15}	2^{14}	------	2^0	SLE/0				
TWOS COMPLEMENT FRACTIONAL	(w/o sl)															
-2^0 2^{-1} ------ 2^{-15}	-2^9	2^8	------	2^2	2^1	2^0	------	2^{-14}	2^{-15}	------	2^{-29}	2^{-30}				
	(w/ sl)															
	-2^8	2^7	------	2^1	2^0	2^{-1}	------	2^{-15}	2^{-16}	------	2^{-30}	SLE/0				
UNSIGNED MAGNITUDE INTEGER																
2^{15} ------ 2^0	2^{39}		------	2^{32}	2^{31}		------	2^{16}	2^{15}	------	2^1	2^0				
UNSIGNED MAGNITUDE FRACTIONAL																
2^{-1} ------ 2^{-16}	2^7		------	2^0	2^{-1}		------	2^{-16}	2^{-17}	------	2^{-31}	2^{-32}				
MIXED MODE INTEGER																
-2^{15} 2^{14} ------ 2^0 & 2^{15} 2^{14} ------ 2^0	-2^{39}	2^{38}	------	2^{32}	2^{31}		------	2^{16}	2^{15}	------	2^1	2^0				
MIXED MODE FRACTIONAL																
-2^0 2^{-1} ------ 2^{-15} & 2^{-1} 2^{-2} ------ 2^{-16}	-2^8	2^7	------	2^1	2^0		------	2^{-15}	2^{-16}	------	2^{-30}	2^{-31}				

Table II. ADSP-1110A Data Formats

Instruction Group	Instruction	Microcode Instruction 5 4 3 2 1 0	Comments
Miscellaneous	NOP	0 0 0 0 x x	No Operation
	CKMR	0 0 0 1 x x	Clock MR
Input	X = BUS	0 0 1 0 x x	
Preload	LS = BUS	0 1 0 0 0 0	
	MS = BUS	0 1 0 1 x 0	
	EX = BUS	0 1 0 0 1 0	
Transfer	LS = MS	0 1 0 0 0 1	Sets SLE register
	MS = EX	0 1 0 1 0 1	
Sign Extend	EX = SIGN EXT MS	0 1 0 0 1 1	
	MS = SIGN EXT LS	0 1 0 1 1 1	
Output	BUS = EX	0 0 1 1 0 1	All output instructions are asynchronous
	BUS = EX (sl)	0 0 1 1 0 0	I5 – I2:
	BUS = MS	0 1 1 0 0 1	0011 = EX
	BUS = MS (sl)	0 1 1 0 0 0	0110 = MS
	BUS = MS (sat)	0 1 1 0 1 1	0111 = LS
	BUS = MS (sl,sat)	0 1 1 0 1 0	I1 – I0:
	BUS = LS	0 1 1 1 0 1	01 = to bus
	BUS = LS (sl)	0 1 1 1 0 0	00 = to bus shifted
	BUS = LS (sat)	0 1 1 1 1 1	10 = to bus shifted w/saturation
	BUS = LS (sl,sat)	0 1 1 1 1 0	11 = to bus w/saturation
Multi-Operation	Y = BUS; CKMR; $X_{US}*Y_{US}$	1 0 0 x 0 0	Require two cycles to complete.
	Y = BUS; CKMR; $-X_{US}*Y_{US}$	1 0 0 x 0 1	Other instructions can be executed
	Y = BUS; CKMR; $X_{US}*Y_{US}$ +MR	1 0 0 0 1 0	on the second cycle.
	Y = BUS; CKMR, $-X_{US}*Y_{US}$ +MR	1 0 0 0 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{US}$ −MR	1 0 0 1 1 0	I5 = Multiplication/MAC operation
	Y = BUS; CKMR; $-X_{US}*Y_{US}$ −MR	1 0 0 1 1 1	I4 = Y twos complement
	Y = BUS; CKMR; $X_{TC}*Y_{US}$	1 0 1 x 0 0	I3 = X twos complement
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$	1 0 1 x 0 1	I2 = Subtract previous result
	Y = BUS; CKMR; $X_{TC}*Y_{US}$ +MR	1 0 1 0 1 0	I1 = Add/subtract previous result
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$ +MR	1 0 1 0 1 1	from product
	Y = BUS; CKMR; $X_{TC}*Y_{US}$ −MR	1 0 1 1 1 0	I0 = Negate product
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$ −MR	1 0 1 1 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$	1 1 0 x 0 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$	1 1 0 x 0 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$ +MR	1 1 0 0 1 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$ +MR	1 1 0 0 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$ −MR	1 1 0 1 1 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$ −MR	1 1 0 1 1 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$	1 1 1 x 0 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$	1 1 1 x 0 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$ +MR	1 1 1 0 1 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$ +MR	1 1 1 0 1 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$ −MR	1 1 1 1 1 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$ −MR	1 1 1 1 1 1	

Mnemonic Definitions

=	Assign right side to left.	sl	Shift left.
BUS	16-bit external data bus used for all I/O operations.	sat	Conditional on overflow, saturate the outputted value.
X	Input register for multiplier.	TC	Two's complement number.
Y	Input register for multiplier.	US	Unsigned magnitude number.
EX	8-bit extension register for accumulator.	SIGN	Sign bit (MSB) of specified register.
MS	16-bit most significant product register.	CKMR	Clock product into EX, MS, and LS.
LS	16-bit least significant product register.	*	Multiply
MR	40-bit accumulator comprising EX, MS and LS.	x	Microcode instruction bit can be either a 0 or 1.

Table III. ADSP-1110A Instruction Set

CLOCK AND TIMING

Figure 1 presents a timing diagram for the ADSP-1110A's operation.

Input data, round controls, and non-output instructions are clocked (synchronous); set-up and hold times are specified accordingly. All multi-operation (two cycle) instructions are clocked, and the internal controls needed for the second cycle are latched internally.

Unlike all other ADSP-1110A instructions, output operations are asynchronous. The relevant timing specification is the delay between control inputs and valid outputs. The use of saturation (sat) slows down the availability of a valid output on the ADSP-1110A's I/O bus; delay times are specified accordingly.

The ADSP-1110A's OVF (overflow) flag is set according to the contents of the MR register. However, upon outputting the MR with the shift-left control, the OVF flag may be modified if the left shift causes overflow. The relevant timing for this case is specified.

The ADSP-1110A's output three-state drivers are not disabled until t_{DIS} ns after an output instruction is removed. Since the ADSP-1110A has just one I/O port, bus contention can occur when an ADSP-1110A input immediately follows an output. For example, an input source (e.g., a data RAM) enabled to drive the bus immediately after an ADSP-1110A output creates the possibility that both drivers are active simultaneously. There are two ways to avoid such conflicts:

1. Set up output instructions well in advance of the clock's rising edge (>t_D set-up time), enabling the data output to complete in time for the data to be latched at the clock edge. Allow t_{DIS} ns after the clock edge before enabling a different device to drive the bus. Note that any system that provides the ADSP-1110A with its instruction from a pipeline register operates in this way. The *Hardware Implementations with the ADSP-1110A* section describes several alternative implementations consistent with this approach.
2. For systems with minimal instruction set-up time, an operation that doesn't use the bus (e.g., a NOP) may need to be inserted after an output instruction. The reason for this is as follows. Output instructions must be held valid for t_D ns, which means that—if instruction set-up time is minimal—output instructions must be held beyond the rising edge of the clock. After the output instruction is removed, another t_{DIS} elapses before the output drivers are disabled. As a result, the three-state output drivers are active well into the next cycle. If the bus is driven with an input in the next cycle, bus contention may occur.

Instruction Sequences

With the ADSP-1110A, single multiplication operations involve three overhead statements in addition to the multiply command, as Figure 7 illustrates.

While a multiplication/accumulation sequence is structurally similar to a single multiplication, overhead as a percentage of computation time is reduced substantially. In the instruction flow diagram shown in Figure 8, a NOP is needed only in the final multiplication/accumulation operation. Also, new X values are loaded as multiplication/accumulation instructions complete. In this sequence, the three cycles of overhead can be spread out over as many multiplication/accumulations as are performed consecutively.

For a series of multiplication/accumulation sequences, I/O operations can be further overlapped. At the end of each multiplication/accumulation string, a new string is initiated. In this instance, overhead cycles become negligible in importance; the multiplication/accumulation rate of the ADSP-1110A approaches 11MHz.

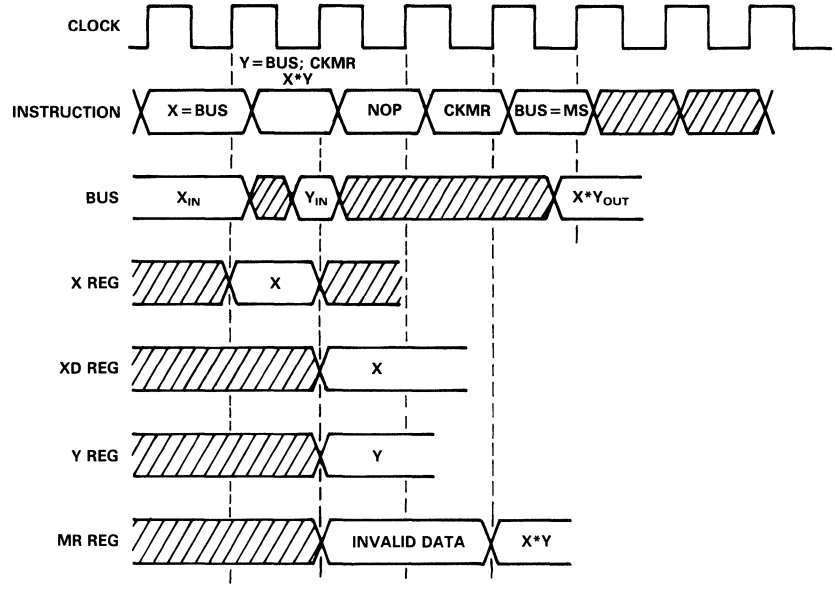


Figure 7. Multiply Operation Timing

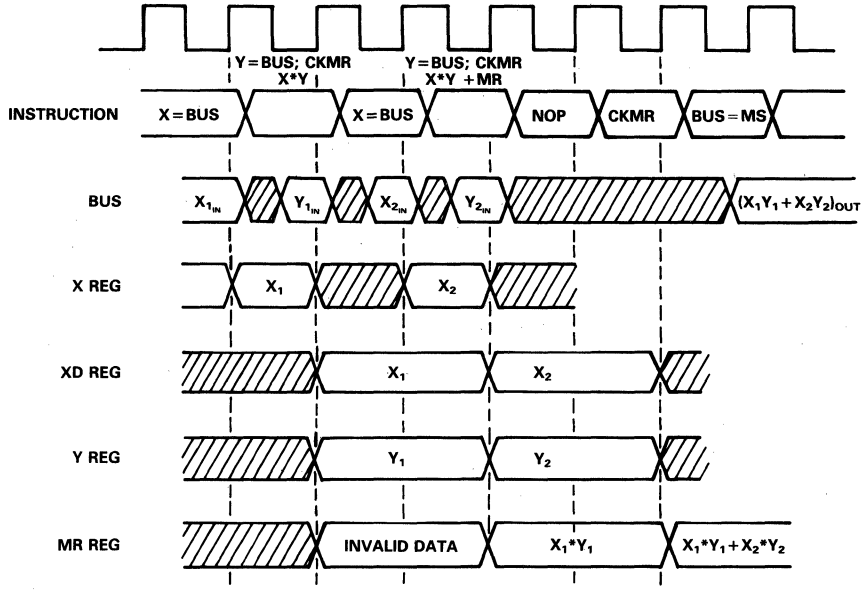


Figure 8. Multiply/Accumulate Operation Timing

Avoid Bus Contentions

Because the ADSP-1110A typically shares its data port with other devices on a common bus, there is a potential for bus contentions at power-up. If the instruction applied to the ADSP-1110A at power-up is random, the multiplier/accumulator could be in an output state. If any other devices are driving the bus at the same time, there will be a bus contention.

The obvious solution is to make sure no other devices are driving the common data bus at power-up. Another approach is to force instruction bit I_5 (pin 18) HI at power-up. This guarantees that the ADSP-1110A will not be in an output state because ADSP-1110A output instructions are asynchronous and all have a zero in instruction bit 5 (I_5).

HARDWARE IMPLEMENTATIONS WITH THE ADSP-1110A

There are many alternative ways of implementing high performance DSP systems with the ADSP-1110A. The following sections illustrate some of the more commonly used approaches using the ADSP-1110A: a microcoded system, a ROM-based sequential machine, a PLA-based state machine, and as a device directly interfaced to a microprocessor. The optimal implementation will depend on the performance, price, and board area requirements of the design.

Microcoded System

Many microcoded systems have the design objective of fast number crunching, while minimizing microcode bits and circuit board area. The ADSP-1110A single port MAC—with just 8 control bits, its single bus structure, and fast cycle time—helps meet these objectives. The ADSP-1110A can be simply connected to the processor data bus and microcode instruction field to provide powerful multiplier/accumulator functions.

A typical Word-Slice® processor with the ADSP-1110A is shown below:

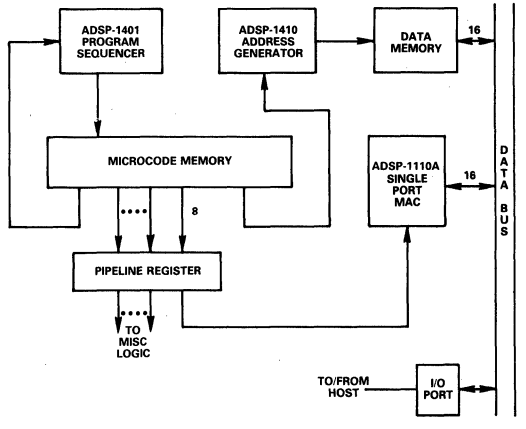


Figure 9.

In most bit-slice designs, the control bits from the microcode memory are latched in a pipeline register. In the above implementation, the ADSP-1110A and all miscellaneous logic are used in conjunction with an external pipeline latch. The pipeline latch guarantees that the microcode bits controlling the circuitry are valid for a complete cycle (see timing diagram below). Note that the ADSP Word-Slice® components (the ADSP-1401 and ADSP-1410) contain an internal pipeline register and are fed directly from the microcode.

Word-Slice is a registered trademark of Analog Devices, Inc.

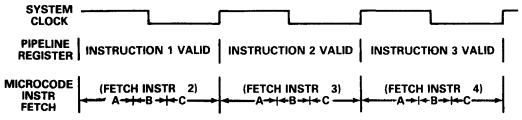


Figure 10.

Segments A, B and C of the above microcode instruction fetch cycle denote different operations. The sequencer starts execution of an instruction at the beginning of a cycle and, during segment A, calculates the microcode memory address for the next instruction. The output of the sequencer (microcode memory address) is valid at the start of segement B and the data is accessed during this segment. In segment C, the data from the microcode memory is valid and is latched into the pipeline register at the end of the cycle (the rising edge of the clock). This rising clock edge is used to latch all registers and devices in the circuit.

Notice from the timing diagram that the instructions in the pipeline register are valid during the complete cycle. Also, while an instruction in the pipeline register is available to circuitry, the next instruction is concurrently being fetched from microcode memory to be subsequently latched into the pipeline register at the start of the next cycle.

To better understand how to program the ADSP-1110A in a pipelined architecture, the ADSP-1110A's instruction set can be divided into three functional classes. The first includes all instructions that cause a register to be loaded. The second class includes output instructions, which are asynchronous. The final class of instructions are the multiply operations. Note that multiply instructions perform multiple operations—they also load the Y register and clock MR.

Class 1 Register Loads	Class 2 Output	Class 3 Multiply
CKMR	Bus = EX	X*Y
X = Bus	Bus = EX (sl)	(All forms)
LS = Bus	Bus = MS	
MS = Bus	Bus = MS (sl)	
EX = Bus	Bus = MS (sat)	
LS = MS	Bus = MS (sl,sat)	
MS = EX	Bus = LS	
EX = SIGN EXT MS	Bus = LS (sl)	
MS = SIGN EXT LS	Bus = LS (sat)	
	Bus = LS (sl,sat)	

Table IV.

Register loads occur at the end of the cycle (rising clock edge) whenever a Class 1 instruction is presented. Output instructions (Class 2) are executed during the same cycle as presented, with the output data becoming valid t_D ns into the cycle and remaining valid throughout the rest of the cycle. This data is available to be latched into an external register, or other device, at the end of the cycle (rising clock edge).

Multiply instructions (Class 3) begin executing at the beginning of the cycle *after* the cycle in which the instruction is presented. These instructions require two cycles to complete. Therefore, when programming the ADSP-1110A with a multiply instruction, it must be noted that the instruction will not start execution until the next cycle, as opposed to Class 1 and 2 instructions, which are executed in the current cycle. This can be illustrated by the following program example.

Cycle	Pipeline Instruction	ADSP-1110A Activity
1	X = BUS	X register loaded with data at end of cycle.
2	Y = BUS; CKMR; X*Y	Y register loaded with data at end of cycle and multiplier control signals latched at end of cycle. Garbage clocked into the MR.
3	X = BUS	Multiply of first operands begins at start cycle, X register loaded with new data at end of cycle.
4	Y = BUS; CKMR; X*Y + MR	Y register loaded with new data. MR loaded with first product and multiplier control signals are latched at end of cycle.
5	NOP	Multiply of second operands begins at start of cycle.
6	CKMR	The sum of the second product and the old MR contents are loaded into the MR at the end of the cycle.
7	BUS = MS	Most significant portion of MR is output to the bus and data is valid t_D ns max into cycle.

Table V.

ROM-Based Sequential Machine

In a similar manner to which the microcode memory of the bit-slice machine provides control bit to circuit components, a ROM can be used in conjunction with a binary counter and a latch to provide these same control bits. Such an approach offers a more compact design, at the expense of versatility. A binary counter is used to sequence through ROM locations, thus implementing a specified algorithm. This architecture is shown below.

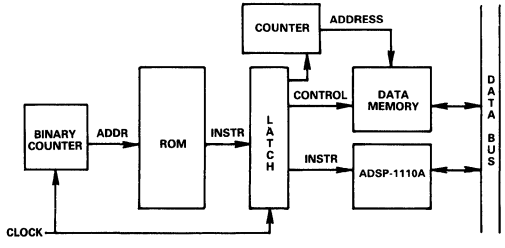


Figure 11.

The ROM contains the necessary bit patterns to control the miscellaneous circuitry and the ADSP-1110A. The binary counter provides sequential addresses to the ROM, resulting in the execution of a specific algorithm. The output of the ROM is latched so that the bits remain stable during the ROM access time associated with the next cycle. A separate counter is used to address the data memory.

This design technique can be expanded to access several functions stored in ROM by using a preloadable counter. The counter is preloaded with the starting address of the desired program in ROM. A dedicated control bit from the ROM is used to flag the counter, denoting the end of the program segment. Note that a latch is placed in front of the preloadable counter so that a host microprocessor can feed the required starting addresses if desired. This design is illustrated below.

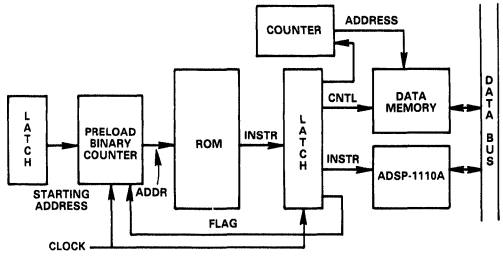


Figure 12.

PLA-Based State Machine

Instead of using the purely sequential approach of the ROM/counter solution, a latched PLA can generate the microcode. This state machine reduces real estate by entirely eliminating the latch (which is internal to the PLA) and the counter. No counter is needed since, in a state machine, the next output state is determined by the current output state. Use of state diagrams and CAD techniques provide the PLA truth table. However, in designs that require many states and complex state diagrams, a ROM-based sequential machine may be easier to implement.

Interfacing the ADSP-1110A to a Microprocessor

Because of its high speed, the ADSP-1110A can be used in an accelerator for microprocessors such as the Intel 286 or the Motorola 68000, performing dedicated macro-routines. The host microprocessor communicates with the ADSP-1110A via memory-mapping or I/O mapping (depending on the microprocessor). Also, the microprocessor and the ADSP-1110A must both have access to data memory. The microprocessor merely triggers the ADSP-1110A circuit and proceeds to perform some other task while the ADSP-1110A executes its macro-routine such as a matrix multiply, inverse function, square-root function, or digital filter. The following diagram illustrates a typical interface to a microprocessor.

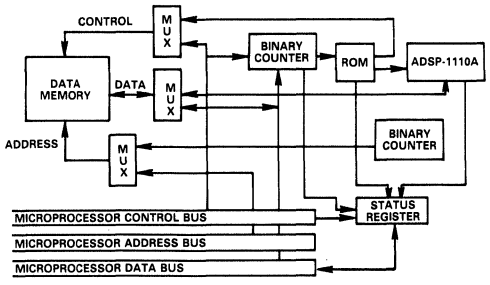


Figure 13.

Any signal that must communicate with the microprocessor is connected to the external status register. The status register is either I/O mapped or memory mapped. The overflow line from the ADSP-1110A, along with the end of program control flag, is also connected to the status register. Also, flags may be used to interrupt the microprocessor. Note that the high-speed data memory is available to both the microprocessor and the ADSP-1110A circuit. The multiplexers performing this selection are controlled by the microprocessor, with multiplexer select lines coming from the external status register.

APPLICATIONS

The ADSP-1110A is a high-performance component for a host of digital signal processing applications including FFT's, digital filters, and double-precision multiplication.

FFT Applications

The fast Fourier transform (FFT) is the principal algorithm used to analyze the frequency content of a signal. The FFT significantly reduces the time required to compute a Fourier transform by taking advantage of patterns in the computations to economize on multiplications. The ADSP-1110A performs the "butterfly," the key arithmetic operation in an FFT, entirely on-chip.

Figure 14 illustrates a decimation-in-time butterfly. As outlined by equations (1)

$$A_0' = A_0 + A_1 e^{j\theta} \qquad (1)$$
$$A_1' = A_0 - A_1 e^{j\theta}$$

the complex number A_1 is multiplied by a rotation term, $R = e^{j\theta}$, and added to the complex number A_0, producing A_0'. A_1' is obtained by subtracting the complex product A_1*R from A_0. The rotation R can be written:

$$R = e^{j\theta} = \cos\theta + j\sin\theta \equiv C + jS \qquad (2)$$

In an FFT A_0 and A_1 are complex numbers. Let

$$A_0 = X_0 + jY_0 \qquad (3)$$
$$A_1 = X_1 + jY_1$$

Then,

$$(A_1)e^{j\theta} = (X_1 + jY_1)(C + jS) \qquad (4)$$
$$= (X_1C - Y_1S) + j(X_1S + Y_1C)$$

allowing A_0' and A_1' to be represented as:

$$A_0' = X_0 + jY_0 + [(X_1C - Y_1S) + j(X_1S + Y_1C)] \qquad (5)$$
$$= X_0' + jY_0'$$
$$A_1' = X_0 + jY_0 - [(X_1C - Y_1S) + j(X_1S + Y_1C)]$$
$$= X_1' + jY_1'$$

Expanding and equating real and imaginary terms yields:

$$X_0' = X_0 + (X_1C - Y_1S) \qquad (6)$$
$$Y_0' = Y_0 + (X_1S + Y_1C)$$
$$X_1' = X_0 - (X_1C - Y_1S)$$
$$Y_1' = Y_0 - (X_1S + Y_1C)$$

Equations 6 can be used by the ADSP-1110A to efficiently implement the butterfly computation. First, X_0 is loaded into

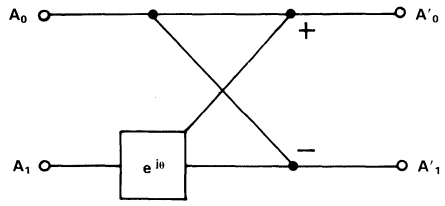


Figure 14. FFT "Butterfly" Diagram

MR by multiplying it by positive full scale (k=0111...1). ("+1" cannot be represented in fractional two's complement, though "−1" can be. Scaling all terms by the factor "0.111111111111111" [binary] fits all values of sine and cosine into fractional two's complement and introduces less error than consistently failing to represent positive unity.) X_0' is obtained as follows:

$$X_0' = kX_0 + X_1C - Y_1S \quad (7)$$

Note that the factor k is not equal to unity. To ensure consistency in results, all stored cosine and sine factors (C and S) should similarly be scaled by k. Then, X_1' can be simply computed:

$$X_1' = kX_0 - (X_1C - Y_1S) \quad (8)$$
$$= 2kX_0 - X_0'$$

where X_0' is the accumulator's contents, and $2kX_0$ results from a mixed-mode multiplication of 2k and X_0 . This operation represents a multiply/subtract, which illustrates an additional feature of the ADSP-1110A. Conventional MAC's cannot perform mixed-mode multiplies or multiply/subtracts.

Table VI provides the details for computing an FFT Butterfly with the ADSP-1110A. Each point requires ten cycles to compute the real component and ten cycles for the imaginary component. A 1024-point FFT requires 5120 butterflies.

A butterfly calculation contains a series of multiply/accumulates and multiply/subtracts. This presents a challenge in rounding the result, because rounded outputs from earlier cycles become inputs in later cycles. The rounding on cycles 4, 6, 14, and 16 ensures that the outputs (lines 7, 10, 17, and 20) are rounded correctly. Lines 4 and 14 round on bit 14, consistent with a left shift during output. However, lines 6 and 16 round on bit 15 to arrive at $X_1'$and Y_1'. In performing the multiply and subtract on these lines, the original round on lines 4 and 14 becomes inverted. To compensate, 2 must be added to the 14th bit, 1 to compensate for the previous round, and then 1 to round the current result. This can be easily accomplished in one step by adding a 1 to bit 15 rather than 2 to bit 14.

5

Cycle	Instruction	Comments
18'.	X = BUS	Load k
19'.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load X_0, MR = Previous
20'.	BUS = MS(sl)	Output Y_1' from previous butterfly
1.	X = BUS	Load C
2.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR	Load X_1, MR = kX_0
3.	X = BUS	Load S
4.	Y = BUS;CKMR; − X_{TC}*Y_{TC} + MR/RND14	Load Y_1, MR = $kX_0 + X_1C$
5.	X = BUS	Load 2k
6.	Y = BUS;CKMR;X_{US}*Y_{TC} − MR/RND15	Load X_0, MR = $kX_0 + (X_1C - Y_1S)$ + RND14
7.	BUS = MS(sl)	Output X_0'
8.	X = BUS	Load k
9.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load Y_0, MR = $kX_0 - (X_1C - Y_1S)$ + RND14
10.	BUS = MS(sl)	Output X_1'
11.	X = BUS	Load S
12.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR	Load X_1, MR = kY_0
13.	X = BUS	Load C
14.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR/RND14	Load Y_1, MR = $kY_0 + X_1S$
15.	X = BUS	Load 2k
16.	Y = BUS;CKMR;X_{US}*Y_{TC} − MR/RND15	Load Y_0, MR = $kY_0 + (X_1S + Y_1C)$ + RND14
17.	BUS = MS(sl)	Output Y_0'
18.	X = BUS	Load k
19.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load new X_0, MR = $kY_0 - (X_1S + Y_1C)$ + RND14
20.	BUS = MS(sl)	Output Y_1'

Table VI. Sample FFT "Butterfly" Sequence

FIR Filters

The ADSP-1110A is readily included in an FIR filter configuration. Figure 15 diagrams an N-tap finite impulse response (FIR) filter. FIR filters perform convolution in the time domain, corresponding to multiplication in the frequency domain. The coefficients h_i represent the filter's impulse response—the time domain equivalent of the filter's desired frequency response.

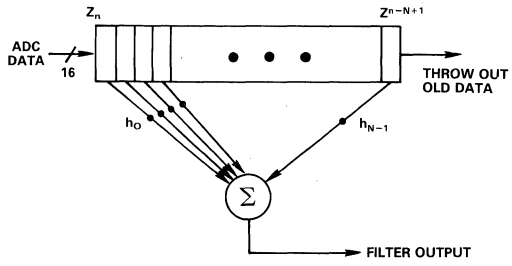


Figure 15. FIR Filter Design

When implemented with the ADSP-1110A, FIR filters employ a single RAM with a memory map as diagrammed in Figure 16. This contrasts with the multiple RAMs usually required in three-port MAC designs. Except in adaptive filters, where the filter response changes to meet changing system requirements, the filter coefficients remain constant.

Input data, on the other hand, is continuously updated. Each new data sample overwrites the oldest data point in RAM, an action that is tracked by an address counter. The Z_{n-N} sample, for instance, is overwritten with the new Z_n sample, and data points are addressed as a circular buffer.

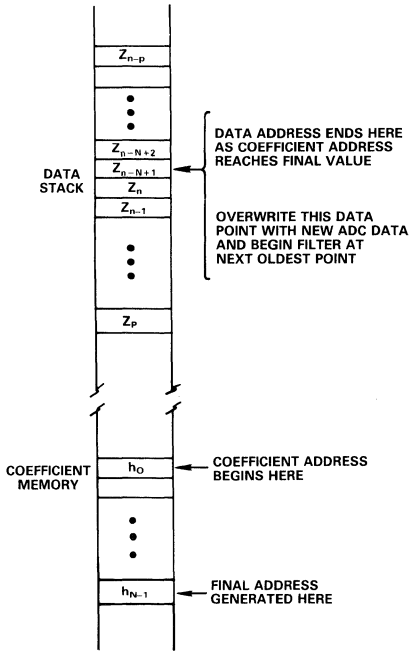


Figure 16. Memory Map

In the time interval between new data samples, the filter multiplies each of the N previously stored data samples, Z_i, by the respective filter coefficients, h_{n-i}. The resulting sum of the products represents the filtered signal output. An overflow flag and optional saturation logic allow long FIR filters to be implemented without risking overflow.

Note that the use of the ADSP-1110A does not require a complicated control circuit. Figure 17, for example, diagrams the controller circuit flow-chart for the FIR filter described above. When implemented in hardware, the controller's complexity remains comparable to that of the controller required for a three-port MAC-based filter design.

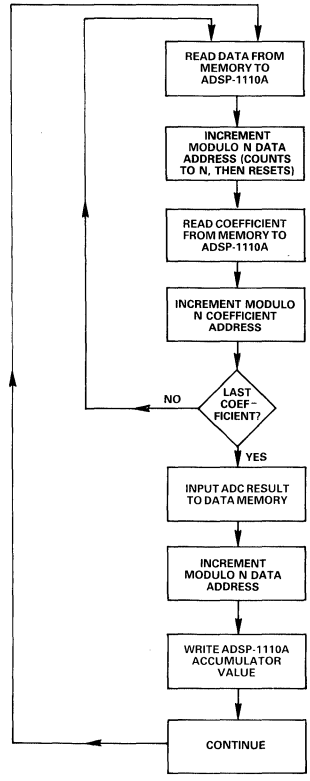


Figure 17. Controller Flow Chart for FIR Filter

IIR Filters

Infinite impulse reponse (IIR) filters use feedback to improve filter performance at the cost of a more complicated design. The principal advantage of an IIR filter is the relatively small number of multiplies needed to achieve a high-performance filter. The ADSP-1110A, unlike conventional MAC's, has architectual features that eliminate some of the disadvantages associated with implementing IIR's.

The time required for the ADSP-1110A to calculate a biquad section of an IIR filter is (5 MAC operations) × (two cycles/operation) + (output cycle), or eleven cycles. In the equation for a biquad,

$$Y_0 = a_0X_0 + a_1X_{-1} + a_2X_{-2} - b_1Y_{-1} - b_2Y_{-2} \quad (9)$$

the coefficient b_1 generally lies between 1 to 2. The most conventional way to represent the coefficients and data is in fractional two's complement notation. Since this numbering system only ranges from -1 to 0.999 . . . , all coefficients and data for the IIR filter have to be divided by 2 to handle b_1 when using a conventional MAC. To compensate, external shifters are needed on output to shift the result up by one bit (multiply by 2).

A coefficient in the $+2$ to -2 range can be handled with the ADSP-1110A by using a mixed-mode multiply. Since the coefficient's sign is known in advance, multiply/add and multiply/subtract operations can supply the sign to an unsigned magnitude number. The MS register is left-shifted as usual on output to obtain the correct result.

Stability is an important issue for IIR filters. The ADSP-1110A's wide accumulator, together with the hard-limiting provided by its saturation circuit, prevent the overflow problems and large-scale oscillations that often plague IIR filters.

Double-Precision Multiplies

In order to handle double-precision multiplication (multiplying two 32-bit two's complement numbers), conventional MACs require additional external logic. The ADSP-1110A, in contrast, performs these operations without external support. Moreover, the device performs a double-precision multiplication in fifteen cycles, seven of which represent overhead.

Equation 10 represents a double-precision multiply:

$$\begin{aligned} P &= (X)(Y) \\ &= (MSW_x + LSW_x \cdot 2^{-16})(MSW_y + LSW_y \cdot 2^{-16}) \\ &= MSW_x \cdot MSW_y + (MSW_x \cdot LSW_y + MSW_y \cdot LSW_x) \cdot 2^{-16} \\ &\quad + LSW_x \cdot LSW_y \cdot 2^{-32} \end{aligned} \tag{10}$$

where P is the 64-bit product of two 32-bit two's complement numbers, X and Y. MSW_x represents the 16 most significant bits of word X, and LSW_x represents the 16 least significant bits. The product P equals the sum of partial products; each partial product's sign and significance must be taken into account in order to obtain the proper result.

A double-precision multiply requires no external logic. Furthermore, as illustrated in the following sequence, the ADSP-1110A performs the operation in 15 cycles.

In this double-precision multiply sequence, shown in Table VII, the four basic multiplications require only eight cycles. Cycles 5, 6, 11, 12, 13, 14, and 15 represent overhead.

Double-Precision MACs

The previous discussion concerned double-precision multiplies. The ADSP-1110A also readily handles double-precision multiply/accumulate operations. For example:

$$AP = \sum_{i=1}^{N} X_i Y_i \tag{11}$$

$$= \sum_{i=1}^{N} (MSW_{xi} + LSW_{xi} \cdot 2^{-16})(MSW_{yi} + LSW_{yi} \cdot 2^{-16})$$

Cycle	Operation	Comments
1.	X = BUS	Load LSW_x.
2.	Y = BUS;CKMR; X_{US}*Y_{US}/RND15	Load LSW_y and multiply (unsigned).
3.	NOP	No op.
4.	Y = BUS;CKMR; X_{US}*Y_{TC} + MR	Load MSW_y and perform MAC (mixed-mode).
5.	LS = MS	MS shifts into LS. The LS of the $(LSW_x)(LSW_y)$ product is discarded.
6.	MS = EX	Shift EX into MS.
7.	X = BUS	Load MSW_x.
8.	Y = BUS;CKMR; X_{TC}*Y_{US} + MR/ RND14	Load LSW_y and perform MAC (mixed-mode) with round in bit 14.
9.	NOP	No op.
10.	Y = BUS;CKMR X_{TC}*Y_{TC} + MR	Load MSW_y and perform MAC (two's complement).
11.	LS = MS	Shift MS into LS.
12.	MS = EX	Shift EX into MS.
13.	CKMR	Clock the output registers. This loads the MAC from cycle 10 into the accumulator.
14.	BUS = MS(sl)	Output MS with left shift.
15.	BUS = LS(sl)/ RND14	Output LS with left shift. The SLE register provides an extra bit of precision.

Table VII.

5

where each X and Y is a 32-bit number, and AP is a 72-bit accumulated product. Note that AP can be expressed as the sum of accumulated partial products as follows:

$$AP = \left[\left\{\sum_{i=1}^{N} (MSW_{xi})(MSW_{yi})\right\}\right] + \tag{12}$$

$$+ \left[\left\{\sum_{i=1}^{N} ((MSW_{xi})(MSW_{yi}) + (LSW_{xi})(MSW_{yi}))\right\}\right] \cdot 2^{-16}$$

$$+ \left\{\sum_{i=1}^{N} (LSW_{xi})(LSW_{yi})\right\} \cdot 2^{-32}$$

Computing the accumulated double-precision product AP requires the same basic sequence as in computing a single-precision MAC. Simply compute a summation of partial products, rather than the summation of products themselves.

The summation of partial products often leads to sums greater than 32-bits. The 8-bit extension register stores any overflow, letting the summation proceed without error. The output and shift cycles occur once each at the end of the appropriate partial product calculation. A 32-point double-precision FIR filter, requires (32-points)(4-multiplies)(2 cycles/multiply) + 9 overhead cycles = 273 total cycles.

An optional procedure, which cuts the multiply/accumulate time by roughly 25%, entails omitting the $LSW_x \times LSW_y$ accumulation and instead adding ¼ of the number of accumulations to the final result. This removes the bias because the LSW's of both words have a mean value of ½ and when multiplied together have a mean product of ¼. Thus any bias in the answer is removed. A simple way to add 1's to the LSB is to assert the round control on the appropriate number of MAC operations.

ABSOLUTE MAXIMUM RATINGS

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

PIN DESIGNATIONS
All Packages

PIN	FUNCTION	PIN	FUNCTION
1	RND15	15	CLK
2	RND14	16	I_3
3	I/O_{14}	17	I_4
4	I/O_{12}	18	I_5
5	I/O_{10}	19	OVF
6	I/O_8	20	I/O_1
7	I/O_6	21	I/O_3
8	I/O_4	22	I/O_5
9	I/O_2	23	I/O_7
10	I/O_0	24	I/O_9
11	I_0	25	I/O_{11}
12	I_1	26	I/O_{13}
13	I_2	27	I/O_{15}
14	GND	28	V_{DD}

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1110AJD	0 to +70°C	28-Pin Ceramic DIP	D-28A
ADSP-1110AKD	0 to +70°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ASD	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ATD	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ASD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ATD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110AJN	0 to +70°C	28-Pin Plastic DIP	N-28A
ADSP-1110AKN	0 to +70°C	28-Pin Plastic DIP	N-28A
ADSP-1110AJP	0 to +70°C	28-Lead Plastic Leaded Chip Carrier	P-28
ADSP-1110AKP	0 to +70°C	28-Lead Plastic Leaded Chip Carrier	P-28

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

ESD SENSITIVITY

The ADSP-1110A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1110A has been classified as a Class 1 device.

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*.

ANALOG DEVICES

Integer Arithmetic Unit

ADSP-1101

FEATURES

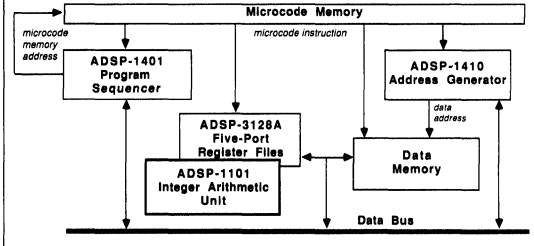
16x16-Bit Parallel Multiplication / 40-Bit Accumulation
80ns Cycle Time
Can Support 2.4ms 1024-Point Complex FFT with Block Floating-Point
40-Bit Adder/Subtracter with Status Flags
16-Bit Logic Unit
Dual 40-Bit Accumulators with Status Flags
Right/Left Shifts on Output Up to 7-Bits
Flexible Load of Six Input Registers
Flexible Preload of Both Accumulators
Feedback from Accumulators to Adder/Subtracter with Left/Right Shift Control
Feedback from Adder/Subtracter to Y Input Registers
On-Chip Block Floating-Point Control
Autonormalized Output with Exponent Output with Saturation
32-Bits-Per-Cycle Data Transfer Rate Through Each 16-Bit Data Port (Two Input and One Output)
Twos-Complement and Unsigned-Magnitude Data Formats
Independent Microcode Control of Each Functional Unit
375mW Power Dissipation in Low-Power TTL-compatible CMOS
100-Pin Grid Array

APPLICATIONS

High-Performance Digital Signal Processing
Digital Filtering
Fourier Transformations
Correlations
General-Purpose Integer Processing
Fast Function Generation

GENERAL DESCRIPTION

The ADSP-1101 Integer Arithmetic Unit (IAU) is a versatile 16-bit integer processor which has at its core a high-speed 16x16 array multiplier, a 40-bit addition/subtraction circuit, and dual 40-bit accumulators (Figure 1). Extensive data paths and support circuitry allow its users to accomplish a broad range of integer processing tasks entirely on-chip, including complex arithmetic. The ADSP-1101 offers a full complement of arithmetic, logic, and shift functions. Block Floating-Point Control logic is also provided. Sustainable single-cycle operations of the form y=mx+b are also supported.



Word-Slice Microcoded System with ADSP-1101

The ADSP-1101 is ideally suited for signal processing applications such as digital filters and FFTs. Multiple ADSP-1101s can be cascaded to perform FIR filters at a single-cycle throughput rate by storing filter coefficients in input registers and passing partial sums of products to one of the Accumulators of the next IAU in the chain. The ADSP-1101 simplifies FFTs by performing six-cycle radix-2 butterfly operations entirely on-chip. Fast function generation (using Taylor/Chebyshev series, etc.) and other algorithms employing series of products can also be performed on-chip.

The ADSP-1101 has two input ports and an output port. Both of the independently controlled Y registers may be loaded from either input port. One pair of X input registers is loadable from the X-Port, the second pair, from the Y-Port. Both Accumulators may be preloaded from the Y-Port. Up to six 16-bit words can be transferred through the ADSP-1101's three data ports in a single cycle, thereby avoiding bottlenecks at the input ports and output port.

Data from the Y input registers can be passed through the Logic Unit prior to entering the Multiplier Array. The Adder/Subtracter, fed by the Multiplier Array and Accumulators, produces a result that may be routed to either one or both of the two Accumulators or to either or both Y input registers. The contents of either Accumulator can be fed back to the Adder/Subtracter or routed to the output port (via the Output Shifter). Block Floating-Point Control is implemented entirely on-chip.

5

The ADSP-1101's 20-bit Z-Port can output a 16-bit data word or Status Register on its lower-order 16-bits and extension data, status flags, or an exponent from an autonormalized output on its high-order 4 bits. (Adder/Subtracter flags are specified only 0-70°C.) The 16-bit data word can come from the Accumulator. Like the X and Y input ports, the Z-Port can transfer data at twice the clock rate.

The ADSP-1101's 39-bit instruction word is divided into subfields that allow independent control of the IAU's various functional elements. Instruction subfields which don't change can be hardwired, thus conserving microcode memory. A number of instructions may be conditioned on internal or external status.

The ADSP-1101 is fabricated in double-metal 1.5 µm CMOS and consumes 375mW maximum, significantly less than comparable bipolar solutions. The differential between the chip's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, unlike similar bipolar devices, the ADSP-1101 can be safely specified for operation at environmental temperatures over its extended temperature range (–55°C to +125°C ambient).

The ADSP-1101 is available for both commercial and extended temperature ranges. Extended temperature range parts are available processed fully to MIL-STD-883, Class B. The ADSP-1101 is available packaged in a ceramic 100-lead pin grid array.

TABLE OF CONTENTS

PIN DESCRIPTIONS

PIN NAME	DESCRIPTION
DATA PORTS	
X_{15-0}	16-bit X Input Data
Y_{15-0}	16-bit Y Input Data
Y_{7-0}	Pass Magnitude Register and Shift Control Register Preload
Y_{3-0}	Shift Control Register Preload
Y_{3-0}	Bit Growth Register Preload
Z_{19-0}	20-bit Z Output Data and Status Registers
Z_{19-16}	4-bit Exponent of Autonomalized Output
Z_{19}	Accumulator A Overflow (OVFLA)
Z_{18}	Adder/Subtracter Zero (ZEROAS)
Z_{17}	Adder/Subtracter Overflow (OVFLAS)
Z_{16}	Accumulator B Overflow (OVFLB)
INSTRUCTION PORT	
IEXT	External Condition Flag
I_{38-33}	X-Buffer Input Control (XBUF)
I_{32-27}	Y-Buffer Input Control (YBUF)
I_{26-24}	Accumulator Feedback Control (FDBK)
I_{23-16}	Arithmetic/Logic Functions (ARITHL)
I_{15-12}	Accumulator A Write Control (ACCA)
I_{11-8}	Accumulator B Write Control (ACCB)
I_{7-0}	Output, BFP, and Shift Control (OUT)
STATUS FLAG	
SIGNAS	Adder/Subtracter Sign
MISCELLANEOUS	
CLK	Clock
GND	Ground (3 lines)
V_{dd}	+5V Power Supply (3 lines)

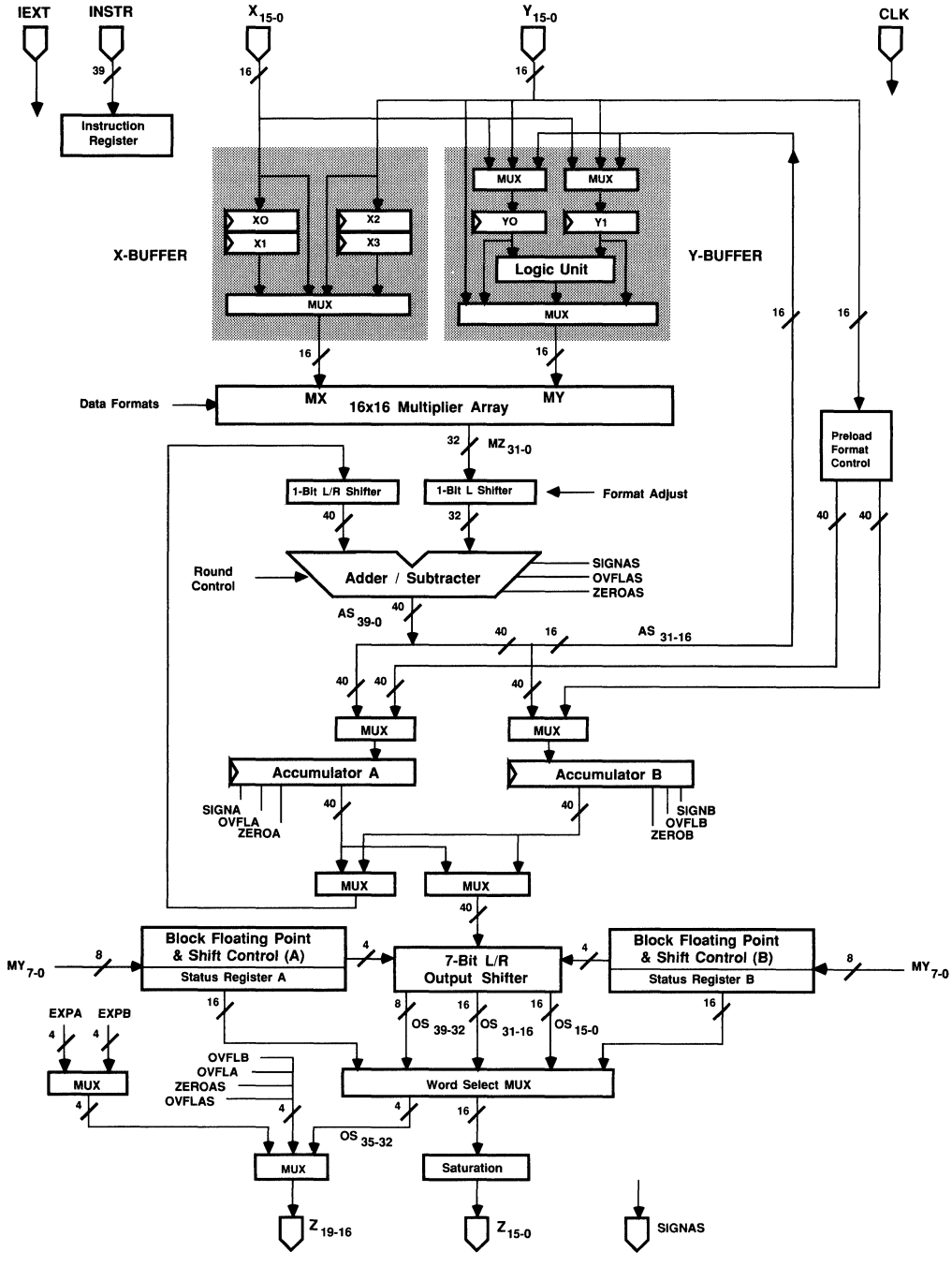


Figure 1. ADSP-1101 Functional Block Diagram

	Input Control		Arithmetic and Logic Control				Output Control
IEXT (IEXT)	XBUF (38:33)	YBUF (32:27)	FDBK (26:24)	ARITHL (23:16)	ACCA (15:12)	ACCB (11:8)	OUT (7:0)
External Condition	X-Buffer	Y-Buffer	Accumulator Feedback	Arithmetic / Logic Functions	Accumulator A write	Accumulator B write	Output, BFP, & Shift

Figure 2. ADSP-1101 Instruction Word Organization

INSTRUCTION ORGANIZATION AND TIMING

The ADSP-1101 features a highly orthogonal instruction set. Its 39 instruction bits are fielded to afford independent control over the key functional blocks of the Integer Arithmetic Unit (Figure 2). As a consequence, the ADSP-1101 can be treated as a single-chip integer processing subsystem. If all the flexibility of the Integer Arithmetic Unit is not required, a user can reduce the width of the instruction word coming from microcode memory by tying unchanging instruction pins to GND or +5V.

Instruction pins are numbered from I_{38} to I_0. To aid in the readability, they are referenced in this data sheet by the relevant instruction field within the instruction word. For example, instruction pins I_{23} through I_{16} are called "ARITHL23:16" since those pins control the IAU's arithmetic and logic functions. The numerical references match the "I" pin numbers.

Several arithmetic, shift, and data-path instructions can be conditioned on the state of an internal programmable flag, IFLAG. IFLAG can reflect sign, zero, or overflow status from the Adder/Subtracter. Alternatively, IFLAG can reflect the state of an external pin (IEXT), allowing for externally controlled conditional instructions. By depending on IFLAG, these conditional instructions can be made dependent on IEXT or on any one of these three Adder/Subtracter internal status conditions.

The ADSP-1101 contains an Instruction Register so that the user does not have to hold microcode instructions valid throughout the clock cycle. All instructions and IEXT share the same setup (t_{IS}) and hold-time (t_{IH}) requirements relative to the clock's rising edge (though not all are in fact registered into the internal Instruction Register). Control lines which select the data paths to registers Y0, Y1, Accumulator A, and Accumulator B at the clock's rising edge are asynchronous, allowing the muxes they control to establish data paths prior to the rising edge. Nonetheless, the user can treat all instructions as if they were registered since the asynchronous controls are no longer needed after their hold-time requirements have been met; any change after the clock edge will have no effect.

No instruction fields are internally pipelined, allowing the user complete control over the sequence of operations. When writing Adder/Subtracter results to an Accumulator and then outputting these results, for example, the instruction fields for arithmetic/logic operations would be presented on one rising edge of the clock and the instruction fields for writing this result to the Accumulators and outputting it would be presented on the next rising edge. (See Fig. 19) Instructions that put the output port into a high-impedance state take effect in the cycle after the rising edge at which they are presented.

Suggested mnemonics for the ADSP-1101's instructions have been chosen to be as short as possible while remaining descriptive. The meta-assembler-level instruction for a single cycle of IAU execution can consist of 18 or 19 independent mnemonics. Readability requires that each mnemonic express its operation in some intuitive manner. In most cases, the class of operations is denoted by the first few characters of each mnemonic. Because of the complex options available in the Accumulator write instruction sets, the mnemonics for these instructions have themselves been fielded. Some commonly used conventions include

Mnemonic	*Meaning*
XP	X-Port
YP	Y-Port
ACC	selected Accumulator
AS	Adder/Subtracter
A	absolute
P	plus
M	minus
N	negate or no change
S	sign extend
PRD	product
PASS	pass
L	shift left one bit
R	shift right one bit
U	unsigned-magnitude
2	twos-complement
Z	zero
D	default sign
LW	least significant word
MW	most significant word.

Data transfer operations have been represented in the format "[*source*]T[*destination*]" whenever anything shorter would be ambiguous, "T" meaning "to." Conditional instructions have been represented as "[*result if true*]E[*result if false*]," "E" meaning "else."

A summary of the ADSP-1101's instruction set can be found at the end of this data sheet.

INPUT BUFFERS AND TIMING

The ADSP-1101's X-Port and Y-Port are 16-bit input ports that provide data to input data buffers, the X-Buffer (Figure 3) and the Y-Buffer (Figure 4). The X-Buffer consists of four 16-bit registers, two feedthrough data paths, and muxes. Registers X0 and X1 accept data from the X-Port. Registers X2 and X3 accept data from the Y-Port. The Y-Buffer consists of two 16-bit registers, one feedthrough path, the Logic Unit, and muxes. Independently controlled registers Y0 and Y1 can both accept data either from the X-Port, from the Y-Port, or from the Adder/Subtracter.

The Y-Port can also provide up to 16-bits of data per clock phase to preload either or both Accumulators. The X-Buffer and the Y-Buffer provide inputs to the Multiplier Array and the Adder/Subtracter. The Logic Unit physically resides in the Y-Buffer, though is controlled primarily by Arithmetic and Logic Control (ARITHL23:16) instructions.

Registers in the X-Buffer can be written on either the rising or

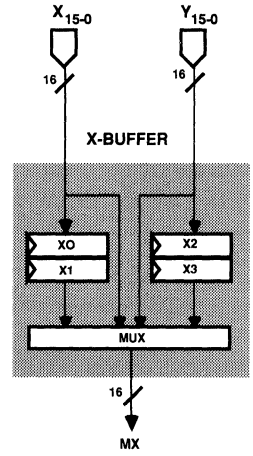


Figure 3. ADSP-1101 X-Buffer

	Input Control		Arithmetic and Logic Control				Output Control
IEXT (IEXT)	XBUF (38:33)	YBUF (32:27)	FDBK (26:24)	ARITHL (23:16)	ACCA (15:12)	ACCB (11:8)	OUT (7:0)

XBUF

38	37	36	35	34	33
X	X	X	X	X	X

	multiplier MX read address	mnemonic
0 0 0	X0	X0
0 0 1	X1	X1
0 1 0	X2	X2
0 1 1	X3	X3
1 0 0	X-Port	XP
1 0 1	Y-Port	YP
1 1 0	If IFLAG, X0; else, X2	X0EX2
1 1 1	If IFLAG, X2; else, X3	X2EX3

	clock edge for write	mnemonic XWxxx
0	rising	default
1	falling	XWFALL

	write address	mnemonic xPTXx
0 0	X0 from X-Port	XPTX0
0 1	X1 from X-Port	XPTX1
1 0	X2 from Y-Port	YPTX2
1 1	X3 from Y-Port	YPTX3

Table I. ADSP-1101 X-Buffer Instruction Set

falling edge of the clock, a mode that can be changed dynamically under microcode control. Y-Buffer registers, however, can be written only at the rising edge. Only one X-Buffer register can be written in a given cycle. But some X-Buffer register must be written to in every cycle. A "dummy" X -Buffer register should be designated to receive garbage X-Port data on cycles when valid data is *not* presented to the X-Port. The Y-Buffer registers, however, can be independently controlled, allowing both Y0 and Y1 to be loaded from any of three sources in the same cycle, if desired, or not loaded at all.

Both input buffers include feedthrough paths which bypass the input registers. Thus, the user can eliminate the level of pipelining normally involved in loading input data, though there is no throughput or latency advantage to doing so. Note that data loaded directly to the multiplier ports, MX or MY, can also be concurrently loaded to one or more available registers in the input buffers and preloaded to one or both Accumulators.

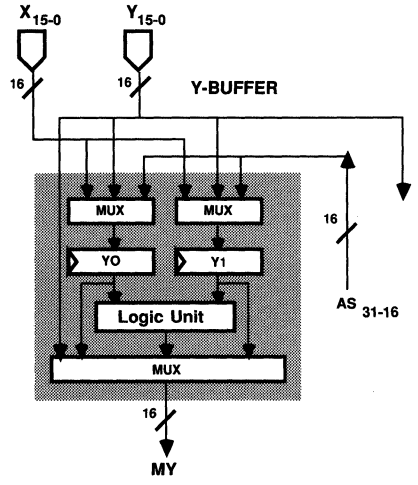


Figure 4. ADSP-1101 Y-Buffer

	Input Control		Arithmetic and Logic Control				Output Control
IEXT (IEXT)	XBUF (38:33)	YBUF (32:27)	FDBK (26:24)	ARITHL (23:16)	ACCA (15:12)	ACCB (11:8)	OUT (7:0)

YBUF

32 31	30 29	28 27
X X	X X	X X

	multiplier MY read address	mnemonic
0 0	Y0	Y0
0 1	Y1	Y1
1 0	If IFLAG, Y0; else Y1	Y0EY1
1 1	Y-Port	YP

	Y1 write control	mnemonic xxxTY1
0 0	write from Y-Port	YPTY1
0 1	don't write	default
1 0	write from Adder/ Subtracter	ASTY1
1 1	write from X-Port	XPTY1

	Y0 write control	mnemonic xxxTY0
0 0	write from Y-Port	YPTY0
0 1	don't write	default
1 0	write from Adder/ Subtracter	ASTY0
1 1	write from X-Port	XPTY0

Table II. ADSP-1101 Y-Buffer Instruction Set

When using the feedthrough data paths, results must be clocked into one of the Accumulators before changing any input data. The data at the input ports must remain valid until t_{DHAS} before the next rising edge of the clock to insure stable results at that clock edge and also to insure stable Adder/Subtracter flags (Figure 19).

X-Buffer

The X-Buffer accepts input from either the X-Port or the Y-Port as indicated in XBUF34:33 (Table I). Note that X0 and X1 are written only from the X-Port, and X2 and X3, only from the Y-Port. XBUF35 determines whether register loading occurs on the rising edge or on the falling edge. The Multiplier Array's input source at its MX port is determined by XBUF38:36. This source can be either port or any X register. The choice of X-Buffer register can be conditional on the state of IFLAG at the beginning of the cycle.

The registers in the X-Buffer all meet the same setup (t_{DSS}) and hold time (t_{DH}) requirements regardless of whether they are clocked on the rising or falling edge. Note that when writing to X-Buffer registers on the falling edge, two additional requirements must be met. First, if an X-Buffer register is loaded mid-cycle and that register is selected as a source to the MX Multiplier Array port in that same cycle, clock LO will have to be extended to at least the minimum value of t_{CLK} as listed in "Specifications" to insure arithmetic circuits have been allowed sufficient time for propagation delays. Second, an X-Buffer register cannot be written mid-cycle on a falling edge and then re-written at the next rising edge. If attempted, the second write will not occur.

Y-Buffer

In addition to X-Port and Y-Port data sources, the Y-Buffer can also accept post-rounded bits AS_{31-16} from the Adder/Subtracter (the Adder/Subtracter's Most Significant Word) as an input (Figure 4). 16-bit results from either Accumulator can be passed through the Adder/Subtracter to load either or both of the Y-Buffer registers. Because of the feedback paths from the Adder/Subtracter to the Y-Buffer registers (and from the Accumulators to the Adder/Subtracter), these input registers can serve as auxiliary 16-bit temporary working registers. This feature is valuable in calculations involving products of sums and/or products of products.

The Y-Buffer registers accept input on the rising edge of the clock from the X-Port, the Y-Port, or the Adder/Subtracter as indicated in YBUF28:27 and YBUF30:29 (Table II). The Multiplier Array's MY port can accept data from either Y-Buffer register. Feedthrough data can come only from the Y-Port. The data source for the Multiplier Array's MY port is determined by YBUF32:31. The choice of Y-Buffer register can be conditional on the state of IFLAG at the beginning of the cycle in which the conditional instruction is executed.

For logic operations (which employ the Logic Unit), YBUF32:31 are redefined as shown in Table IV. A logic operation is defined by ARITHL18:16 = "111" (non-multiplication instruction) and ARITHL23:21 = "111" (logic instruction). The Logic Unit's operands always come from Y0 and Y1. Thus, the source for the logic operation as the Y-Buffer registers is implicit in the very fact that a logic operation is being executed. Hence, in a logic operation there would be no need to use YBUF32:31 to specify the Logic Unit's data source, and these instruction bits can be (and are) reused.

DATA FORMATS

The ADSP-1101 Integer Arithmetic Unit can process twos-complement, unsigned-magnitude, or mixed-mode fixed-point data in multiplication operations. The data formats for twos-complement and unsigned-magnitude input data are shown in Figure 5. The variable "k" determines the user's placement of the implicit binary point, which can be placed wherever desired. Integers are represented when k=0. Fractional twos-complement numbers are represented when k=–15; fractional unsigned-magnitude numbers are represented when k=–16. "Mixed-mode" operations are those with one twos-complement operand and a second unsigned-magnitude operand.

WEIGHT	Sign -2^{k+15}	2^{k+14}	2^{k+13}	...	2^{k}
VALUE	i_{15}	i_{14}	i_{13}	...	i_{0}
POSITION	15	14	13	...	0

16-Bit Twos-Complement Fixed-Point

WEIGHT	2^{k+15}	2^{k+14}	2^{k+13}	...	2^{k}
VALUE	i_{15}	i_{14}	i_{13}	...	i_{0}
POSITION	15	14	13	...	0

16-Bit Unsigned-Magnitude Fixed-Point

Figure 5. ADSP-1101 Input Data Formats

Data formats for arithmetic inputs to the Multiplier Array are specified with ARITHL23:22 pins in the Arithmetic and Logic Control / Multiplication instruction set. (See Table IV. Data formats are specified at the inputs to the Multiplier Array, *not* within the Input Buffers.) The Accumulator control instructions also support multiple formats for data preloaded from the Y-Port. Internally, the IAU tracks the data format of every result. This format tracking is essential for proper shifting, saturation at output, extension of data to wider fields, and block floating-point control. Both Accumulators and the Adder/Subtracter generate flags indicating whether their most recent contents were twos-complement or unsigned-magnitude. These flags are available in the Status Registers and can be read through the Z-Port. (See "Status Flags.")

In general, if any term in a multiplication or multiplication/accumulation operation is formatted for twos-complement, its result will be flagged as a twos-complement number. Unsigned-magnitude results are obtained only from logic operations and from Multiplication Instructions when all input and Accumulator terms are unsigned-magnitude. Mixed-mode and twos-complement operations yield twos-complement results. Mixed-mode multiplications can be useful for

increasing the precision of calculations, since twos-complement multiplication results normally contain a redundant sign bit. Mixed-mode is also useful for intermediate cross-terms in double-precision multiplications.

ACCUMULATOR FEEDBACK CONTROL

Many instructions controlling the Multiplier Array and the Adder/Subtracter (ARITHL23:16) make use of feedback data from one of the two 40-bit Accumulators. These 40-bit data paths are illustrated in Figure 6. The Accumulator Feedback instructions (Table III) select the feedback path for a particular arithmetic operation. Thus, any arithmetic operation referencing an Accumulator will use the Accumulator specified in the feedback instruction. This data can be shifted right or left by one bit before entering the Adder/Subtracter. Several instructions are conditional on the state of IFLAG.

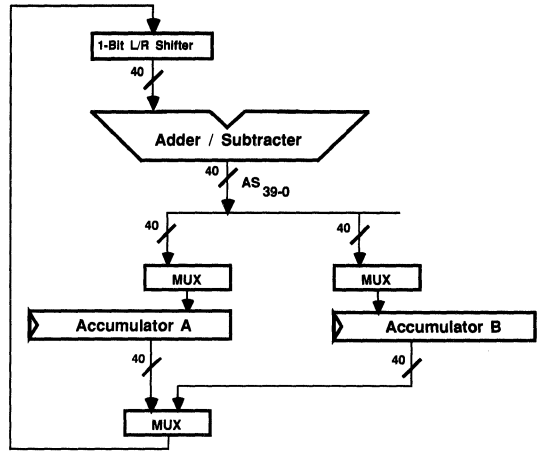


Figure 6. ADSP-1101 Accumulator Feedback Data Paths

	Input Control		Arithmetic and Logic Control				Output Control
IEXT (IEXT)	XBUF (38:33)	YBUF (32:27)	FDBK (26:24)	ARITHL (23:16)	ACCA (15:12)	ACCB (11:8)	OUT (7:0)

FDBK

26 25 24

X X X

	accumulator selection	shift selection at AS input	mnemonic FBxxx
0 0 0	AccA	no shift	FBA
0 0 1	AccB	no shift	FBB
0 1 0	If IFLAG, AccA; else AccB	no shift	FBAEB
0 1 1	If IFLAG, AccB; else AccA	no shift	FBBEA
1 0 0	AccA	If IFLAG, shift left; else no shift	FBALE
1 0 1	AccB	If IFLAG, shift left; else no shift	FBBLE
1 1 0	AccA	If IFLAG, shift right; else no shift	FBARE
1 1 1	AccB	If IFLAG, shift right; else no shift	FBBRE

Table III. ADSP-1101 Accumulator Feedback Instruction Set

Bit Position		
39 38 … 33 32	31 30 … 17 16	15 14 … 2 1 0
MSB EXT LSB	MSB MSW LSB	MSB LSW LSB

Figure 7. Data Fielding for Accumulators

The instruction field, FDBK26:24, determines the feedback option applicable to the current arithmetic operation. Left shifts are logical. (When Accumulator data is shifted left one bit, the Least Significant Bit (LSB) entering the Adder/Subtracter will be zero.) Right shifts are arithmetic. (When Accumulator data is shifted right one bit, the Most Significant Bit [MSB] entering the Adder/Subtracter will be sign-extended from Accumulator bit 39 in the case of twos-complement data or zero in the case of unsigned-magnitude data). These shift options are useful for effectively multiplying or dividing the contents of an Accumulator by two before adding it to or subtracting it from a product from the Multiplier Array.

ARITHMETIC AND LOGIC CONTROL

Overview

The arithmetic/logic blocks of the ADSP-1101 Integer Arithmetic Unit consist of the 16-bit Logic Unit, located in the Y-Buffer, the 16x16 Multiplier Array, and a 40-bit Adder/Subtracter. See Figure 8 for the functional arithmetic/logic blocks of the IAU. All operations using these blocks begin execution with the clock's rising edge and require t_{CLK} for completion. (When outputting twice per cycle or executing the autonormalize instruction, the clock period may have to be extended, however, to allow for the data output delay time. See "Output Control and Timing" below.) Operations are controlled primarily by the ARITHL23:16 instruction pins (Table IV). The Arithmetic and Logic Control instruction set consists of Multiplication Instructions and Non-Multiplication Instructions, as determined by ARITHL18:16.

The Logic Unit residing in the Y-Buffer performs logical operations on the contents of Y0 and Y1 and supplies the result to the MY input of the Multiplier Array.

The parallel Multiplier Array accepts 16-bit inputs through its MX port from the X-Buffer and 16-bit inputs through its MY port from either the Y-Buffer or the Logic Unit. Twos-complement, unsigned-magnitude, and mixed-mode are supported input data formats for multiplication operations. All results are internally tagged as either twos-complement or unsigned-magnitude. This format information is available in the two 16-bit Status Registers, one for each Accumulator. Unbiased rounding is supported for Multiplication Instructions

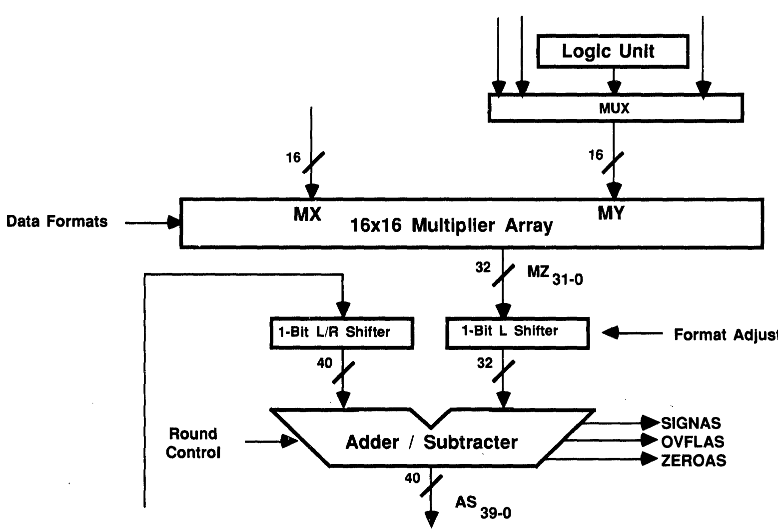


Figure 8. ADSP-1101 Logic Unit, Multiplier Array, and Adder/Subtracter

at any one of three Adder/Subtracter bit positions (AS_{16}, AS_{15}, or AS_{14}). The Multiplier Array's 32-bit product (MZ_{31-0}) can be left-shifted (logically) by one bit (Format Adjusted) to eliminate the normally redundant extra sign bit in twos-complement products before entering the Adder/Subtracter or, more generally, to scale by two.

The Adder/Subtracter accepts a 32-bit result from the Multiplier Array and a 40-bit Accumulator result from either Accumulator. The 40-bit Accumulator result can be shifted right arithmetically or left logically one bit before entering the Adder/Subtracter, as described in "Accumulator Feedback Control." The Adder/Subtracter's 40-bit output can be routed to either or both Accumulators and/or directly and asynchronously to the Output Shifter and Z-Port. The 16-bit Most Significant Word (MSW) from the Adder/Subtracter (AS_{31-16}) can also be routed to either or both registers in the Y-Buffer.

The dual 40-bit Accumulators, A and B, accept inputs from two sources: either the Adder/Subtracter or the Y-Port (via the Preload Format Control). The Accumulators are each fielded into a 16-bit Least Significant Word (LSW), a 16-bit Most Significant Word, and an 8-bit Extension (EXT) byte (Figure 7). The wide EXT byte guarantees no true data loss for at least 256 multiplication/accumulation operations. Each Accumulator can be independently written with Y-Port data. The Preload Format Control directs this data to any of the three Accumulator subfields. The other two fields in each register not receiving Y-Port data can be simultaneously cleared, sign-extended, or left unchanged. (See "Accumulator Write Control" for a complete description of the available options.)

ARITHMETIC AND LOGIC CONTROL

Multiplication Instructions

(ARITHL18:16 ≠ "111")

The Multiplication Instructions, shown in Table IV, control the IAU's multiplication and multiplication/accumulation operations. Inputs to the Multiplier Array's MX and MY ports are specified by the X-Buffer and Y-Buffer instruction fields. Input data formats for Multiplication Instructions are specified by ARITHL23:22. The Multiplier Array produces a 32-bit product (MZ_{31-0}) from these two 16-bit inputs.

For "X•Y" and "AccA/B + X•Y" instructions (ARITHL18:17="00"), the 32-bit product will be in twos-complement format if any operand is twos-complement. In Table IV, these format results are indicated by "2sC if any." If all operands are unsigned-magnitude, the product will be unsigned-magnitude. The remaining Multiplication Instructions always produce a twos-complement result. This fact is indicated by "2sC"s in Table IV for these remaining four instructions.

The 32-bit products leaving the Multiplier Array can be "format adjusted," that is, uniformly left-shifted by one bit. The Format Adjust operation is most useful for twos-complement products since they normally contain redundant sign bits in MZ_{31-30}. Unlike with first-generation array multipliers, Format Adjust on the ADSP-1101 uniformly left shifts (logical) all 32 bits one position before entering the 40-bit Adder/Subtracter. Unsigned-magnitude and mixed-mode products can also be left shifted one position using Format Adjust, an operation equivalent to multiplying by two.

Data entering the Adder/Subtracter on MZ_{31-0} is extended to 40-bits according to its data format. Twos-complement (and mixed-mode) data is signed-extended; unsigned-magnitude data is zero-extended. Because twos-complement data is sign-extended, format-adjusted full-scale negative times full-scale negative products are fully representable; there is no true data overflow in the sense of lost data. The Adder/Subtracter's overflow flag (OVFLAS) will indicate that its result has overflowed into the EXT field (AS_{39-32}) if this is the case after its operation. (See "Status Flags and Registers.")

The ADSP-1101 offers four rounding options for multiplication operations, controlled by ARITHL20:19. Rounding occurs in the Adder/Subtracter, regardless of whether it is the result of a multiplication or a multiplication/accumulation operation. The no-rounding option leaves the output from the Adder/Subtracter unaltered. The three remaining options allow unbiased rounding at one of three bit positions in the Adder/Subtracter's output field: AS_{16}, AS_{15}, or AS_{14} (Figure 9).

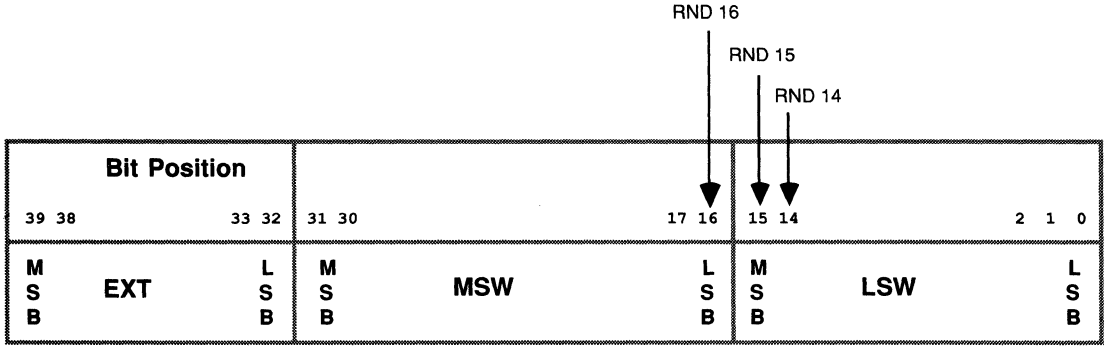
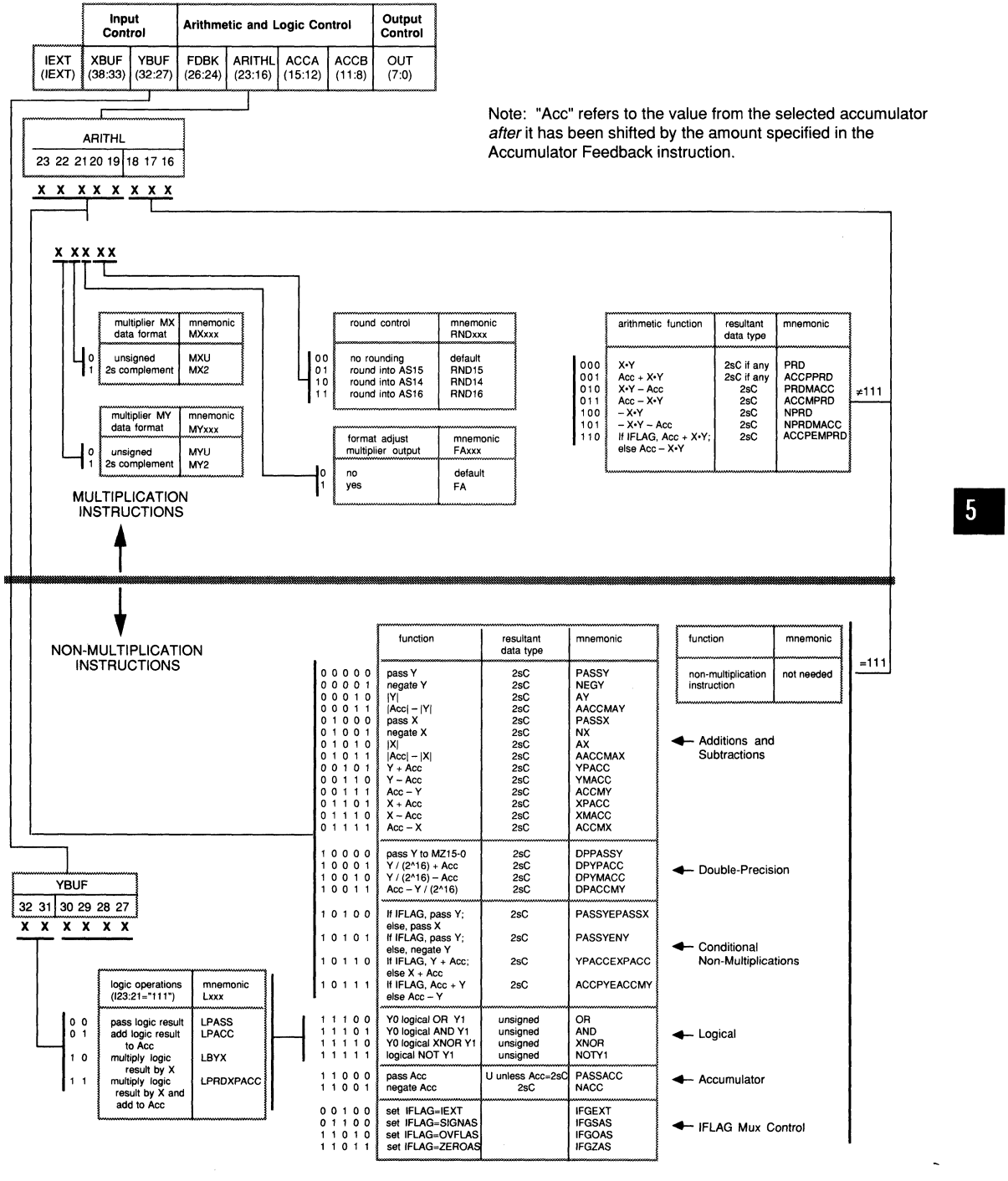


Figure 9. ADSP-1101 Rounding Positions



Note: "Acc" refers to the value from the selected accumulator *after* it has been shifted by the amount specified in the Accumulator Feedback instruction.

5

Table IV. ADSP-1101 Arithmetic and Logic Control

Unbiased rounding adds a binary one to the chosen bit position (with carry), *except* when the chosen bit position contains a "1" and all bits of lesser significance are "0." At this "mid-scale" condition, the ADSP-1101 will round to even, which may or may not imply that a "1" be added. What "round to even" means is that when the bit field bounded on the left by the chosen rounding position equals "1000...000" binary, the post-rounded, higher-order bits comprising the result field will be even, i.e. its LSB (the bit immediately to the left of the chosen rounding position) will be "0." As a consequence, in a large, statistically random sample, the IAU will round up as often as it rounds down. The processed data will not exhibit the large-sample statistical bias of +1/2 LSB of the LSW characteristic of industry-standard multipliers and multiplier/ accumulators, which *always* add a binary one to the MSB of the LSW.

Three choices of bit position make rounding consistent with 1-bit shifts left or right or with no shifting on output. If output data is not shifted, the user will normally round at AS_{15}. With a 1-bit left shift, the user will normally round at AS_{14}; with a 1-bit right shift, at AS_{16}. Note that rounding occurs subsequent to Format Adjust; the bit positions are defined in the Adder/Subtracter (AS) fields, *not* in the Multiplier Array output field (MZ).

If the user desires to round at bit positions other than AS_{16}, AS_{15}, or AS_{14}, a binary one can be added to any bit position in the MSW and LSW of the AS field using the appropriate Non-Multiplication Instruction. Of course, this rounding won't be unbiased.

ARITHMETIC AND LOGIC CONTROL

Non-multiplication Instructions

(ARITHL18:16 = "111")

The Non-Multiplication Instructions are invoked whenever ARITHL18:16 = "111" is specified (Table IV). Instruction bits ARITHL23:19 are redefined for Non-Multiplication Instructions. This class of instructions can be further subdivided into Additions-and-Subtractions, Double-Precision Instructions, Conditional Non-Multiplications, Logical Instructions, Accumulator Instructions, and IFLAG Mux Control Instructions.

Note that Additions-and-Subtractions, Double-Precision Instructions, and Conditional Non-Multiplications (Table IV) always produce twos-complement results. Logical Instructions always produce unsigned-magnitude results. Pass Accumulator produces an unsigned-magnitude result unless the last value written to the selected Accumulator was twos-complement. Negate Accumulator always produces a twos-complement result. IFLAG Mux Control Instructions produce no data results at all.

Additions-and-Subtractions and Conditional Non-Multiplications accept inputs from either Multiplier Array port (MX or MY, but not both). These 16-bit inputs are simply passed through the Multiplier Array output port to lines MZ_{31-16}. (Lines MZ_{15-0} are cleared.) They are thus scaled by 2^{16} to the MSW of the Accumulators and Adder/Subtracter. (Format Adjust is not an option in the Non-Multiplication Instruction set.) In some cases, particularly Double-Precision calculations, it is useful alternatively to scale values through the Multiplier Array to lines MZ_{15-0}, the output LSW. Four Double-Precision instructions read the value at MY into these LSW positions. Double-Precision instructions also clear lines MZ_{31-16}. The Double-Precision instructions can pass this down-scaled value from MY or add/subtractit to or from a selected Accumulator.

The four Logical Operations are indicated by ARITHL23:21 = "111" (and ARITHL18:16 = "111"). As explained above in "Input Buffers and Timing," the Y-Buffer YBUF32:31 instruction bits are not needed to specify a data source to the Logic Unit, since it always takes Y1 or both Y0 and Y1 as its sources. These YBUF instruction bits allow four permutations of the four Logical Operations: the output of the Logic Unit can be passed on to AS_{31-16} unchanged, multiplied by X, added to Accumulator A or B, or both multiplied by X and added to Accumulator A or B. (Note that these multiplications are classified as "Non-Multiplication Instructions.")

The IFLAG Mux Control Instructions determine which of four possible flags IFLAG will represent on the current and subsequent cycles (Figure 10). The four inputs to the IFLAG Mux are IEXT (the external condition flag), SIGNAS (the sign of the Adder/Subtracter), OVFLAS (the overflow flag from the Adder/Subtracter), and ZEROAS (the zero flag from the Adder/Subtracter). Once set, the path through the IFLAG Mux remains set until changed. It has no default value and must be initialized at power up.

The IFLAG multiplexer can select one of three Adder/Subtracter flags. When it does, IFLAG is updated at the end of every cycle with that cycle's Adder/Subtracter flags. The Adder/Subtracter flags can therefore be used, via IFLAG, as conditions for the next cycle's execution. For example, if IFLAG is set to SIGNAS and

X – AccA

generated a negative result, the very next instruction

If IFLAG, then AccA + Y; else AccA – Y

would yield the first result, AccA + Y, from the Adder/Subtracter. (SIGNAS is set true for negative twos-complement results.) The IFLAG multiplexer can also select the external condition, IEXT. IEXT, if selected as IFLAG, becomes the condition for the instruction with which it is set up. See "Status Flags" for a further discussion of using the ADSP-1101's various status flags.

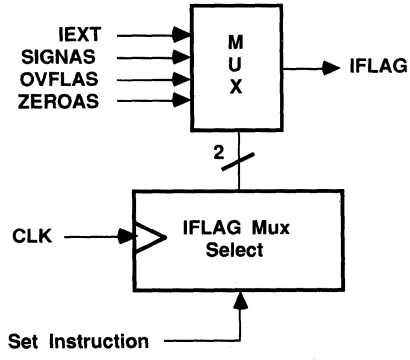


Figure 10. ADSP-1101 IFLAG Mux

ACCUMULATOR WRITE CONTROL

The ADSP-1101's dual Accumulators have independent, identical write control instruction sets. Accumulator A's Write Control Instruction set, ACCA15:12, is shown in Table V; Accumulator B's, ACCB11:8, in Table VI. They control the Accumulators and data paths as shown in Figure 11.

This instruction set controls three distinct fields within the Accumulators (EXT, MSW, and LSW), as well as an Accumulator sign flag. The two input sources to the Accumulators are the Adder/Subtracter and the Preload Format Control, which takes data from the Y-Port. Because the two instruction sets are independent, a large number of possible Accumulator Write Control combinations are possible in a single cycle. Unless otherwise indicated in Tables V and VI, the Accumulator fields are loaded on the rising edge of the clock. Fields that are loaded on the falling edge are indicated by "@falling" in these tables; to emphasize the contrast, fields in double-cycle preload instructions loaded on the rising edge are indicated by "@rising." Note that the double-cycle preload instructions are specified to match the double-cycle output instruction Table (VIII) for simple cascading of multiple IAUs. Also note that data preloaded at mid-cycle on a falling edge should not be fed back to the Adder/Subtracter input in that same cycle.

SHIFT CONTROL

A 7-Bit Left/Right Output Shifter accepts 40-bit data from the Adder/Subtracter or either Accumulator. The Output Shifter is controlled by either of two 4-bit Shift Control Registers, SCRA and SCRB (which reside in Status Registers A and B [Figure 14]). SCRA controls the shifting *on output* of values from Accumulator A. SCRB controls the shifting on output of values from Accumulator B. The Shift Control Registers are a part of the ADSP-1101's Block Floating-Point Control circuitry. However, the Shift Control Registers can be used independently to shift seven or fewer positions in either direction.

The twos-complement value in the SCR selected determines the number of positions the 40-bit field will be shifted on output. Left shifts are logical; right shifts are arithmetic. The value in Shift Control Register A (SCRA) determines the number of positions the data from Accumulator A is shifted on output. Table VII details the relationship when the Shift by SCRA Instruction is executed. Shift Control Register B (SCRB) works in exactly the same way with data output from Accumulator B. Note that even though –8 is representable, it only produces a 7-bit left shift. Additional Shift Instructions are available that can force single-bit shifts left or right, independent of the SCRs' contents.

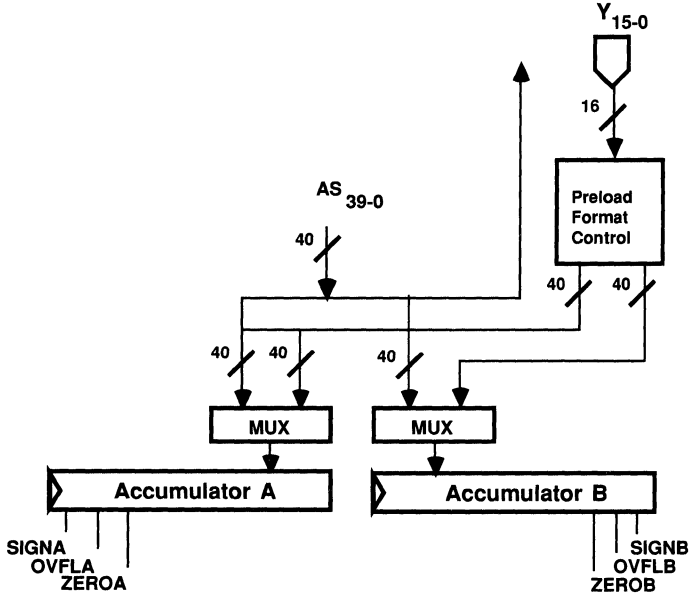
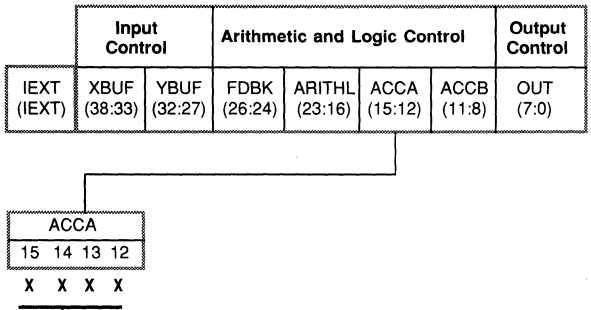
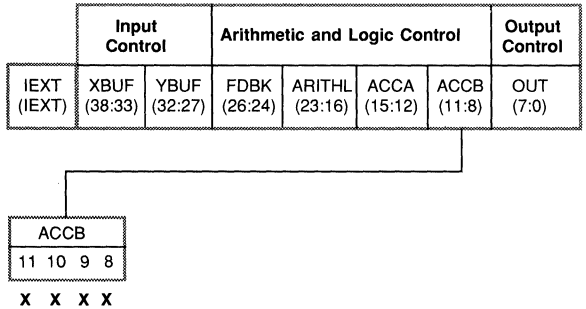


Figure 11. ADSP-1101 Accumulators



	Accumulator A 2sC flag	Accumulator A EXT load	Accumulator A MSW load	Accumulator A LSW load	mnemonic WA(2s,E,M,L)
0 0 0 0	set if AS is 2sC	AS39–32	AS31–16	AS15–0	WADASASAS
0 0 0 1	unsigned	zero	zero	Y-Port	WAUZZYP
0 0 1 0	unsigned	zero	Y-Port	no change	WAUZYPN
0 0 1 1	no change	Y-Port	no change	no change	WANYPNN
0 1 0 0	2s complement	sign extend	Y-Port	zero	WA2SYPZ
0 1 0 1	2s complement	sign extend	Y-Port	no change	WA2SYPN
0 1 1 0	unsigned	zero	Y-Port	zero	WAUZYPN
0 1 1 1	no change	no change	no change	no change	default
1 0 0 0	unsigned	zero	zero	zero	WAZERO
1 0 0 1	unsigned	zero	AS31–16	AS15–0	WAUZASAS
1 0 1 0	2s complement	sign extend	sign extend	Y-Port	WA2SSYP
1 0 1 1	no change	Y-Port @ rising	Y-Port @ falling	zero	WANYPYPZ
1 1 0 0	unsigned	zero	Y-Port @ rising	Y-Port @ falling	WAUZYPYP
1 1 0 1	2s complement	sign extend	Y-Port @ rising	Y-Port @ falling	WA2SYPYP
1 1 1 0	unsigned	zero	Y-Port @ falling	zero	WAUZYPZ
1 1 1 1	2s complement	sign extend	Y-Port @ falling	zero	WA2SYPZ

Table V. ADSP-1101 Accumulator A Write Control Instructions



	Accumulator B 2sC flag	Accumulator B EXT load	Accumulator B MSW load	Accumulator B LSW load	mnemonic WB(2s,E,M,L)
0000	set if AS is 2sC	AS39–32	AS31–16	AS15–0	WBDASASAS
0001	unsigned	zero	zero	Y-Port	WBUZZYP
0010	unsigned	zero	Y-Port	no change	WBUZYPN
0011	no change	Y-Port	no change	no change	WBNYPNN
0100	2s complement	sign extend	Y-Port	zero	WB2SYPZ
0101	2s complement	sign extend	Y-Port	no change	WB2SYPN
0110	unsigned	zero	Y-Port	zero	WBUZYPN
0111	no change	no change	no change	no change	default
1000	unsigned	zero	zero	zero	WBZERO
1001	unsigned	zero	AS31–16	AS15–0	WBUZASAS
1010	2s complement	sign extend	sign extend	Y-Port	WB2SSYP
1011	no change	Y-Port @ rising	Y-Port @ falling	zero	WBNYPYPZ
1100	unsigned	zero	Y-Port @ rising	Y-Port @ falling	WBUZYPYP
1101	2s complement	sign extend	Y-Port @ rising	Y-Port @ falling	WB2SYPYP
1110	unsigned	zero	Y-Port @ falling	zero	WBUZYPZ
1111	2s complement	sign extend	Y-Port @ falling	zero	WB2SYPZ

Table VI. ADSP-1101 Accumulator B Write Control Instructions

Independent of the SCRs, the Output Shifter can also autonormalize data from either Accumulator to a twos-complement or unsigned-magnitude MSW. (See "Autonormalization.") An Accumulator result can be autonormalized up to seven bit positions in either direction. The exponent (EXPA or EXPB) corresponding to the autonormalized MSW (from Accumulator A or B) can be simultaneously output through the Z-Port's extension field.

The user can write the two SCRs directly through the Y-Port with either the Write Status Register from Y_{7-0} Instruction or with the Write SCR from Y_{3-0}. (See Table VIII.)

SCR Value	*Accumulator Data Shift*
–8 (1000 B)	left by 7 bits (Note: not 8)
–7 (1001 B)	left by 7 bits
–6 (1010 B)	left by 6 bits
–5 (1011 B)	left by 5 bits
–4 (1100 B)	left by 4 bits
–3 (1101 B)	left by 3 bits
–2 (1110 B)	left by 2 bits
–1 (1111 B)	left by 1 bit
0 (0000 B)	no shift
1 (0001 B)	right by 1 bit
2 (0010 B)	right by 2 bits
3 (0011 B)	right by 3 bits
4 (0100 B)	right by 4 bits
5 (0101 B)	right by 5 bits
6 (0110 B)	right by 6 bits
7 (0111 B)	right by 7 bits

Table VII. Shift Control Registers' Effect on Output Shifter

5

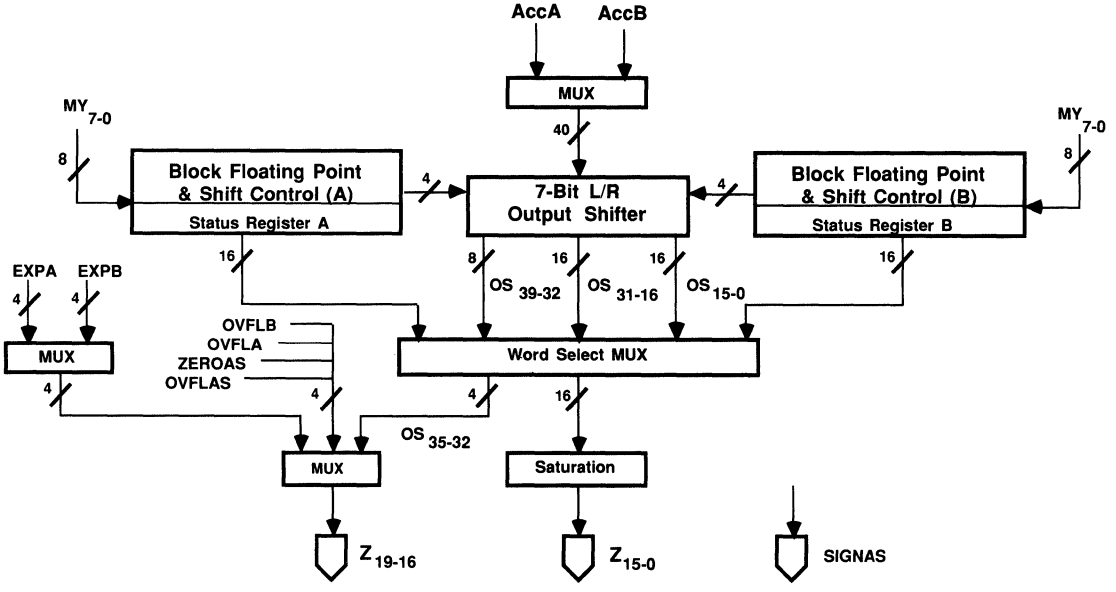


Figure 12. ADSP-1101 Shift Control, Block Floating-Point, Status Registers, Saturation, and Output Port

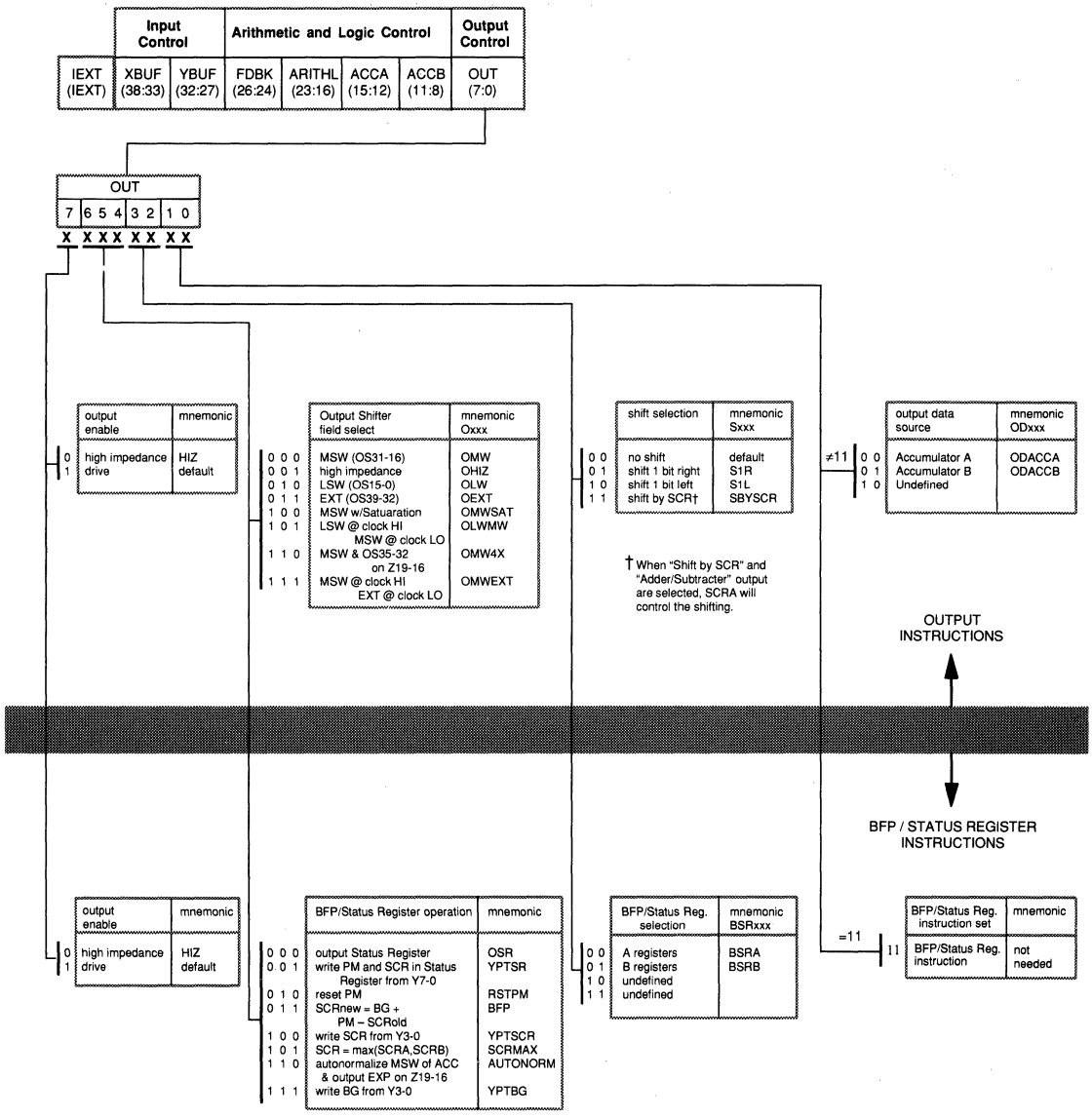


Table VIII. Output, Shift, Block Floating-Point, and Status Register Instructions

BLOCK FLOATING-POINT

Extensive iterative computation with fixed-point numbers can cause either an overflow of fixed-point data fields as results grow in magnitude or a loss of precision as results decrease in magnitude. Block Floating-Point (BFP) arithmetic is a method for scaling a block of fixed-point data by a common exponent to prevent either occurrence. It prevents overflow while preserving precision. It is most useful when algorithms can be structured to process data in multiple stages or "passes." Examples include the Fast Fourier Transform, Infinite Impulse Response Filters, and some matrix operations.

The ADSP-1101 contains all the circuitry necessary to implement Block Floating-Point Control on-chip. The Pass Magnitudes representable in a single pass range from 2^{-7} to 2^{7}, that is, the magnitude of data can change by up to seven binary orders of magnitude in a single pass and be handled properly. These Pass Magnitudes are consistently formatted as 4-bit twos-complement numbers. ("1000" binary ["–8" decimal] is used only to reset the Pass Magnitude registers.) The ADSP-1101's BFP circuitry works in conjunction with the 7-bit Left/Right Output Shifter (see Figure 12) to insure that changes in magnitude during a data pass cause compensating data shifts on output. These data shifts on output represent *changes* in the externally-referenced Block Floating-Point Exponent.

The IAU's Block Floating-Point circuitry includes a set of three 4-bit registers for Accumulator A: a Shift Control Register (SCRA), a Pass Magnitude (PMA) register, and a Bit Growth (BGA) register. Accumulator B has a parallel set of three 4-bit registers: SCRB, PMB, and BGB. Each set of BFP circuitry can operate entirely independently of the other set.

The SCRs control the Output Shifter described in the last section and can either be written by the user directly through the Y-Port or updated internally using the Block Floating-Point Instruction in the Block Floating-Point Instruction set (Table VIII). Like the SCRs, the Pass Magnitude (PM) registers reside as fields in the respective Status Registers (Figure 14). Note that any data written to the SCR or PM fields of the Status Registers must meed the setup time requirements for synchronous inputs, t_{DSS}. The PMs can be reset by the user to full-scale negative or written from the Y-Port, but are otherwise under the control of the IAU's internal BFP circuitry (Table VIII). (The only time a user is likely to want to write the PMs is when restoring the state of BFP processing that has been interrupted.) The BGs are written only from the Y-Port (Y_{3-0}) (Table VIII). As with the SCRs and PMs, data written to the BGs must meet setup time, t_{DSS}.

During a block of data's pass through the IAU, the Pass Magnitude register A tracks the magnitude of the largest number output from Accumulator A *as it is positioned in Accumulator A*. PMB similarly tracks the largest number output from Accumulator B. This tracking takes place on numbers ***before*** *they have been shifted by the amount specified in the SCRs.* To be tracked, however, an Accumulator value must be output.

"Magnitudes," as that word is used in this data sheet, are calculated relative to fully normalized Accumulator contents (Figure 13); a fully normalized number is defined here to have a magnitude of zero. For a twos-complement number this means both that the number has *not* overflowed into the EXT field (all EXT bits are the same) and that bit 30 differs from bit 31, the sign bit. "Full normalization" for an unsigned-magnitude number means both that it has not overflowed into the EXT field (all bits are zero) and that bit 31 is one. If an Accumulator value would be fully normalized if shifted left exactly one bit position, its magnitude is negative one. This is the exponent it would have were it fully normalized. If the Accumulator contents have overflowed the MSW into the EXT field by exactly one bit (i.e., a one-bit right shift would fully normalize the Accumulator value), its magnitude will be positive one, and so on.

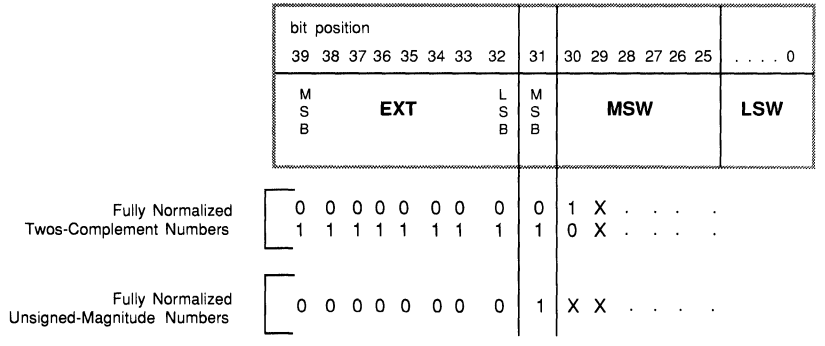


Figure 13. Fully Normalized Numbers

At the beginning of a pass, the Reset PM Instruction (Table VIII) should be issued twice to set PMA and PMB both to full-scale negative ("1000" binary). Every time an Accumulator A output instruction is executed, the magnitude of the value leaving Accumulator A is compared to the current contents of the PMA register. If this magnitude is greater than that represented in PMA, then register PMA is updated to this new magnitude. Similarly, every time a value is output from Accumulator B, its magnitude is compared to the current contents of PMB and the PMB register is updated with the larger. At the end of the pass, the PM registers will contain the magnitudes of the largest values output, here called the "Pass Magnitudes." These values can then be used to calculate the output shifting required on the *next* pass. Twos-complement and unsigned-magnitude data are both handled properly in evaluating magnitude.

Output data will be shifted by the amounts specified in the Shift Control Registers. Hence, the Pass Magnitude of the output data *in external memory* will be the *differences* between the contents of the PMs and the SCRs.

To use the ADSP-1101's BFP logic, the maximum bit growth (2^{-7} to 2^{7}) possible in a given pass must have been previously calculated. This bit growth is usually apparent from the structure of the algorithm to be implemented. The value for Accumulator A's worst-case bit growth should be written to the BGA register from Y_{3-0} at the beginning of a pass. The value for Accumulator B should also be written to BGB at the beginning of a pass. (See Table VIII.) If, for example, the magnitude of data could grow by as much as 2^3 (three bit positions), the relevant BG Registers should be preloaded with "+3." These registers will retain these values until written again. Since in many algorithms these values don't change from pass to pass, the BG Registers will often only need a single initialization.

On the first pass, the SCRs should be written with the same pair of values as the BG Registers, respectively, if the input data is fully normalized. If the Pass Magnitude of the input block is other than zero (not normalized), this Pass Magnitude should be added to the Bit Growth, i.e.,

$$\text{SCR} \leftarrow \text{BG} + \text{Pass Magnitude of the input block.}$$

This insures that the data output from the first pass will not overflow.

After the first pass, the values in the Pass Magnitude registers can be used to calculate precisely the values that should be in the SCRs during the second pass to prevent overflow while retaining maximum data precision for the block. In general, the values in the PMs at the end of pass *i* can be used to calculate the SCRs for pass *i+1*. An instruction in the Block Floating-Point and Output Instruction set will automatically do this as follows for a selected set of BFP circuitry (A or B):

$$\text{SCR}_{i+1} \leftarrow \text{BG} + (\text{PM} - \text{SCR}_i).$$

The reason this update works is that $(\text{PM} - \text{SCR}_i)$ represents the Pass Magnitude of the data from the ADSP-1101 *as formatted in memory* after pass *i*. If we add that Pass Magnitude to the worst-case bit growth, the data output on the *next* pass will be shifted precisely the amount required to both prevent overflow and maximize precision.

Thus, if the Pass Magnitude of data at the end of a given pass is less than the worst-case bit growth, the IAU will shift by a lesser amount on the next pass. Without this BFP circuitry, a user would have no option but to shift by the worst-case bit growth on every pass. In an algorithm with multiple passes, the ADSP-1101's Block Floating-Point logic can preserve several bits of precision in the final results, as demonstrated in the example below (Table IX). If the user desires to scale data output from both Accumulators by the same block exponent, the ADSP-1101 also provides an instruction (Table VIII) that sets SCRA or SCRB to the greater value currently in SCRA and SCRB.

The shifted output data on the final pass through the IAU will be fully normalized after that pass only if either a) both PMA=SCRA and PMB=SCRB or b) when all output results are scaled together, max(PMA,PMB)=SCRA=SCRB. If the final pass results aren't normalized, the data can be passed again through the IAU and its Output Shifter one last time to fully normalize the results (preload accumulator and output). For this post-processing normalization pass, the SCRs should be set as follows:

$$\text{SCR}_{\text{normalization pass}} \leftarrow \text{PM} - \text{SCR}_{\text{final processing pass}}$$

since there will be no bit growth on this scaling operation.

The SCRs (as well as the PMs) can be read from the Status Registers (Table VIII and Figure 14) at the end of each pass. The SCRs are useful for adjusting the externally referenced block floating-point exponent(s). Suppose that the block of fixed-point data prior to IAU processing has an exponent of X_0 and that the results of this processing will be output from Accumulator A. After the first pass, the data written back to memory will have been shifted by SCRA. Hence, the externally referenced block floating point(s) should be updated as follows:

$$X_1 \leftarrow X_0 + \text{SCRA}_0$$

or in general,

$$X_{i+1} \leftarrow X_i + \text{SCRA}_i.$$

If all data in memory are kept scaled to the same block exponent, for an N-pass algorithm,

$$X_N \leftarrow X_0 + \Sigma\,(\text{SCRA}_i).$$

Identical calculations apply to data output from Accumulator B and the SCRB register.

Block Floating-Point Example

An example may help clarify how the BFP logic in the IAU works (Table IX). Consider a radix-4 FFT with data initially scaled by block exponent, $N_0 = +2$. The basic radix-4 butterfly computes each output point by adding together 8 values. For fully normalized input data, the worst-case bit growth is therefore 3 bits. Consequently, both BGA and BGB are loaded with "three" ("0011" binary) prior to processing, in this illustration. We initialize SCRA and SCRB with the same pair of values. For simplicity here, assume that the data is all scaled by a single block exponent and remains so by setting

$$\text{SCRA}_{i+1} \leftarrow \max\,(\text{SCRA}_i, \text{SCRB}_i)$$

and

$$\text{SCRB}_{i+1} \leftarrow \max\,(\text{SCRA}_i, \text{SCRB}_i)$$

after each pass. The SCR=max(SCRA,SCRB) Instruction (Table VIII) executed selecting Accumulator A and then repeated selecting Accumulator B will do this.

Note that if only Accumulator B were used in a pass, these operations would only apply to Accumulator B's BFP circuitry. If Accumulator B were never used, they would only apply to the Accumulator A BFP circuitry.

Note that the core block floating-point instruction,

$$SCR_{i+1} \leftarrow BG + (PMB - SCR_i),$$

updates the Shift Control Registers at the end of each pass in Table IX. In this example, the last-pass block of data out was fully normalized, as indicated by a zero Pass Magnitude of data out. That means that the largest datum in the block is fully normalized. More generally, data will require an additional, final pass through the IAU if fully normalized block results are desired.

The external block exponent in this example grew from 2 to 13, i.e., by 11. Without the ADSP-1101's Block Floating-Point Control, a user would have had to shift by 3 bits on every pass to insure against overflow, for a total of 18 right shifts. This would have caused a loss of 7 bits of data precision, precision preserved by the IAU.

AUTONORMALIZATION

The ADSP-1101 Integer Arithmetic Unit will also autonormalize a single datum output from a selected Accumulator (see instruction in Table VIII) up to seven bit positions in either direction. Left shifts are logical; right shifts are arithmetic. The values from Accumulator A or B are normalized and the most significant 16 bits (the MSW) are output on Z_{15-0}. Its exponent relative to its Accumulator position is output on Z_{19-16} as shown in Table X. The ADSP-1101 handles both twos-complement and unsigned-magnitude numbers in autonormalization. Autonormalization does not affect the contents of the SCRs.

If the magnitude of the value to be autonormalized is either greater than +7 or less than –7, the value will be shifted to the seven position limit. Exponents will be as indicated in Table X. Note that an exponent of –8 (like –7) corresponds to a value that has been left-shifted by seven positions.

Autonormalization on output causes a longer output delay (t_{ACOD}) than any other operation and is specified separately (Figure 19). Depending on system requirements, the IAU's clock may need to be slowed down to accommodate this extended data delay for the autonormalization operation.

SATURATION

The ADSP-1101 Integer Arithmetic Unit can optionally saturate on output an overflowed, post-shifted MSW (OS_{31-16}) from either Accumulator or from the Adder/Subtracter to full scale. Saturation circuitry can prevent wrap-around errors. (See Saturation Instruction in Table VIII.) If the sign bit or any significant bits have overflowed to Output Shifter lines OS_{39-32}, the output is considered to have overflowed. Note that the Saturation logic operates on data leaving the Output Shifter.

5

Pass	*External Block Exponent Data In* [assumed in example]	*Pass Magnitude of Results In Acc (PM)* [assumed in example]	*SCR During Pass* [new SCR]	*Pass Magnitude of Data Out* [PM – SCR]	*External Block Exponent Data Out* [Extnl In + SCR]	*New SCR After Pass* [BG+(PM– old SCR)]
1	2	2	3=(BG)	–1	5 ← 2+3	2 ← 3+(2–3)
2	5	1	2	–1	7 ← 5+2	2 ← 3+(1–2)
3	7	2	2	0	9 ← 7+2	3 ← 3+(2–2)
4	9	1	3	–2	12 ← 9+3	1 ← 3+(1–3)
5	12	–2	1	–3	13 ← 12+1	0 ← 3+(–2–1)
6	13	0	0	0	13 ← 13+0	3 ← 3+(0–0)

Table IX. An Example of the IAU's BFP Control

Accumulator Contents	*Location of Most Significant Accumulator Data Bit Prior to Autonomalization*		*Bits of Shift*	*Exponent (EXP)*
	unsigned	*2sC*		
Accumulator overflowed bit position 31 by at least 7 bits	38	37	+7	7 (0111 B)
Accumulator overflowed bit position 31 by 6 bits	37	36	+6	6 (0110 B)
Accumulator overflowed bit position 31 by 5 bits	36	35	+5	5 (0101 B)
Accumulator overflowed bit position 31 by 4 bits	35	34	+4	4 (0100 B)
Accumulator overflowed bit position 31 by 3 bits	34	33	+3	3 (0011 B)
Accumulator overflowed bit position 31 by 2 bits	33	32	+2	2 (0010 B)
Accumulator overflowed bit position 31 by 1 bit	32	31	+1	1 (0001 B)
Accumulator contents are fully normalized	31	30	0	0 (0000 B)
Accumulator underflowed bit position 31 by 1 bit	30	29	–1	–1 (1111 B)
Accumulator underflowed bit position 31 by 2 bits	29	28	–2	–2 (1110 B)
Accumulator underflowed bit position 31 by 3 bits	28	27	–3	–3 (1101 B)
Accumulator underflowed bit position 31 by 4 bits	27	26	–4	–4 (1100 B)
Accumulator underflowed bit position 31 by 5 bits	26	25	–5	–5 (1011 B)
Accumulator underflowed bit position 31 by 6 bits	25	24	–6	–6 (1010 B)
Accumulator underflowed bit position 31 by 7 bits	24	23	–7	–7 (1001 B)
Accumulator underflowed bit position 31 by at least 8 bits	23	22	–7	–8 (1000 B)

Table X. Exponents (EXPA and EXPB) from Autonormalization

Thus, whether an Accumulator or Adder/ Subtracter value gets saturated will depend in part on how it is shifted. If right shifting brings the most significant 16 bits of the fully normalized value back into OS_{31-16}, the output will not be saturated; similarly, left shifts can cause a value to overflow into OS_{39-32} and thereby saturate. Since Saturation operates on post-shifted data, it affects no register contents.

The IAU handles both twos-complement and unsigned-magnitude shifted outputs. The Saturation values for conditions with both formats are shown in Table XI. Saturation on negative numbers is defined as full-scale negative plus one. This guarantees that saturated values re-entering the Multiplier Array never cause overflow (by themselves) in the Adder/Subtracter even when Format Adjusted prior to entering the Adder/Subtracter.

OUTPUT CONTROL AND TIMING

The 20-bit output Z-Port is fielded into a 16-bit result field and a 4-bit extension field, which can contain a) data bits 35 through 31 from the Output Shifter, b) the 4-bit exponent from an autonormalized output, or c) 4 status flags. The general-purpose output instructions (OUT1:0 ≠ "11") are shown in Table VIII.

The low-order 16 bits of the Z-Port (Z_{15-0}) can be put in a high-impedance state either by setting OUT7 to LO or by selecting "high impedance" with OUT6:4 (only when OUT1:0 ≠ "11"). The high-order 4 bits of the Z-Port (Z_{19-16}) will reflect the four status flags described in "Status Flags and Registers" except when executing OMW4X and AUTONORM, which use these four bits. When the 8-bit extension register is output on Z_{15-0}, it will be sign-extended according to its data type to a 16-bit field. Single-cycle double-output instructions have been chosen to coordinate with the single-cycle double-input instructions, allowing for efficient cascading of multiple IAUs. Note their timing requirements in Figure 19. Cycle times may need to be extended to accommodate the data delays of double-output operations. Shifting options are specified by OUT3:2. Output data source is selected by OUT1:0.

An output from an Accumulator will become available t_{ACOD} after the rising edge of the clock. This value had to have been computed during the previous cycle. The output instruction is presented to the IAU at the end of the processing cycle. The minimum clock cycle time is t_{CLK}. Note that is Adder/ Subtracter flags are needed off chip in the same cycle as the Adder/Subtracter operation that generates them, then the cycle time of the ADSP-1101 will have to be extended to accommodate that flag delay (t_{FDAS}).

STATUS FLAGS AND REGISTERS

Five status flags are updated every cycle and can be continuously asserted off chip. In addition to these five status flags, each Accumulator has associated with it a 16-bit Status Register.

The five status flags are:

Name	*Description*	*Output Pin*
OVFLAS	Adder/Subtracter result has overflowed into AS_{39-32}	Z_{17}
ZEROAS	Adder/Subtracter result AS_{39-0} is zero	Z_{18}
OVFLA	Accumulator A has overflowed into its EXT byte	Z_{19}
OVFLB	Accumulator B has overflowed into its EXT byte	Z_{16}
SIGNAS	Sign (AS39) of Adder/Subtracter result	SIGNAS

SIGNAS is always available at a dedicated status pin of the same name. The other four flags are generally available on Z_{19-16}. However, for two instructions the Z_{19-16} pins serve other functions: the MSW&OS35-32 Output Instruction will output the Output Shifter fields OS_{35-32} on Z_{19-16}, and the Autonormalization Instruction will output the autonormalized value's exponent on Z_{19-16}. (See instructions in Table VIII.)

OVFLAS, ZEROAS, and SIGNAS are transparent in the second half of the clock cycle (clock LO) and are latched internally in clock HI of the subsequent cycle. They become valid externally t_{FDAS} into the cycle. They therefore can be used externally for control of the operation of the next cycle. (Note that Adder/Subtracter flags are specified only over the commercial temperature range, 0-70°C. They should not be used in systems with extended temperature range requirements.) Note, however, that t_{FDAS} is greater than the minimum specified cycle times. If you use the Adder/ Subtracter flags for external control of the next cycle, you must extend the clock period to accommodate this flag delay. OVFLA and OVFLB become valid early (t_{FDAC}) in the cycle in which their respective Accumulators are written, i.e., the cycle after processing.

For unsigned-magnitude Adder/Subtracter results, SIGNAS is always cleared (LO). For twos-complement Adder/ Subtracter results, SIGNAS is set (HI) if the sign bit (AS_{39}) is one. For unsigned-magnitude Accumulator results, OVFLA and OVFLB will be cleared (LO) if the respective extension byte (bits 39-32) is all zeros. For twos-complement Accumulator results,

Overflow Condition	*OS_{31-16} after Saturation Instruction*
positive twos-complement overflow	0111111111111111 B
negative twos-complement overflow	1000000000000001 B
positive unsigned-magnitude overflow	1111111111111111 B

Table XI. ADSP-1101 Saturation Values

OVFLA and OVFLB will be set (HI) if any respective Accumulator bits 38-31 (EXT byte) differ from the sign bit, bit 39. OVFLAS is defined analogously for Adder/ Subtracter results. ZEROAS is set (HI) only when all AS_{39-0} are zero.

The two Status Registers are fielded as shown in Figure 14. All 16 bits of a given Status Register can be read out Z_{15-0} using the Output Status Register Instruction (Table VIII). Only the low-order 8 bits are writable. The Write Status Register Instruction (Table VIII) writes Y_{7-0} to the PM and SCR fields, allowing restoration of the IAU if interrupted. The Write Shift Control Register Instruction (Table VIII) transfers data from the Y-Port (Y_{3-0}) to SCRA or SCRB without affecting the PM registers. Note that any data written to the Status Registers must meed the setup time requirements for synchronous inputs, t_{DSS}. Reset Pass Magnitude Register Instructions (Table VIII) force PMA or PMB to full-scale negative.

Bits 15 in the respective Status Registers are HI if the datum most recently written to Accumulator A or B, respectively, was twos-complement. Bits 14 are identical to OVFLA and OVFLB described above. Bits 13 are HI if the result most recently written to Accumulator A or B was both twos-complement and negative. Bits 12 are HI if the contents of Accumulator A or B, respectively, are zero. Bits 11-8 are defined analogously for the previous results from the Adder/Subtracter. Note that bits 11-8 in Status Register A and Status Register B are identical. Also bits 15-12 will match bits 11-8 when the Adder/Subtracter results are written to the Accumulator associated with the Status Register in question.

DESIGN CONSIDERATIONS: POWER SUPPLY DECOUPLING

The ADSP-1101 is designed with high-speed drivers on all output pins. This means that large peak currents may pass through the ground and V_{DD} pins, particularly when all output port lines are simultaneously charging their load capacitance in transition, whether from LO to HI or vice versa. These peak currents can cause a large disturbance in the ground and supply lines. For printed circuit boards, the ADSP-1101's GND and V_{DD} pins must be tied directly to solid ground and V_{DD} planes, respectively, with 0.1µf ceramic and 20µf tantalum bypass capacitors as close as possible to the tie points. Lead lengths and trace lengths should be as short as possible. The ground plane should tie to driver GND in particular with a very low inductance path.

For breadboarding with wirewrap construction, V_{DD} should be bypassed to GND with 0.1µf ceramic and 20µf tantalum capacitors. Both sets of capacitors should then be common at a point with a low impedance path to the power supply. Lead lengths should be as short as possible. This will reduce coupling of output driver current spikes into the logic supply.

OUTPUT DISABLE AND ENABLE INFORMATION

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:

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

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.

5

bit	15	14	13	12	11	10	9	8	7-4	3-0
contents	Acc A/B 2sC?	Acc A/B OVFL?	Acc A/B sign	Acc A/B zero?	A/S 2sC?	A/S OVFL?	A/S sign	A/S zero?	PMA/B register	SCRA/B register

Figure 14. ADSP-1101 Status Registers A and B

SPECIFICATIONS [1]

RECOMMENDED OPERATING CONDITIONS

	ADSP-1101				
	J & K Grades		S & T Grades		
Parameter	Min	Max	Min	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

		ADSP-1101				
		J & K Grades		S & T Grades		
Parameter	Test Conditions	Min	Max	Min	Max	Unit
V_{IH} High-Level Input Voltage	@V_{DD} =max	2.0		2.2		V
V_{IHC} High-Level Input Voltage, CLK	@V_{DD} =max	2.4		2.6		V
V_{IL} Low-Level Input Voltage	@V_{DD} =min		0.8		0.8	V
V_{ILC} Low-Level Input Voltage, CLK	@V_{DD}=min		0.7		0.7	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		0.4		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}=0.0V		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		75		75	mA
I_{DD} Supply Current-Quiescent	All V_{IN}=2.4V		40		60	mA

SWITCHING CHARACTERISTICS[3]

	ADSP-1101								
	J Grades		K Grades		S Grades		T Grades		
	0 to 70°C		0 to 70°C		-55°C to 125°C		-55°C to 125°C		
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Unit
t_{CLK} Clock Period	90		80		105		95		ns
t_{LO} Clock LO Period	38		35		47		44		ns
t_{HI} Clock HI Period	38		35		47		44		ns
t_{IS} Instruction Setup	19		17		23		21		ns
t_{IH} Instruction Hold	3		3		3		3		ns
t_{DSS} Synchronous Data Setup	22		20		26		24		ns
t_{DH} Synchronous Data Hold	7		7		7		7		ns
t_{DSAS} Asynchronous Data Setup	90		80		105		95		ns
t_{DHAS} Asynchronous Data Hold		5		5		5		5	ns

Parameter		J Grades 0 to 70°C Min	J Grades 0 to 70°C Max	K Grades 0 to 70°C Min	K Grades 0 to 70°C Max	S Grades -55°C to 125°C Min	S Grades -55°C to 125°C Max	T Grades -55°C to 125°C Min	T Grades -55°C to 125°C Max	Unit
t_{ACOD}	ACC&SR&EXP Output Delay	58		53		69		64		ns
t_{ACOD}	ACC&EXP Output Delay Autonormalize Instruction	65		60		73		68		ns
t_{ACH}	ACC&SR&EXP Output Hold	10		10		10		10		ns
t_{ENA}	Three-State Enable Delay	5	45	5	40	5	49	5	44	ns
t_{DIS}	Three-State Disable Delay	35		30		37		32		ns
t_{FDAC}	Accumulator Flag Delay	45		40		47		42		ns
t_{FHAC}	Accumulator Flag Hold	10		10		10		10		ns

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-1101 can be found in Analog Devices' Military Data Book.
[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 Figures 18 and 19, and 2) t_{DS} and t_{DH} which are measured from clock V_{IHA} to data input V_{IH} or V_{IL} crossing points.

Specifications subject to change without notice.

5

ABSOLUTE MAXIMUM RATINGS

Supply Voltage..... –0.3V to 7V
Input Voltage..... –0.3V to V_{DD}
Output Voltage Swing... –0.3V to V_{DD}
Operating Temperatute Range (ambient)..... –55°C to +125°C
Storage Temperature Range..... –65°C to +150°C
Lead Temperature (10 seconds)..... 300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1101JG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-1101KG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-1101SG	–55 to +125°C	100-Pin Grid Array	G-100A
ADSP-1101TG	–55 to +125°C	100-Pin Grid Array	G-100A
ADSP-1101SG/883B	–55 to +125°C	100-Pin Grid Array	G-100A
ADSP-1101TG/883B	–55 to +125°C	100-Pin Grid Array	G-100A

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

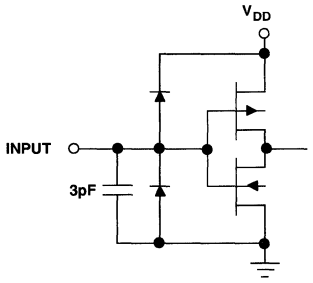


Figure 15. Equivalent Input Circuits

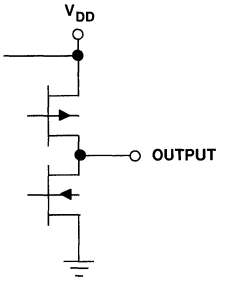


Figure 16. Equivalent Output Circuits

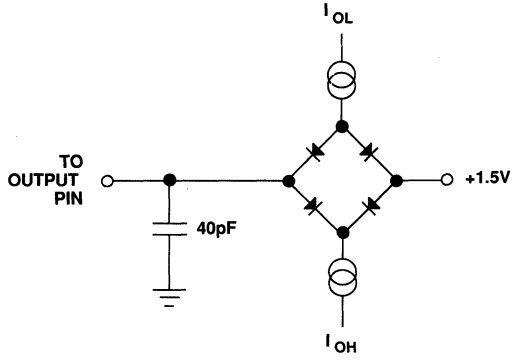


Figure 17. Normal Load for ac Measurements

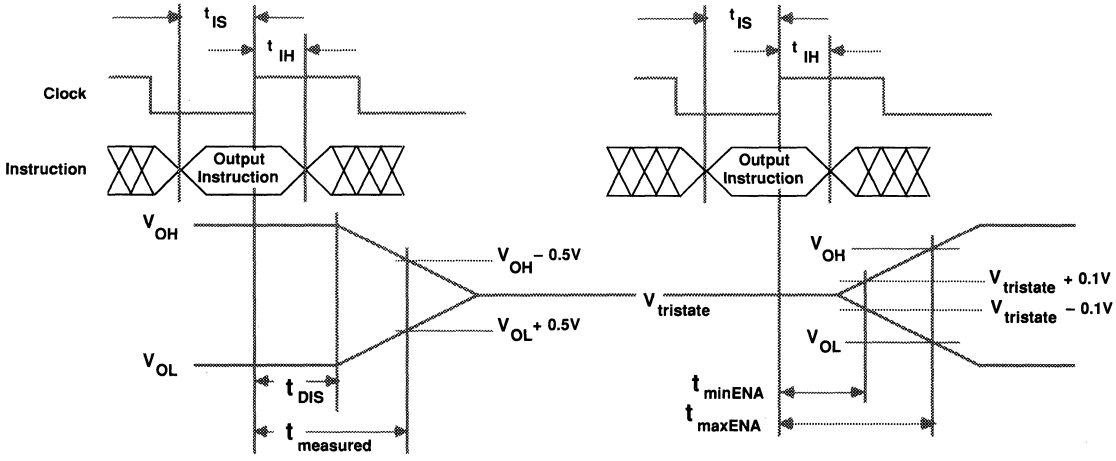


Figure 18. Output Disable and Enable Time Measurement

ESD SENSITIVITY

The ADSP-1101 features proprietary input protection circuitry to dissipate high-energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883C, the ADSP-1101 has been classified as a Category A device.

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.

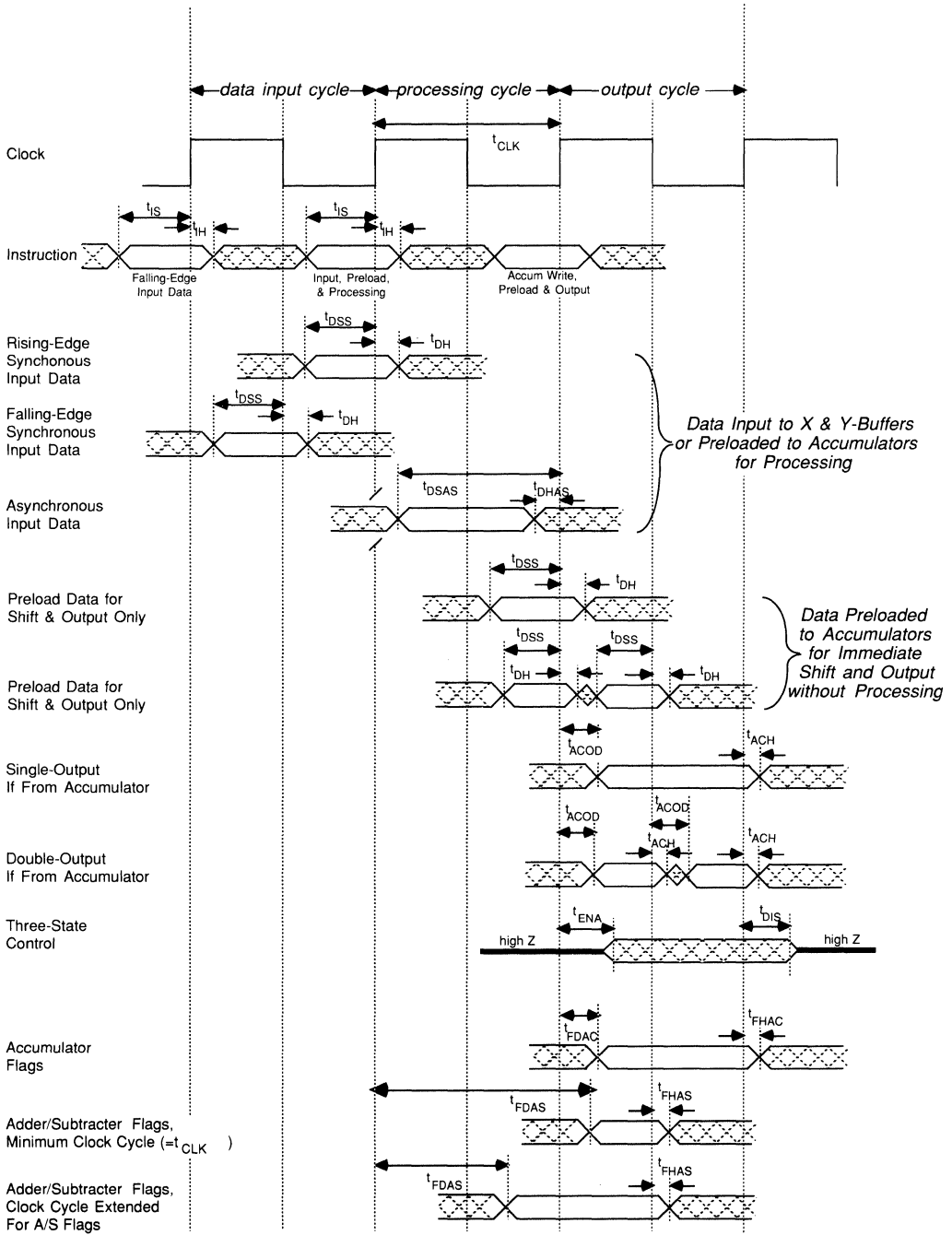
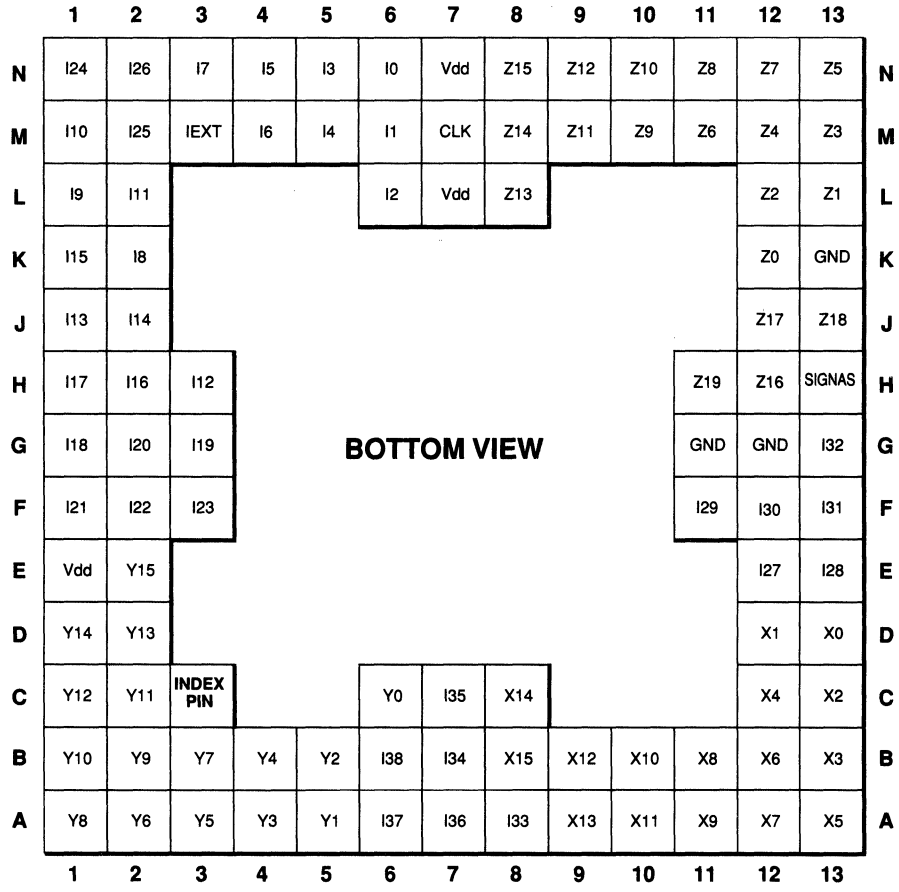


Figure 19. ADSP-1101 Timing For Synchronous Accumulator Outputs



ADSP-1101 Pin Grid Array Pinout

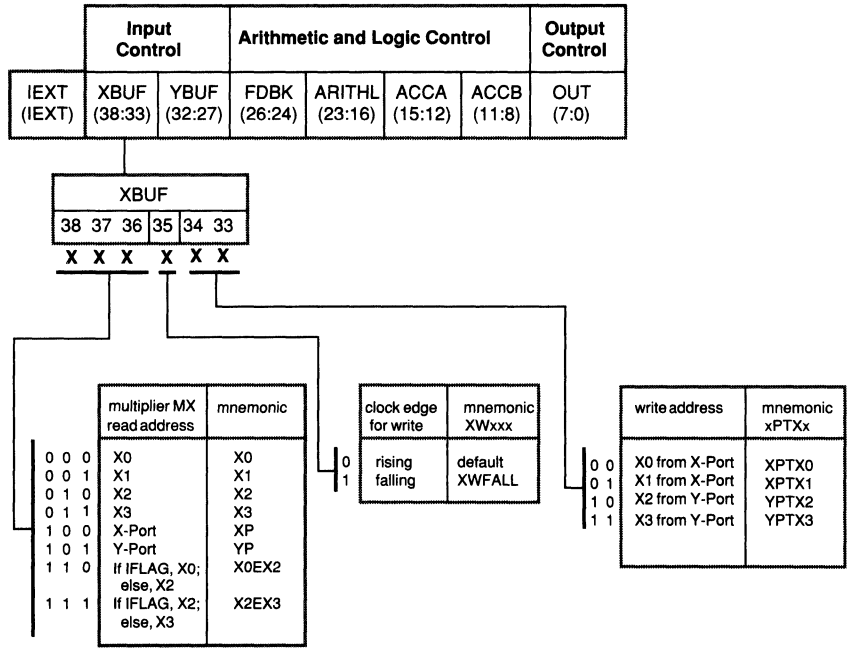


Table I. ADSP-1101 X-Buffer Instruction Set

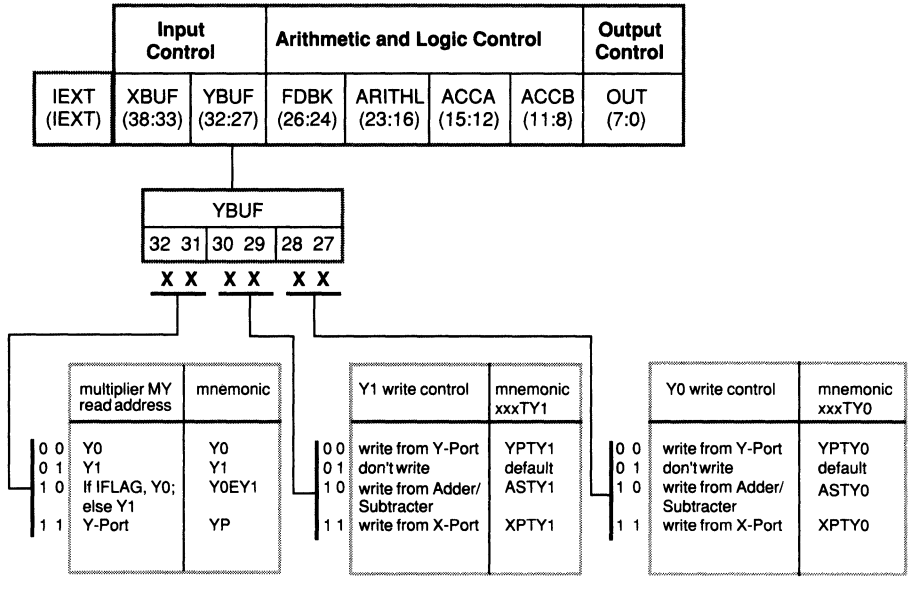
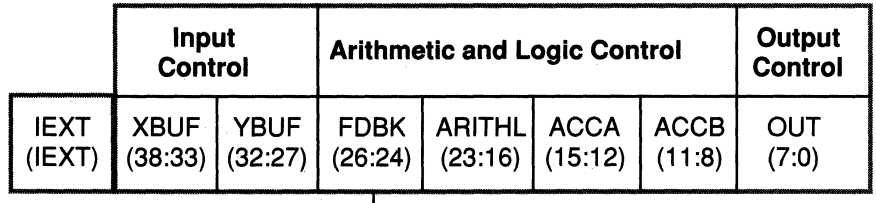
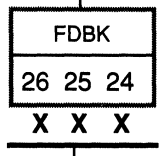


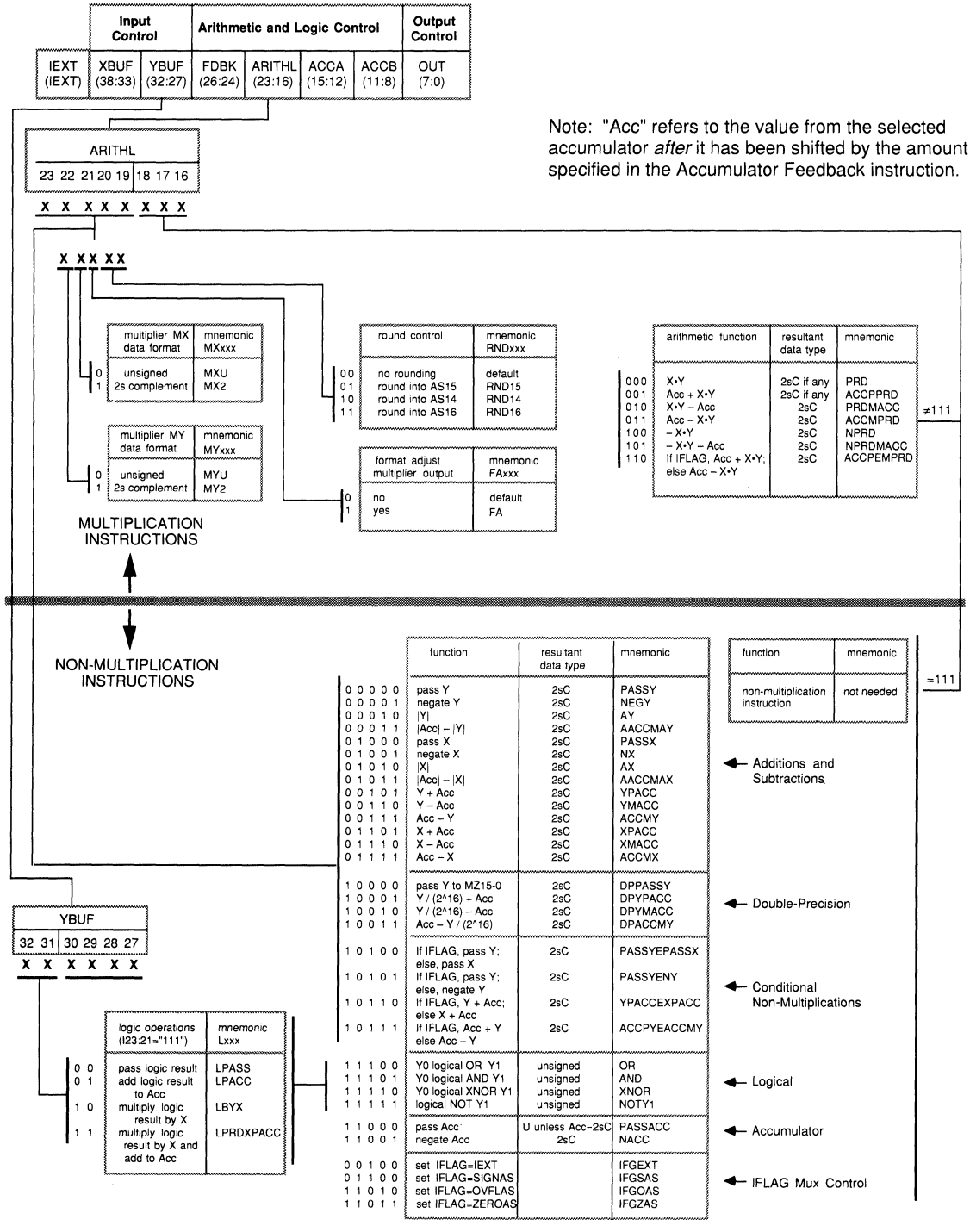
Table II. ADSP-1101 Y-Buffer Instruction Set





	accumulator selection	shift selection at AS input	mnemonic FBxxx
0 0 0	AccA	no shift	FBA
0 0 1	AccB	no shift	FBB
0 1 0	If IFLAG, AccA; else AccB	no shift	FBAEB
0 1 1	If IFLAG, AccB; else AccA	no shift	FBBEA
1 0 0	AccA	If IFLAG, shift left; else no shift	FBALE
1 0 1	AccB	If IFLAG, shift left; else no shift	FBBLE
1 1 0	AccA	If IFLAG, shift right; else no shift	FBARE
1 1 1	AccB	If IFLAG, shift right; else no shift	FBBRE

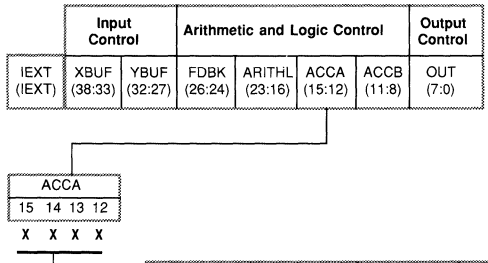
Table III. ADSP-1101 Accumulator Feedback Instruction Set



Note: "Acc" refers to the value from the selected accumulator *after* it has been shifted by the amount specified in the Accumulator Feedback instruction.

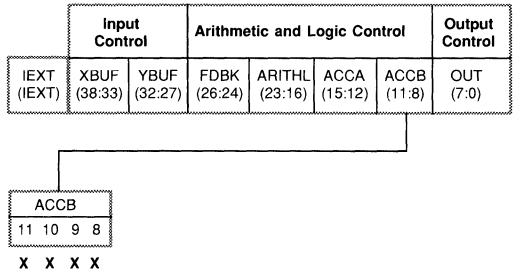
5

Table IV. ADSP-1101 Arithmetic and Logic Control



	Accumulator A 2sC flag	Accumulator A EXT load	Accumulator A MSW load	Accumulator A LSW load	mnemonic WA(2s,E,M,L)
0 0 0 0	set if AS is 2sC	AS39–32	AS31–16	AS15–0	WADASASAS
0 0 0 1	unsigned	zero	zero	Y-Port	WAUZZYP
0 0 1 0	unsigned	zero	Y-Port	no change	WAUZYPN
0 0 1 1	no change	Y-Port	no change	no change	WANYPNN
0 1 0 0	2s complement	sign extend	Y-Port	zero	WA2SYPZ
0 1 0 1	2s complement	sign extend	Y-Port	no change	WA2SYPN
0 1 1 0	unsigned	zero	Y-Port	zero	WAUZYPN
0 1 1 1	no change	no change	no change	no change	default
1 0 0 0	unsigned	zero	zero	zero	WAZERO
1 0 0 1	unsigned	zero	AS31–16	AS15–0	WAUZASAS
1 0 1 0	2s complement	sign extend	sign extend	Y-Port	WA2SSYP
1 0 1 1	no change	Y-Port @ rising	Y-Port @ falling	zero	WANYPYPZ
1 1 0 0	unsigned	zero	Y-Port @ rising	Y-Port @ falling	WAUZYPYP
1 1 0 1	2s complement	sign extend	Y-Port @ rising	Y-Port @ falling	WA2SYPYP
1 1 1 0	unsigned	zero	Y-Port @ falling	zero	WAUZYPZ
1 1 1 1	2s complement	sign extend	Y-Port @ falling	zero	WA2SYPZ

Table V. ADSP-1101 Accumulator A Write Control Instructions



	Accumulator B 2sC flag	Accumulator B EXT load	Accumulator B MSW load	Accumulator B LSW load	mnemonic WB(2s,E,M,L)
0000	set if AS is 2sC	AS39–32	AS31–16	AS15–0	WBDASASAS
0001	unsigned	zero	zero	Y-Port	WBUZZYP
0010	unsigned	zero	Y-Port	no change	WBUZYPN
0011	no change	Y-Port	no change	no change	WBNYPNN
0100	2s complement	sign extend	Y-Port	zero	WB2SYPZ
0101	2s complement	sign extend	Y-Port	no change	WB2SYPN
0110	unsigned	zero	Y-Port	zero	WBUZYPN
0111	no change	no change	no change	no change	default
1000	unsigned	zero	zero	zero	WBZERO
1001	unsigned	zero	AS31–16	AS15–0	WBUZASAS
1010	2s complement	sign extend	sign extend	Y-Port	WB2SSYP
1011	no change	Y-Port @ rising	Y-Port @ falling	zero	WBNYPYPZ
1100	unsigned	zero	Y-Port @ rising	Y-Port @ falling	WBUZYPYP
1101	2s complement	sign extend	Y-Port @ rising	Y-Port @ falling	WB2SYPYP
1110	unsigned	zero	Y-Port @ falling	zero	WBUZYPZ
1111	2s complement	sign extend	Y-Port @ falling	zero	WB2SYPZ

Table VI. ADSP-1101 Accumulator B Write Control Instructions

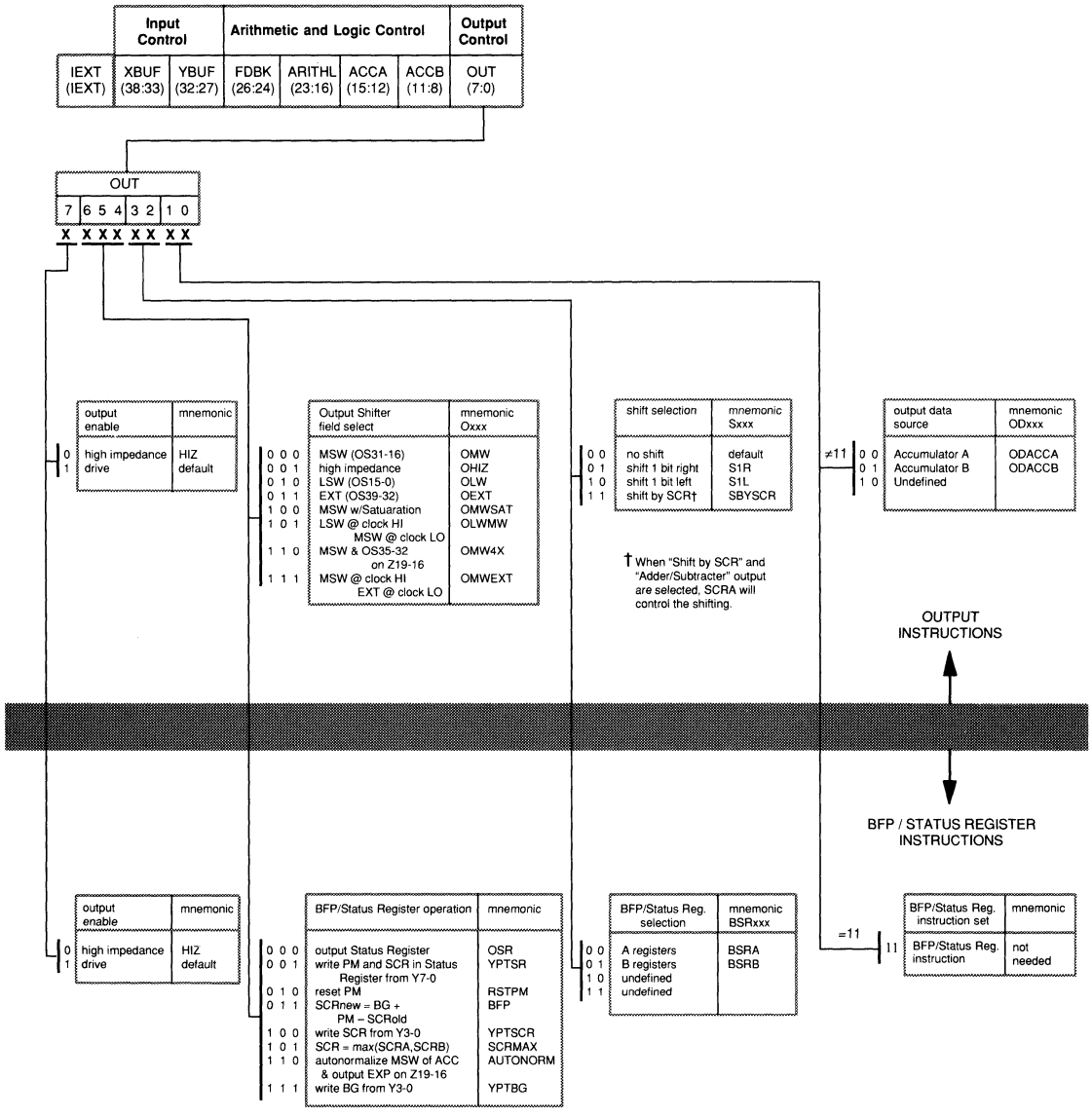
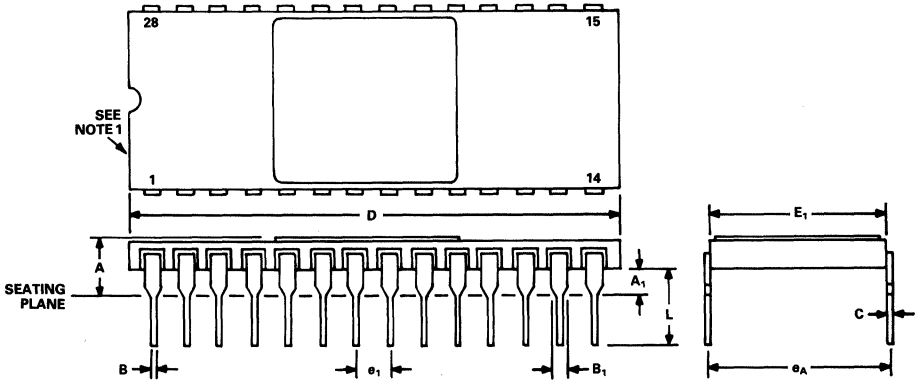


Table VIII. ADSP-1101 Output, Shift, Block Floating-Point, and Status Register Instructions

Package Information
Contents

D-28A
28-Pin Side Brazed

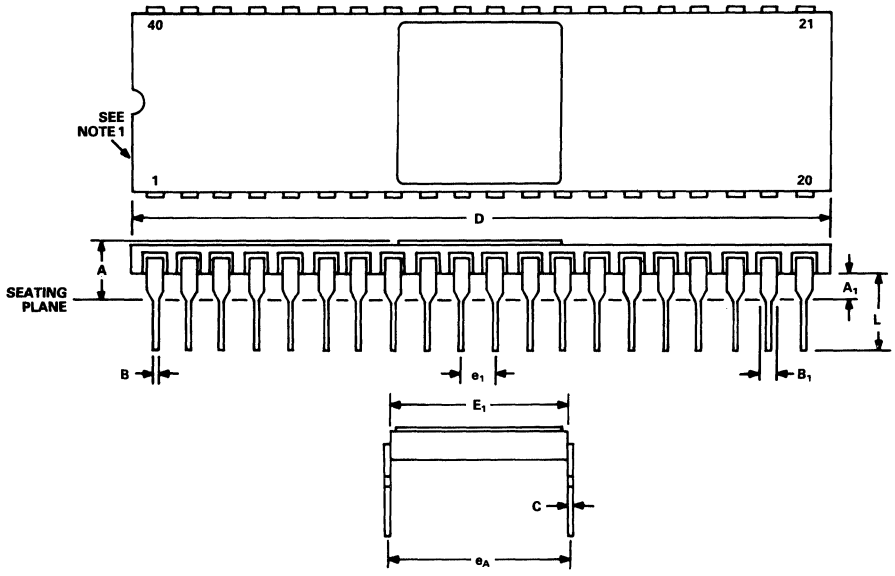


SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A		0.175		4.45	
A_1	0.040		1.02		3
B	0.015	0.020	0.38	0.51	5
B_1	0.045	0.055	1.14	1.40	2, 5
C	0.008	0.012	0.20	0.30	5
D		1.420		36.07	4
E_1	0.580	0.605	14.73	15.37	4
e_A	0.600 TYP		15.24 TYP		
e_1	0.095	0.105	2.41	2.67	6
L	0.200		5.08		

NOTES

1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension B_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
6. Twenty-six spaces.

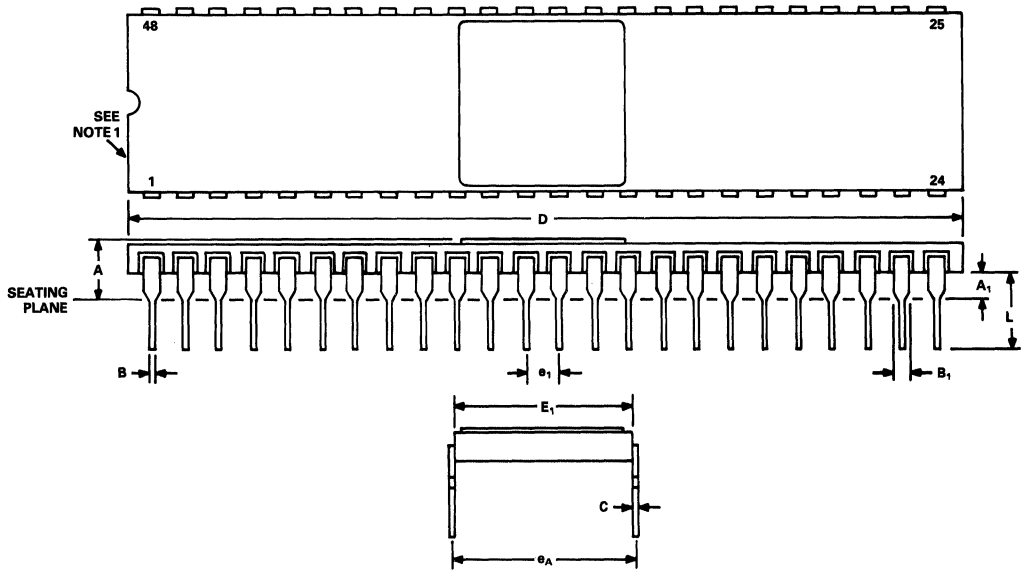
D-40A
40-Pin Side Brazed



SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A		0.169		4.29	
A_1	0.040	0.060	1.02	1.52	3
B	0.016	0.020	0.41	0.51	5
B_1	0.045	0.055	1.14	1.40	2,5
C	0.009	0.012	0.23	0.30	5
D	1.980	2.020	50.29	51.31	4
E_1	0.590	0.620	14.99	15.75	4
e_A	0.600 TYP		15.24 TYP		
e_1	0.095	0.105	2.41	2.67	6
L	0.210	0.240	5.33	6.10	

NOTES
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension B_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
6. Thirty-eight spaces.

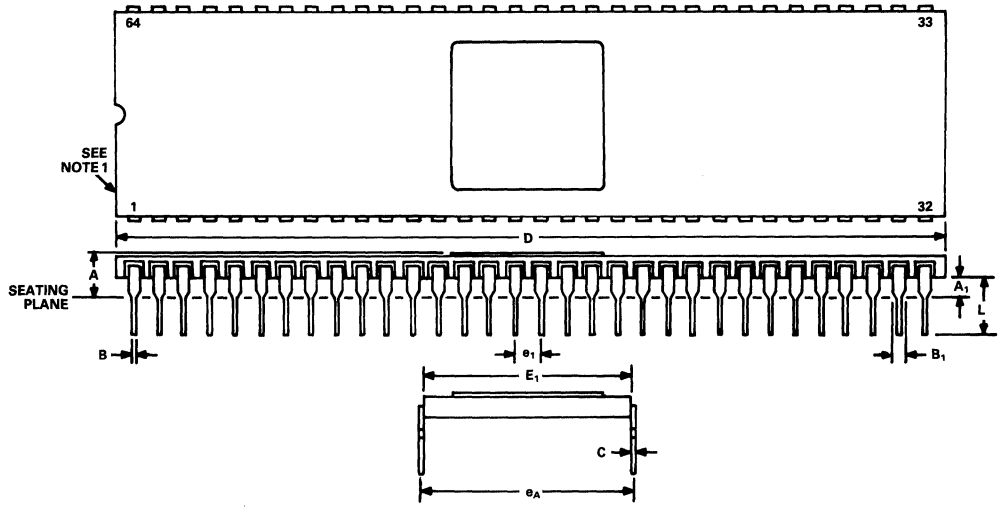
D-48A
48-Pin Side Brazed



SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A		0.169		4.29	
A_1	0.040	0.060	1.02	1.52	3
B	0.016	0.020	0.41	0.51	5
B_1	0.045	0.055	1.14	1.40	2,5
C	0.009	0.012	0.23	0.30	5
D	2.376	2.424	60.35	61.57	4
E_1	0.580	0.600	14.73	15.24	4
e_A	0.600 TYP		15.24 TYP		
e_1	0.095	0.105	2.41	2.67	6
L	0.210	0.240	5.33	6.10	

NOTES
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension B_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
6. Forty-six spaces.

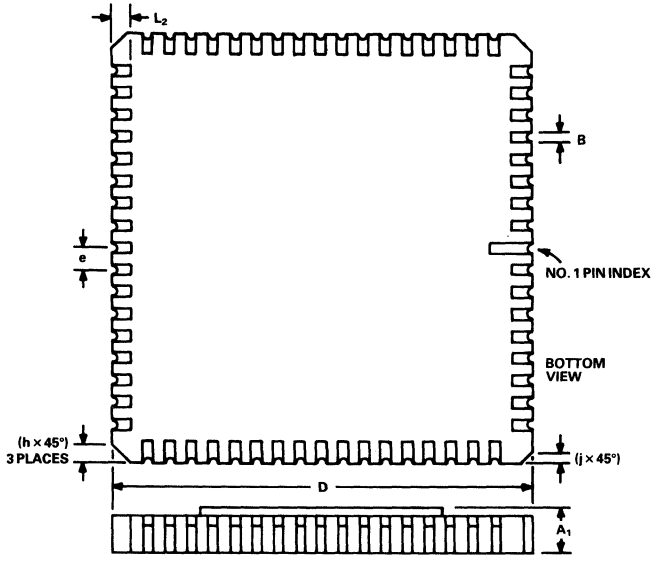
D-64A
64-Pin Side Brazed



SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A		0.169		4.29	
A_1	0.040	0.060	1.02	1.52	3
B	0.016	0.020	0.41	0.51	5
B_1	0.045	0.055	1.14	1.40	2,5
C	0.009	0.012	0.23	0.30	5
D	3.170	3.230	80.52	82.04	4
E_1	0.880	0.905	22.35	22.99	4
e_A	0.900 TYP		22.86 TYP		
e_1	0.095	0.105	2.41	2.67	6
L	0.210	0.240	5.33	6.10	

NOTES
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension B_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
6. Sixty-two spaces.

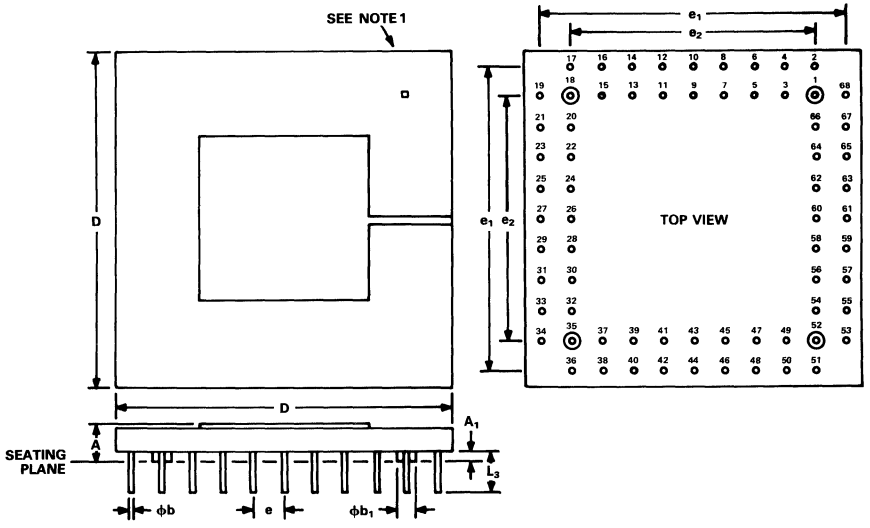
E-68A
68-Terminal Leadless Chip Carrier



SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A_1	0.065	0.103	1.65	2.62	1
B	0.020	0.030	0.51	0.76	
D	0.940	0.965	23.88	24.51	2
e	0.045	0.055	1.14	1.40	
h	0.040 TYP		1.02 TYP		
j	0.020 TYP		0.51 TYP		
L_2	0.045	0.055	1.14	1.40	

NOTES
1. Dimension controls the overall package thickness.
2. Applies to all 4 sides.
3. All terminals are gold plated.

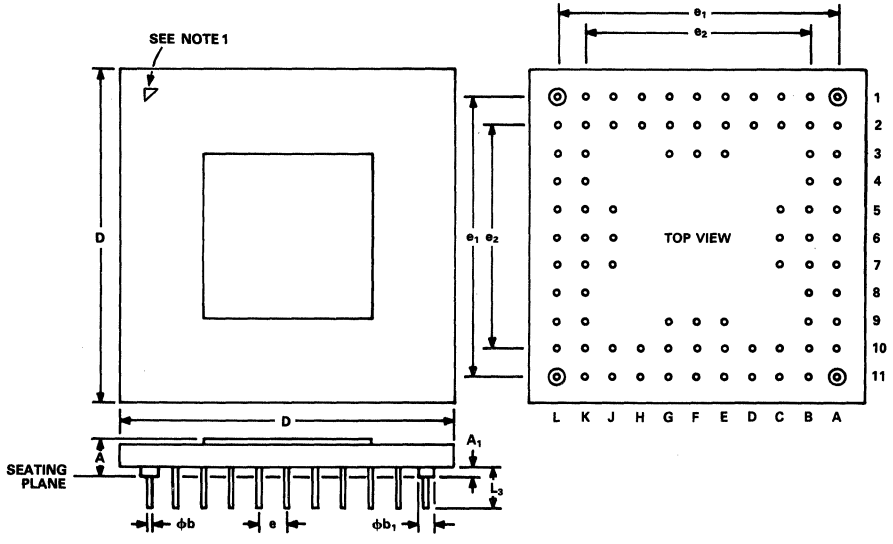
G-68A
68-Pin Grid Array



SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A	0.123	0.164	3.12	4.17	3
A_1	0.035	0.055	0.89	1.40	3
ϕb	0.016	0.021	0.41	0.53	8
ϕb_1	0.045	0.060	1.14	1.52	2, 8
D	1.080	1.110	27.43	28.19	4, 9
e_1	0.988	1.012	25.10	25.70	7
e_2	0.788	0.812	20.02	20.62	7
e	0.095	0.105	2.41	2.67	5
L_3	0.145	0.190	3.68	4.83	

NOTES
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension ϕb_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. The basic pin spacing is 0.100" (2.54mm) between centerlines.
6. Applies to all four corners.
7. Lead center when α is 0°; e_1 shall be measured at the centerline of the leads.
8. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
9. All four sides.

G-84A
84-Pin Grid Array

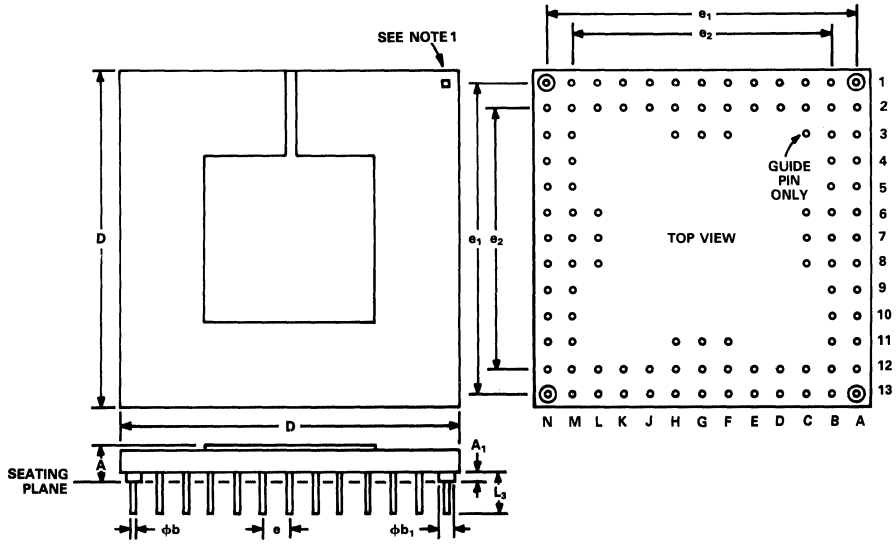


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A	0.118	0.169	3.00	4.29	3
A_1	0.025	0.055	0.64	1.40	3
ϕb	0.018	0.022	0.46	0.56	8
ϕb_1	0.045	0.065	1.14	1.65	2, 8
D	1.080	1.120	27.43	28.45	4, 9
e_1	0.988	1.012	25.10	25.70	7
e_2	0.788	0.812	20.02	20.62	7
e	0.095	0.105	2.41	2.67	5
L_3	0.175	0.185	4.45	4.70	

NOTES

1. **Index area; a notch or a lead one identification mark is located adjacent to lead L1.**
2. **The minimum limit for dimension ϕb_1 may be 0.023" (0.58mm) for all four corner leads only.**
3. **Dimension shall be measured from the seating plane to the base plane.**
4. **This dimension allows for off-center lid, meniscus and glass overrun.**
5. **The basic pin spacing is 0.100" (2.54mm) between centerlines.**
6. **Applies to all four corners.**
7. **Lead center when α is 0°; e_1 shall be measured at the centerline of the leads.**
8. **All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.**
9. **All four sides.**

G-100A
100-Pin Grid Array

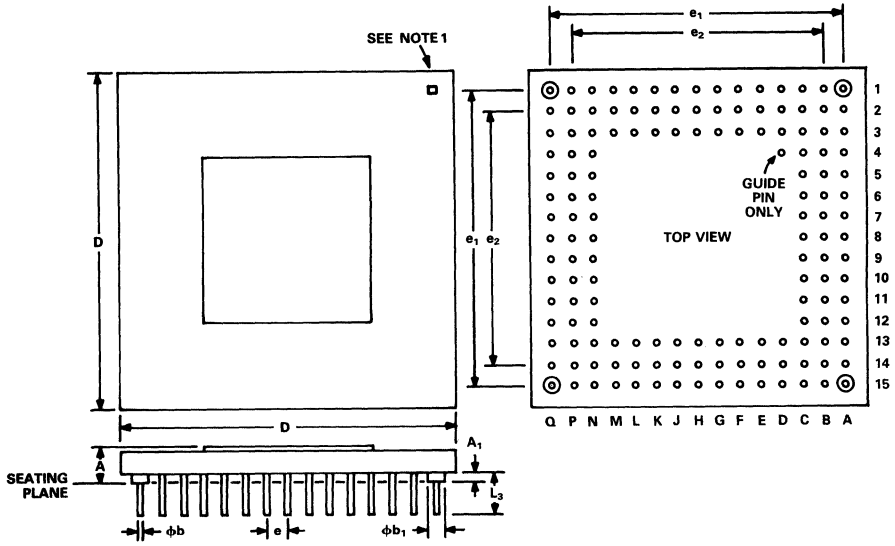


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A		0.169		4.29	3
A_1	0.025	0.055	0.64	1.40	3
ϕb	0.016	0.020	0.41	0.51	8
ϕb_1	0.040	0.055	1.02	1.40	2, 8
D	1.308	1.332	33.22	33.83	4, 9
e_1	1.188	1.212	30.18	30.78	7
e_2	0.988	1.024	25.10	26.01	7
e	0.095	0.105	2.41	2.67	5
L_3	0.165	0.190	4.19	4.83	

NOTES

1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension ϕb_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. The basic pin spacing is 0.100" (2.54mm) between centerlines.
6. Applies to all four corners.
7. Lead center when α is 0°; e_1 shall be measured at the centerline of the leads.
8. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
9. All four sides.
10. Gold plating 50μ inches over 100μ inches ref. Thickness of nickel.

G-144A
144-Pin Grid Array

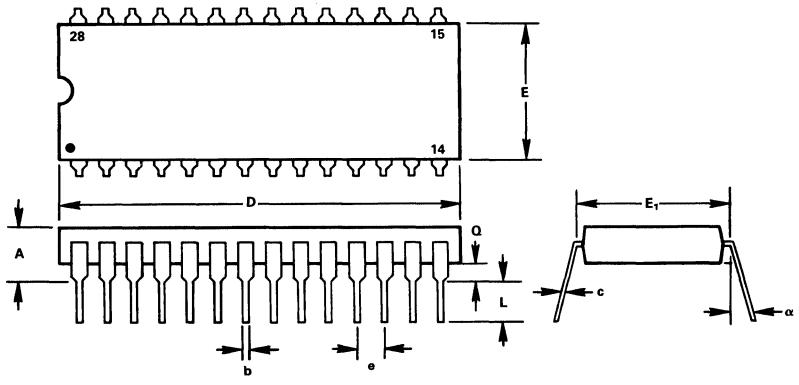


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A	0.132	0.165	3.35	4.19	3
A_1	0.045	0.055	1.14	1.40	3
ϕb	0.016	0.020	0.41	0.51	8
ϕb_1	0.045	0.055	1.14	1.40	2, 8
D	1.559	1.591	39.60	40.41	4, 9
e_1	1.388	1.412	35.26	35.86	7
e_2	1.188	1.212	30.18	30.78	7
e	0.095	0.105	2.41	2.67	5
L_3	0.175	0.185	4.45	4.70	

NOTES

1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension ϕb_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, meniscus and glass overrun.
5. The basic pin spacing is 0.100" (2.54mm) between centerlines.
6. Applies to all four corners.
7. Lead center when α is 0°; e_1 shall be measured at the centerline of the leads.
8. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
9. All four sides.
10. Gold plating 50µ inches over 100µ inches ref. Thickness of nickel.

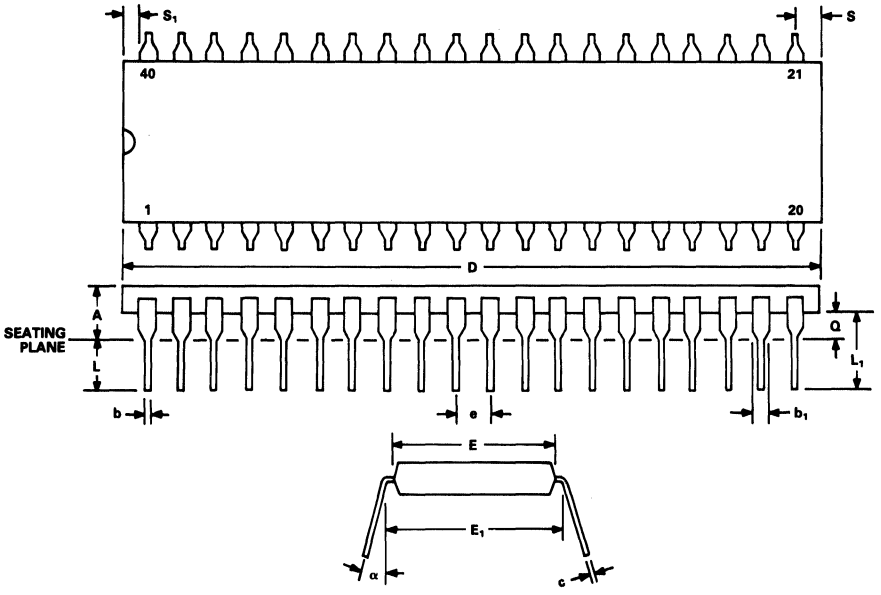
N-28A
28-Pin Plastic DIP



SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A		0.200		5.080	
b	0.015	0.020	0.381	0.508	3
c	0.008	0.012	0.203	0.305	3
D	1.440	1.450	35.580	36.830	
E	0.530	0.550	13.470	13.970	
E_1	0.594	0.606	15.090	15.400	2
e	0.096	0.105	2.420	2.670	4
L	0.120	0.175	3.050	4.450	
Q	0.020	0.060	0.560	1.580	
α	0°	15°	0°	15°	

NOTES
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. Lead center when α is 0°. E_1 shall be measured at the centerline of the leads.
3. All leads – increase maximum limit by 0.003" (0.08mm) measured at the center of the flat, when hot solder dip lead finish is applied.
4. Twenty-six spaces.

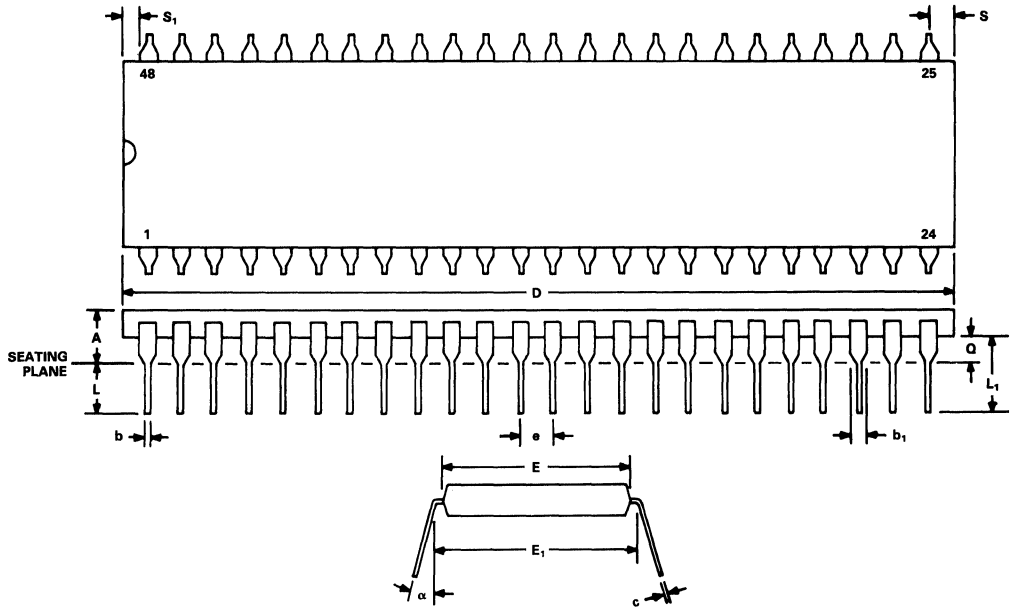
N-40A
40-Pin Plastic DIP



NOTE:
LEADS ARE SOLDER-PLATED KOVAR OR ALLOY 42

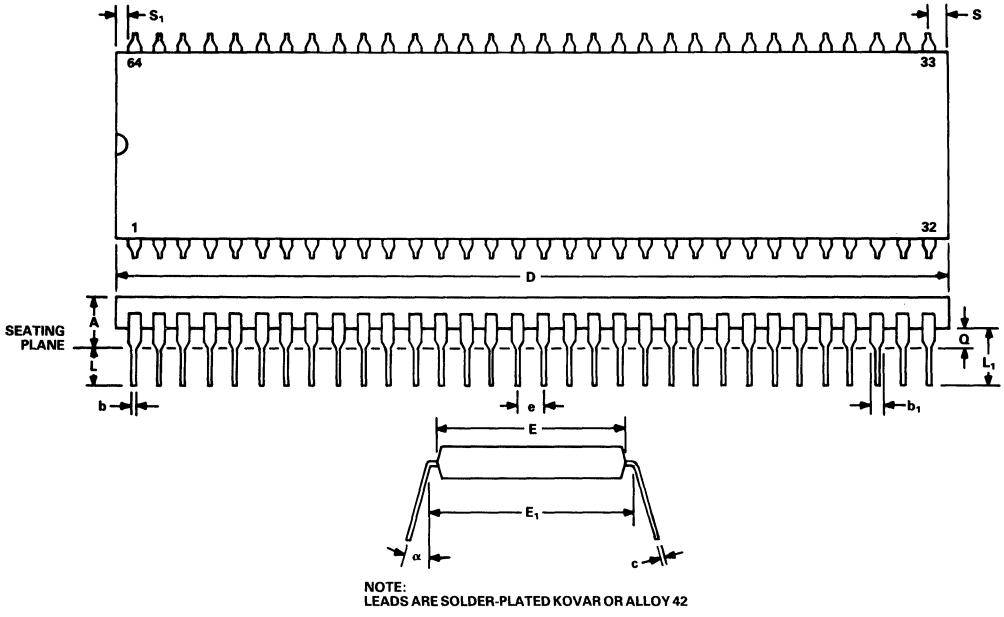
SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	–	0.200	–	5.08
b	0.015	0.025	0.38	0.64
b_1	0.040	0.060	1.02	1.52
c	0.008	0.015	0.20	0.38
D	–	2.08	–	52.83
E	0.550	0.550	13.46	13.97
E_1	0.580	0.620	14.73	15.75
e	0.100 BSC		2.54 BSC	
L	0.120	0.175	3.05	4.45
L_1	0.140	–	3.56	–
Q	0.015	0.060	0.38	1.52
S	–	0.110	–	2.79
S_1	0.005	–	0.13	–
α	0°	15°	0°	15°

N-48A
48-Pin Plastic DIP



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	–	0.200	–	5.08
b	0.015	0.025	0.38	0.64
b_1	0.040	0.060	1.02	1.52
C	0.008	0.015	0.20	0.38
D	–	2.475	–	62.87
E	0.530	0.550	13.46	13.97
E_1	0.580	0.620	14.73	15.75
e	0.100 BSC		2.54 BSC	
L	0.120	0.175	3.05	4.45
L_1	0.135	–	3.43	–
Q	0.015	0.060	0.38	1.52
S		0.110	–	2.79
S_1	0.005	–	0.13	–
α	0°	15°	0°	15°

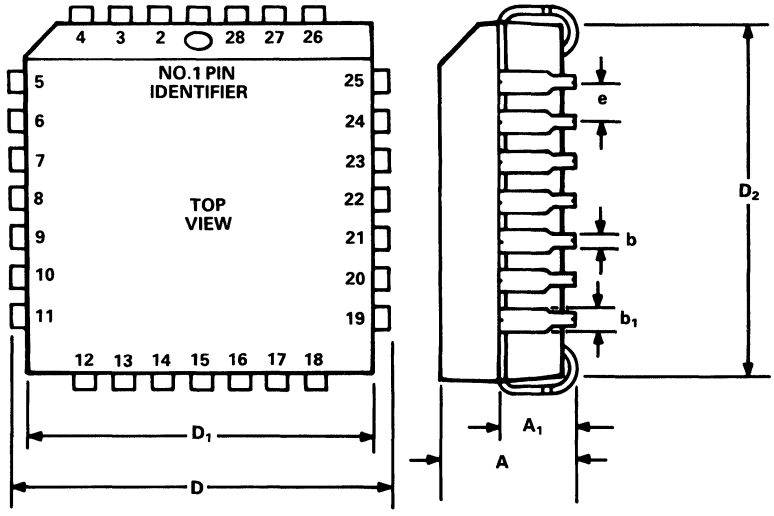
N-64A
64-Pin Plastic DIP



NOTE:
LEADS ARE SOLDER-PLATED KOVAR OR ALLOY 42

SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A		0.250		6.35
b	0.015	0.025	0.38	0.64
b_1	0.030	0.070	0.76	1.78
c	0.008	0.015	0.20	0.38
D		3.25		82.55
E	0.790	0.810	20.07	20.57
E_1	0.880	0.920	22.35	23.37
e	0.100 BSC		2.54 BSC	
L	0.125	0.200	3.18	5.08
L_1	0.150		3.81	
Q	0.015	0.060	0.38	1.52
S		0.098		2.49
S_1	0.005		0.13	–
α	0°	15°	0°	15°

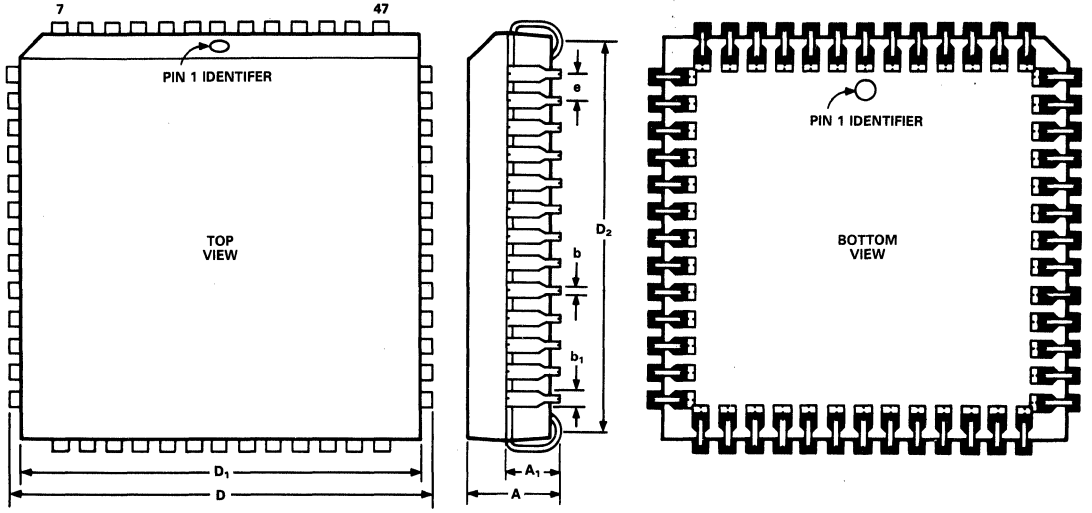
P-28
28-Lead Plastic Leaded Chip Carrier



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.165	0.180	4.19	4.57
A_1	0.090	0.120	2.29	3.05
b	0.013	0.021	0.33	0.53
b_1	0.026	0.032	0.66	0.81
D	0.485	0.495	12.32	12.57
D_1	0.450	0.456	11.43	11.58
D_2	0.390	0.430	9.91	10.92
e	0.045	0.055	1.14	1.40

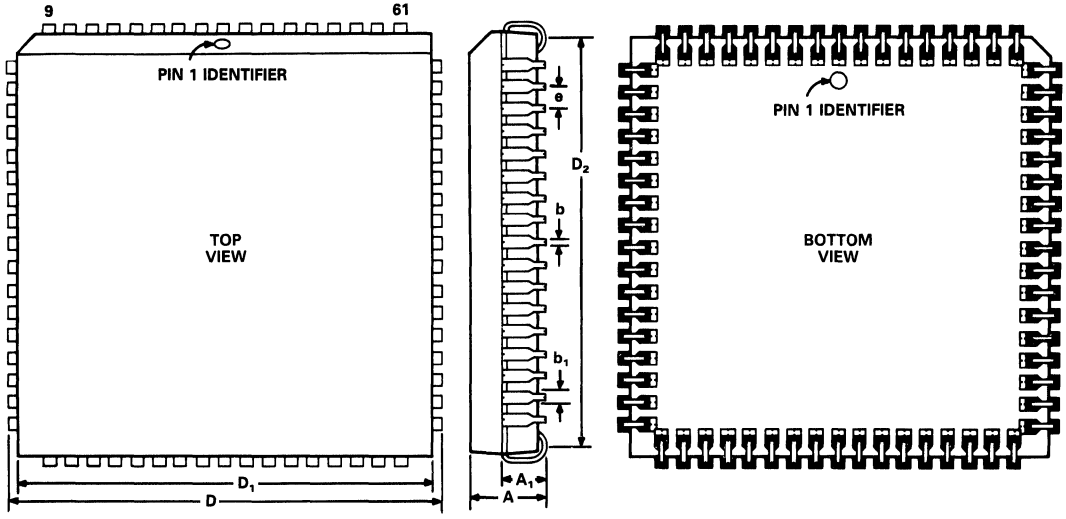
6

P-52
52-Lead Plastic Leaded Chip Carrier



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.165	0.180	4.19	4.57
A_1	0.090	0.120	2.29	3.05
b	0.013	0.021	0.33	0.53
b_1	0.026	0.032	0.66	0.81
D	0.785	0.795	19.94	20.19
D_1	0.750	0.756	19.05	19.20
D_2	0.690	0.730	17.53	18.54
e	0.045	0.055	1.14	1.40

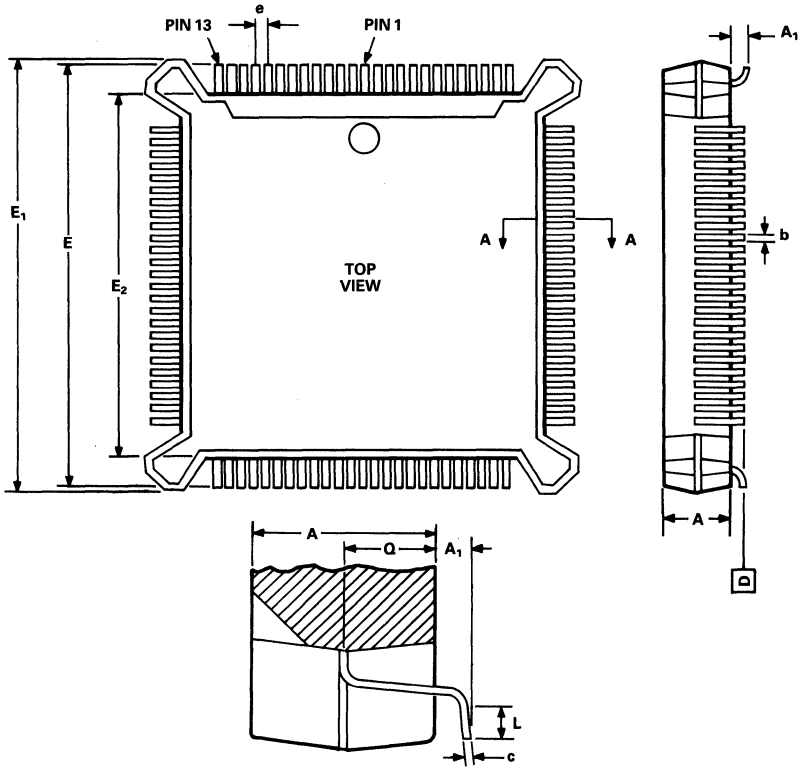
P-68
68-Lead Plastic Leaded Chip Carrier



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.169	0.175	4.29	4.45
A_1	0.104 TYP		2.64 TYP	
b	0.017	0.019	0.43	0.48
b_1	0.027	0.029	0.69	0.74
D	0.885	0.995	22.48	25.27
D_1	0.950	0.954	24.13	24.23
D_2	0.895	0.925	22.73	23.50
e	0.050 TYP		1.27 TYP	

P-100

100-Lead Plastic Quad Flat Pack



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.160	0.180	4.06	4.57
A_1	0.020	0.040	0.51	1.02
b	0.010	0.013	0.25	0.33
c	0.006	0.008	0.15	0.20
E	0.875	0.885	22.23	22.48
E_1	0.897	0.903	22.78	22.94
E_2	0.747	0.753	18.97	19.13
e	0.020	0.030	0.51	0.76
L	0.020	0.030	0.51	0.76
Q	0.065	0.075	1.65	1.91
⍺		0.008		0.20

Application Notes
Contents

Introduction

This chapter contains application notes representative of the applications support provided by the Digital Signal Processing Division. Additional application notes are always being generated by Applications Engineering. The complete list of current applications literature is printed in the divisional newsletter, *DSPatch*. Consult your Analog Devices Sales Office for more information.

The applications in this databook deal with microcode components, technical product comparisions, DSP processors and general design practices. A great deal of applications information specifically supporting the ADSP-2100 family of processors has been published in our series of Applications Handbooks. Volume One and Volume Two are now available and Volume Three is in preparation and should be available shortly after publication of this databook. Because of the code compatibility between the ADSP-2100 microprocessor and the ADSP-2101 microcomputer, much of the information in earlier volumes continues to be pertinent.

Here is a chapter-by-chapter summary of the available volumes. Together they total over 400 pages. An IBM PC diskette containing the listed programs is also available.

ADSP-2100 Applications Handbook, Volume One

1. *Fixed-Point Arithmetic*
   Describes how basic fixed-point and multiprecision arithmetic operations are implemented in the hardware of the processor.
2. *Floating-Point Arithmetic*
   Describes how to convert from fixed-point to floating-point and back and how to perform basic floating-point operations. (Block floating-point is described under FFTs.)
3. *Function Approximation*
   Shows how to approximate several functions such as sine and acrtangent and describes a random number generator algorithm.
4. *Fixed-Coefficient Digital Filters*
   Describes the implementation of several finite impulse response (FIR) and infinite impulse response (IIR) filters with fixed coefficients.
5. *Fast Fourier Transforms*
   Details several FFT algorithms (DIT, DIF, radix-2, radix-4) and the related operations of bit-reversed addressing, digit reversing, block floating-point scaling and windowing.
6. *Adaptive Filters*
   Describes filters with time-varying coefficients.
7. *Image Processing*
   Describes the processing of digitized images and related algorithms, such as computing the histogram.
8. *Linear Predictive Speech Coding*
   Presents techniques used to analyze, encode and synthesize speech signals.
9. *High Speed Modems*
   Describes several algorithms (stochastic gradient, etc.) used in implementing a high speed modem.

ADSP-2100 Applications Handbook, Volume Two

1. *Graphics*
   Presents a graphics subsystem based on the ADSP-2100, complete with all software routines and support circuitry.
2. *Multirate Digital Filters*
   Describes filters which change the sampling rate of digitally represented signals.
3. *Pulse Code Modulation*
   Presents an implementation of the CCITT standard pulse-code modulation (PCM) algorithm. Encoding and decoding are shown, and both μ-law and A-law companding methods are used.
4. *Adaptive Differential Pulsej Code Modulation*
   Presents an ADSP-2100 implementation of the CCITT standard adaptive differential pulse-code modulation (ADPCM) algorithm. A nonstandard program that is suitable for some applications is also described.
5. *Dual Tone Multifrequency*
   Describes the generation and detection of the CCITT standard dual-tone multifrequency (DTMF) signals.

Sharing the Output Bus of the ADSP-1401 Microprogram Sequencer

by Bob Fine

INTRODUCTION

In some applications, such as fast context switching or multitasking, multiple ADSP-1401 Program Sequencers may be used or microcode memory addresses may come from a source other than a program sequencer. Due to the three-state output feature of the ADSP-1401 Program Sequencer, other devices can be added to the microprogram memory address bus without the need for special buffering of the program sequencer output. This becomes very important when progagation delay of the microprogram address is critical and microprogram memory access time must be conserved.

BUS SHARING IMPLEMENTATION

The basic architecture for multi-source microprogram addressing is shown in Figure 1.

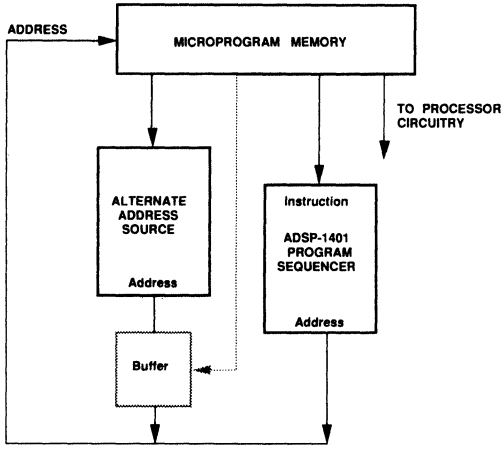


Figure 1. Basic Bus Sharing Architecture

In normal program execution, the program sequencer supplies addresses to the microprogram memory. In special cases, an alternate address source may take over by gaining access to the microprogram address bus. If the alternate address source does not have an output that can be disabled, a three-state buffer is required. When switching from one address source to the other, adequate disable time must be allowed for the device turning off and adequate enable time must be allowed for the device turning on. Careful timing analysis in the design stage should be done to avoid a bus contention, where both devices are trying to drive the microprogram address bus.

Use of IDLE Instruction

A method is required for systematically turning one device on and the other off. In the case of the ADSP-1401 Program Sequencer, this method involves the use of the IDLE instruction together with the Instructional Hold Control mode (IHC).

The output of the ADSP-1401 Program Sequencer may be disabled by using the IDLE instruction. The IDLE instruction places the address port into the high-impedance state and inhibits the program counter (PC) from incrementing. The ADSP-1401 behaves as if the clock had stopped. The IDLE instruction may be latched internally by using the Instruction Hold Control mode (IHC), freeing microcode for use by another device. Note that while idle, external interrupts will continue to be latched and should therefore be masked or disabled. (See ADSP-1401 Data Sheet)

IHC Mode

Before the alternate address source can be enabled, the program sequencer should be put into the instruction hold control mode with the IHC instruction which is clocked into the sequencer at the rising edge of the clock. Executing the IHC instruction sets status register bits 5 and 4 to a '10' and redefines the function of interrupt input IR_1 allowing subsequent instructions to be held for repeated execution, regardless of the instruction port. Use of the IHC mode requires that the mask bit for IR_1 be set, otherwise the assertion of the IHC signal will result in a faulty interrupt. (Since IR_5 shares the same input pin as IR_1, care should be taken when using all eight external interrupts with the IHC mode.)

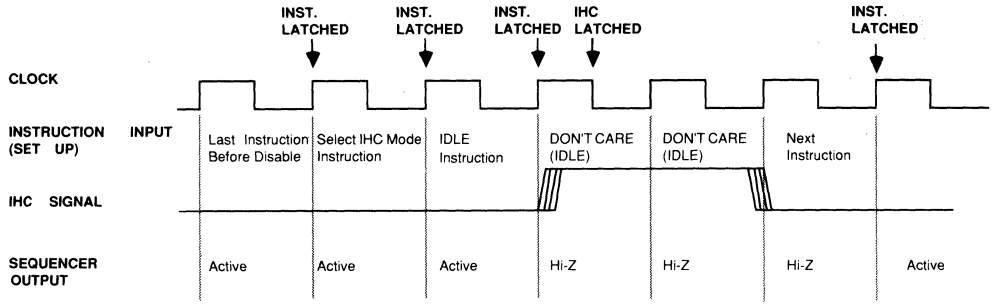


Figure 2. IHC Mode - IDLE Instruction Timing

Program Execution

Once in the IHC mode (having executed the IHC instruction), the IDLE instruction is presented to the sequencer at the instruction port, prior to the rising edge of the clock, and IR_1 asserted HI (prior to the falling edge of the clock). The IDLE instruction will be held with all interrupts disabled (although they will continue to be latched) until IR_1 is brought LO again (prior to the falling edge of the clock).
It is recommended that IR_1 be dedicated to control of the IHC mode. However, if it must also be used for subsequent interrupting, then the CAIR instruction should be executed before unmasking IR_1 (to clear the interrupt request resulting from use of IR_1 as the IHC control).

Figure 2 shows a timing diagram illustrating the series of events described above. The ADSP-1401 will continue to re-execute the IDLE instruction for each cycle until the IHC signal goes low.

When IR_1 is to be dynamically assigned to an interrupt input or IHC input, a multiplexer may be used to externally switch an interrupt line to the IHC input (See Figure 3). It should be noted that when the IHC mode is set, the multiplexer should also be selected so that the IHC signal is passed and not the interrupt. (If IHC is used, it is best to dedicate IR_1 to that function and sacrifice the interrupt.)

Interrupts cannot be used while the IHC is used. Once the IHC mode is invoked, the IDLE instruction is presented to the sequencer and is latched at the rising edge of the clock. The sequencer recognizes the IHC input (reassigned IR_1) at the falling edge of the clock. If this input is high, the instruction being executed (IDLE) is latched and the instruction input is ignored. The sequencer remains in this state until the IHC input goes low. This implementation allows both program sequencer and the alternate address source (be it another ADSP-1401 or other device) to share many of the same microcode instruction bits. If many sequencers are used, the IHC controls may come from an addressable latch driven by the microcode. This will further conserve microcode bits. The IHC may be disabled by performing a SELECT RELATIVE ADDRESS WIDTH instruction (REL16, REL12, or REL8). When the sequencer executes this instruction, the IHC is cleared. (See ADSP-1401 Data Sheet)

CONCLUSION

The addition of buffers in a high-speed bus path requires additional time overhead with the addition of propagation delay. The ability to put the output of the ADSP-1401 in a high-impedance state in conjunction with the use of the IDLE instruction and the IHC mode eliminates the need for external buffers when the address bus of the microprogram memory is to be shared. In addition, the sequencer operation is suspended avoiding the need for clock stop circuitry.

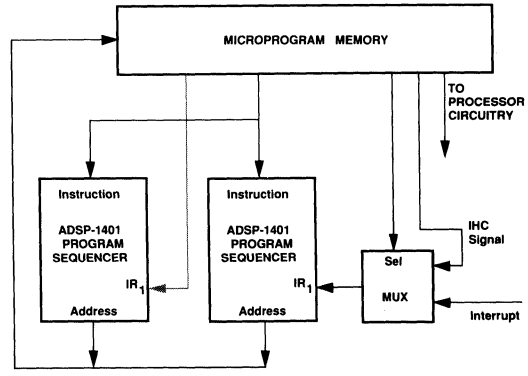


Figure 3. Block Diagram with IHC Connections

Implement a Writeable Control Store in Your Word-Slice® System

by Bob Fine

INTRODUCTION

The ADSP-1402 Program Sequencer is used to sequence through microprogram instructions residing in a microprogram memory. The microprogram instructions in turn control all the circuitry of the processor. Figure 1 shows the basic interconnection of the program sequencer to the microprogram memory.

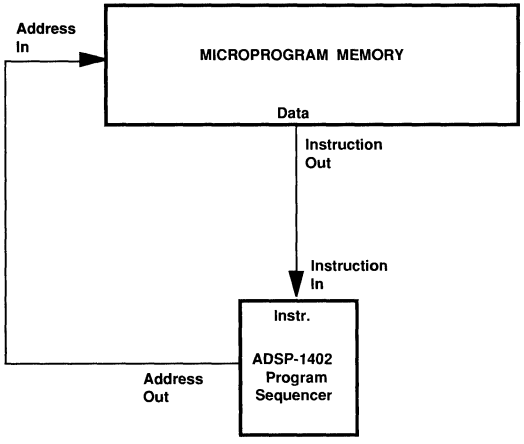


Figure 1. Program Sequencer Connection to Microprogram Memory

Typically, the microprogram is developed and "burned" or programmed into a read only memory (ROM) device so that the program is always there regardless of whether the circuit is powered or not. This method of storing microcode (fixed control store) has an advantage in that data is not lost when power is removed from the circuit and no program memory initialization is required.

The disadvantages, though, outweigh the advantages for ROM type microprogram storage. A masking process or use of a ROM programming device is needed in order to store data and once the device is programmed, it cannot be changed. This adds extra handling in the manufacturing process, increasing cost and potential for error, and also limits the flexibilty of the system.

WRITEABLE CONTROL STORE

With the availability of fast, CMOS static RAMs (15 - 35 ns access time), a writeable control store may be a better solution for microprogram memory. A writeable control store is basically a random access (RAM) read/write memory which serves the same purpose as the fixed ROM control store but adds the advantage of flexibility since new code can be written into the RAM at any time allowing dynamic configuration of the microcode system.

The writeable control store (RAM devices) is normally in a read only mode, acting as the microprogram memory. It receives its address from the program sequencer while its data output (microcode control bits) feeds the processor circuitry. Unlike the ROM based microprogram memory, the writeable control store can be downloaded with new program code at any time. This provides the advantage of flexibility but at the expense of extra circuitry needed to support the loading of RAM. A data path must be made available from a host or DMA circuit to the data input of the writeable control store. Since the host may supply data at a rate slower than the program sequencer clock rate, some handshaking is required. Also, addresses must be provided, pointing to the appropriate writeable control store location, while memory write and enable signals are properly supplied.

ARCHITECTURAL DESCRIPTION

The architecture of a writeable control store, used in conjunction with the ADSP-1402 microprogram sequencer, is very similar to that of a fixed control store except that there must be an access path from a host or DMA circuit to the microcode memory for download purposes as shown in Figure 2, which can be found on the following page.

Upon initialization of the Word-Slice system, the writeable control store must be filled with microprogram code before operation of the circuit can begin.

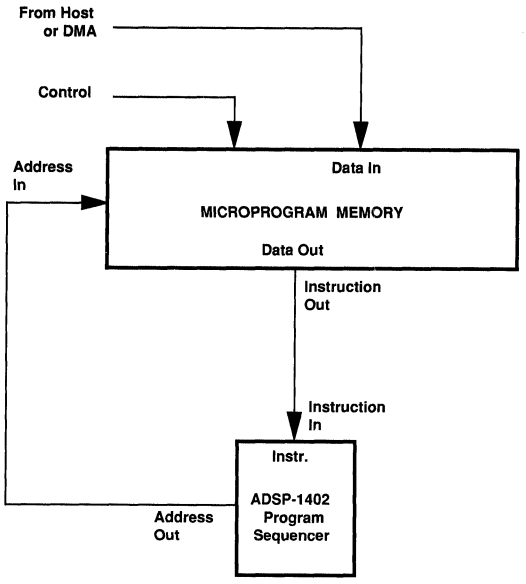


Figure 2. Host Data and Control Connections for WCS

The following description contains the details of implementing a writeable control store with the ADSP-1402 and some RAM, and details the initialization procedures for loading and starting program execution.

Two examples are described showing:

1) A system that only contains RAM or writeable control store and must be initialized upon power-up, since memory contents of the RAM are lost when power is taken away. Basically, the system at power-up is "dumb" and must be initialized by a host or DMA circuit.

2) A system that contains conventional ROM based microcode memory and in addition, has a section of writeable control store. This preferred case is often found where a fixed program, residing in ROM, is used to provide a power-up program or boot procedure.

The ADSP-1402 provides addresses to the microcode memory during program download and normal program execution. Microcode instruction control bits come from the data output of the microprogram memory. The data input of the microprogram memory is fed by a host or DMA circuit during microcode download and the control lines of the microcode memory (RD/$\overline{WR}$ and CS) are controlled by both the host and the microcode system circuits.

BOOT INPUT

The ADSP-1402 fully supports the WCS instruction of the ADSP-1401; however, it also provides a pin that allows external hardware control of a download. The BOOT input of the ADSP-1402 controls the operation for downloading microcode in much the same way as the WCS instruction of the ADSP-1401. The Boot operation, although slightly more restricted compared to the WCS operation, requires no external circuitry. A system using the BOOT input for WCS is shown in Figure 3.

In the cycle that BOOT is asserted, the ADSP-1402 outputs H#0000 on the address port and sets the PC to H#0000. When $FLAG_0$ is asserted, the PC is incremented and its new value is output on the address port. The ADSP-1402 remains in this mode until the BOOT pin is deasserted. Thus, no interrupt is required to end the download. Program execution will start after a BOOT at location H#0000.

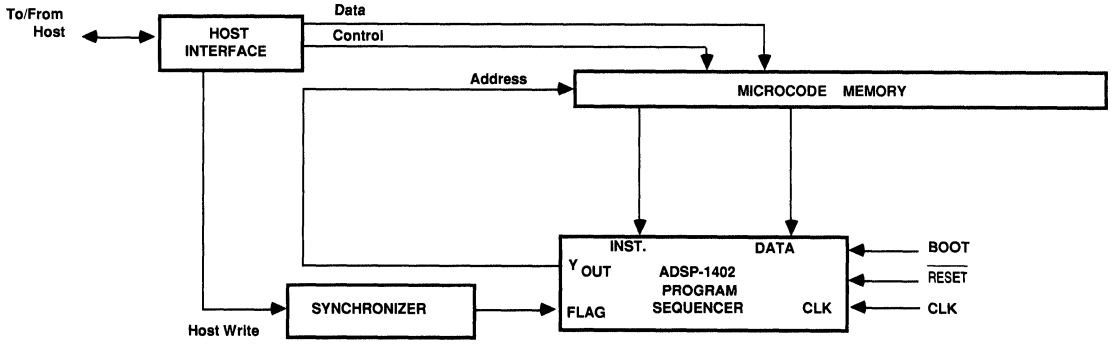
The system clock must be stable and $\overline{RESET}$ must be asserted for a minimum number of cycles before BOOT is asserted and after BOOT is deasserted. $\overline{IDLE}$ must be HI and TRAP must be LO for the entire time that BOOT is asserted. When BOOT is active, $FLAG_0$ is edge-sensitive. It must meet setup and hold times to the CLK rising edge, and it cannot be asserted more than every other cycle. BOOT timing is shown in Figure 4. See the ADSP-1402 data sheet for timing specifications.

WRITEABLE CONTROL STORE IN CONJUNCTION WITH FIXED CONTROL STORE

A microcode system that has a fixed control store (ROM) can be enhanced by adding a writeable control store (RAM) thus providing added flexibility. The ROM contains programs that are unlikely to change, such as a boot program, and the RAM is used for general purpose program area. The microcode ROM must contain a writeable control store download program that sets up the interrupt vector address and, if used, sets up the internal counter with the appropriate value. An example of the instruction sequence to download 1024 microcode instructions starting at location 256 is shown in Table 1.

Mnemonic	*Opcode*	*Data*	*Comment*
SLRIVP	1D	0020	Initialize stack limit register and interrupt vector pointer.
WRIV	0D	0100	Load interrupt vector address (100) into IV0.
WRCNTR	38	03FE	Load counter with 2 less than the number of memory locations to be filled (3FE).
WCS	20	0100	Start download at address 0100.

Table 1. Instruction Sequence For WCS Download



Note

$\overline{\text{IDLE}}$ must be HI and TRAP must be LO while the Boot function is being used. $\overline{\text{RESET}}$ must be active when BOOT is asserted and remain active until BOOT is deasserted.

Figure 3. Block Diagram of WCS Circuit with No ROM

The first instruction (SLRIVP) loads the stack limit register and the interrupt vector pointer. The interrupt vector pointer is loaded with a 0 to point to IV0, which is used for WCS, and a stack limit of 32 (20 Hex) is used. The interrupt vector pointer value is put into bits 12 - 15 and the stack limit is put into bits 2 - 5 to yield a 20 Hex.

The next instruction (WRIV) loads the interrupt vector address, which is 100 Hex in this example. This address is the starting address for program execution once microcode download is complete.

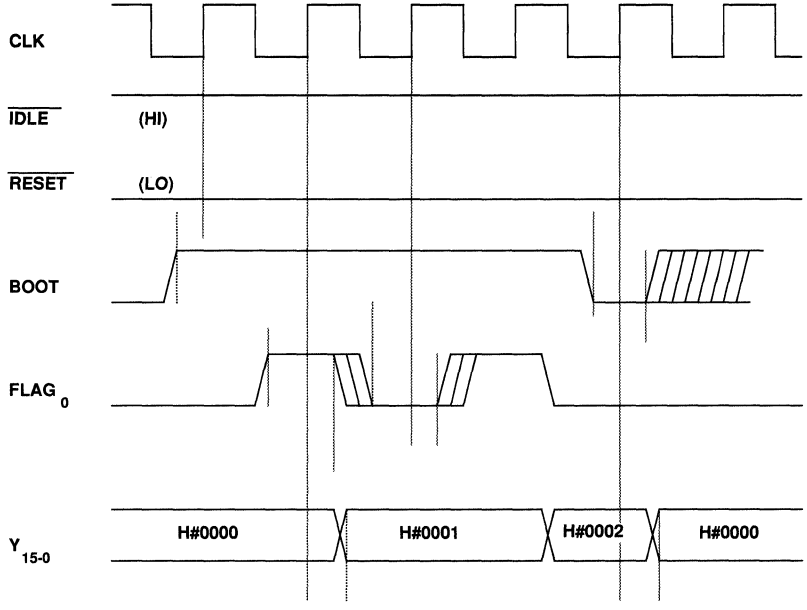


Figure 4. Boot Timing

The WRCNTR instruction loads the counter for use with the internal interrupt of the sequencer. The value loaded is two less than the number of instructions to be downloaded. 400 Hex instructions are to be downloaded so a count of 3FE Hex is used.

Finally, the WCS instruction initiates the download. A data value of 100 Hex is used to denote that download will start at location 100 Hex of the microcode memory and will continue sequentially until the counter underflows.

If an external interrupt is to be used instead of the internal interrupt from the counter, the WRCNTR instruction may be omitted. No added hardware is needed for this implementation and is the recommended configuration.

DATA SEGMENTING

In most cases the width of the microcode word is larger than the data width of the host bus. Therefore, the writeable control store must be loaded in segments where the number of segments needed is equal to the microcode field width divided by the host data bus width. For example, if the microcode word is 48 bits wide and the host data bus is 16 bits wide, three segments or three passes are needed to completely fill the microcode memory.

The host must divide the microprogram to be downloaded into segments and fill all locations of microcode memory one segment at a time. Thus, in the three segment example, the sequencer must perform the WCS instruction three times or the hardware boot sequence is followed 3 times. The circuit shown can be retriggered by the host for subsequent segment loading. In the case of a system containing ROM, the program must allow for multisegment loading.

A counter in conjunction with a decoder can be used to provide the appropriate write enable signals for the banks of RAM chips during each WCS pass. The host write, which initiates the WCS, is also used to increment the counter. The counter is reset by the host.

CONCLUSION

By implementing a writeable control store in your Word- Slice system, great flexibility is added in comparison to a system using fixed control store. The WCS feature of the ADSP-1402 Program Sequencer simplifies the task of downloading to a writeable control store and minimizes the need for added external circuitry.

Replacing the Am2910 with the ADSP-1402 Program Sequencer

by Gordon A. Sterling

Although the Am2910 has in the past been used in a variety of designs, it is now an old design. Its single counter, small addressing capability and minute stack are inadequate when compared to the current generation of sequencers available. In particular, the ADSP-1402 eclipses the Am2910 on every front. The ADSP-1402 provides on-chip interrupt handling, 64-word stack, four independent counters, and significantly more instructions.

This application note describes replacing the well-known but old Am2910 with the superior ADSP-1402. It presents the details necessary to design with the ADSP-1402 instead of the Am2910. A general feature comparison is followed by a specific Am2910 instruction replacement section. It presents each instruction with its ADSP-1402 equivalent(s). A brute force translation of Am2910 code is not the most efficient way to use the ADSP-1402. It does, however, provide the framework for understanding the ADSP-1402.

INSTRUCTION SET

The ADSP-1402 has an instruction set consisting of 60 powerful instructions. Seven microcode bits provide conditional jumps, returns, loop control, register manipulation and a variety of other operations. As seen below, the ADSP-1402 instruction set can be used as an extensive superset to the limited number of instructions (16) of the Am2910.

INTERNAL MEMORY

With its 64-word on chip RAM, the ADSP-1402 has the space to handle automatic interrupt vectoring and deep subroutine nesting. Using the Global and Local stack pointers into this memory, a large number of registers can be accessed. Four separate down-counters are dedicated to storing count values used for looping and other purposes.

On the other hand, the Am2910 is severely limited by its five location stack. (The Am2910A supports a nine word stack). Its single register is further hindered by its dual role as a counter. Additional hardware is required for AMD's pseudo-interrupt scheme.

ADDRESS SPACE

The ADSP-1402 Program Sequencer provides the addresses needed to sequence a microcoded system through the instructions stored in microcode memory. It can output a 16-bit address (65,536 words) at a high speed because its internal Look-Ahead pipeline coordinates the timing of its instruction input and address output, which are both latched on-chip. The Am2910 provides only 12 addressing bits, so it is limited to only 4096 microcode words. In addition, an external pipeline latch is required for the instruction bits of the Am2910. The ADSP-1402 provides a full 16 bits in its counters and registers, while the Am2910 is limited to only 12 bits in its single register/counter.

INTERRUPT HANDLING

Interrupt handling is provided in hardware on the ADSP-1402. Up to eight external and two internal interrupts can be used, without any additional hardware. The two internal interrupts are used for stack limit violations and counter underflow. The Am2910 attempts interrupt handling by checking the condition bit and using its VECT output. A single instruction is provided to jump to an address provided by an external vectoring PROM based on an external condition. In addition to the vector PROM, this scheme requires a priority interrupt encoder.

REPLACING THE Am2910 WITH THE ADSP-1402

The following information should only be used for comparison purposes. The ADSP-1402 provides a variety of features that can perform many of the Am2910 operations in a more efficient fashion. The ADSP-1402 allows the programmer more freedom with its significantly larger instruction set. For a complete description of the ADSP-1402, a copy of its data sheet should be obtained from your local Sales Office.

Since the ADSP-1402 does not require any source control outputs, such as the MAP, VECT and PL lines on the Am2910, they are not present. As mentioned above, under interrupt handling the ADSP-1402 has no need of a vectoring PROM and the PL output is unnecessary since the ADSP-1402 can

take its data input directly from the microcode word. The MAP output is also unnecessary since the ADSP-1402 64-word internal RAM can provide the required addresses. If it is necessary to simulate the MAP output, it can be accomplished with one microcode bit. Whenever an address was desired from the mapping PROM, its output going to the ADSP-1402's data port could be enabled with this bit. The buffer necessary to tristate any other lines going to the data port would be controlled with the same bit as the mapping PROM.

Whenever an Am2910 instruction is listed as conditional, it is based on the value of the $\overline{CC}$ input pin. In addition, the $\overline{CCEN}$ pin can be used. Whenever $\overline{CCEN}$ is high, checking of the $\overline{CC}$ pin is disabled, and the part functions as if $\overline{CC}$ were low. It is possible to simulate this setup on the ADSP-1402. The circuit of Figure 1 generates a high signal for a passed test and a low for a failure. The output of this circuit should be tied directly to the FLAG input of the ADSP-1402. All instructions can now be conditioned on the FLAG input.

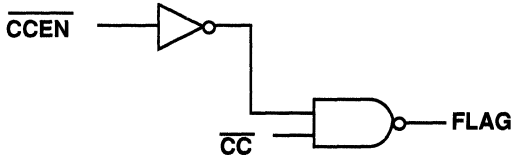


Figure 1. ADSP-1402 Equivalent Conditional Logic

The $\overline{CCEN}$ line was included on the Am2910 because many of its instructions are always conditional. The ADSP-1402 supports conditional execution of certain instructions with four different options. They are FLAG, NOT FLAG, SIGN, and UNCONDITIONAL. The FLAG and NOT FLAG options are based on the state of the external flag input to the ADSP-1402. The SIGN condition is generated by either forcing its state with an instruction or by the sign bit of the last counter decremented. The UNCONDITIONAL flag causes the instruction to be executed regardless of the state of any of the flags. As you can see, this can be used to eliminate the $\overline{CCEN}$ line, since the instructions can be made unconditional in software.

Am2910 Instruction

JZ Jump Zero (Reset)

This instruction is used to generate zero as the next address. It also clears the stack pointer. Its function can be duplicated on the ADSP-1402 by generating a RESET. If it was only necessary to output location zero as the next address, the ADSP-1402 would execute an unconditional jump to data absolute (JDA) instruction while zero was present at its data port.

CJS Conditional Jump to Subroutine

The ADSP-1402 has an identical instruction (JSA).

JMAP Jump Map

This instruction enables the MAP output of the Am2910 and jumps to the address present at its data port. The ADSP-1402 can be made to jump to an address at its data port using the jump data absolute instruction (JDA). Optionally, the internal RAM of the ADSP-1402 can be used to store mapping addresses. In this case, the Jump Register (JRC) instruction would be executed. If it is necessary to jump to a location supplied by a mapping PROM, this could be accomplished using a single microcode bit to enable the output of the PROM while disabling the microcode data field.

CJP Conditional Jump Pipeline

The ADSP-1402 can execute an identical instruction (JDA).

PUSH Push/Conditional Load Counter

When using the Am2910, you must place the jump address for looping on the stack. This is necessary because only one register is available for addressing and counting. This instruction is an attempt to get around this restriction. This restriction is not present on the ADSP-1402 since it has four separate counters, and up to 64 different registers. When you desire to load a counter on the ADSP-1402, a single instruction, Write Counter from Data Port (WRCNTR), is executed. It can easily be made conditional by preceding it with an If Condition (NOT FLAG) Jump PC+2 (JTWO) instruction.

JSRP Conditional Jump to Subroutine

Used on the Am2910 for conditional jumping to one of two subroutines, it is necessary for the instruction immediately preceding this to set the register/counter to the proper value for the jump. For proper operation, this instruction is part of a two-instruction sequence. This could be accomplished on the ADSP-1402 with two Conditional Jump to Subroutine (JSA) instructions. The first instruction would jump to the first subroutine if its condition is true, otherwise, the next instruction will cause a jump to the alternate routine. If the first subroutine is executed, it must insure, upon completion, that the second condition is false, avoiding execution of the second subroutine.

CJV Conditional Jump Vector

This command is used by the Am2910 to simulate interrupts. When the condition is true, the VECT output is asserted. An additional register or PROM is required to generate the appropriate address. No return address is pushed on the stack, so some technique must be used to return program flow. The ADSP-1402 is able to handle up to eight interrupts and can generate an interrupt vector automatically. When the ADSP-1402 is initially set up, there are various options setting the interrupt sensitivity, masking, and vectoring. After that, interrupt handling is automatic. The return address is automatically pushed onto the stack for program flow control.

JRP Conditional Jump

This instruction is very similar to the Am2910 JSRP instruction. The ADSP-1402 can perform the same operation with the BRANCH instruction. In addition to jumping to one of two locations, BRANCH decrements one of the four counters. This provides a simple technique for looping and branching with minimal overhead. If the SIGN condition of the ADSP-1402 is true, the next address will be taken from a register; otherwise if the specified condition is true, the address will come from the data bus.

RFCT Repeat Loop, Counter≠0

This Am2910 instruction causes the next address to be taken from the top of the stack if the counter is not equal to zero. When the counter is zero, the loop is exited by using PC+1 as the next address. The ADSP-1402 instruction that performs this operation is JRS. This instruction checks the previous sign of the counter, decrements the specified counter, and jumps to the specific register value, if the sign condition was true. For a loop to be executed N times, the counter should be loaded with 2^{15}-2+N. Of course, the Am2910 only provides one register/counter, while the ADSP-1402 provides four separate counters.

RPCT Repeat Pipeline Register, Counter≠0

This instruction is an attempt to compensate for the single register/counter of the Am2910. It allows the jump address to come from the microcode instruction word (through the data port). It uses the address at the data port until the counter is zero, then uses PC+1 as the next address. The JDRST instruction of the ADSP-1402 performs this operation. It checks the sign condition, decrements the counter, and jumps to the address at the data port if the condition is false. When the sign condition is true, it resets the counter and proceeds to PC+1.

CRTN Conditional Return

If the Am2910 condition is true, the return from subroutine is executed and the address is taken from the top of the stack. Otherwise, the next instruction is PC+1. The ADSP-1402 contains an identical instruction (RTN).

CJPP Conditional Jump Pipeline and Pop

The Am2910 specifies that this instruction is used for loop termination and stack maintenance. It provides a method of popping the stack while jumping to another address. This can be accomplished in a variety of ways on the ADSP-1402. The most direct would be to decrement the stack pointer (DSSP) and then perform the conditional jump (JDA).

LDCT Load Counter and Continue

As stated in the name, this instruction loads the counter of the Am2910 with the value at the data port and continues to the next instruction. The ADSP-1402 provides an identical instruction (WRCNTR) to load any of its four counters. A variety of other instructions allow different sources for the counter load.

LOOP Test End of Loop

This Am2910 instruction is used to conditionally exit a loop. If the condition is false, it will jump to the address on the top of the stack. When the condition is true, it will pop the stack and proceed to PC+1. This can be done on the ADSP-1402 using the Conditional (NOT FLAG) Jump Register (JRC) instruction. This will cause the next address to be generated from the specified register, or PC+1, depending on the condition.

CONT Continue

This simple instruction of the Am2910 causes the program counter to increment, generating the next sequential address. The ADSP-1402 Continue (CONT), not only sounds the same, but performs the same function.

TWB Three Way Branch

This Am2910 instruction performs a loop until zero function. If the counter is not zero, it will branch to the address on the top of the stack; otherwise, it will jump to the address at the data port. If during its execution its condition is passed, it will generate PC+1 as the next address. The BRANCH instruction of the ADSP-1402 operates in a similar fashion. If the sign condition is true, it will jump to the address specified by the register; otherwise if the condition (NOT FLAG) is true, it will jump to the address specified at the data port. If neither condition is true, PC+1 will provide the next address. The desired counter should be loaded with 2^{15}-2+N, to loop N times.

In addition to the ten or so ADSP-1402 instructions described above, there are many more that provide a variety of advanced functions that cannot be duplicated by the Am2910. For complete descriptions of all the instructions, you should request the *Word-Slice® User's Manual* from your local Analog Devices Sales Office.

Loading an ADSP-2101 Program via the Serial Port

by Gerald McGuire

INTRODUCTION

For many DSP applications, it is desirable to have a DSP processor under the control of a host computer. In these situations, the host computer would download a program for the DSP to execute. The ADSP-2101 provides two serial ports suitable for program download from a host computer. This application note details the ADSP-2101 monitor program for downloading from a serial port. The monitor program itself would be booted from EPROM or other boot memory. While this example uses serial port zero, the principal could be extended to download via a memory-mapped parallel port.

A MONITOR

The task of the host computer is to download a series of instructions to the ADSP-2101 for execution. The ADSP-2101 receives the incoming instructions, loads them into program memory and when all instructions have been received, executes them. Prior to and during the download from the host, the ADSP-2101 executes a monitor program. This monitor activates the serial port, receives the instructions and places them in program memory for execution.

The ADSP-2101 instruction is twenty-four bits wide but many hosts, including eight-bit processors, more readily handle byte-wide data. Since the serial port can accommodate serial words from three to sixteen bits in length, byte-length data words are easily received.

Whenever a program memory write occurs, the sixteen most significant bits are supplied by the source register, explicitly named in the instruction, and the eight LSBs are supplied by the PX register. The basic tactic of the monitor program is to assemble the two most significant bytes in a data register (using the Shifter) and load PX explicitly with the least significant byte. A program memory write then writes the correct twenty-four bit instruction.

In addition to the transfer of instructions through the serial port into program memory, the monitor program must also know when the download is complete and execution can begin. A straight forward method is to count the number of instructions sent to the serial port. A count value is sent to the ADSP-2101 before the first instruction. This is the count of the instructions to follow. After each instruction is downloaded, the count can be decremented.

The downloaded program must avoid overwriting the monitor program while the monitor executes. The last instruction of the monitor program is identified by a global label which also identifies the beginning of the available space for downloaded code. The monitor program must be linked with the downloaded program so that the downloaded program makes the correct address references including the reference to this global label.

The indirect addressing capabilities of the Data Address Generators on the ADSP-2101 make it easy to cycle through the correct sequential locations starting with the label.

The final concern is the interrupt table. If the downloaded program is interrupt-driven, the interrupt table (program memory H#0000 to H#001C) must contain valid instructions for servicing expected interrupts.

There are several ways to do this. First, the monitor program itself could contain the valid interrupt table for the program to be downloaded. This assumes that the interrupt structure of the downloaded program is known when the monitor program is created. Second, the interrupt table may be downloaded through the serial port just as the rest of the program is. The DAG can loaded with the start address of the interrupt table and the instructions can be loaded, but you may not overwrite the interrupt being used to receive the data on the serial port until all instructions have been received.

The monitor program example does not load an interrupt table. The best approach is dependent on your application.

IMPLEMENTATION

The first task of the monitor program is to setup and enable the

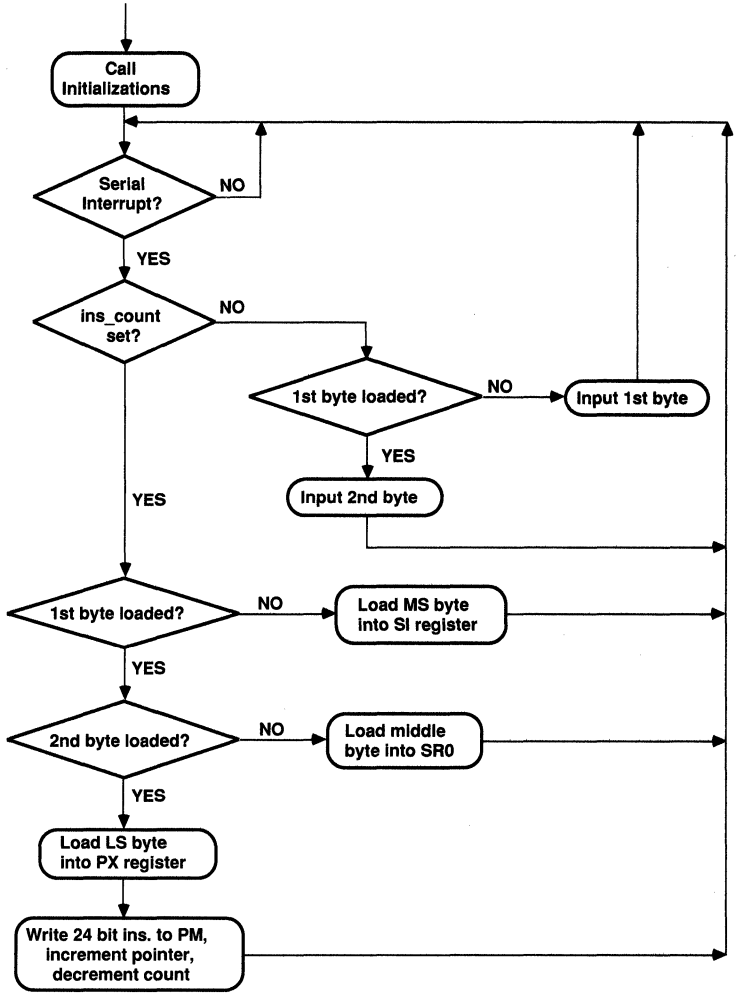


Figure 1. Boot Program Flow Diagram

serial port. Serial ports on the ADSP-2101 are extremely flexible in terms of framing options, word lengths and timing. The ADSP-2101 serial ports may receive the frame synch and serial clock from the host processor or generate them internally.

As the program is downloaded from a host computer, the ADSP-2101 looks to the host for serial port information. That is, the serial port frame synchronization and serial port clock are supplied by the host computer. For purposes of illustration, the code that appears at the end of this application note uses normal framing and external receive frame synchronization. For externally generated serial clocks the ADSP-2101 can support frequencies up to the processor instruction rate.

The flow for the monitor program is shown in Figure 1.

Once the serial ports are enabled, the monitor program waits for a serial port interrupt signifying that a serial word has been received. The first two serial words received are the instruction count. As the serial word is eight bits, two serial port words make up the instruction count. The separate bytes of the instruction count are combined in the shifter and loaded into data memory. This count represents the number of instructions to be downloaded from the host and does not include the interrupt table. The interrupts are handled automatically, as the interrupt table has a fixed length.

With the count downloaded, the ADSP-2101 is ready to accept instructions through the serial port. Instructions are downloaded a byte at a time just as the instruction count was. The most significant byte is first. It is loaded into the SI register and the byte count ("count") is decremented. The middle byte

of the instruction is loaded into the SR0 register. These two bytes are combined in the shifter with the results residing in the SR0 register. Once again the byte count is decremented. Finally, the least significant byte is loaded into the PX register. Now that all three bytes are loaded into registers on the ADSP-2101, the downloaded instruction can be written to program memory.

When all is downloaded, a jump to the new downloaded program is all that is necessary to begin execution.

The monitor program is listed at the end of this note.

SUMMARY

A monitor program initializes the serial port and receives instructions, writing them into program memory, then beginning execution. This method of booting is useful when the ADSP-2101 is under the control of a host computer or controller. Any size program may be downloaded (up to the full addressing capability of the ADSP-2101) with this particular method of implementation. Only the program memory used by the monitor program cannot be loaded. That space could, however, be used for program memory data storage.

```
.MODULE/RAM/BOOT=0                  serial_boot_monitor;
.VAR/DM                             count;              {counts bytes}
.VAR/DM                             ins_count;          {counts instructions}
.GLOBAL                             code_start;         {end of monitor space}


                    JUMP restarter; NOP; NOP; NOP;      {restart vector}
                    RTI; NOP; NOP; NOP;                 {IRQ2 not used}
                    RTI; NOP; NOP; NOP;                 {sport0 TX not used}
                    JUMP serial; NOP; NOP; NOP;         {sport0 RX }
                    RTI; NOP; NOP; NOP;                 {sport1 TX not used}
                    RTI; NOP; NOP; NOP;                 {sport1 RX not used}
                    RTI; NOP; NOP; NOP;                 {no timer used}

restarter:          CALL initializations;
wait_loop:          IDLE;
                    JUMP wait_loop;

initializations:    I4 = H#3ff3;                        {pointer to mem map reg}
                    I5 = ^code_start;                   {pointer to start of prog}
                    M4 = 0;
                    M5 = 1;
                    L4 = 0;
                    L5 = 0;
                    SR0 = 0;
                    SR1 = 0;

                    AX1 = 1;
                    DM(count) = AX1;                    {count val for # of bytes}
                    DM(I4,M5) = 0;                      {disable autobuffer}
                    DM(I4,M5) = 0;                      {no frame divide modulus}
                    DM(I4,M5) = 0;                      {no clk divide modulus}
                    DM(I4,M5) = H#2007;                 {extrnl RFS & SCLK, no compand}
                                                        {SLEN 8, no multichannel}
                    AX0 = H#1000;
                    DM(H#3fff) = AX0;                   {enable sport0}
                    AX0 = H#FFFF
                    DM(ins_count) = AX0;

                    IMASK = 8;                          {sport0 rec interrupt only}
                    RTS;

serial:             AY1 = DM(ins_count);
                    AR = PASS AY1;
                    IF GT JUMP next_instruction;        {get next instruction}
                    IF LT JUMP load_word_count;         {get number of instructions}
                    IF EQ JUMP code_start;              {start downloaded program}
```

```
{load the count, that is, the number of instructions to be downloaded}
{this happens in two bytes  The most significant byte first}

load_word_count:    AY0 = DM(count);                    {is this 1st or 2nd byte}
                    AR = PASS AY0;
                    IF NE JUMP first_byte;
                    IF EQ JUMP second_byte;

first_byte:         SI = RX0;                           {first byte decrem. count}
                    AR = AY0 - 1;
                    DM(count) = AR;
                    RTI;

second_byte:        SR0 = RX0;                          {second byte...}
                    SR = SR OR LSHIFT SI BY 8 (LO);     {put two bytes together}
                    DM(ins_count) = SR0;                {store in ins_count}
                    AX0 = 3;
                    DM(count) = AX0;                    {load count for ins.}
                    RTI;


{load the next instruction.  Instructions are 24 bits long and appear}
{at the serial port in 8 bit fragments.  The most significant byte 1st}

next_instruction:   AX0 = 2;                            {decide which byte is due}
                    AY0 = DM(count);
                    AR = AX0 - AY0;

                    IF LT JUMP most_sig_byte;
                    IF EQ JUMP middle_byte;
                    IF GT JUMP least_sig_byte;

most_sig_byte:      SI = RX0;                           {load MS byte into SI}
                    AR = AY0 - 1;                       {decrement count}
                    DM(count) = AR;
                    RTI;

middle_byte:        SR0 = RX0;                          {load Middle into SR}
                    SR = SR OR LSHIFT SI BY 8 (LO);     {put MS and middle together}
                    AR = AY0 - 1;                       {decrement count}
                    DM(count) = AR;
                    RTI;

least_sig_byte:     PX = RX0;                           {put LS byte into PX}
                    PM(I5,M5) = SR0;                    {write SR0 into PM}
                                                        {PX provides 8 LS bits}
                    AX0 = 3;
                    DM(count) = AX0;                    {reset byte count}
                    AR = AY1 - 1;                       {decrement ins count}
                    DM(ins_count) = AR;
                    RTI;

code_start:         NOP;

.ENDMOD;
```

Disk Drive Head Positioning with the ADSP-2101

by Kapriel Karogozyan

INTRODUCTION

A hard disk servo control system must do two main things. First, it must position the read/write heads onto the desired tracks and maintain their position until the data is read or written. Second, it must precisely control the speed of the spindle motor. Although these control tasks are traditionally handled by analog systems, these circuits have difficulty coping with the increasing track densities and shorter access times in today's high performance disk drives. High-speed digital servo control circuits are the solution to this problem. However, this places high demands on the computational powers of the digital circuitry and traditional microcontrollers fall short. These servo control demands can be easily handled by the ADSP-2101 DSP Microcomputer.

We briefly look at how a high performance digital servo head positioning system for a hard disk drive can be implemented using the ADSP-2101. More detailed description of the design and implementation of this servo control system is available as an application note upon request.

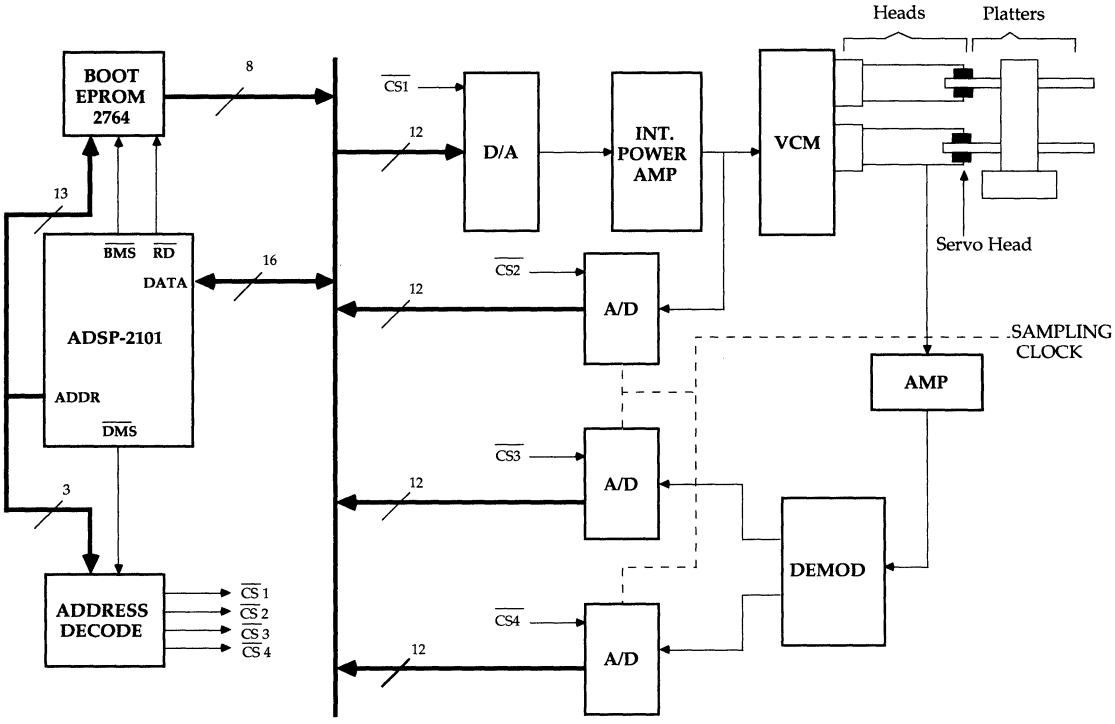


Figure 1. Hard Disk Digital Servo Head Positioning System

SYSTEM OVERVIEW

The servo system that is described here is designed for disk drives that use a "voice coil motor" (VCM) as their head actuator and that also use "dedicated" servo information. The VCM actuator consists of a coil movable through the magnetic field of a permanent magnetic stator. The heads, which are attached to the coil, can be moved radially by applying current to the VCM. All heads move in unison. This allows us to dedicate one of them to send position feedback to the control circuitry. Pre-recorded servo information is located on a platter dedicated for this purpose. The servo information is sensed by the dedicated servo head and fed back to the controller.

SYSTEM ARCHITECTURE

A block diagram of the ADSP-2101 based servo control system is shown in Figure 1. The head-disk assembly to be controlled is shown on the upper right hand corner. The heads are actuated by the VCM which takes its controlling current input from an integrating power amplifier. The input to the power amplifier is provided by a 12-bit digital to analog converter (DAC). The digital word converted by this DAC is the output of the control system. The servo information sensed by the dedicated servo head is demodulated in order to generate a position error signal (PES) which is converted to two 12-bit words and read by the ADSP-2101. The PES indicates the position error of the servo head from the nearest track center line. The output of the integrating power amplifier is also fed back to the ADSP-2101 as a 12-bit word. The ADSP-2101 that is shown on the left side of the figure runs a number of algorithms in order to accomplish its control tasks. It uses a 250ns 8Kx8 EPROM as a boot memory and it also uses a decoder chip for addressing the proper converters. The total number of components needed to build this system based around an ADSP-2101 is small. The control system firmware is also very flexible if parameter or feature modifications are necessary.

OPERATION

The digital servo control system shown in Figure 1 receives a command from a host processor (possibly over a SCSI interface) and positions the read/write heads onto the desired track in the minimum possible time. It can also keep the heads precisely over the center line of a track until a new positioning command is received from the host. The control system accomplishes these duties by having the ADSP-2101 run several algorithms.

The most involved digital control algorithm used here is referred as the state estimator. This algorithm calculates the estimated head position, velocity and acceleration for each digital sample, based on the predicted head position, velocity and acceleration along with the measured head position and the measured VCM current. This algorithm uses a large number of multiplications and additions that involve a large number of characteristic system constants, previous samples and predicted samples of the desired variables. ADSP-2101's single cycle multiply-accumulate instruction with two concurrent memory fetches facilitates the execution of this algorithm and saves time for other functions.

Another algorithm used in the system is a lead-lag network which also requires a number of multiply and accumulate operations.

The next important algorithm is the polynomial approximation function. This function is needed to make an instantaneous approximation of the velocity trajectory curve. This is a function for the optimal head velocity for a given remaining distance to target position. The ADSP-2101's 40-bit multiplier-accumulator extends the dynamic range of the polynomial computations and allows the system to have more accurate control outputs. All of the algorithms mentioned above need to use several constants in sequence for a large number of iterations. The circular addressing capabilities of the ADSP-2101 allows efficient execution of these iterations. The on-board memories also serve as data and coefficient storage and eliminate the need for additional memory components. The ADSP-2101 is easily interfaced with memory-mapped peripherals. This allows it to access the needed A/D and D/A converters in a simple manner. Overall, the ADSP-2101 is well suited to handle digital servo control system tasks.

NOTE

The digital servo head positioning system described in this article is an ADSP-2101 based implementation of a patented design by M. L. Workman of IBM. The title of the patent is "Digital Servo Control System For A Data Recording Disk File" and is filed under U.S. Patent No. 4,679,103 (1987).

Digital Filtering with the ADSP-2100A

by Bob Fine

INTRODUCTION

The ADSP-2100A is a single-chip microprocessor optimized for DSP processing tasks. This note describes how to take advantage of the internal architecture of the ADSP-2100A to implement stand-alone digital filtering functions. A theoretical overview of FIR filters is presented followed by a hardware configuration description and software program development where actual code is presented.

FIR FILTER IMPLEMENTATION

An FIR filter must perform the following convolution equation:

$$y(n) = h(n) * x(n) = \sum_{i=0}^{N-1} h(i)\, x(n-i)$$

where h(i) is the filter coefficient array and x(n-i) is the input data array to the filter. This input would typically come from an A/D converter. The number N, in the equation, represents the number of taps of the filter and relates to filter performance. (See *ADSP-2100 Applications Handbook, Volume One*, available upon request.)

Memory Addressing

As an example of convolution calculations, the equations for three output points from a four-tap filter are shown below:

y(5) = h(0)x(5) + h(1)x(4) + h(2)x(3) + h(3)x(2)
y(6) = h(0)x(6) + h(1)x(5) + h(2)x(4) + h(3)x(3)
y(7) = h(0)x(7) + h(1)x(6) + h(2)x(5) + h(3)x(4)

In the series of equations, the N coefficient locations are always accessed sequentially from h(0) to h(N-1). The associated data points circulate through memory; new samples are added replacing the oldest each time a filter output is computed.

A fixed boundary RAM can be used to achieve this circulating buffer effect. The following diagram shows the contents of the data RAM for several data samples.

Input	Memory Location	Contents	Output
	0	X(1)	
	1	X(2)	
	2	X(3)	
X(4) ———	3	X(4)	
			Y(4)
X(5) ———	0	X(5)	
	1	X(2)	
	2	X(3)	
	3	X(4)	
			Y(5)
	0	X(5)	
X(6) ———	1	X(6)	
	2	X(3)	
	3	X(4)	
			Y(6)
	0	X(5)	
	1	X(6)	
X(7) ———	2	X(7)	
	3	X(4)	
			Y(7)

Figure 1. Circular Buffer RAM Contents

The oldest data sample is replaced by the newest after each convolution. A "time history" of the four most recent data samples is kept in which the four delay taps give rise to the name "four tap filter."

This delay line can be implemented with a fixed boundary RAM if new data values are written into memory as shown in Figure 1. To facilitate memory addressing, old data values are read from memory starting with the value one location after the value that was just written. For example; x(5) is written into memory location 0 and data values are then read from locations 1, 2, 3, and 0 as shown in Figure 2.

Of course, this example can be expanded to accommodate any number of taps. By addressing data memory locations in this manner, the address generator need only supply sequential addresses regardless of whether the operation is a memory read or write. This data memory buffer is called circular because when the last location is reached, the memory pointer must be reset to the beginning of the buffer.

Output	y(4)	y(5)	y(6)	y(7)	y(8)
Write address	3	0	1	2	3
1st read address	0	1	2	3	0
2nd read address	1	2	3	0	1
3rd read address	2	3	0	1	2
4th read address	3	0	1	2	3

Figure 2. RAM Address Sequence

The coefficients are fetched simultaneously with the data. Due to the addressing scheme chosen, the oldest data sample is fetched first. Therefore, the last coefficient must be fetched first. The coefficients can be stored backwards in memory (h(N-1) is in the first location and h(0) is in the last) with the address generator providing incremental addresses, or coefficients can be stored in a normal manner with accessing of coefficients starting at the end of the buffer and the address generator being decremented. In the design example below, the coefficients are to be stored in a reversed manner.

HARDWARE CONFIGURATION

The ADSP-2100A, along with a program memory and data memory, can be connected directly to an A/D and D/A converter for real-time digital filtering. The example which follows describes a 15-tap FIR filter. Of course this example can be expanded to accommodate any number of taps. The ADSP-2100A supports an external Harvard architecture so that both program and data memories can be simultaneously accessed. By storing coefficients in the program memory and input samples in the data memory, input values for the convolution can be fetched in one cycle, as long as the next instruction resides in the instruction cache (See *ADSP-2100 User's Manual.)*

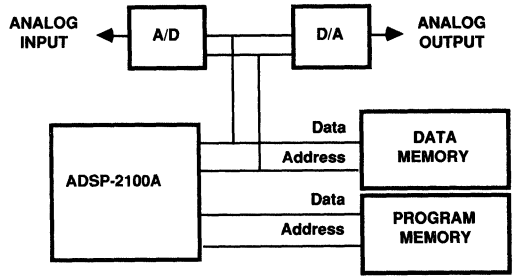


Figure 3. Minimum Hardware FIR Configuration

The minimum hardware configuration for the FIR filter is shown in Figure 3. Note that, due to the external Harvard architecture, both program and data memories have an individual data and address bus. Also, the A/D and D/A converters are mapped into the data memory address space.

SOFTWARE DEVELOPMENT

The software development for the ADSP-2100A microprocessor consists of several stages which are greatly simplified by the use of the ADSP-2100A development system (See *ADSP-2100 Cross-Software User's Manual.*) An architecture description must be created, the FIR program modules written and assembled and the modules linked.

Architecture Description

The purpose of the architecture description is to let the development system know the context of the program within a specific hardware configuration. To describe the hardware in Figure 3, we construct the following architecture description file.

```
.SYSTEM                          filter_system;
.SEG/ROM/ABS=0/PM/CODE           prog_mem[4096];
.SEG/ROM/ABS=4096/PM/DATA coeff_stor[256];
.SEG/RAM/ABS=0/DM/DATA           delay_line[256];
.PORT/ABS=H#3FFE                 ad_sample;
.PORT/ABS=H#3FFF                 da_data;
.ENDSYS;
```

Figure 4. Architecture Description

This program sets up the architecture description called *filter_system.* A 4K program memory ROM segment is placed at absolute location 0 and is called *prog_mem.* This space is reserved for the actual program instructions. A program memory data ROM section of 256 locations is placed after the program ROM. This segment is called *coeff_stor* and is used for the storage of filter coefficients. A data memory RAM segment of 256 locations is reserved for the data buffer and is called *delay_line.* Finally, two ports are reserved in the names *ad_sample* and *da_data* for the A/D and D/A converters.

FIR ASSEMBLY PROGRAM

The next step in our development is to create the program containing the instructions which carry out the FIR algorithm. In this example, a main routine containing initialization commands is developed separately from the FIR routine. These modules will later be linked together with the Linker. (See *ADSP-2100 Cross-Software Manual.*)

Since the FIR filter calculations run much faster than the A/D sample interval used in this example, ADSP-2100A interrupts are used. The main routine runs in the background until an A/D sample is ready. The A/D converter interrupts the ADSP-2100A and the FIR filter routine is called.

The assembly language program for the main routine module, called *main_routine*, is shown in Figure 5.

Declarations

In the main routine, the constant *taps* is defined which represents the number of taps. Circular buffers called *data_buffer* and *coefficient* are declared for data memory and program memory segments with a length equal to the number of taps of the FIR filter. Note that only fifteen of the 256 available locations are used. The FIR interrupt routine begins at label *fir_start* in the FIR module and is declared as an external reference. The coefficient buffer is initialized with the

```
.MODULE/ROM/ABS=0 main_routine;
.CONST taps=15;                          {Number of Filter Taps}
.VAR/DM/RAM/CIRC data_buffer[taps];      {Buffers declared as circular; will be }
.VAR/PM/RAM/CIRC coefficient[taps];      {automatically placed on a Modulo N}
                                         {boundary, where N is the buffer length}
.EXTERNAL   fir_start;                   {Declare External Reference}
.INIT coefficient : <COEFF.DAT>;         {Initialize coefficients.}

{Interrupt Service Instructions for Interrupts 0 - 3}
                        JUMP FIR_START;
                        RTI;
                        RTI;
                        RTI;

{Initialize Memory Pointers}
                        I0=^data_buffer;
                        I4=^coefficient;

{Initialize Memory Pointer Modifiers}
                        M0=1;
                        M4=1;

{Set Buffer Lengths}
                        L0=%data_buffer;
                        L4=%coefficient;

{Clear Delay Line}
                        CNTR=taps;
                        DO delay_clear UNTIL CE;
delay_clear:            DM(I0,M0)=0;

                        IMASK=B#0001;           {Enable Interrupts}
                        ICNTL=H#0F;
mainloop:          JUMP mainloop;               {Loop and Wait for Interrupt}

.ENDMOD;
```

Figure 5. Main Routine

filter coefficients that were previously calculated and stored in a file called COEFF.DAT. This initialization actually takes place at link time but is declared in the assembly program.

Interrupt Vector Table

The first four locations of the program memory are for the interrupt service instructions which are usually jump instructions to interrupt routines. In this example, interrupt 0 is used for the A/D converter and so the first interrupt vector is a jump to *fir_start*, the beginning of the FIR interrupt routine. No other interrupts are used and the interrupt vector space contains RTI instructions for the remaining interrupts. This is done so that if by some mistake other interrupts are allowed to occur, a return from interrupt will be executed and the program can resume.

Program Initialization

Upon reset, the interrupts are disabled and the registers are ready to be initialized. The address register, I0, of the data memory address generator is initialized to the starting location of the data buffer. Note that this is done symbolically (See *ADSP-2100 Cross-Software Manual.*) The address register, I4, of the program memory address generator is initialized in a similar manner to the starting location of the coefficient buffer. Next, the address modifier registers, M0 and M4, are set to a 1 so that after a memory location is accessed, the address register will be incremented by 1, pointing to the next location. Since these buffers are circular, the address pointers automatically reset to the top of the buffer after the last buffer location is accessed. The last part of the initialization deals with the L (length) registers of the data memory address generator and the program memory address generator. Registers L0 and L4 are set to the length of the data buffer and the coefficient buffer.

Main Routine

Once this initialization is complete, the interrupts are enabled and the main routine waits for an interrupt to occur. In this example, the main routine merely loops.

```
.MODULE/ROM fir_routine;

.CONST TAPS=15;                    {Set Constant to Number Of Filter Taps}
.PORT ad_sample;                   {Declare I/O Ports}
.PORT da_data;
.ENTRY fir_start;                  {Declare program Entry Point}

fir_start:  CNTR=taps-1;           {Set Program Sequencer Counter}
            SI=DM(ad_sample);      {Store A/D Sample in SI Register}
            DM(I0,M0)=SI;          {Load Delay Line With Data Sample}
                                   {Load Data Sample and Coefficient into Multiplier
                                   Input Registers}
            MR=0,MY0=PM(I4,M4),MX0=DM(I0,M0);

            DO convolution UNTIL CE;

{Perform Multiply And Fetch Next Operands}

convolution: MR=MR+MX0*MY0(SS), MY0=PM(I4,M4), MX0=DM(I0,M0);
                        MR=MR+MX0*MY0(RND);     {Do Last Multiply With RND
                                                Specification}
                        IF MV SAT MR;           {If overflow, saturate result.}
                        DM(da_data)=MR1;        {Send Results To D/A.}
                        RTI;                    {Return From Interrupt.}
.ENDMOD;
```

Figure 6. FIR Routine

FIR ROUTINE

The FIR interrupt routine has been written as a separate module called *fir_routine* and is shown in Figure 6.

Note that since coefficients are stored in program memory and the delay line resides in data memory, the two operands are simultaneously fetched in a single cycle. The filter loop fits into the internal cache memory and the instructions are fetched from the cache after execution of the first loop.

Declarations

The module *fir_routine* performs the actual digital filter task. The number of taps is defined by the constant *taps* and two I/O ports are declared, one for the A/D converter and a second for the D/A. Note that the constant *taps* is defined in both the main routine and the FIR routine. An alternate solution would be to use .INCLUDE, where both modules reference one constant in an include file (See *ADSP-2100 Cross-Software Manual.*)

The statement label *fir_start* is declared as an entry point from the main routine.

Filter Calculation

The FIR routine starts by setting the counter of the ADSP-2100A program sequencer (CNTR) to one less than the number of taps of the filter (the last calculation is performed outside the loop). The next instruction takes the data from the memory mapped A/D and stores it in a register internal to the ADSP-2100A (SI). This data word is then transferred from the SI register to a location in the circular buffer, which resides in the external data memory, specified by the value of the address generator register, I0. The address generator register I0 is then modified by the contents of modifier register M0.

DO Loop

Once the circular buffer is updated, the actual convolution can begin. A coefficient is fetched from the program memory and a data value is fetched from the data memory in the same cycle. A DO loop is then entered which repeatedly performs a multiply/accumulate and a fetch of the next operands from the data and program memories until fifteen coefficients and fifteen data points are read. The addressing scheme that was outlined in the previous sections is adhered to. Once a data value is read, the address generator merely increments the address value. The last multiply/accumulate is performed outside the loop with a RND specification so that the result is properly formatted for output. This value is then transferred to the memory mapped I/O port called *da_data*. The physical addresses of the I/O ports must come from the architecture description.

Interrupt Return

A return from interrupt instruction, RTI, finishes the FIR routine so that the main routine execution can be properly resumed.

CONCLUSION

The ADSP-2100A can produce a compact solution to real-time digital filtering. With the aid of the ADSP-2100A Development System, microprocessor code is easily and quickly developed. Developing code for the ADSP-2101 is essentially identical but typically has a different I/O structure.

Power and Ground Connection Guidelines for Pin Grid Arrays

by Bob Fine

INTRODUCTION

As integrated circuits provide more functions and I/O performance, the number of pins for each device usually increases. Also, as the speeds of these devices increase, so do the current surges caused by the fast switching of internal transistors. These facts are evident in the highly integrated, high-speed devices that are available in pin grid array (PGA) packages with pin counts of several hundred pins.

To guarantee reliable operation of these devices in a circuit, special care must be used when connecting power and ground. This application note gives guidelines for printed circuit board connections of power and ground to devices in a pin grid array package.

BACKGROUND

High-speed integrated circuits which must drive data buses consisting of many data lines require high levels of power supply current surges as well as a low level of steady state current.

The high-level surges of power supply current are required for the bus driver circuits to charge any bus capacitance. This capacitance can be as large as 100pf per data line and the device could be driving thirty-two data lines. In a worst case scenario, where thirty-two data lines are changing from all logic '0's to all logic '1's, the capacitance on each data line must be charged. The current necessary to do this is supplied by the output drivers of the device. The bus capacitance will at first look like a short circuit to the output driver and a large amount of current will flow. As the bus capacitance quickly charges, the amount of current diminishes.

The low level of steady state current is required for the internal logic circuits. Since the internal bus capacitance of a digital device with very small geometry (such as a 1.0μm device) is relatively small, the current surges required are relatively small.

IDEAL POWER SUPPLY CONDUCTOR

Since an integrated circuit will require very quick surges of current, there is a potential for power supply problems. This current must be supplied through whatever conductor connects the device to the power supply. An ideal power supply conductor would have zero resistance and inductance and would carry the surges of current with no affects to the power supply voltage at the device. A non-ideal power supply conductor would pass these currents through some finite resistance and inductance and would change the level of the power supply voltage at the device.

A printed circuit board with a power and ground plane is the closest practical application of an ideal power supply conductor and is recommended for all high-speed digital circuits.

PIN GRID ARRAYS

Figure 1 shows a PGA footprint in a ground plane of a printed circuit board. The holes shown are drilled through the ground plane so that the pins of the PGA can be connected to etch or traces on other layers of the printed circuit board without being shorted to ground. The drilling of these holes results in the ground plane being perforated. This perforation divides the ground plane (or power plane) into two sections as shown in the figure. The inner ground plane is isolated from the outer ground plane by the perforated area.

The holes that are drilled through the ground plane reduce the amount of surface area. This reduction in surface area makes that section of the ground plane less ideal, possessing a larger amount of resistance and inductance. A model of this is shown in Figure 2. The perforated area now looks like a distributed resistance and inductance connecting the inner ground plane to the outer ground plane. Any two points (from point A to point B in Figure 2) in the outer ground plane have essentially no inductance or resistance between them and will be forced to be at the same voltage potential. Similarly, any two points (from point C to point D in Figure 2) in the inner ground plane will be at the same voltage level. From any point on the inside ground plane to any point on the outside ground plane (point A to point C in Figure 2), however, will be separated by the resistance and inductance of the perforated area.

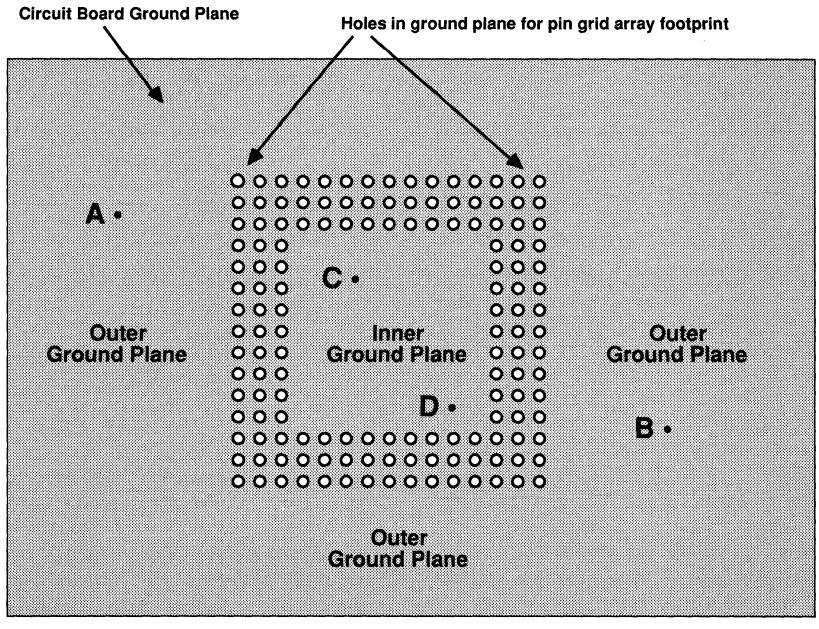


Figure 1. PGA Footprint in a Printed Circuit Board Ground Plane

Even though this resistance and inductance is small, the effects of high speed current surges can be large enough to cause the device to work improperly.

GROUND "BOUNCE"PROBLEMS

With the information discussed so far in mind, it becomes evident that any voltage potential developed across the distributed inductance and resistance of the model in Figure 2

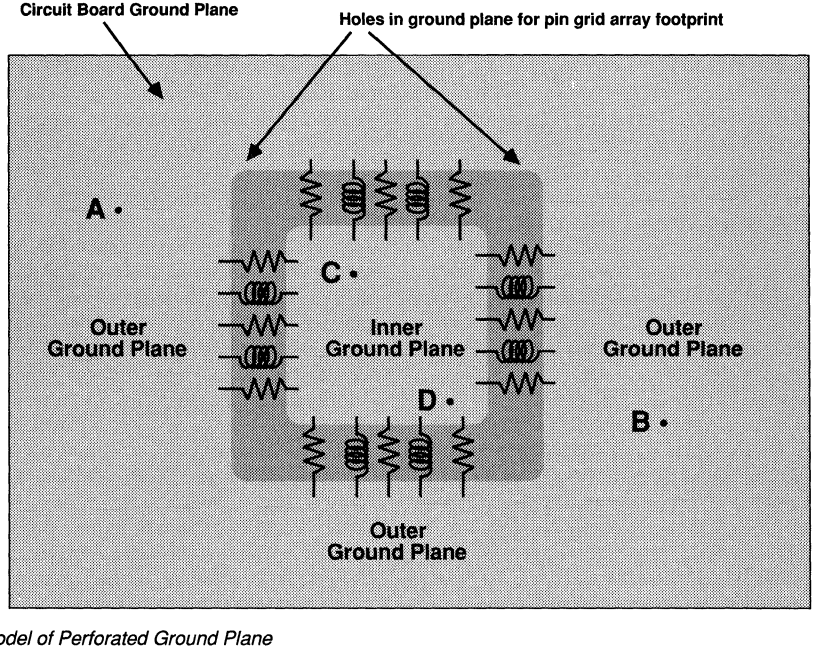


Figure 2. Circuit Model of Perforated Ground Plane

can raise the voltage potential at a logic ground pin of the device located in the inner ground plane. Since this voltage change is proportional to the level of any power supply current surges, the ground can "bounce." If input levels are referenced by the device to the logic ground, this bounce destroys the integrity of the input logic level and the device will not work correctly.

This can be seen more clearly by looking at the pin configuration of the ADSP-3128A Register File. Figure 3 shows this pin diagram with logic grounds and driver grounds called out separately. It can be seen that the inner ground plane connects directly to a driver ground pin in the lower left hand corner and an internal logic ground pin in the upper right hand corner.
ince these two pins are connected through the inner ground plane, they must stay at the same voltage potential. Remembering that the internal ground plane is connected to the outer ground plane through some inductance and resistance, it can be seen that current which flows from the outer ground plane through the inductance and resistance to the inner ground plane (Driver ground connection) will raise the potential of the inner ground plane. Since the internal logic ground pin is also connected to the inner ground plane, its potential will also change resulting in corruption of the logic reference.

THE SOLUTION

One solution would be to provide a short circuit around the inductance and resistance formed by the perforated section of ground plane. This could be accomplished by connecting a heavy wire from the inner ground plane to the outer ground plane. This solution would insure that the two ground planes stayed at the same voltage (hopefully true zero volts) but it may be impractical in the manufacture of a multi-layer printed circuit board.

Another solution is possible which deals with the fact that both the driver ground and the internal logic ground are connected to the internal ground plane. If the internal logic ground could somehow be isolated from the driver ground so that they were not forced to be at the same potential, the driver ground could "bounce" without affecting the internal logic ground. The only element that forces these two pins to be at the same potential is the direct connection to the internal ground plane. If the internal ground plane could be split, the connection would be broken.

Figure 4 shows the inner ground plane split. As the driver ground "bounces," it is isolated from the internal logic grounds through the resistance and inductance paths. The internal logic ground can remain at true zero volts while the "bounce" voltage potential is found across the isolating resistance and inductance.

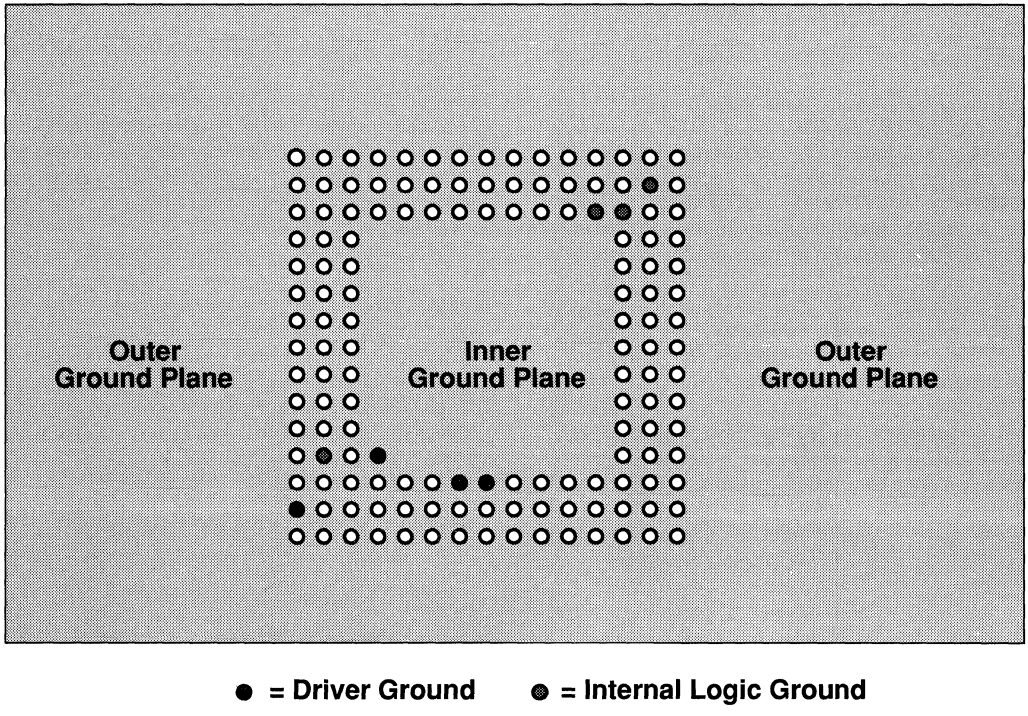


Figure 3. Ground pins of the ADSP-3128A Register File

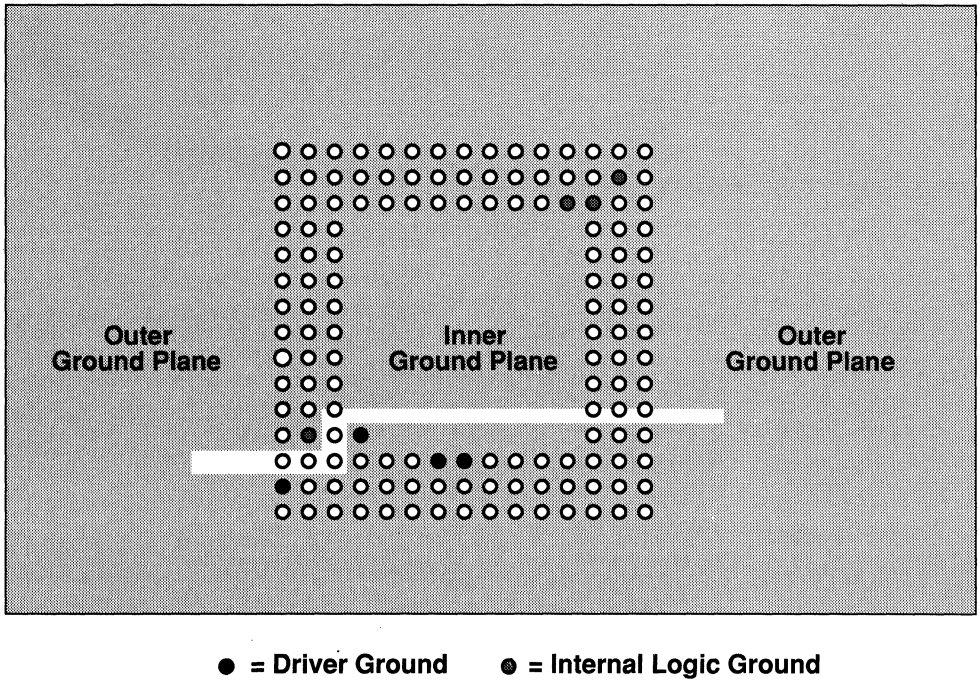


Figure 4. Split Inner Ground Plane for Isolation of Driver and Internal Logic Grounds

POWER PLANE AND BYPASSING

A similar splitting technique can be used to add extra margin against noise in the power plane also. In addition, adequate bypass and filter capacitors should be used between the power and ground plane. A 0.1 μf capacitor should be used to bypass high frequency power supply noise to ground. This capacitor should be placed close to the V_{dd} pin of the device and in the path of the power supply current. A larger capacitor in the range of 20 – 40 μf should also be used to act as a current reservoir for current surges. This capacitor should also be placed as close to the device as possible.

One word of caution when placing these capacitors. A capacitor will allow high frequency noise to pass through it. This is good when the noise is being bypassed to ground and is bad when the noise is being passed to an isolated area. When placing capacitors with the split ground plane, make sure that the capacitor does not add a noise path across any isolation barriers.

CONCLUSION

Even though high-speed digital devices such as the ADSP-3128A Register File will work fine with a solid power and ground plane and normal bypassing practices, following the techniques in this application note will assure an extra degree of immunity from large current surges and abrupt voltage changes.

Appendix
Contents

8

Technical Publications

Analog Devices provides a wide array of FREE technical publications, including Data Sheets for all products; Catalogs; Databooks; Application Notes and Guides; and a set of serial publications. In addition to the free publications, technical Handbooks are available at reasonable cost. Examples of our publications are described below.

DIGITAL SIGNAL PROCESSING
In addition to the DSP Application Notes included in this volume, the following publications are available:

ARTICLE REPRINT
"DSP Microcomputers Cut Pin Count and Retain Performance," by Greg Coker. *Electronic Products*, May 1, 1988 (42-47). The ADSP-2101 and ADSP-2102 do it with Harvard architecture, on-chip memory, and dual serial interfaces.

MANUALS (Obtain from your nearby sales office)
ADSP-2100 USER'S MANUAL—Introduction, Computational Units, Data Moves, Program Control, System Interface, Instruction Set Overview, Appendixes—162 pages.

ADSP-2100 CROSS-SOFTWARE MANUAL—Overview, System Builder, Assembler, Linker, Simulator, PROM Splitter, C Compiler, Instruction Set, Appendixes—240 pages plus Programmer's Reference Card.

ADSP-2100 APPLICATIONS HANDBOOK, Volume 1—Introduction, Fixed-Point Arithmetic, Floating-Point Arithmetic, Fixed-Coefficient Digital Filters, FFTs, Adaptive Filters, Image Processing, Linear Predictive Speech Coding, High Speed MODEM Algorithms, Bibliography—178 pages.

ADSP-2100 APPLICATIONS HANDBOOK, Volume 2—Overview, Graphics, Multirate Filters, PCM, ADPCM, Dual-Tone Multi-Frequency (DTMF)—248 pages.

ADSP-2101 USER'S MANUAL—Introduction, Computational Units, Data Moves, Program Control, Timer, Serial Ports, System Interface, Memory Interface, Instruction Set Overview, Appendices—184 pages.

WORD-SLICE® USER'S MANUAL (ADSP-1401/ADSP-1402 Program Sequencers and ADSP-1410 Address Generator)—Introduction; Program Sequencers: Internal Architecture, Jumps, Interrupt Processing, System Interface, Instruction Set; ADSP-1410: Internal Architecture, Addressing Operations, Precision Modes, System Interface, Instruction Set—218 pages.

NEWSLETTER: ***DSPatch***—a quarterly newsletter about digital signal-processing products and their applications. Subscriptions are free upon request.

OTHER TECHNICAL PUBLICATIONS
Brief descriptions of typical publications appear below. For copies of any item, to subscribe to any of our free serials, or to request any other publications, please get in touch with Analog Devices or the nearest sales office.

CATALOGS
CONVERSION PRODUCTS DATABOOK—1988. Data Sheets, Selection Guides, and Application Notes on D/A, A/D, V/F, and F/V Converters, Sample-Track/Hold Amplifiers, Voltage References, Multiplexers & Switches, Synchro-Resolver Converters, Data-Acquisition Subsystems, Application-Specific ICs.

LINEAR PRODUCTS DATABOOK—1988. Data Sheets, Selection Guides, and Application Notes on Op Amps, Instrumentation Amplifiers, Isolators, Multipliers/Dividers, Log/Antilog Amplifiers, RMS-to-DC Converters, Comparators, Temperature-Measuring Components and Transducers, Special Function Components, Digital Panel Instruments, Signal-Conditioning Components and Subsystems.

1988 SHORT-FORM DESIGNERS' GUIDE—Selection Guides and Other Information on Data Converters, Amplifiers, Analog Signal Processing, Transducers, Voltage References, Switches and Multiplexers, Data-Acquisition Subsystems, and Digital Signal-Processing Components—74 pages.

APPLICATION NOTES AND GUIDES
Application Notes. All are available individually upon request.

A/D Converters:
- "AD670 8-Bit A/D Converter Applications."
- "Interfacing the AD7572 to High Speed DSP Processors."
- "The AD7574 Analog-to-Microprocessor Interface."
- "The AD9502 Video Signal Digitizer and Its Application."

Amplifiers:
- "Applications of High Performance BiFET Op Amps."
- "How to Select Operational Amplifiers."
- "How to Test Basic Operational-Amplifier Parameters."
- "Low-Cost Two-Chip Voltage-Controlled Amplifier and Video Switch." (AD539)
- "Using the AD9610 Transimpedance Amplifier."

D/A Converters:
- "AD7528 Dual 8-Bit CMOS DAC."
- "Analog Panning Circuits Provide Almost Constant Output Power."
- "Changing Your VGA Design from a 171/176 to an ADV471."
- "CMOS D/A Converter Circuits for Single +5-Volt Supplies."
- "CMOS DACs and Operational Amplifiers Combine to Build Programmable-Gain Amplifiers." *In two parts.*
- "8th Order Programmable Low-Pass Filter Using Dual 12-Bit DACs."
- "Exploring the AD667 12-Bit Output Port."
- "14-Bit DACs (AD7534/AD7535) Maintain High Performance over Extended Temperature Range."
- "Gain Error and Tempco of CMOS Multiplying DACs."
- "Improved PCB layouts for Video RAM-DACs Can Use Either PLCC or DIP Package Types."
- "Interfacing the AD7549 Dual 12-Bit DAC to the MCS-48 and MCS-51 Microcomputer Families."
- "Microstepping Drive Circuits for Single-Supply Systems."

"Simple DAC-Based Circuit Implements Constant Linear Velocity (CLV) Motor Speed Control."
"Simple Interface Between D/A Converter and Microcomputer Leads to Programmable Sine-Wave Oscillator." (AD7542)
"The AD7224 DAC Provides Programmable Voltages over Varying Ranges."
"Video Formats and Required Load Terminations."

Resolver (Synchro) to Digital Conversion:
"Dynamic Characteristics of Tracking Converters."
"Dynamic Resolution-Switching on the 1S74 Resolver-to-Digital Converter."
"Why the Velocity Output of the 1S74 and 1S64 Series R/D Converters is Continuous and Step-Free Down to Zero Speed."

Sample-Holds:
"Applying IC Sample-Hold Amplifiers." (AD585)
"Generate 4 Channels of Analog Output Using AD7542 12-Bit D/A Converters and Control It All with Only Two Wires."

Switches:
"ADG201A/202A and ADG221/222 Performance with Reduced Power Supplies."
"Overvoltage Protection for the ADG5XXA Multiplexer Series."

V/F Converters:
"Operation and Applications of the AD654 IC V-to-F Converter."
"Analog-to-Digital Conversion Using Voltage-to-Frequency Converters."

Application Guides. All are available upon request.

Analog CMOS Switches and Multiplexers. A 16-page short-form guide to high-speed CMOS switches, CMOS switches with dielectric isolation, and CMOS multiplexers. Also included are reliability data and information on single-supply operation.

Applications Guide for Isolation Amplifiers and Signal Conditioners. A 20-page guide to specifications and applications of galvanically isolated amplifiers and signal conditioners for industrial, instrumentation, and medical applications.

CMOS DAC Application Guide 2nd Edition by Phil Burton (1986—63 pages). Introduction to CMOS DACs, Inside CMOS DACs, Basic Application Circuits in Current-Steering Mode, Single-Supply Operation Using Voltage-Switching Mode, The Logic Interface, Applications.

ESD Prevention Manual – Protecting ICs from electrostatic discharges. Thirty pages of information that will assist the reader in implementing an appropriate and effective program to assure protection against electrostatic discharge (ESD) failures.

High Speed Data Conversion – A 24-page short-form guide to video and other high-speed A/D and D/A converters and accessories, in forms ranging from monolithic ICs to card-level products.

RMS-to-DC Conversion Application Guide 2nd Edition by C. Kitchin and L. Counts (1986—61 pages). RMS-DC Conversion: Theory, Basic Design Considerations; RMS Application Circuits; Testing Critical Parameters; Input Buffer Amplifier Requirements; Programs for Computing Errors, Ripple, and Settling Time.

Surface Mount IC—A 280 page guide to ICs in SO and PLCC packages. Products include op amps, rms-to-dc converters, DACs, ADCs, VFCs, sample-holds, and CMOS switches.

SERIAL PUBLICATIONS

Analog Briefings—Newsletter: Current information about products for military/avionics and the status of reliability at ADI.

Analog Dialogue—Technical magazine: in-depth discussions of products, technologies, and applications.

BOOKS—Can be purchased from Analog Devices, Inc.; send check for indicated amount to One Technology Way, P.O. Box 796, Norwood, MA 02062. VISA charges are welcome; phone (617) 461-3392. If more than one book is ordered, deduct a discount of $1 from the price of each book.

ANALOG-DIGITAL CONVERSION HANDBOOK: Third Edition, by the Engineering Staff of Analog Devices, edited by Daniel H. Sheingold. Englewood Cliffs, NJ: Prentice-Hall (1986). A comprehensive guide to A/D and D/A converters and their applications. This third edition of our classic is in hardcover and has more than 700 pages, an Index, a Bibliography, and much new material, including: video-speed, synchro-resolver, V/F, high-resolution, and logarithmic converters, ICs for DSP, and a "Guide for the Troubled." Seven of its 22 chapters are totally new. $32.95

NEW—*DIGITAL SIGNAL PROCESSING IN VLSI*, by Richard J. Higgins. Englewood Cliffs, NJ: Prentice-Hall (1989). Six hundred pages of theory, hardware, software, and applications information for engineers and scientists who need to understand DSP algorithms and the special-purpose DSP hardware ICs and software tools developed to carry them out efficiently. To be published in mid-1989; call (617) 461-3392 for price and availability.

NONLINEAR CIRCUITS HANDBOOK: Designing with Analog Function Modules and ICs, by the Engineering Staff of Analog Devices, edited by Daniel H. Sheingold. Norwood MA: Analog Devices, Inc. (1974). A 540-page guide to multiplying and dividing, squaring and rooting, rms-to-dc conversion, and multifunction devices. Principles, circuitry, performance, specifications, testing, and application of these devices. Contains 325 illustrations. $5.95

SYNCHRO & RESOLVER CONVERSION, edited by Geoff Boyes. Norwood, MA; Analog Devices, Inc. (1980). Principles and practice of interfacing synchros, resolvers, and Inductosyns to digital and analog circuitry. $11.50

TRANSDUCER INTERFACING HANDBOOK: A Guide to Analog Signal Conditioning, edited by Daniel H. Sheingold. Norwood MA: Analog Devices, Inc. (1980). A book for the electronic engineer who must interface transducers for temperature, pressure, force, level, or flow to electronics, these 260 pages tell how transducers work—as circuit elements—and how to connect them to electronic circuits for effective processing of their signals. $14.50

8

Ordering Guide

INTRODUCTION

This Ordering Guide should make it easy to order Analog Devices DSP products, whether you're buying one multiplier or 1,000 each of 15 different items. It will help you:

1. Find the correct part number for the options you want.
2. Get a price quotation and place an order with us.
3. Know our warranty for components and subsystems.

For answers to further questions, call the nearest sales office (listed at the back of the book) or our main office in Norwood, MA, U.S.A. (617-329-4700).

MODEL NUMBERING

Many of the data sheets in the Databook have an Ordering Guide. Use it to specify the correct part number for the exact combination of options you want. Part numbers are created for our DSP products using this system:

The figure shows the form of model number used. It consists of an "ADSP" (Analog Devices Signal Processing) prefix, a 4-digit model number, an alphabetic performance/temperature-range designator and a package designator. An additional letter may immediately follow the digits ("A" for second-generation redesigned ICs). A suffix is used to indicate optional processing.

SECOND SOURCE

In addition to our many proprietary products, we also manufacture devices that are fit-, form-, and function-compatible (and often superior in performance and reliability) to popular products that originated elsewhere. For such products, we typically add the prefix "ADSP" to the familiar model number (example: ADSP-1016A).

ORDERING FROM ANALOG DEVICES

When placing an order, please provide specific information regarding model type, number, option designations, quantity, ship-to and bill-to address. Prices quoted are list; they do not include applicable taxes, customs, or shipping charges. All shipments are F.O.B. factory. Please specify if air shipment is required.

Place your orders with our local sales office or representative, or directly with our customer service group located in the Norwood facility. Orders and requests for quotations may be telephoned, sent via TWX or TELEX, or mailed. Orders will be acknowledged when received; billing and delivery information is included.

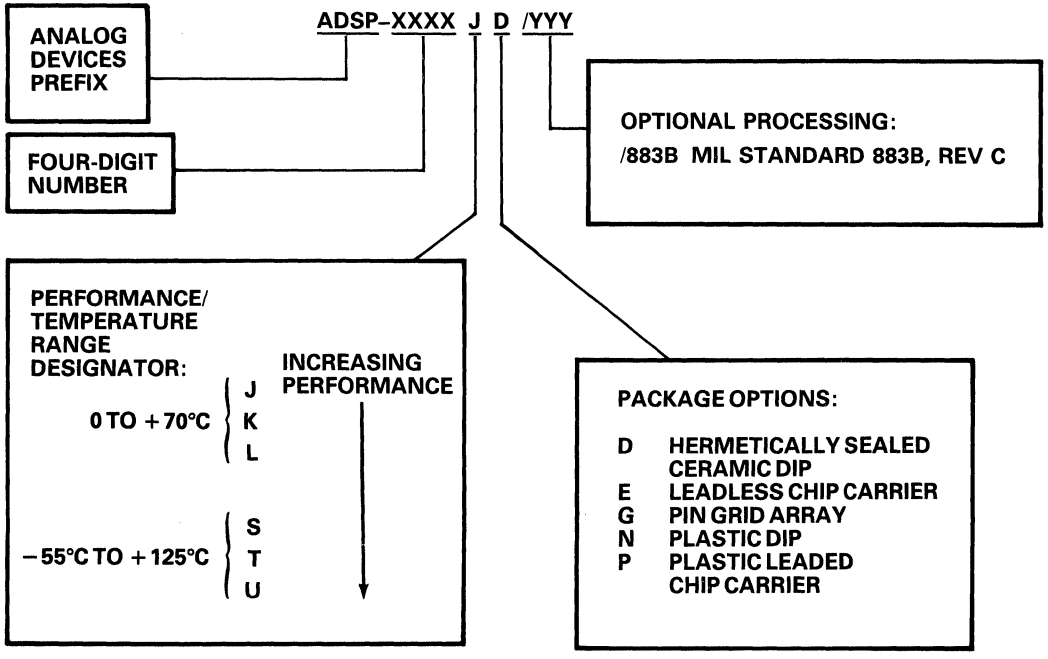
Payments for new accounts, where open-account credit has not yet been established, will be C.O.D. or prepaid. On all orders under fifty dollars ($50.00), a five-dollar ($5.00) processing charge is required.

When prepaid, orders should include $2.50 additional for packaging and postage (and a 5% sales tax on the price of the goods if you are ordering for delivery to a destination in Massachusetts).

WARRANTY AND REPAIR CHARGE POLICIES

All Analog Devices, Inc., products are warranted against defects in workmanship and materials under normal use and service for one year from the date of their shipment by Analog Devices, Inc., except that components obtained from others are warranted only to the extent of the original manufacturers' warranties, if any. This warranty does not extend to any products which have been subjected to misuse, neglect, accident, or improper installation or application, or which have been repaired or altered by others. Analog Devices' sole liability and the Purchaser's sole remedy under this warranty is limited to repairing or replacing defective products. (The repair or replacement of defective products does not extend the warranty period. This warranty does not apply to components which are normally consumed in operation or which have a normal life inherently shorter than one year.) Analog Devices, Inc., shall not be liable for consequential damages under any circumstances.

THE FOREGOING WARRANTY AND REMEDY ARE IN LIEU OF ALL OTHER REMEDIES AND ALL OTHER WARRANTIES, WRITTEN OR ORAL, STATUTORY, EXPRESS, OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.


ADSP-XXXX J D /YYY
ANALOG DEVICES PREFIX
FOUR-DIGIT NUMBER
OPTIONAL PROCESSING:
/883B MIL STANDARD 883B, REV C
PERFORMANCE/ TEMPERATURE RANGE DESIGNATOR:
INCREASING PERFORMANCE
0 TO +70°C
J
K
L
–55°C TO +125°C
S
T
U
PACKAGE OPTIONS:
D HERMETICALLY SEALED CERAMIC DIP
E LEADLESS CHIP CARRIER
G PIN GRID ARRAY
N PLASTIC DIP
P PLASTIC LEADED CHIP CARRIER

Worldwide Service Directory

North America

SERVICE DIRECTORY

Alabama
(205) 536-1506

Alaska
*(206) 575-6344
*(714) 641-9391

Arizona
(602) 949-0048
*(303) 590-9952

Arkansas
*(214) 231-5094

California
*(714) 641-9391
*(408) 559-2037
*(619) 268-4621

Colorado
(303) 443-5337
*(719) 590-9952

Connecticut
(516) 673-1900
*(617) 329-4700

Delaware
*(215) 643-7790

Florida
(407) 855-0843
(407) 724-6795
(813) 963-1076

Georgia
(404) 497-9404

Hawaii
*(714) 641-9391

Idaho
(303) 443-5337
*(303) 590-9952
*(206) 575-6344

Illinois
(312) 520-0710

Indiana
(317) 244-7867

Iowa
(319) 373-0200

Kansas
(913) 829-2800

Kentucky
(615) 459-0743
*(617) 329-4700

Louisiana
*(214) 231-5094

Maine
*(617) 329-4700

Maryland
*(301) 992-1994

Massachusetts
*(617) 329-4700

Michigan
(313) 559-9700
(616) 949-7400

Minnesota
(612) 835-2414

Mississippi
(205) 536-1506

Missouri
(314) 521-2044
(913) 829-2800

Montana
(801) 466-9336
*(714) 641-9391

Nebraska
(913) 829-2800

Nevada
(505) 828-1300
*(408) 559-2037
*(714) 641-9391

New Hampshire
*(617) 329-4700

New Jersey
(516) 673-1900
*(617) 329-4700
*(215) 643-7790

New Mexico
(505) 828-1300
*(303) 590-9952

New York
(516) 673-1900
(716) 425-4101

North Carolina
(919) 373-0380

North Dakota
(612) 835-2414

Ohio
(216) 248-4995
*(614) 764-8795

Oklahoma
*(214) 231-5094

Oregon
*(206) 575-6344

Pennsylvania
*(215) 643-7790
*(614) 764-8795

Rhode Island
*(617) 329-4700

South Carolina
(919) 373-0380

South Dakota
(612) 835-2414

Tennessee
(205) 536-1506
(615) 459-0743

Texas
*(214) 231-5094

Utah
(801) 466-9336
*(303) 590-9952

Vermont
*(617) 329-4700

Virginia
*(301) 992-1994

Washington
*(206) 575-6344

West Virginia
*(614) 764-8795

Wisconsin
(414) 784-7736

Wyoming
(801) 466-9336

Puerto Rico
*(617) 329-4700

Canada
(416) 821-7800
(613) 729-0023
(514) 697-0804
(604) 941-7707

Mexico
*(617) 329-4700

International

CALL

Australia
(02)4383900

Austria
*(222)885504-0

Belgium
*(3)2371672

Brazil
(11)531-9355

Denmark
*(2)845800

Finland
(0)8041041

France
*(1)46662525
*(76)222190
*(61)408562
*(99)535200

Holland
*(1620)81500

Hong Kong
(5)8339013

India
(212)53880
(11)6862460
(812)560506

Ireland
*(932)253320
(United Kingdom Sales)

Israel
*(052)911415
*(052)913551

Italy
*(2)6883831
*(6)8393405
*(11)6504572
(2)9520551
(51)555614
(49)633600
(6)390083
(11)599224
(55)894105

Japan
*(3)2636826
*(6)3721814

Korea
(2)7841144

Malaysia
(65)2848537

Mexico
(83)351721
(83)351661

New Zealand
(9)592629

Norway
(3)847099

People's Republic of China – Beijing
(1)890721, Ext. 120

Romania
*(222)885504
(Austria)

Singapore
(65)2848537

South Africa
(11)882-1620

Spain
(1)7543001
(3)3007712

Sweden
*(8)282740

Switzerland
*(22)315760
*(1)8200102

Taiwan
(2)501-8170

Turkey
(1)3372245

United Kingdom
*(932)232222
*(932)253320
(Sales)
*(01)9411066
*(635)35335
*(506)30306
*(021)5011166
*(0279)418611

United States of America
*(617) 329-4700

West Germany
*(89)570050
*(4181)8051
*(721)48567
*(30)316441
*(221)686006

Yugoslavia
*(222)885504
(Austria)

8

*Analog Devices, Inc. Direct Sales Offices

WORLDWIDE HEADQUARTERS
One Technology Way, P.O. Box 9106, Norwood, Massachusetts 02062-9106 U.S.A.
Tel: (617) 329-4700, TWX: (710) 394-6577; FAX: (617) 326-8703, Telex: 924491
Cable: ANALOG NORWOODMASS

Product Index

WORLDWIDE HEADQUARTERS

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106 U.S.A.
Tel: (617) 329-4700, Twx: (710) 394-6577, Telex: 924491, Cable: ANALOG NORWOODMASS

PRINTED IN U.S.A.

G1125a–75–4/89