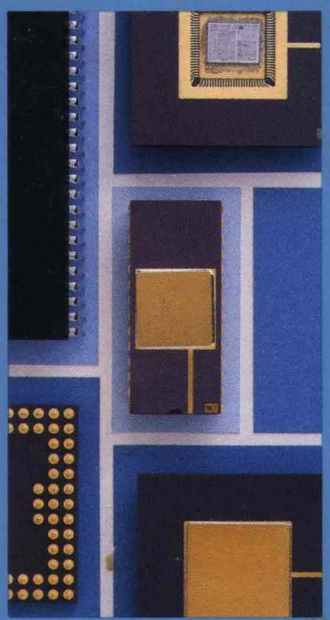
ANALOG DEVICES



DSP PRODUCTS DATABOOK

DSP MICROPROCESSORS
MICROCODED SUPPORT COMPONENTS
FLOATING POINT COMPONENTS
FIXED POINT COMPONENTS

DSP PRODUCTS DATABOOK 1987

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 1–5 and 1–6. 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
Data Acquisition Subsystems
V/F Converters
F/V Converters
Sample-Track/Hold Amplifiers
Voltage References
Multiplexers and Switches
Synchro/Resolver Converters
Application Specific ICs
Power Supplies

LINEAR PRODUCTS DATABOOK

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

1987 DSP PRODUCTS DATABOOK

General Information
Contents

1

Introduction

ANALOG DEVICES

Analog Devices is the industry's leading supplier of high-performance signal processing integrated circuits. Since 1965, when we introduced innovative modular op amps, we have consistently advanced the state of the art in components technology. Today, with 80% of our sales generated by integrated circuits, we have the industry's most widely recognized line of advanced signal processing chips. Our strengths include proprietary wafer fabrication processing technologies, outstanding product quality, strong internal manufacturing, and innovative, high-performance product architectures.

THE GROWING IMPORTANCE OF DSP

While signal processing was an exclusively analog phenomenon in the early 1970s, it is increasingly performed in the digital domain. Signals from the real world are digitized and fed to a digital signal processing (DSP) subsystem where fast arithmetic processors allow operations such as filtering and spectral analysis to be performed at real-time rates.

THE ROLE OF DSP AT ANALOG DEVICES

As the leader in signal processing circuit technology, Analog Devices was quick to recognize the important opportunities made possible by DSP. Our customer base spans a wide area, including instrumentation, defense/avionics, scientific computing, industrial, medical, and telecommunications. These areas have in common the need to continually advance the performance of their systems. Increasingly they have turned to fast number-crunching digital hardware to complement their analog processing. A key strategy of Analog Devices is to offer customers the industry's best family of high-performance signal processing devices – ranging from those strictly in the analog domain, to converters, to high-speed digital processors.

THE DSP DIVISION

Analog Devices was first in the industry to bring CMOS semiconductor technology to the area of fast arithmetic processors. In 1983, our DSP Division introduced a set of seven CMOS fixed-point multipliers that matched the speed of bipolar alternatives, while cutting power requirements by a factor of twenty. This breakthrough redirected the focus of the industry from bipolar to CMOS for high-speed VLSI circuits.

Our innovations in CMOS wafer fabrication technology continued as we pushed processing geometries from the 5μm level down to 1.5μm. In parallel, we brought out a complete building block family of VLSI processors for high-end DSP and numeric processing systems including a 64-bit IEEE floating-point chipset, the ADSP-3210 and ADSP-3220. We again advanced the state of the art in DSP with the introduction of the industry's first microprocessor optimized for DSP, the ADSP-2100.

ANALOG DEVICES' DSP CAPABILITIES

Currently the DSP Division is in production with a 1.5μm double-layer metal CMOS process. We are now moving to a 1.0μm CMOS process, continuing our innovation in process technology. Analog Devices continues to develop advanced processes (such as a specialized bipolar process and gallium arsenide) both internally and through strategic investments. For example, we recently introduced our first bipolar DSP product, a 16 × 16 multiplier that operates at cycle times in excess of 50MHz.

During 1986 we brought on-line a new VLSI wafer fabrication module to handle the volumes and advanced geometries of our DSP product line. Our assembly facilities include factories in Wilmington, Massachusetts and the Philippines. Our VLSI test capability is located in Norwood, Massachusetts, the Division's headquarters.

Analog Devices supports its DSP products with a technically strong direct salesforce and readily available applications assistance. Included in our applications support is a quarterly newsletter – *DSPatch* – that brings you up-to-date applications information on our products and on the general field of DSP. If you're not already receiving *DSPatch*, call or write us.

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, 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. More than 20 of our products—both proprietary and second-source—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.

Our PLUS program makes available standard devices (commercial and industrial grades, plastic or ceramic packaging) for *any* user with demanding application environments, at a small premium. Subjected to stringent screening, similar to MIL-STD-883 test methods, they are often suffixed "/+" and are available from stock.

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 price lists and/or quotations are available upon request from our sales offices.

Table of Contents

DSP Microprocessors
Contents

2

The ADSP-2100 is a single-chip processor optimized for digital signal processing (DSP) and other high-speed numeric processing applications. The chip has many special features which facilitate a fast design cycle. The ADSP-2100 offers:

1. **Easy-to-Attain High Performance**
   The ADSP-2100 integrates arithmetic/logic unit(ALU), multiplier/accumulator(MAC), barrel shifter, data address generators, and a program sequencer in a single device. Its architecture offers single-cycle access to both the external program and data memories. The resulting architecture combines the functions and performance of a bit-slice/building block system with the ease-of-design and development of a general-purpose microprocessor.
2. **Easy-to-Understand Instruction Set**
   The ADSP-2100 instruction set uses an algebraic syntax, similar to high-level languages, making it easy to write and understand code for the processor. This results in easier and faster code development and maintenance.
3. **Easy-to-Use Development Tools**
   The complete set of development tools available for the ADSP-2100 (Cross-Software, Simulator, In-Circuit Emulator, and Evaluation Board) minimizes both design time and effort. Your application is up and running faster with this powerful development system.
4. **Easy-to-Design System Interface**
   The advanced design of the ADSP-2100 allows simple interconnection of memories and I/O devices, minimizes the external logic required to handle interrupts, and supports straightforward host interface and multiprocessing of multiple ADSP-2100s.

The ADSP-2100 is available in 6 or 8MHz versions and is fabricated in a high-speed 1.5 micron double-layer metal CMOS process. It dissipates less than 600mW. The part is available in both commercial and military versions.

Faster versions of the ADSP-2100 are planned. Check with your local sales office for current information.

Some of the applications for which the ADSP-2100 has been designed-in are image processing, speech processing, high-speed modems, telecommunications, radar, sonar, graphics, and numerical processing.

ADSP-2100 Benchmarks

Algorithm	Performance @ 8MHz
FIR Filter	125ns per Tap (1 Cycle per Tap)
Complex FIR Filter	500ns per Tap (4 Cycles per Tap)
Biquad Filter Section	875ns per Section (71 Cycles per Section)
Lattice Filter Section	625ns per Section (5 Cycles per Section)
1024-Point Complex FFT (Radix-2)	6.61ms
4096-Point Complex FFT (Radix-2)	33.3ms

ADSP-2100 Features and Benefits

Feature	Benefit
Separate Program and Data Buses	Efficient Data Transfer, in Parallel with Computations
Dual Purpose Program Memory for Both Instruction and Data Storage	Dual Operand Fetch in One Cycle
Single-Cycle Direct Access to Both Program and Data Memory	No Time Penalty for Off-Chip Memory Access
Single Cycle Instruction Execution with 125ns Cycle Time	High Throughput
Multifunction Instructions	High Degree of Parallelism
Addresses 16K × 16 of Data Memory and 16K × 24 of Program Memory (a 16K × 24 Expansion Available for Data)	Room for Complex Programs and Up to 32K of Data
Three Independent Computational Units (ALU, MAC, Barrel Shifter)	Programming Flexibility; Output of Any Computational Unit Can Be Input to Any Computational Unit
Two Independent Data Address Generators	Dual Operand Fetch in One Cycle; No Overhead Modulo Addressing; Bit-Reversing
Powerful Program Sequencer	Zero-Overhead Looping and Single-Cycle Conditional Branches
Internal Instruction Cache	Three Bus Performance During Program Loops
Provisions for Multiprecision Computation and Saturation Logic	Processing Flexibility
Four External Interrupts	Easy System Design; Minimizes External Logic
Complete Development System	Ease of Program Development
1.5μm CMOS	High-Speed and Low-Power Dissipation
Simple Bus Interface	Simplifies Interconnections of Memories and I/O Devices
Bus Request and Bus Grant Signals	Supports Multiprocessing of Several ADSP-2100s and Easy Host Interface
Background Registers on All Computational Units	Fast Context Switching for Ease of Interrupt Handling

ADSP-2100 Development System

FEATURES

Cross-Software Modules:
System Builder
Assembler
Linker
PROM Splitter
Simulator
Stand-Alone In-Circuit Emulator

Cross-Software Available for Use on
IBM PC† Under PC-DOS†
DEC VAX‡ Under VMS‡
Macros for Modular Code Development
Interactive and Symbolic User-Friendly Interface
Stand-Alone Evaluation Board

2

GENERAL DESCRIPTION

The ADSP-2100 Development System is a complete set of development tools for systems using the ADSP-2100 DSP microprocessor. This powerful and easy-to-use set of tools is extremely valuable in implementing an ADSP-2100 system design: the software tools shorten the software design cycle and the Emulator and Evaluation Board facilitate the debug cycle.

The ADSP-2100 Development System includes:

- A complete Cross-Software System consisting of five modules:
  - System Builder – to define the target hardware environment
  - Assembler – to assemble code/data modules
  - Linker – to link separately assembled modules
  - PROM Splitter – to generate PROM burner compatible files
  - Simulator – to perform an instruction-level software simulation
- A stand-alone in-circuit hardware Emulator
- A stand-alone Evaluation Board

With the Cross-Software System, the user defines the target system, writes a program using the ADSP-2100 Assembly Language, assembles each program module; links all modules to form a running system, simulates execution of the code, and downloads the program into PROMs for hardware implementation. VAX VMS and IBM PC versions of the Cross-Software are available. The System Builder, Assembler, Linker, and PROM Splitter for the IBM PC are sold together in a single package, with the Simulator available separately. The VAX VMS version of the Cross-Software System is sold in a single package.

The In-Circuit Emulator provides the user with complete monitoring and control capabilities for debugging code in the actual target system. With its pod connected to the target system's ADSP-2100 socket, the Emulator operates at the processor's full cycle rate. Through the Emulator's RS-232 connections, the user can also download program files created with the Cross-Software Tools from the host computer system.

The ADSP-2100 Evaluation Board is an additional tool for evaluating the processor in real time and an aid to the development process. This board also provides analog I/O and an expansion port for user prototyping.

ORDERING INFORMATION

Refer to Development System Selection Guide on page 2-13.

†IBM PC and PC-DOS are registered trademarks of International Business Machines Corp.
‡DEC VAX and VMS are trademarks of Digital Equipment Corp.

ADSP-2100 Cross-Software Modules

FEATURES

System Builder
- *Allows the User to Specify Target Hardware*

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

Linker
- *Supports Multi-Module Linking*
- *Maps Assembler Output to Target Hardware*

PROM Splitter
- *Formats the ROM Memory Image for Uploading to PROM Burners*

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 Modules allow the user to easily develop applications software for implementation on an ADSP-2100 system. The Cross-Software Modules include the following:

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 ADSP-2100 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 ADSP-2100 System Builder then translates this into an Architecture Description File which is read by the other ADSP-2100 Cross-Software Modules.

Assembler

The ADSP-2100 Assembler translates user-written source code modules into relocatable object-code modules. The user creates an assembler source code module by writing a program 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 ADSP-2100 Linker.

Linker

The ADSP-2100 Linker generates a complete executable program, the Program Memory/Data Memory Image File, by linking together object-code modules which were assembled separately. This output file is used by the Simulator, the PROM Splitter, and the Emulator. 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 the user.

PROM Splitter

The ADSP-2100 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. The PROM image file is generated in either Motorola S Record, Intel Hex Record, or Daisy VLA format.

Simulator

The ADSP-2100 Simulator simulates the operation of the ADSP-2100 and allows the user to observe the contents of the 2100's registers, buses, stacks and program and data memories as a program is being executed. The Simulator is user-friendly, interactive, and screen-oriented. By utilizing the Architecture Description File output of the System Builder, the Simulator configures itself to match the target system hardware. The Simulator supports full symbolic disassembly via the Debug Symbol Table File output of the Linker.

The Simulator supports three execution modes:

- Emulator Mode, which runs at the fastest simulation speed and updates the screen every 256 cycles.
- Extend Mode, which updates the screen every cycle during program execution.
- Single Step Mode, which executes a single instruction per run command.

The Simulator has six major display modes:

- Register Display, which displays the contents of the ADSP-2100 primary and alternate registers.
- Program Memory Display
- Data Memory Display
- Stack Display, which displays the contents of the ADSP-2100's program sequencer stacks.
- Trace Buffer Display, which displays the past external bus states of the ADSP-2100.
- Cache Memory Display

In addition, the Simulator allows you to:

- Modify the contents of Registers, Program or Data Memory, or the Program Counter
- Set break points in program memory
- Set watch points in data memory
- Utilize command files
- Display user-defined addresses or values symbolically
- Save and restore the state of the Simulator
- Dump program and data memory contents to files
- Patch, delete and execute code
- Utilize decimal or hexidecimal numeric format
- Plot the contents of data memory on the terminal screen

ADSP-2100 Emulator

FEATURES

- *Performs In-Circuit Emulation*
- *Operates at the Full Clock Rate of the ADSP-2100 (8MHz)*
- *Self-Emulation of Processor*
- *Same Interactive, Symbolic User Interface as the ADSP-2100 Simulator*
- *Multiple-Run Modes – Full Speed, Extend Mode or Single-Step*
- *Supports Breakpoints*
- *User Selectable Program Memory Source – Emulator or Target System*
- *User Selectable System Clock Source – Emulator, Target System, or External*
- *Two RS-232-C Connectors for Interfacing to a Host System and Terminal*
- *Optional Trace Board*

GENERAL DESCRIPTION

The ADSP-2100 Hardware Emulator is an In-Circuit Emulator which allows the user to debug code developed with the ADSP-2100 Cross-Software Modules in the actual target system. The Emulator, which utilizes an ADSP-2100 to self-emulate the processor, plugs into the target system's ADSP-2100 socket and operates at the ADSP-2100's cycle time.

The Emulator supports three execution modes:

- Emulator Mode, which runs at full processor speed.
- Extend Mode, which updates the screen every cycle during program execution.
- Single-Step Mode, which executes a single instruction per carriage return.

The Emulator has five major display modes:

- Register Display, which displays the contents of the ADSP-2100 primary and alternate registers.
- Program Memory Display
- Data Memory Display
- Stack Display, which displays the contents of the ADSP-2100's program counter stack.
- Trace Buffer Display (for optional Trace Board), which displays the past external bus states of the ADSP-2100.

With the same interactive, symbolic user interface as the ADSP-2100 Simulator, the Emulator allows the user to:

- Modify the contents of Registers, Program or Data Memory, or the Program Counter
- Set breakpoints in Emulator-based program memory
- Display user-defined addresses or values symbolically
- Patch, delete and execute code
- Utilize decimal or hexidecimal numeric format

In addition, the Emulator allows you to:

- Specify the baud rate and parity settings required by the user's terminal

- Activate or deactivate the Emulator Pod through software control
- Select the program memory source from either the Emulator's internally available Program Memory RAM or the target system's Program Memory
- Download files from a host system
- Select the system clock source from either the Emulator's internal clock, the target system's clock or an external clock

ADSP-2100 Evaluation Board

FEATURES

Stand-Alone ADSP-2100 System
125ns Cycle Time Operation
4K × 24 of Program Memory (with Sockets for Expansion to 32K)
2K × 16 of Data Memory (with Sockets for Expansion to 16K)
Analog Interface via A Bidirectional Codec Channel
Undedicated 12-Bit D/A Converter
Microphone and Speaker Connections for Audio Applications
Four BNC Connectors Interfacing to External Instrumentation
Expansion Port for Customization by User
Same Interactive, Symbolic User Interface as the ADSP-2100 Emulator and Simulator
Multiple Run Modes – Full Speed, Extend Mode or Single-Step
Supports Breakpoints
Two RS-232C Connectors for Interfacing to a Host System and Terminal

GENERAL DESCRIPTION

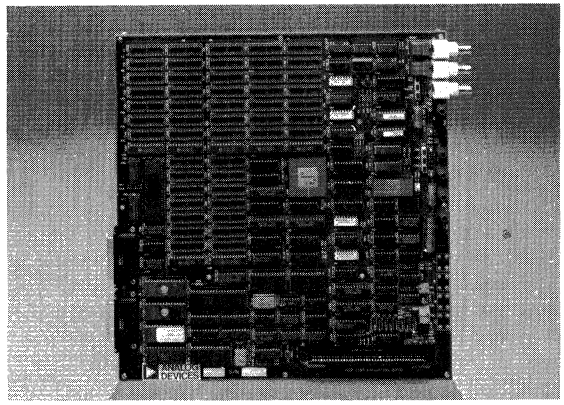
The ADSP-2100 Evaluation Board is an easy-to-use, low-cost way for evaluating the ADSP-2100 DSP Microprocessor in real-time applications. The ADSP-2100 Evaluation Board can play several different roles over the course of an ADSP-2100 design cycle:

- **As a Real-time Demonstration System**
  The Evaluation Board allows the user to observe the ADSP-2100's real-time performance in executing standard DSP benchmarks.
- **As a Real-time Evaluation System**
  The Evaluation Board can be utilized prior to design of proprietary hardware for the real-time execution of user-developed application routines.
- **As a Simulation Accelerator**
  ADSP-2100 application code can be executed in real time for increased productivity of software developers.

The ADSP-2100 Evaluation Board is a stand-alone system and consists of an ADSP-2100 Digital Signal Microprocessor, 4K × 24 of Program Memory, and 2K × 16 of Data Memory. Additional program and data memory can be added as desired up to the full 32K program and 16K data memory address space via factory installed sockets.

A bidirectional codec channel and an undedicated 12-bit linear D/A converter allow for processing of real-world signals, and a prototyping expansion bus allows for customization to reflect the eventual hardware environment. In addition, four BNC Connectors are available for interfacing to external instrumentation. An integral microphone jack and input pre-amplifier, along with a speaker jack and output amplifier, allow for easy implementation of speech and telecommunication applications.

The Evaluation Board's ADSP-2100 runs under the control of an onboard host processor enabling the user to access a variety of powerful debugging tools. Interfacing to an external host computer system running the ADSP-2100 Cross Software (available in VAX/VMS and PC-DOS versions), the Evaluation Board serves as a valuable real-time development tool. The Evaluation Board connects to a terminal and host computer via two RS-232C serial connectors.



Utilizing the same user interface as the ADSP-2100 Emulator and Simulator, the Evaluation Board supports three execution modes:

- Emulator Mode, which runs at full processor speed.
- Extend Mode, which updates the screen every cycle during program execution.
- Single-Step Mode, which executes a single instruction per carriage return.

The Evaluation board has four major display modes:

- Register Display, which displays the contents of the ADSP-2100's primary and alternate registers.
- Program Memory Display.
- Data Memory Display.
- Stack Display, which displays the contents of the ADSP-2100's program counter stack.

The Evaluation Board allows the user to:

- Download ADSP-2100 Cross-Software generated executable code from a host computer.
- Modify the contents of Registers, Program or Data Memory, or the Program Counter.
- Set breakpoints in Program Memory.
- Display user-defined addresses or values symbolically.
- Patch, delete, and execute code.

The Evaluation Board requires an external ±12 and +5 volt power supply (not included).

ADSP-2100 User's Manual

This manual provides the information necessary for an engineer to understand the operation of the ADSP-2100 and to write programs for the ADSP-2100. Together with the ADSP-2100 data sheet, the manual provides all the information required to design a hardware system with the ADSP-2100.

Chapter	Topic	Summary
1	Overview	Overview of the ADSP-2100 and its development system.
2	Internal Architecture	Describes the overall architecture and functional relationships of the major units. The internal operation of each functional unit is described in detail: the arithmetic/logic unit (ALU), the multiplier/accumulator (MAC), the barrel shifter, the data address generator, the program sequencer, the instruction cache and the program-data (PMD-DMD) bus exchange.
3	System Interface	Discusses all major interfaces to the ADSP-2100: the program memory (PM) interface, the data memory (DM) interface, the control interface and the interrupt lines. This chapter gives a functional description of the interfaces and their sequences of operation. For actual timing parameters, refer to the ADSP-2100 data sheet. The pin descriptions are given in this chapter.
4	Instruction Set	Organized as a reference section. All instructions are grouped by major type. These major groups (for example, ALU, MAC, or MULTIFUNCTION) are used as thumb heads in this chapter. The operation and assembly syntax of each instruction is fully described.
A	Instruction Codes	Shows the actual 24-bit instruction coding of each instruction.

ADSP-2100 Cross-Software Manual

This manual guides the engineer in the use of the ADSP-2100 Cross-Software on either the VAX/VMS, or the IBM PC or PC compatible.

Chapter	Topic	Summary
1	Overview	Overview of the ADSP-2100 Cross-Software System.
2	System Builder	Describes the System Builder, showing its input and output files and describing its syntax, operation, and error messages.
3	Assembler	Describes the Assembler, showing its input and output files and describing the Assembler directives, operation and error messages.
4	Linker	Describes the Linker, showing its input and output files and describing its operation and error messages.
5	Simulator	Describes the Simulator. The complete set of interactive commands and typical displays is shown. The Simulator provides on-line help for additional help.
A	File Format	Contains the file format for all input and output files used by the Cross-Software System.
B	System Requirements	List the hardware and software requirements for the computer systems that can host the Cross-Software System.

ADSP-2100 Emulator Manual

This manual guides the engineer in the use of the ADSP-2100 Emulator with the target hardware designed for the specific application.

Chapter	Topic	Summary
1	Overview	Overview of the ADSP-2100 Emulator.
2	Installation	Describes the installation procedure and I/O connections (RS-232 and BNC).
3	Configuration	Describes the initial configuration of the Emulator such as setting the baud rates for communications and activating the ADSP-2100 processor.
4	Operation	Describes the Emulator's basic operation and error messages, highlighting the differences between the Emulator and Simulator.
5	Development Examples	Contains step-by-step examples of Emulator sessions.
A	Specifications	Contains the specifications of the Emulator.
B	Timing Comparison	Compares the timing of the ADSP-2100 in the Emulator with that of an ADSP-2100 running in a non-Emulator system.
C	Accessing more than 16K of Target-Based Program Memory	Tells how to use the Emulator with more than 16K of program memory in your target system.
D	Pin Diagrams	Shows the pin arrangement of the ADSP-2100.
E	Replace Emulator Hardware	Describes how to replace Emulator PROMS, pod board and ADSP-2100 chip.
F	Terminal Emulation Software	Provides information about specific terminal emulation software that can be used with the emulator

ADSP-2100 Applications Handbook
Volume 1

The ADSP-2100 Applications Handbook covers a wide variety of examples of numerical and digital signal-processing algorithms. Each Chapter contains an overview of the chapter topic and then follows with specific examples, including complete ADSP-2100 programs.

Chapter	Topic	Summary
1	Introduction	Introduction to the Handbook.
2	Fixed Point Arithmetic	Describes using the ADSP-2100 for fixed-point single and multiprecision arithmetic.
3	Floating Point Arithmetic	Covers block floating point and full floating point arthitmetic, and fixed-to-floating and floating-to-fixed conversions.
4	Function Approximation	Covers function approximation, including sine, arctangent, square root, and logarithm. Uniform random number generation is also described.
5	Digital Filters	Describes fixed coefficient digital filters including single and double precision FIR filters, two-dimensional FIR filters, and complex FIR filters. Also covered are direct form and cascaded biquad IIR filters and lattice filters.
6	FFTs	Covers Fast Fourier Transforms, both radix-2 and radix-4. There are also examples of both decimation-in-time and decimation-in-frequency algorithms. Windowing, bit reversal and magnitude and phase are also described.
7	Adaptive Filters	Describes adaptive filtering using a single- or double-precision stochastic gradient algorithm.
8	Image Processing	Describes several image processing algorithms including two-dimensional convolutions, matrix multiplications and histograms.
9	Speech Algorithms	Describes a number of linear predictive speech coding algorithms including linear predictive coding, auto and cross correlation, Levinson recursion, and pitch detection. A linear predictive coding synthesizer is also described.
10	Modems	Describes several algorithms used with high-speed modems including the complex-valued gradient and the weighted Euclidean Distance.

ADSP-2100 Training Course

This intensive three day workshop familiarizes participants with the features and capabilities of the ADSP-2100 digital signal microprocessor and its associated development tools. Each topic is discussed thoroughly and illustrated with practical examples. Hands-on laboratory exercises with individualized instruction are an integral part of the workshop. Because of the intensive nature of the workshop, participants will gain the most from it if they are already familiar with the general concepts of programming in any high level or assembly language.

The workshop begins with an introduction to the system architecture and terminology, followed by an in-depth discussion of the major functional units of the processor. Next, interfacing the ADSP-2100 to other circuits is considered. The instruction set is described in detail, with an opportunity to use the Simulator to observe the data flow and operating units. Several examples of arithmetic using different data formats help illustrate simple ADSP-2100 programs.

A large portion of the Workshop is devoted to seeing how the ADSP-2100 can be used in actual applications. Examples used include several types of filters, Fast Fourier Transforms and matrix operations. Although the course does not teach digital signal processing theory, each example is presented by first reviewing the general features of the algorithm, discussing its programming requirements, and finally tracing the operation of a sample program using the Simulator. After examining the available development tools, participants will have an opportunity to experiment with the programming of the ADSP-2100 either with predefined exercises or with problems they have brought to class.

This workshop will be a detailed and practical experience for anyone who will use the ADSP-2100. The price includes tuition, workshop materials, and the use of the development stations for laboratory exercises. Class size is limited to a maximum of fifteen to facilitate individual instruction and a practical hands-on experience.

Please call the Analog Devices Training Coordinator at (617) 461-3622 for an current schedule of workshops, a complete workshop brochure, and pricing.

ADSP-2100 Software Model for Board-Level Simulation

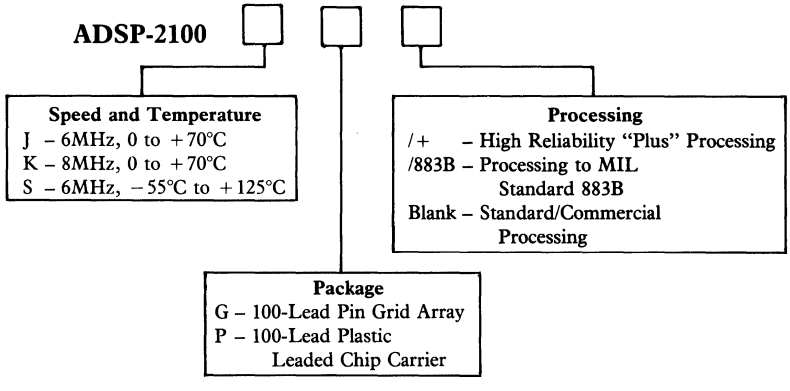
A behavioral model of the ADSP-2100 for use in software simulations of board or system-level products is available from Logic Automation of Beaverton, Oregon. The behavioral model, called a SmartModel*, permits engineers to analyze their designs at a system level with a workstation and a logic simulator. The ability to proceed directly from the simulated design to the final product eliminates costly and time-consuming prototypes and multiple design revisions. Every SmartModel has extensive error checking capabilities. When it detects an error during a simulation, the model describes the type of error and when it occurred. Analog Devices has provided technical data on the ADSP-2100 to Logic Automation. The model has been validated with Analog Devices' production test vectors, insuring its correctness. SmartModels work on a variety of simulators, including Mentor Graphics. For more information, contact Logic Automation, Inc., Sales Engineering, P.O. Box 310, 19545 N.W. Von Neuman Dr., Beaverton, Oregon 97075, (503) 690-6900.

Board Level ADSP-2100 Products

The Industrial Automation Division of Analog Devices offers two board level products which incorporate the ADSP-2100. The RTI-680 is a VME board and the RTI-980 is a Multibus II board. Both boards are programmable array processors and are designed to facilitate the implementation of signal processing algorithms. Development tools are available for each board. For more information, contact Industrial Automation Division Marketing, (617) 329-4700.

***SmartModel is a trademark of Logic Automation.**

ADSP-2100
Use the following guide to select the version of the ADSP-2100 which is right for your application.



Note: Extended temperature range parts (S grade, –55°C to +125°C) and MIL Standard 883B parts are available only in pin grid array packages (G packages).

Examples:

ADSP-2100KG
An 8MHz part, specified for performance at 0 to +70°C, packaged in a 100-lead pin grid array, with standard commercial processing.

ADSP-2100SG/883B
An 6MHz part, specified for performance at –55°C to +125°C, packaged in a 100-lead pin grid array, and subject to MIL Standard 883B processing.

Ordering Cross-Software:
For a VAX – order ADDS-2110.

For a PC or compatible – order both ADDS-2121 and ADDS-2122.

For information on Cross-Software for other Operating Systems, please contact the factory.

Ordering an Emulator or Evaluation Board:
You will need either the ADDS-2100 or ADDS-2121 Cross-Software with your Emulator or Evaluation Board. If you have already purchased one of these packages, you need not order another.

ADSP-2100 Development System

Model No.	Description	Price
ADDS-2110	Cross-Software for DEC VAX/VMS Computers (System Builder, Assembler, Linker, PROM Splitter and Simulator Modules)	$2,850
ADDS-2121	Cross-Software (System Builder, Assembler, Linker, PROM Splitter Modules) for IBM PC and Compatibles under PC-DOS	$ 450
ADDS-2122	Cross-Software Simulator Module ONLY for IBM PC and Compatibles	$ 975
ADDS-2150	Emulator-110V ac (Requires ADDS-2110 or ADDS-2121)	$8,450
ADDS-2150E	Emulator-220V ac (Requires ADDS-2110 or ADDS-2121)	$8,450
ADDS-2151	Emulator-110V ac with Optional Trace Board (Requires ADDS-2110 or ADDS-2121)	*
ADDS-2151E	Emulator-220V ac with Optional Trace Board (Requires ADDS-2110 or ADDS-2121)	*
ADDS-2160	Emulator Board (Requires ADDS-2110 or ADDS-2121)	*
ADDS-2161	Emulator Trace Board (Option for ADDS-2150 or ADDS-2150E)	*
ADDS-2190	Three Day Training Course	$ 900

*Available late 1987.

ADSP-2100

FEATURES

Separate Program and Data Buses, Extended Off-Chip
Single-Cycle Direct Access to 16K × 16 of Data Memory
Single-Cycle Direct Access to 16K × 24 (Expandable to 32K × 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
125ns Cycle Time
600mW Maximum Power Dissipation with CMOS Technology
100-Pin Grid Array

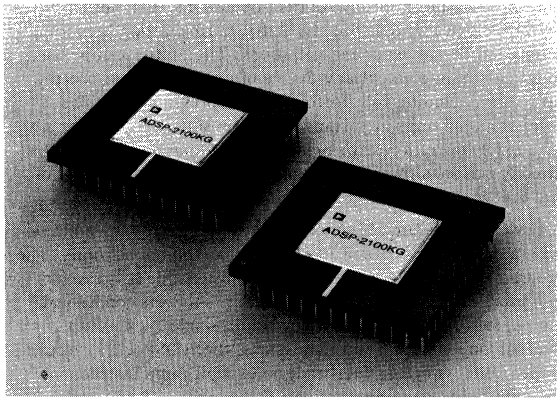
APPLICATIONS

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

GENERAL DESCRIPTION

The ADSP-2100 is a single-chip microprocessor optimized for digital signal processing (DSP) and other high-speed numeric processing applications. It integrates computational units, data address generators and a program sequencer in a single device.

The ADSP-2100 makes efficient use of external memories for program and data storage, freeing silicon area for increased processor performance. The resulting architecture combines the functions and performance of a bit-slice/building block system with the ease of design and development of a general-purpose microprocessor. The ADSP-2100 (K Grade) operates at 8.192MHz. Every instruction executes in a single 125ns cycle. Fabricated in a high-speed 1.5 micron double-layer metal CMOS process, the ADSP-2100 dissipates less than 600mW.



The ADSP-2100's flexible architecture and comprehensive instruction set support a high degree of operational parallelism. 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 is 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 systems under development, an Assembler, a Linker and a Simulator. The Simulator provides an interactive instruction-level simulation. A PROM Splitter generates PROM burner compatible files. An Emulator is available for hardware debugging of ADSP-2100 systems.

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 the Development System, refer to the *ADSP-2100 Cross-Software Manual* and the *ADSP-2100 Emulator Manual*. For examples of a variety of ADSP-2100 applications routines, refer to the *ADSP-2100 Applications Handbook, Volume 1*. Manuals are available from your local Analog Devices sales office. See ordering information.

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

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 to 16K × 24 of external memory, expandable to 32K × 24 by using the program memory data access (PMDA) signal as the 15th address line.

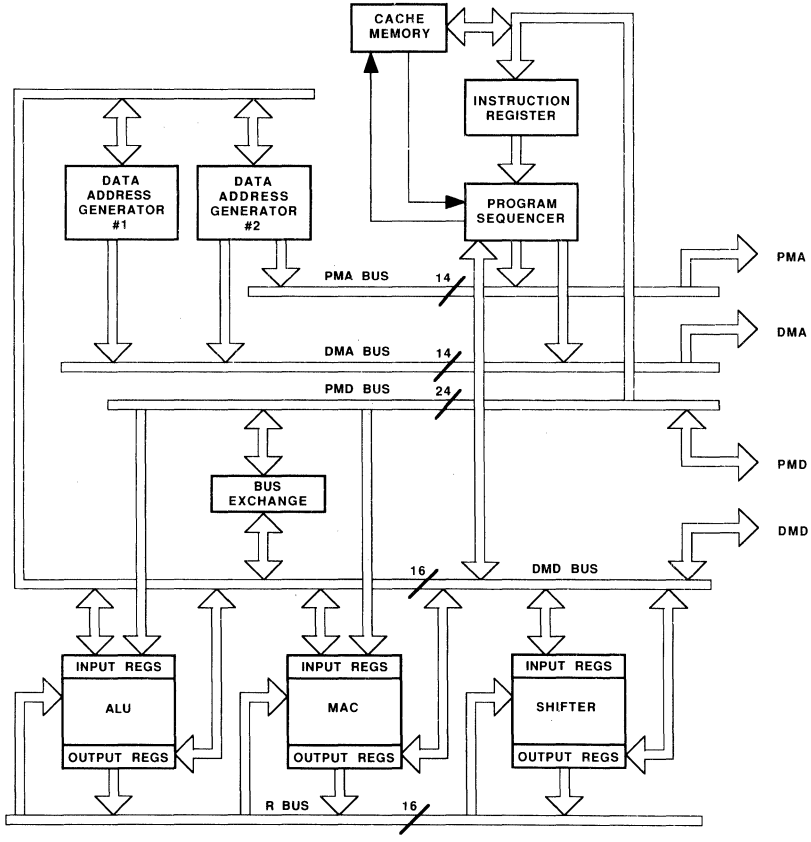


Figure 1. ADSP-2100 Block Diagram

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.

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. J and S grades operate at 6.144MHz and K grade operates at 8.192MHz. CLKIN must be four times these frequencies.
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. Also usable as a chip select signal for external memories. Tristated when $\overline{BG}$ is asserted.
$\overline{PMRD}$	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, is 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 the a Data Memory Access on the Data Memory interface. Also usable as a chip select signal for external memories. 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.

Arithmetic/Logic Unit

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

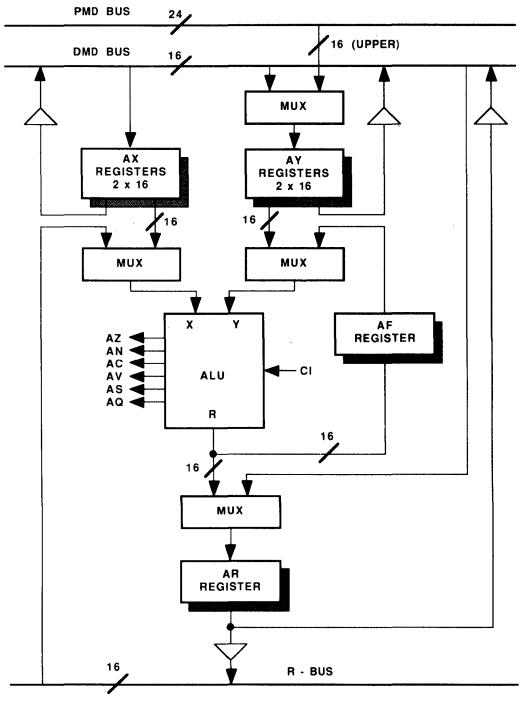


Figure 2. ALU Block Diagram

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.

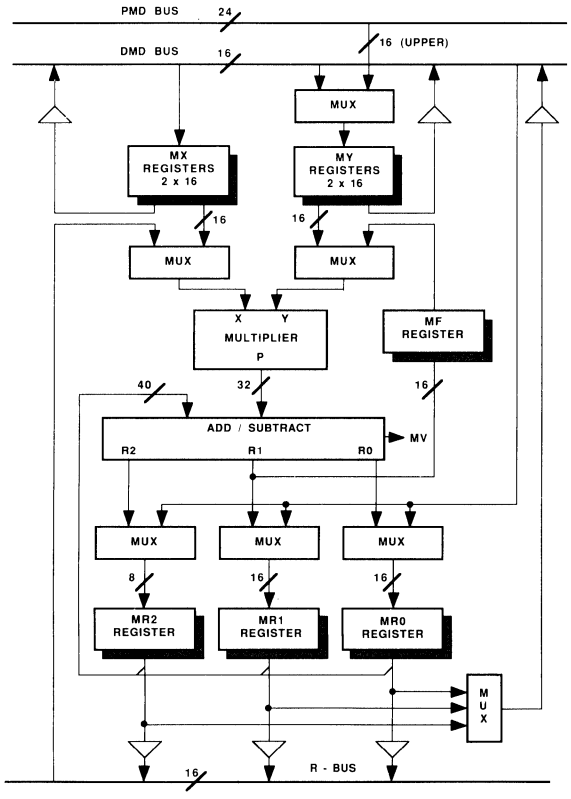


Figure 3. MAC Block Diagram

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.

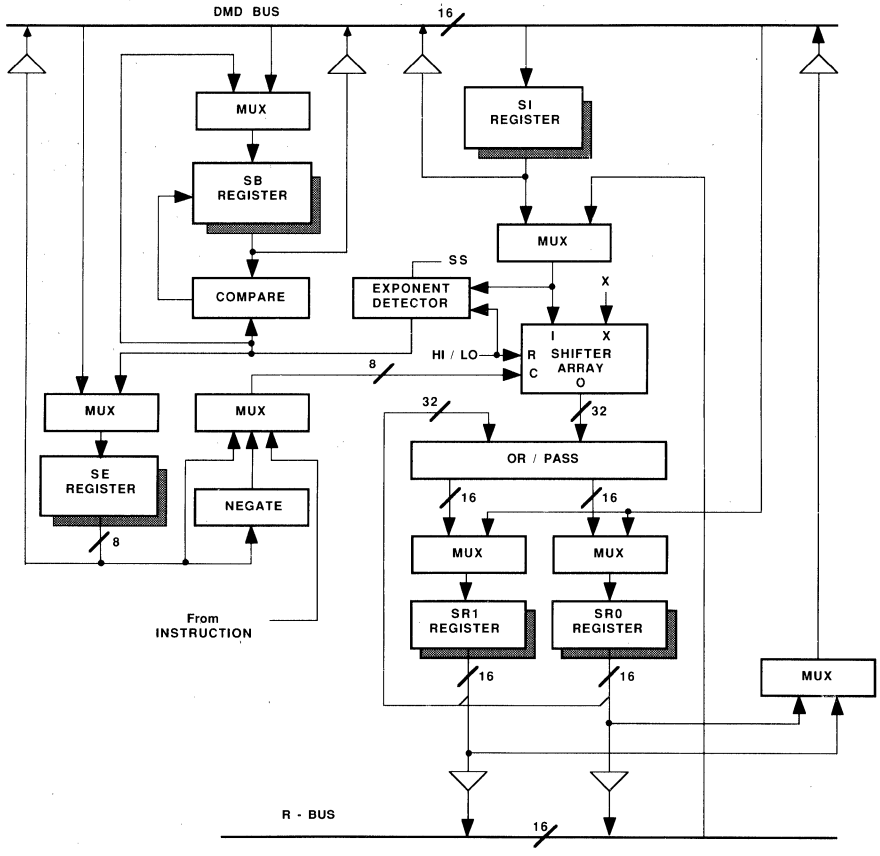


Figure 4. Shifter Block Diagram

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

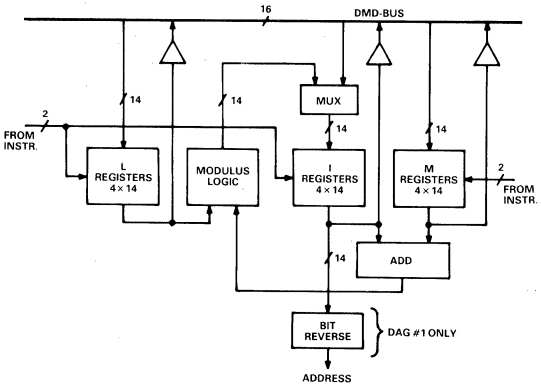


Figure 5. Data Address Generator

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

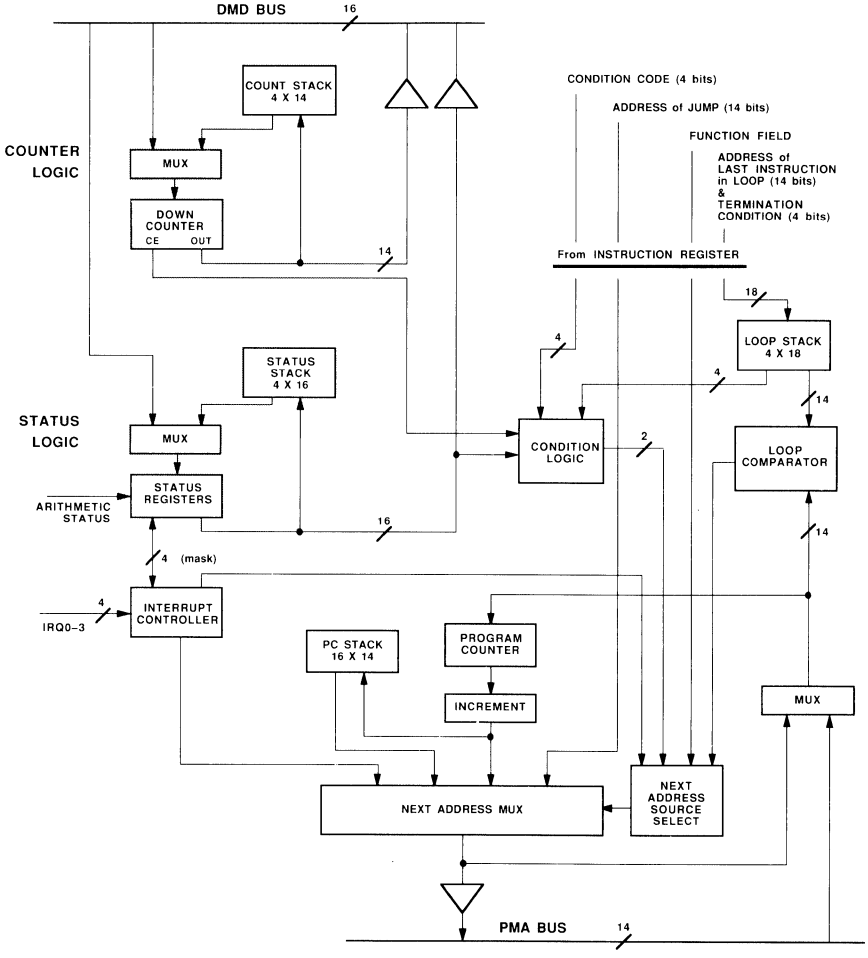


Figure 6. Program Sequencer

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. The CLKIN signal must have a duty cycle of 50% to insure that all eight states are equal in duration.

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.

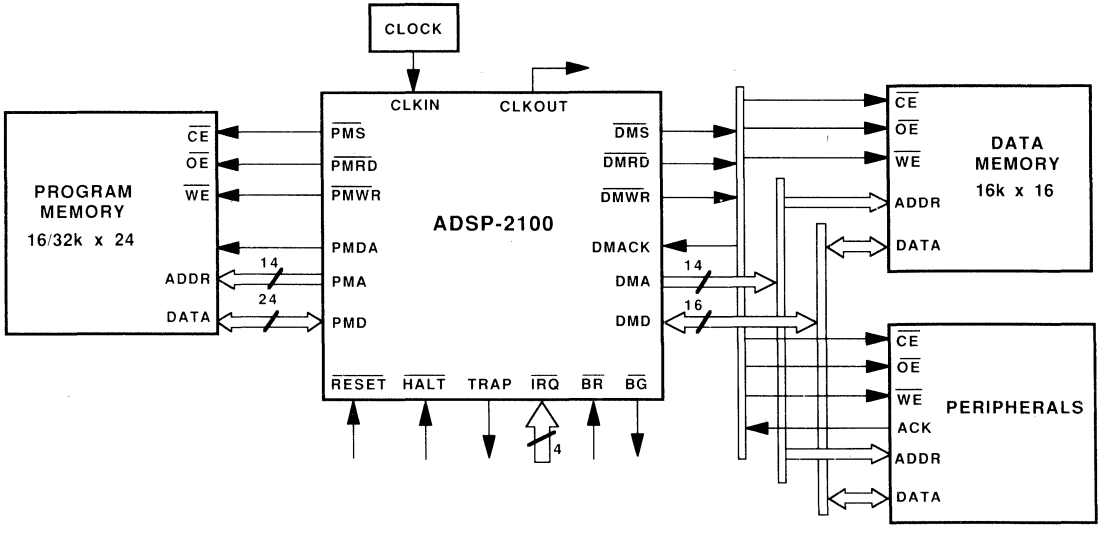


Figure 7. Basic System Configuration

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.

These conventions are used in Table IV on the following pages:

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

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

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 ;

MULTIFUNCTION INSTRUCTIONS

Computation with Memory Read

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

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 / I4 / I5 / I6 / I7 | , | M0 / M1 / M2 / M3 / M4 / M5 / M6 / M7 |) = dreg, <comp> ;

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

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 |) ;

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 ;

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

ADSP-2100

Parameter		J & K Grades Min	J & K Grades Max	S Grade Min	S Grade 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	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		2.0		V
V_{IL}	Lo-Level Input Voltage[1]	@V_{DD} = min		0.8		0.8	V
V_{OH}	Hi-Level Output Voltage[3]	@V_{DD} = min, I_{OH} = −1mA	2.4		2.4		V
V_{OL}	Lo-Level Output Voltage[3]	@V_{DD} = min, I_{OL} = 4mA		0.4		0.6	V
I_{IH}	Hi-Level Input Current[2]	@V_{DD} = max, V_{IN} = 5V		10		10	μA
I_{IL}	Lo-Level Input Current[2]	@V_{DD} = max, V_{IN} = 0V		10		10	μA
I_{OZH}	Tristate Leakage Current[4]	@V_{DD} = max, V_{IN} = max[7]		50		50	μA
I_{OZL}	Tristate Leakage Current[5]	@V_{DD} = max, V_{IN} = 0V[7]		50		50	μA
I_{OZL}	Tristate Leakage Current[6]	@V_{DD} = max, V_{IN} = 0V[7]		150		150	μA
I_{DD}	Supply Current (Power-Down)[6&9]	@V_{DD} = max, V_{IN} = 0V[7]		10		10	mA
I_{DD}	Supply Current (Dynamic)	@V_{DD} = max, max clock rate[8]		90		100	mA

NOTES

[1]Applies to pins: PMD_{0-23}, DMD_{0-15}, $\overline{BR}$, $\overline{IRQ}_{0-3}$, DMACK, $\overline{RESET}$, $\overline{HALT}$, CLKIN (39 input pins)

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

[3]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)

[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 = 6.144mHz for J and S grade, 8.192mHz for K grade.

[9]"Power-down" refers to an idle state. While the ADSP-2100 does not have any special standby or low-power mode, these conditions represent the lowest power consumption state.

ABSOLUTE MAXIMUM RATINGS

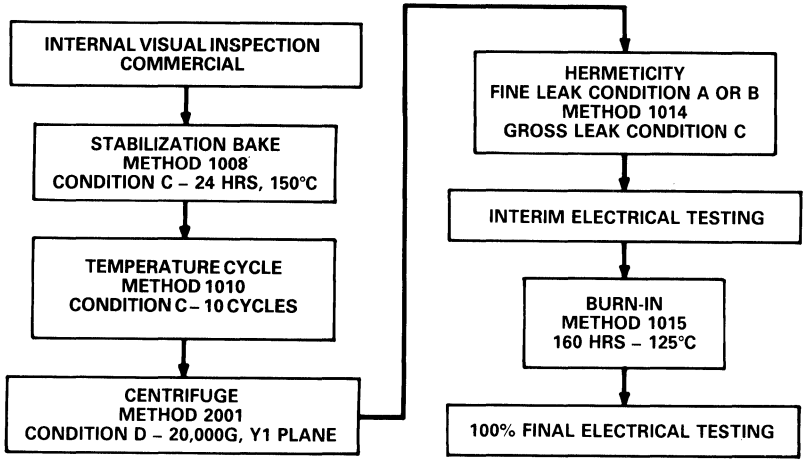
Supply Voltage	−0.3V to +7V
Input Voltage	−0.3V to V_{DD}
Output Voltage Swing	−0.3V to V_{DD}
Load Capacitance	200pF
Operating Temperature Range (Ambient)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10 Seconds)	+300°C

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-2100JG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100KG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100SG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100JG/+	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100KG/+	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-2100SG/+	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100SG/883B	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-2100JP	0 to +70°C	100-Lead PLCC	P-100
ADSP-2100KP	0 to +70°C	100-Lead PLCC	P-100
ADSP-2100JP/+	0 to +70°C	100-Lead PLCC	P-100
ADSP-2100KP/+	0 to +70°C	100-Lead PLCC	P-100

ADSP-2100 Development Tools

Part Number	Description
ADDS-2110	ADSP-2100 Cross-Software (VAX/VMS)
ADDS-2121	ADSP-2100 Assembler, Linker, Utilities (IBM-PC)
ADDS-2122	ADSP-2100 Simulator (IBM-PC)
ADDS-2150	ADSP-2100 In-Circuit Emulator (North American version)
ADDS-2150E	ADSP-2100 In-Circuit Emulator (European version)
ADDS-2151	ADSP-2100 In-Circuit Emulator with Trace Board (North American version)
ADDS-2151E	ADSP-2100 In-Circuit Emulator with Trace Board (European version)
ADDS-2161	Trace Board Option for ADSP-2100 In-Circuit Emulator



Plus Processing

ESD SENSITIVITY

The ADSP-2100 features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-2100 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.

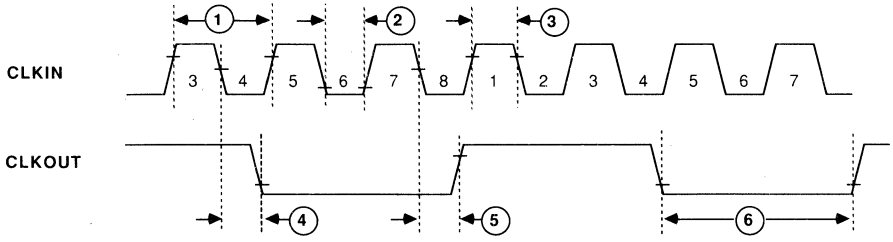
SWITCHING CHARACTERISTICS

GENERAL NOTES

Use the exact timing information given and 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.

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Clock Signals	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
	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	13	34	ns
5	CLKIN Low (7-8) to CLKOUT High	B	6	24	6	20	6	24	ns
6	CLKOUT Width Low[2]	A	60		45		60		ns

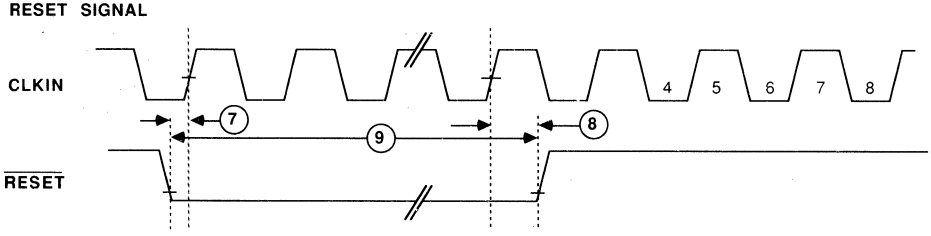


NOTE
The Processor Cycle is Divided into 8 Internal States Determined by the Rising and Falling Edges of CLKIN. CLKOUT is Synchronized to the Processor States as Shown Above.

Figure 8. Clock Signals

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Control Signals	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
	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
9	$\overline{\text{RESET}}$ Width Low[2]	A	162		122		162		ns



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

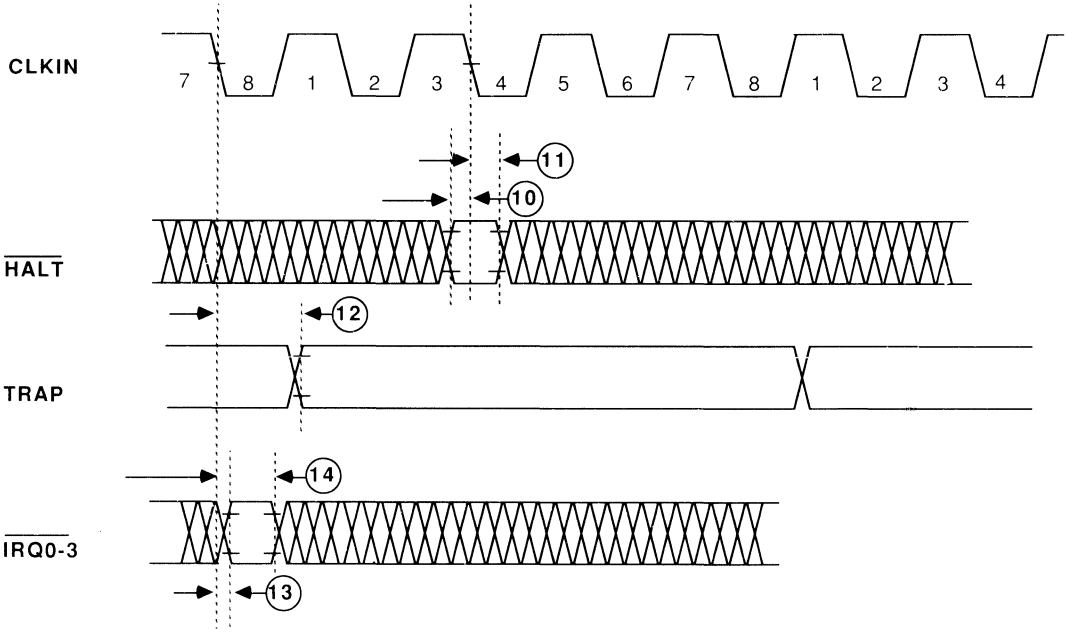
Figure 9. $\overline{\text{RESET}}$ Signal

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

Control Signals *Timing Requirements*	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
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	6	25	6	20	6	25	ns
Interrupts *Timing Requirements*								
13 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Valid	B		2		2		2	ns
14 CLKIN Low (7-8) to $\overline{\text{IRQ}}$ Invalid	B	21		17		21		ns

2

CONTROL SIGNALS



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 $\overline{\text{HALT}}$, TRAP, and $\overline{\text{IRQ}}_{0-3}$.

Figure 10. Control Signals

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Bus Request Asserted *Timing Requirements*	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
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
Switchng Characteristics									
17	CLKIN Low (3-4) to $\overline{BG}$ Low	B	13	38	13	30	13	38	ns
19	$\overline{BG}$ Low to xMxx Disable[3]	D		22		17		22	ns

BUS REQUEST ASSERTED

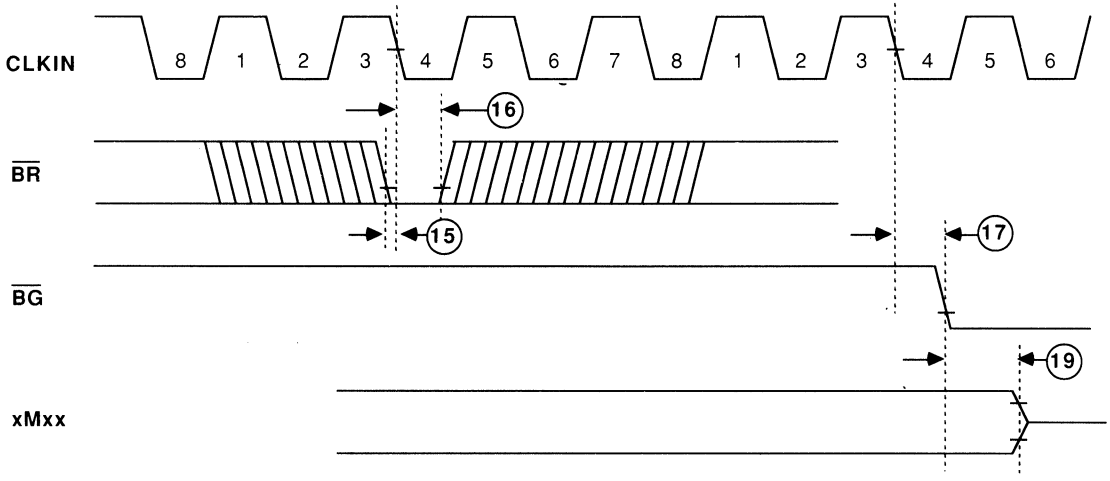


Figure 11. Bus Request Asserted

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

Bus Request Negated	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
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	10	31	10	25	10	31	ns
20 xMxx Enable to $\overline{BG}$ High[3]	F		12		10		12	ns

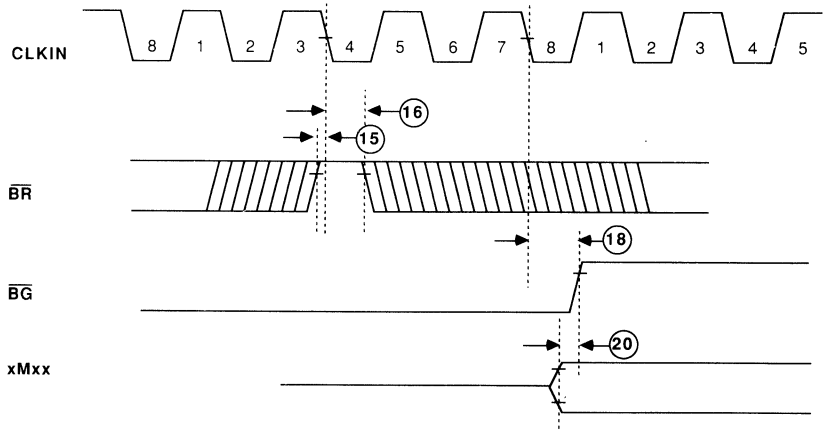
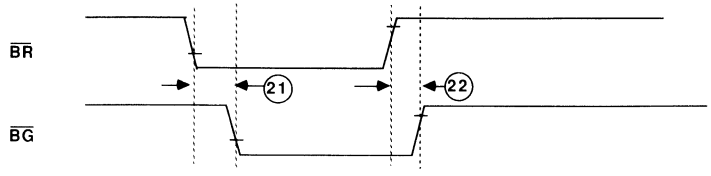


Figure 12. Bus Request Negated

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

Bus Request/Grant with $\overline{RESET}$ Low	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
Switching Characteristics								
21 $\overline{BR}$ Low to $\overline{BG}$ Low during reset	A	10	28	10	23	10	28	ns
22 $\overline{BR}$ High to $\overline{BG}$ High during reset	A	6	21	6	18	6	21	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 $\overline{RESET}$ Low

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Program Memory Read	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
	Switching Characteristics								
23	CLKIN High (8-1) to PMA Valid	B	18	72	18	58	18	72	ns
24	CLKIN High (8-1) to PMA Invalid	B	10		10		10		ns
25	CLKIN High (8-1) to PMDA Valid	B	18	56	18	44	18	56	ns
26	CLKIN High (8-1) to PMDA Invalid	B	12		12		12		ns
27	CLKIN High (8-1) to $\overline{\text{PMS}}$ Valid	B	12	32	12	25	12	32	ns
28	CLKIN High (8-1) to $\overline{\text{PMS}}$ Invalid	B	8		8		8		ns
29	CLKIN Low (3-4) to $\overline{\text{PMRD}}$ Low	B	11	36	11	29	11	36	ns
30	CLKIN Low (7-8) to $\overline{\text{PMRD}}$ High	B	6	25	6	20	6	25	ns
31	$\overline{\text{PMRD}}$ Width Low[2]	A	60		45		60		ns
32	PMA Valid to $\overline{\text{PMRD}}$ Low[2]	A	18		11		18		ns
33	$\overline{\text{PMRD}}$ High to PMA Invalid[2]	A	20		16		20		ns
34	PMDA Valid to $\overline{\text{PMRD}}$ Low[2]	A	41		31		41		ns
35	$\overline{\text{PMRD}}$ High to PMDA Invalid[2]	A	23		18		23		ns
36	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMRD}}$ Low[2]	A	57		40		57		ns
37	$\overline{\text{PMRD}}$ High to $\overline{\text{PMS}}$ Invalid[2]	A	16		12		16		ns
	Timing Requirements								
56	PMD In Valid to CLKIN Low (7-8)	B	4		4		4		ns
57	CLKIN Low (7-8) to PMD In Invalid	B	12		11		12		ns
58	$\overline{\text{PMRD}}$ Low to PMD In Valid[2]	A		45		37		45	ns
59	PMA Valid to PMD In Valid[2]	A		57		50		57	ns
60	$\overline{\text{PMS}}$ Valid to PMD In Valid[2]	A		90		65		90	ns
97	$\overline{\text{PMRD}}$ High to PMD In Invalid	A	0		0		0		ns

PROGRAM MEMORY READ

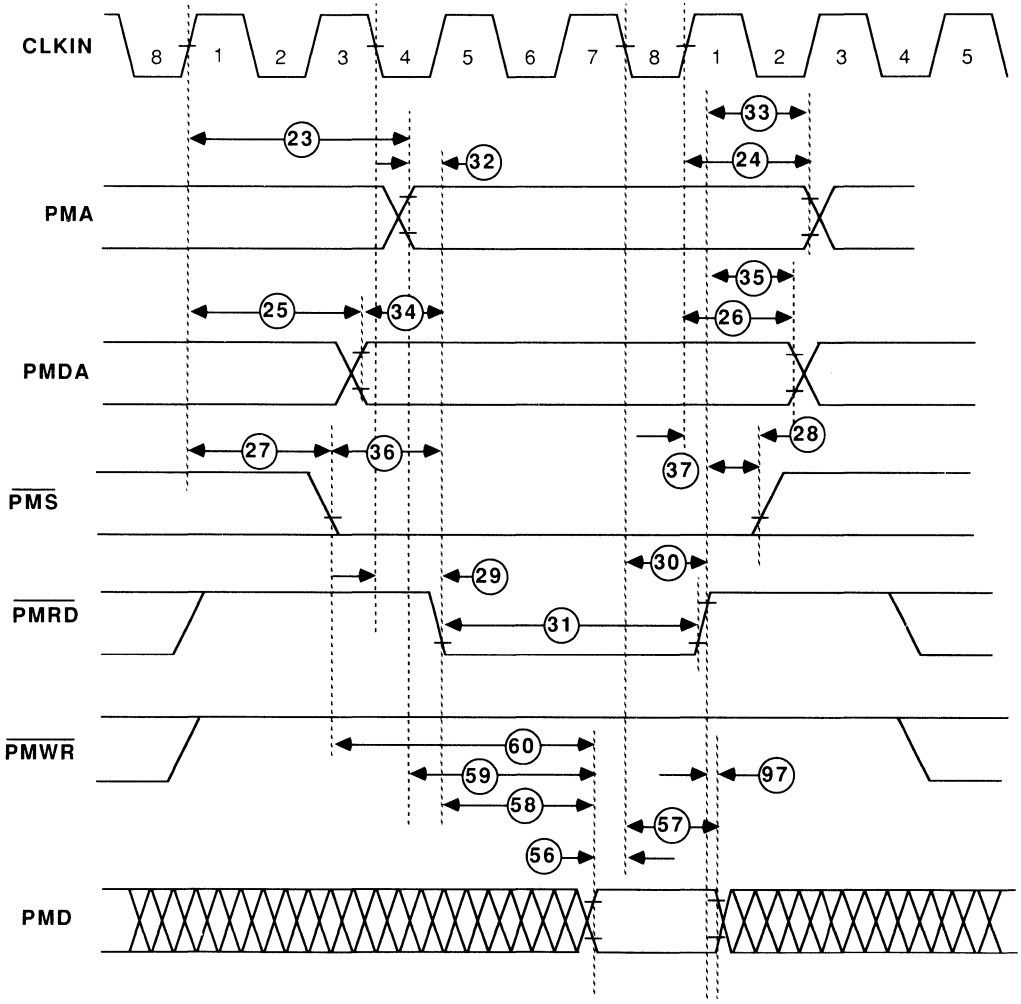


Figure 14. Program Memory Read

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Program Memory Write *Switching Characteristics*	**Test Code**	**J Grade**		**K Grade**		**S Grade**		**Units**
			Min	**Max**	**Min**	**Max**	**Min**	**Max**	
23	CLKIN High (8-1) to PMA Valid	B	18	72	18	58	18	72	ns
24	CLKIN High (8-1) to PMA Invalid	B	10		10		10		ns
25	CLKIN High (8-1) to PMDA Valid	B	18	56	18	44	18	56	ns
26	CLKIN High (8-1) to PMDA Invalid	B	12		12		12		ns
27	CLKIN High (8-1) to $\overline{\text{PMS}}$ Valid	B	12	32	12	25	12	32	ns
28	CLKIN High (8-1) to $\overline{\text{PMS}}$ Invalid	B	8		8		8		ns
38	CLKIN Low (3-4) to $\overline{\text{PMWR}}$ Low	B	8	34	8	25	8	34	ns
39	CLKIN Low (7-8) to $\overline{\text{PMWR}}$ High	B	8	27	8	22	8	27	ns
40	$\overline{\text{PMWR}}$ Width Low[2]	A	60		45		60		ns
41	PMA Valid to $\overline{\text{PMWR}}$ Low[2]	A	16		10		16		ns
42	$\overline{\text{PMWR}}$ High to PMA Invalid[2]	A	19		15		19		ns
43	PMDA Valid to $\overline{\text{PMWR}}$ Low[2]	A	39		29		39		ns
44	$\overline{\text{PMWR}}$ High to PMDA Invalid[2]	A	22		16		22		ns
45	$\overline{\text{PMS}}$ Valid to $\overline{\text{PMWR}}$ Low[2]	A	54		40		54		ns
46	$\overline{\text{PMWR}}$ High to $\overline{\text{PMS}}$ Invalid[2]	A	15		11		15		ns
47	CLKIN High (4-5) to PMD Out Enable	E	18		15		18		ns
48	CLKIN High (8-1) to PMD Out Disable	C		40		35		40	ns
49	CLKIN High (4-5) to PMD Out Valid	B		49		39		49	ns
50	CLKIN High (8-1) to PMD Out Invalid	B	16		16		16		ns
51	$\overline{\text{PMWR}}$ Low to PMD Out Enable[2]	F	19		15		19		ns
52	$\overline{\text{PMWR}}$ High to PMD Out Disable[2]	D		43		37		43	ns
53	$\overline{\text{PMWR}}$ Low to PMD Out Valid[2]	A		40		32		40	ns
54	$\overline{\text{PMWR}}$ High to PMD Out Invalid[2]	A	23		18		23		ns
55	PMD Out Valid to $\overline{\text{PMWR}}$ High[2]	A	33		25		33		ns

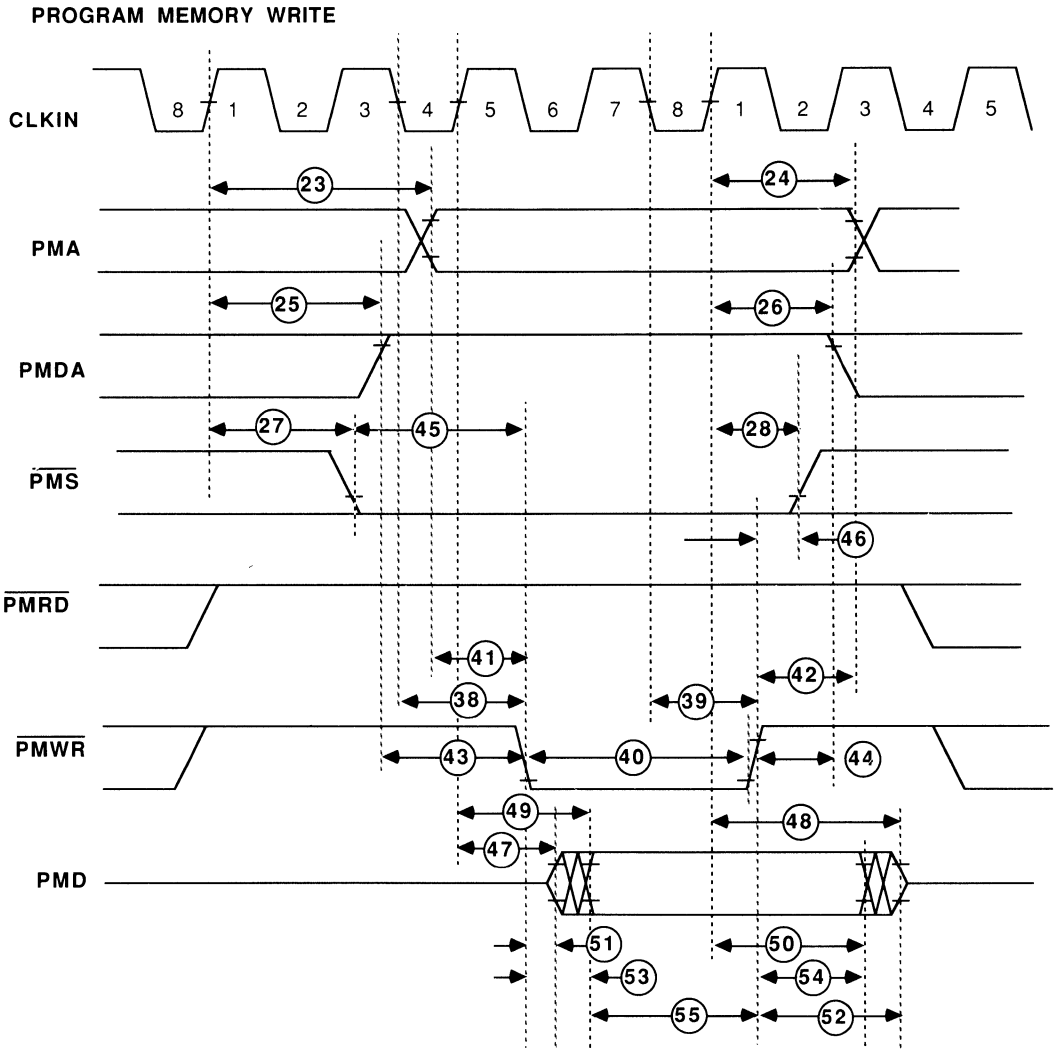


Figure 15. Program Memory Write

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Data Memory Read *Switching Characteristics*	Test Code	J Grade Min	J Grade Max	K Grade Min	K Grade Max	S Grade Min	S Grade Max	Units
61	CLKIN High (8-1) to DMA Valid	B	16	65	16	52	16	65	ns
62	CLKIN High (8-1) to DMA Invalid	B	10		10		10		ns
63	CLKIN High (8-1) to $\overline{\text{DMS}}$ Valid	B	19	53	19	40	19	53	ns
64	CLKIN High (8-1) to $\overline{\text{DMS}}$ Invalid	B	12		12		12		ns
65	CLKIN Low (3-4) to $\overline{\text{DMRD}}$ Low	B	11	33	11	28	11	33	ns
66	CLKIN Low (7-8) to $\overline{\text{DMRD}}$ High	B	6	25	6	21	6	25	ns
67	$\overline{\text{DMRD}}$ Width Low[2]	A	60		45		60		ns
68	DMA Valid to $\overline{\text{DMRD}}$ Low[2]	A	21		16		21		ns
69	$\overline{\text{DMRD}}$ High to DMA Invalid[2]	A	19		15		19		ns
70	DMS Valid to $\overline{\text{DMRD}}$ Low	A	35		27		35		ns
71	$\overline{\text{DMRD}}$ High to $\overline{\text{DMS}}$ Invalid	A	22		18		22		ns
	Timing Requirements								
72	DMACK Valid to CLKIN High (6-7)	B	1		1		1		ns
73	CLKIN High (6-7) to DMACK Invalid	B	9		8		9		ns
74	$\overline{\text{DMRD}}$ Low to DMACK Valid[2]	A		31		21		31	ns
75	DMA Valid to DMACK Valid[2]	A		57		42		57	ns
92	DMD In Valid to CLKIN Low (7-8)	B	0		0		0		ns
93	CLKIN Low (7-8) to DMD In Invalid	B	17		14		17		ns
94	$\overline{\text{DMRD}}$ Low to DMD In Valid[2]	A		57		41		57	ns
95	DMA Valid to DMD In Valid[2]	A		82		61		82	ns
96	$\overline{\text{DMS}}$ Valid to DMD In Valid	A		96		70		96	ns
98	$\overline{\text{DMRD}}$ High to DMD In Invalid	A	0		0		0		ns

DATA MEMORY READ

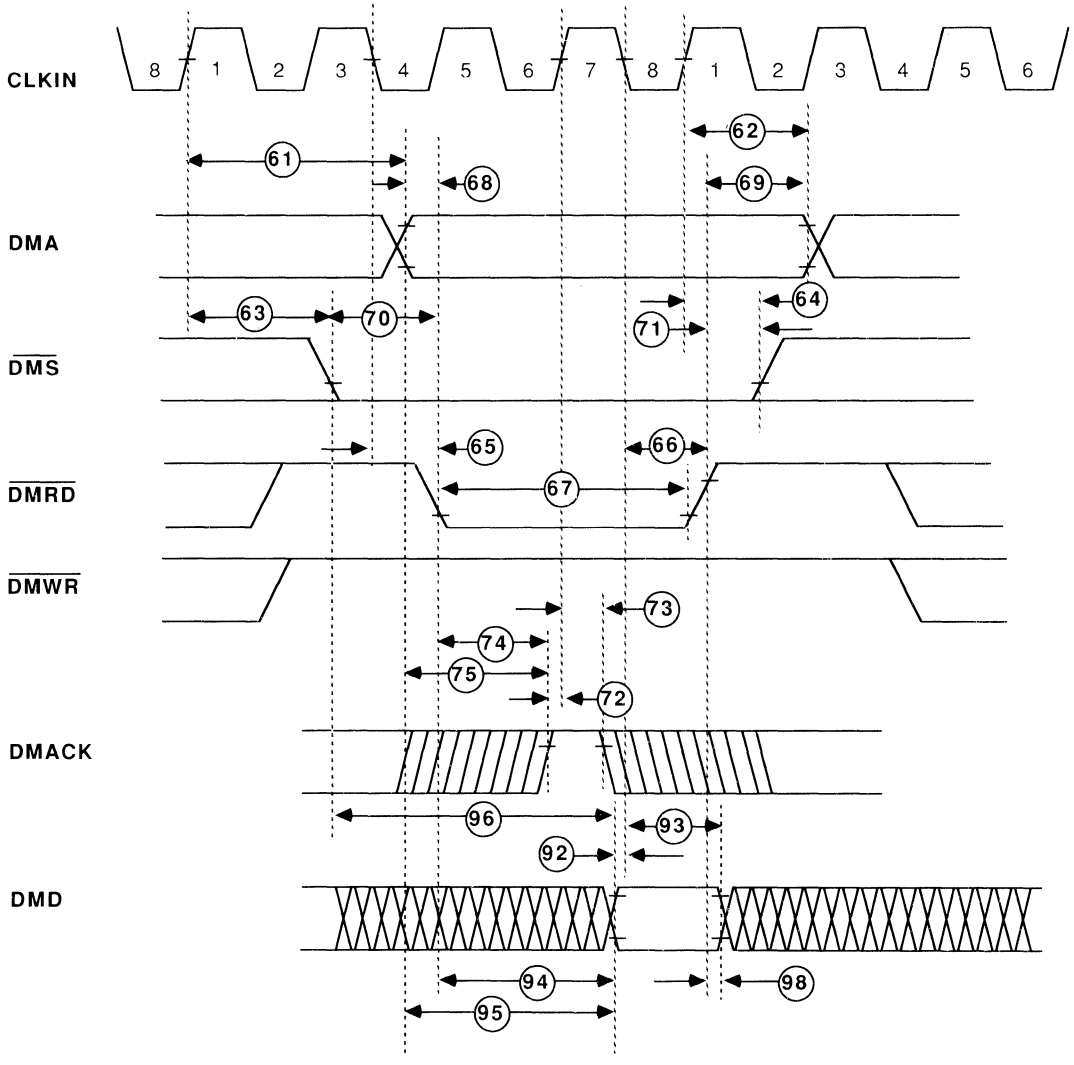


Figure 16. Data Memory Read

Notes 1, 2 and 3 and a table explaining the Test Codes appear further on in this data sheet.

	Data Memory Write	Test Code	J Grade		K Grade		S Grade		Units
			Min	Max	Min	Max	Min	Max	
	Switching Characteristics								
61	CLKIN High (8-1) to DMA Valid	B	16	65	16	52	16	65	ns
62	CLKIN High (8-1) to DMA Invalid	B	10		10		10		ns
63	CLKIN High (8-1) to $\overline{\text{DMS}}$ Valid	B	19	53	19	40	19	53	ns
64	CLKIN High (8-1) to $\overline{\text{DMS}}$ Invalid	B	12		12		12		ns
76	CLKIN Low (3-4) to $\overline{\text{DMWR}}$ Low	B	11	35	11	28	11	35	ns
77	CLKIN Low (7-8) to $\overline{\text{DMWR}}$ High	B	6	26	6	21	6	26	ns
78	$\overline{\text{DMWR}}$ Width Low[2]	A	60		45		60		ns
79	DMA Valid to $\overline{\text{DMWR}}$ Low[2]	A	24		17		24		ns
80	$\overline{\text{DMWR}}$ High to DMA Invalid[2]	A	20		15		20		ns
81	$\overline{\text{DMS}}$ Valid to $\overline{\text{DMWR}}$ Low	A	37		28		37		ns
82	$\overline{\text{DMWR}}$ High to $\overline{\text{DMS}}$ Invalid	A	22		19		22		ns
83	CLKIN High (4-5) to DMD Out Enable	E	18		15		18		ns
84	CLKIN High (8-1) to DMD Out Disable	C		38		32		38	ns
85	CLKIN High (4-5) to DMD Out Valid	B		49		40		49	ns
86	CLKIN High (8-1) to DMD Out Invalid	B	11		11		11		ns
87	$\overline{\text{DMWR}}$ Low to DMD Out Enable	F	18		14		18		ns
88	$\overline{\text{DMWR}}$ High to DMD Out Disable[2]	D		40		35		40	ns
89	$\overline{\text{DMWR}}$ Low to DMD Out Valid	A		38		32		38	ns
90	$\overline{\text{DMWR}}$ High to DMD Out Invalid[2]	A	21		16		21		ns
91	DMD Out Valid to $\overline{\text{DMWR}}$ High[2]	A	33		21		33		ns
	Timing Requirements								
72	DMACK Valid to CLKIN High (6-7)	B	1		1		1		ns
73	CLKIN High (6-7) to DMACK Invalid	B	9		8		9		ns
74	$\overline{\text{DMRD}}$ Low to DMACK Valid[2]	A		31		21		31	ns
75	DMA Valid to DMACK Valid[2]	A		57		42		57	ns

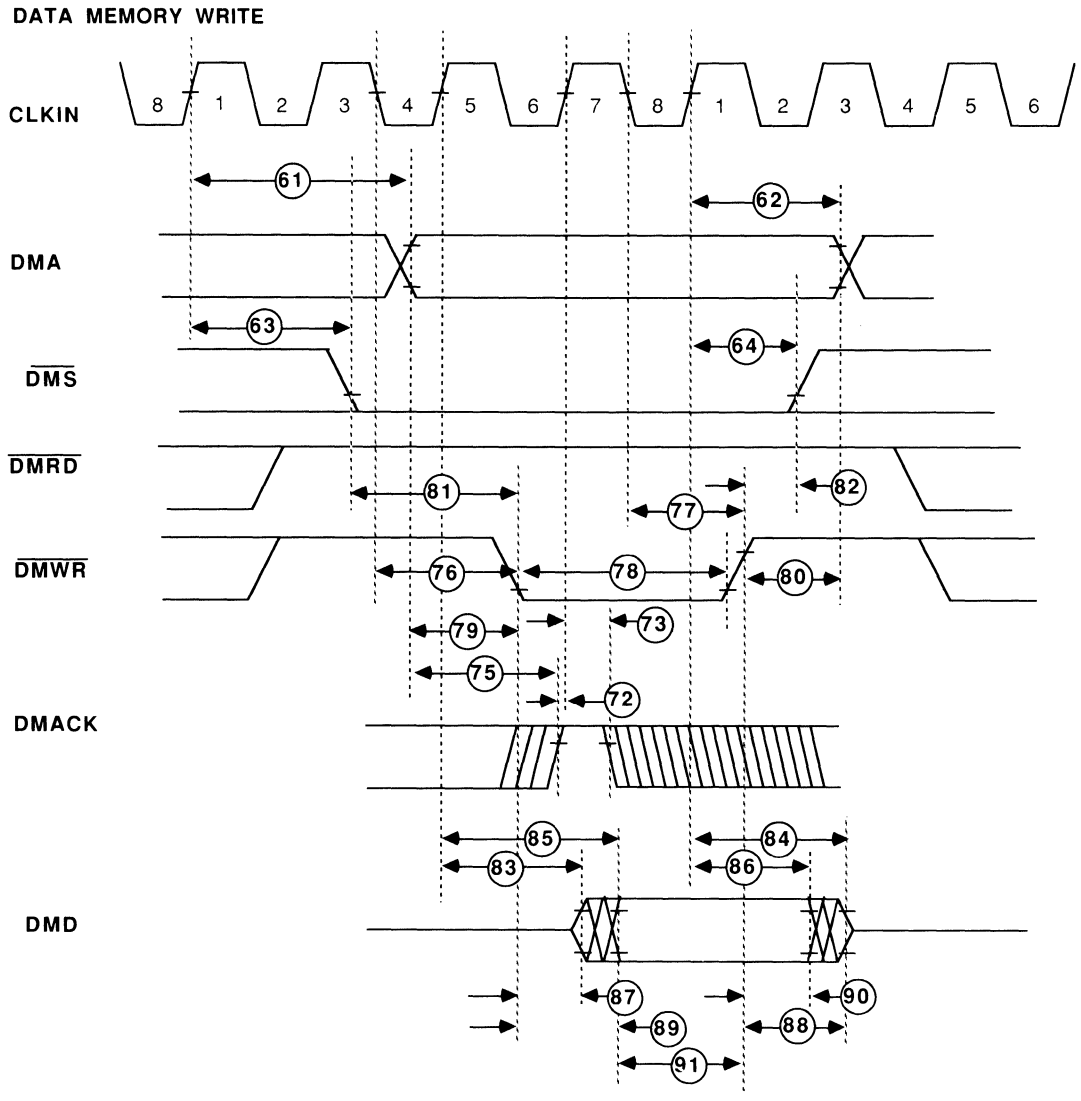


Figure 17. Data Memory Write

NOTES
[1]Rise and fall times ≤5ns.
[2]These items are cycle time dependent.
[3]"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	1.5V
	to/from	
	Inputs, Outputs	Low = 0.8V, High = 2.0V
C	CLKIN	1.5V
	to	
	Output Disable	Low = V_{OL} + 0.5V, High = V_{OH} − 0.5V
D	Output	Low = 0.8V, High = 2.0V
	to	
	Output Disable	Low = V_{OL} + 0.5V, High = V_{OH} − 0.5V
E	CLKIN	1.5V
	to	
	Output Enable	Low = VT − 0.1V, High = VT + 0.1V
F	Output	Low = 0.8V, High = 2.0V
	to/from	
	Output Enable	Low = VT − 0.1V, High = VT + 0.1V

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

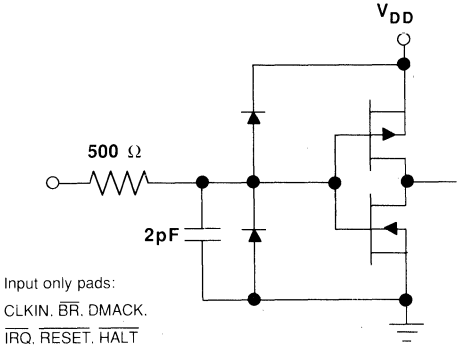


Figure 18. Equivalent Input Circuit

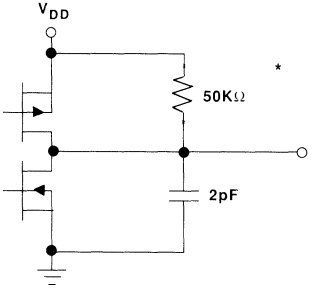


Figure 19. Equivalent Output Circuit

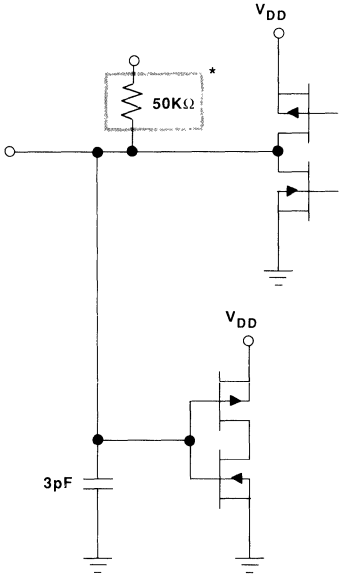


Figure 20. Equivalent Bidirectional Circuit

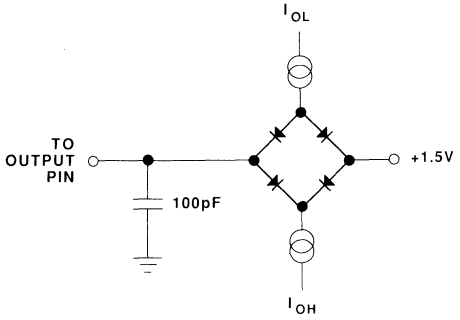


Figure 21. Normal Load for AC Measurements

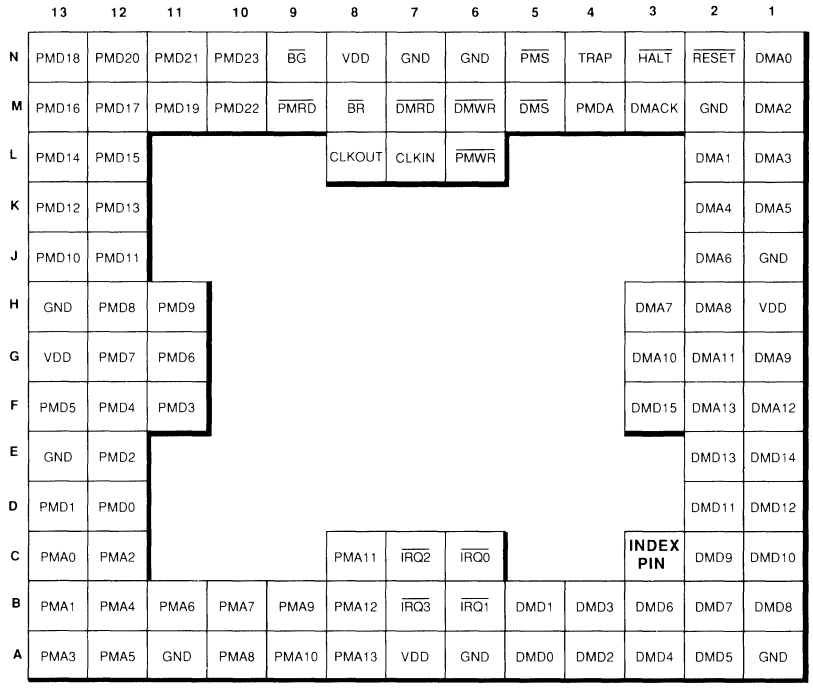


Figure 22. ADSP-2100 Pins, Top View, Pins Down

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

ADSP-2100 PIN CONFIGURATION

P-100

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

Microcoded Support Components
Contents

Introduction

The current members of the Word-Slice™ family of microcode components include the ADSP-1401 Sequencer and the ADSP-1410 Address Generator. (You will also want to look at the ADSP-3128 Register File in the Floating-Point Chapter.) Both feature the Look-Ahead™ pipeline which eliminates the need for an external microcode pipeline register by internally latching instructions and addresses. Both are fabricated in a fast 1.5μm CMOS process and are available in 48-pin DIPs and 52-lead PLCCs. Together they improve performance and reduce board space substantially relative to bit-slice solutions. A 225-page *Word-Slice User's Manual* covers all aspects of programming with these parts.

The ADSP-1401 is a high-speed, 16-bit microprogram controller optimized for the demanding sequencing tasks found in digital signal processors and general-purpose computers. In addition to high speed, this sequencer features 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-1410 is a fast, flexible address generator that rapidly generates the data memory addresses required by routines such as digital filters, FFTs, matrix operations and DMAs. The ADSP-1410 features a 16-bit ALU, a comparator and thirty 16-bit registers organized into four files: sixteen address registers, six offset registers, four compare registers and four initialization 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.

Third-party support is available in the form of meta-assemblers, development systems and behavioral models. Contact the Analog Devices for further information on development tools.

Selection Guide

ADSP-1410 ADDRESS GENERATOR

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

ADSP-1401 PROGRAM SEQUENCER

		Program Address Size	Clock-to-Address Valid Delay[1]	Minimum Cycle Time	Stack Depth	Number of Interrupts	I_{DD}[2]	Package Options[3]	Process	Logic Type
Commerical	J	16 Bits / 64K Words	35 ns	90ns	64 Words	10	75mA	D, N, P	CMOS	TTL
	K	16 Bits / 64K Words	25 ns	70ns	64 Words	10	75mA	D, N, P	CMOS	TTL
Military	S	16 Bits / 64K Words	45 ns	110ns	64 Words	10	100mA	D	CMOS	TTL
	T	16 Bits / 64K Words	35 ns	90ns	64 Words	10	100mA	D	CMOS	TTL

NOTES

[1]Maximum at + 70°C for commercial, + 125°C for military.

[2]mA maximum, f_{CLK} = max, over full V_{DD} range, at + 70°C for commercial, + 125°C for military.

[3]Packages are identified as follows: D = ceramic 48-pin DIP; N = Plastic 48-pin DIP; P = 52-contact PLCC.

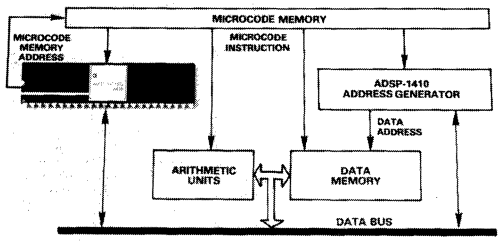
ANALOG DEVICES

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.

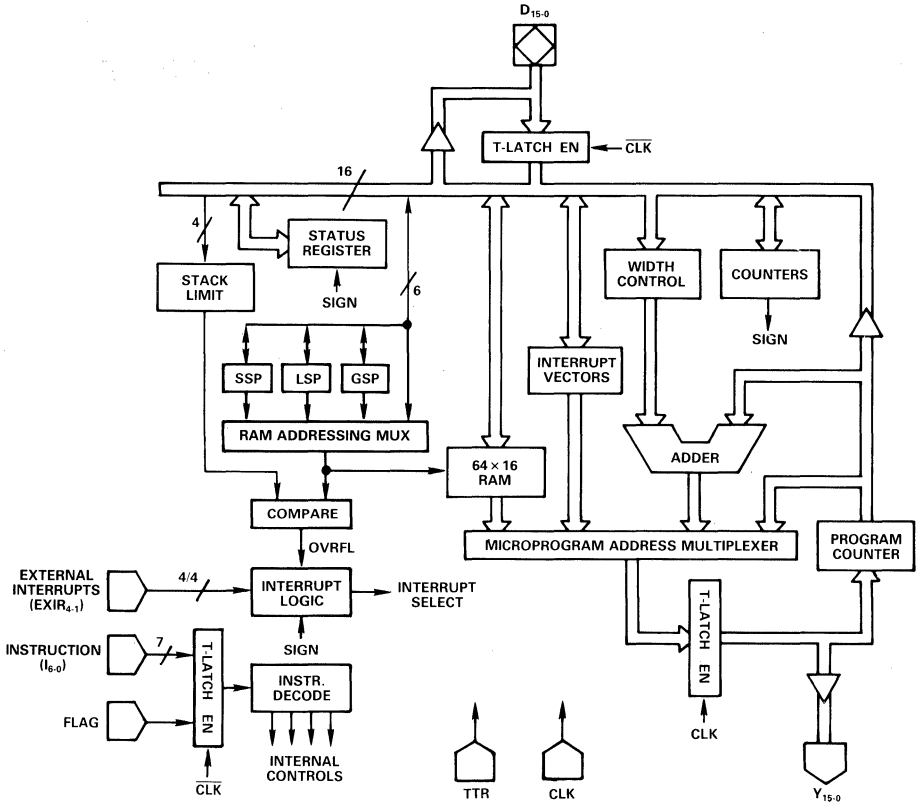


Figure 1. ADSP-1401 Block Diagram

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 (to 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.

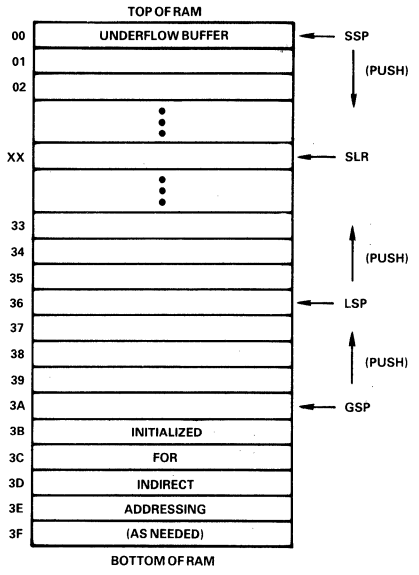


Figure 2. Typical RAM Initialization

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).

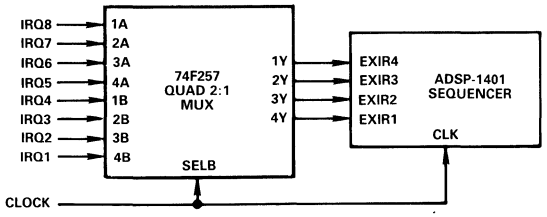


Figure 3. Expanding External Interrupts

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.

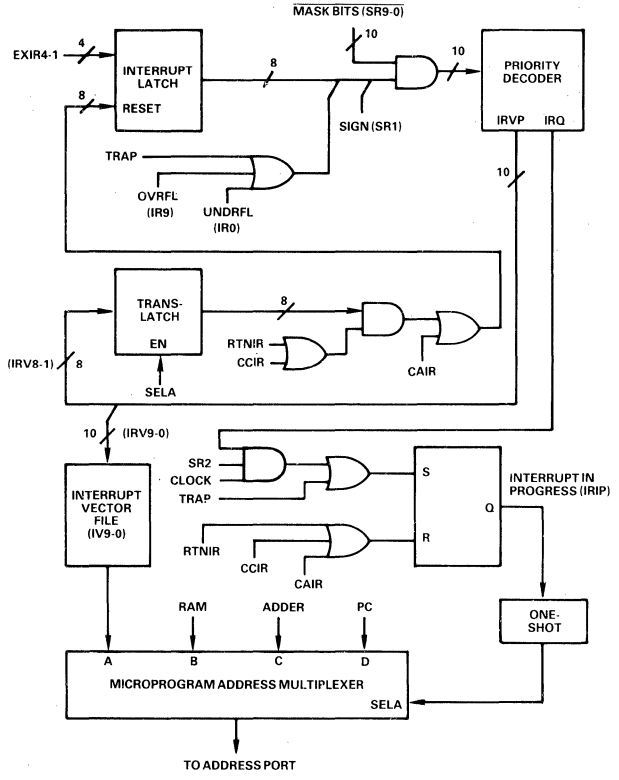


Figure 4. Internal Interrupt Control Logic

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.

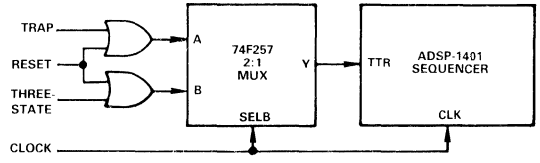


Figure 5. External Logic for TTR Pin

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 downloaded 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.

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.

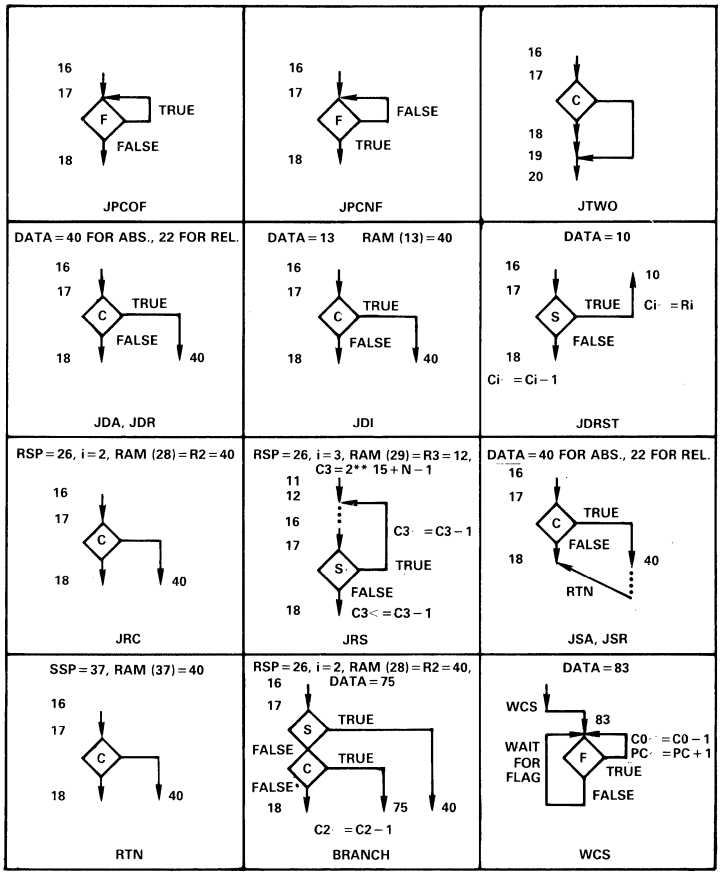


Figure 6. Instruction Flow Charts

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.

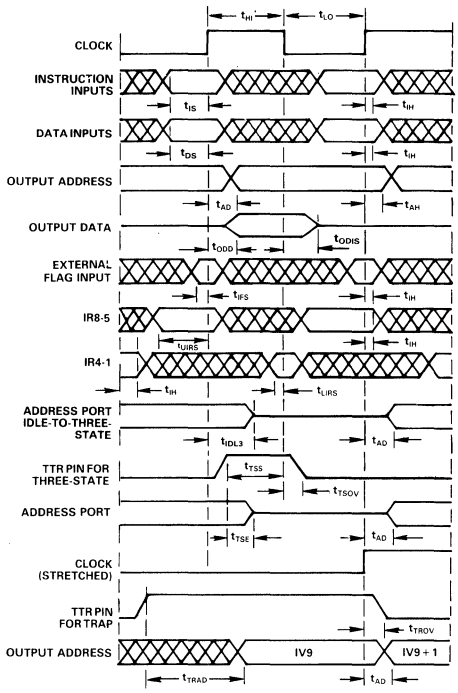


Figure 7. ADSP-1401 Timing Diagram

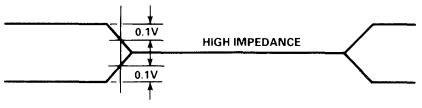


Figure 8. Three-State Reference Levels

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/+	−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		3.0		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		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

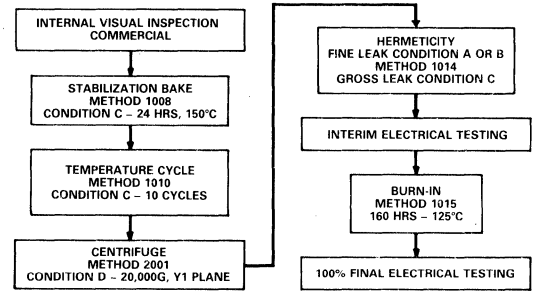


Figure 9.

CAUTION:

1. ESD sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts.
2. Do not insert this device into powered sockets. Remove power before insertion or removal.

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)		15		*		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. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 9.
[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.

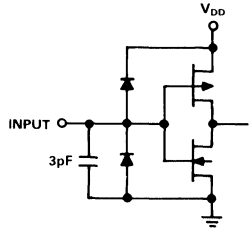


Figure 10. Equivalent Input Circuit

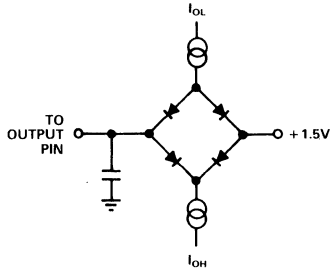


Figure 11. Normal Load for ac Measurements

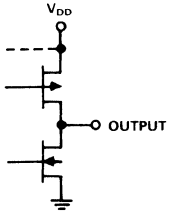


Figure 12. Equivalent Output Circuit

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 $IV_3 = 21$.

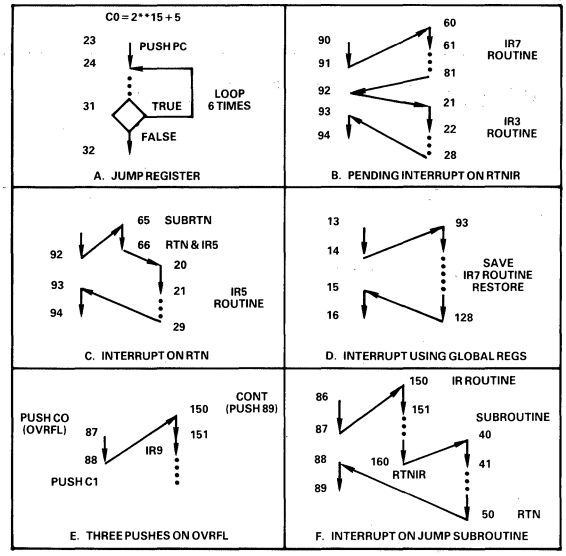


Figure 13. Programming Examples

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.

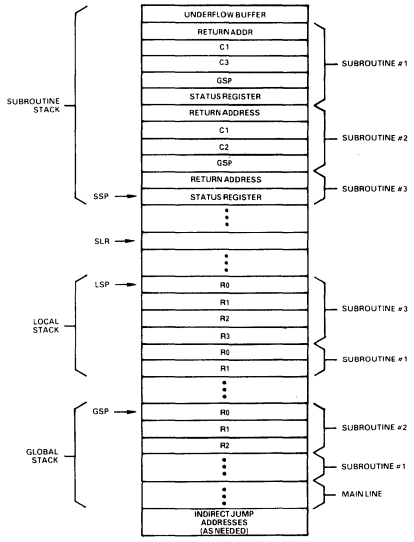


Figure 14. RAM Status after Subroutine Calls

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 "*".

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

Mnemonic	Opcode (I_{9-0})	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 11i i	IF SIGN OF C_i: JUMP DATA, $C_i <= R_i$; ELSE, $C_i <= C_i - 1$
*JRC	110 cc i i	IF COND: JUMP R_i
JRS	110 11i i	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 cc i i	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 10i i	ADD i TO RSP
S1RSP	000 1111	SUBTRACT 1 FROM RSP
S4RSP	011 1100	SUBTRACT 4 FROM RSP
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 10i i	WRITE C_i
CLRS	001 0100	CLEAR SIGN BIT
SETS	011 0100	SET SIGN BIT
PSCNTR	000 10i i	PUSH C_i ONTO SS
PPCNTR	001 10i i	POP C_i FROM SS
DCCNTR	011 00i i	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

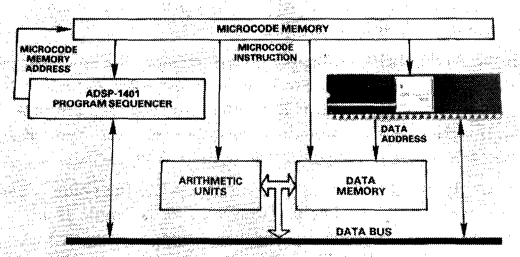
PIN	FUNCTION	PIN	FUNCTION
1	D7	52	D6
2	D8	51	D6
3	D9	50	D4
4	D10	49	D3
5	D11	48	D2
6	D12	47	D1
7	GND	46	GND
8	D13	45	D0
9	D14	44	CLK
10	D15	43	FLAG
11	EXIR1	42	I6
12	EXIR2	41	I5
13	GND	40	V_{DD}
14	EXIR3	39	I4
15	EXIR4	38	I3
16	TTR	37	I2
17	Y15	36	I1
18	Y14	35	I0
19	Y13	34	Y0
20	GND	33	GND
21	Y12	32	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 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

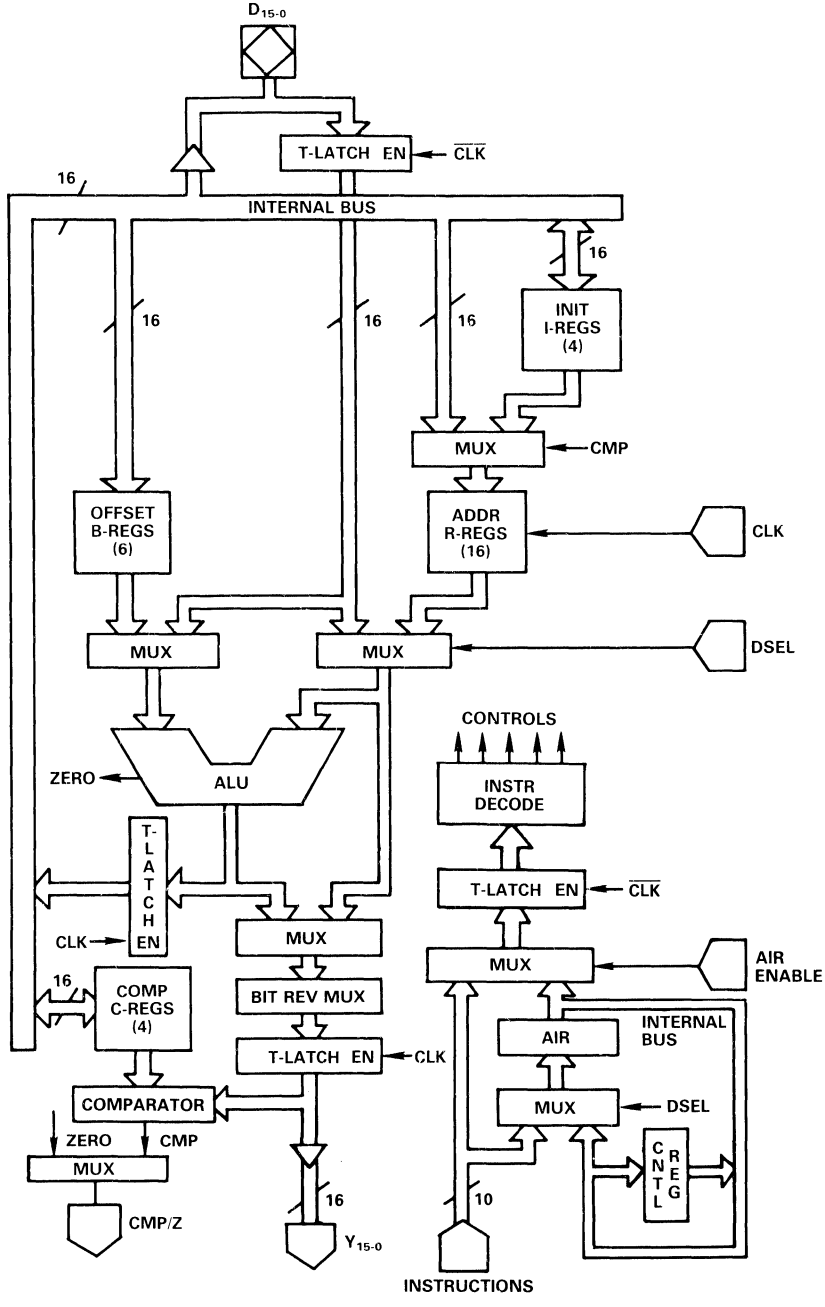


Figure 1. ADSP-1410 Functional Block Diagram

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 and 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.

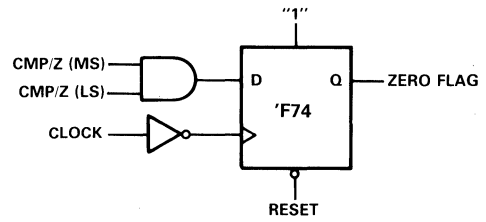


Figure 2. Valid Two-Chip Double-Precision ZERO (Logical Instructions)

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}.

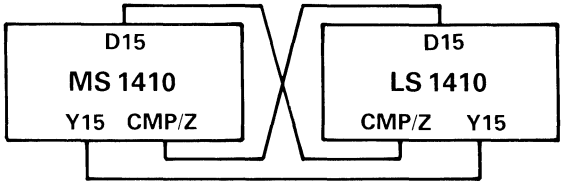


Figure 3. Two-Chip/Double Precision Handshaking

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.

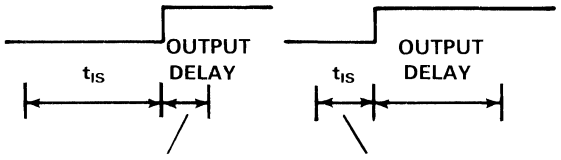
The clock input to the ADSP-1410 is a single, two-phase clock with cycle time: t_{CY}.

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 $t_{ISmin} = 15$ns and $30\text{ns} \leq t_{LAn} \leq 35\text{ns}$):

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 3–38)

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		3.0		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/+	−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

CAUTION:

1. ESD sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts.
2. Do not insert this device into powered sockets. Remove power before insertion or removal.

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	3		3		3		3		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		5		5		ns
t_{ODD}	Guaranteed Clock-to-Data Delay[5]	35	55	30	50	45	70	40	60	ns
t_{DENA}	Output Data Enable Times[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		55		57		70		55	ns
t_{CSS}	Carry/Shift Input Setup Time	45		40		50		45		ns
t_{MSD}	Carry/Shift Input to Valid MS Address (Post-Update Only)		55		40		65		55	ns
t_{CADc}	Valid Instruction to MS CMP/Z (Compare) Flag Delay									ns
t_{CZI}	Clock High to CMP/Z (Compare) Invalid Delay									ns
t_{CZDz}	Valid Instruction to CMP/Z (Zero) Flag Delay									ns
t_{IID}	Instruction Invalid to CMP/Z (Zero) Invalid Delay									ns

NOTES

*Specification 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-1410 can be found in Analog Devices' Military Databook. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 10.

[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.

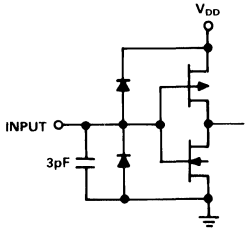


Figure 5. Equivalent Input Circuits

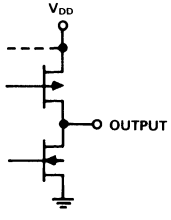


Figure 6. Equivalent Output Circuits

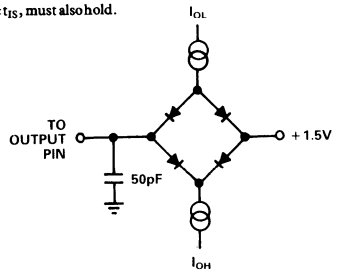


Figure 7. Normal Load for AC Measurement

(continued from page 3–35)

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; $t_{HI} \geq (t_{CSD} + t_{MSD}) - t_{IS}$.

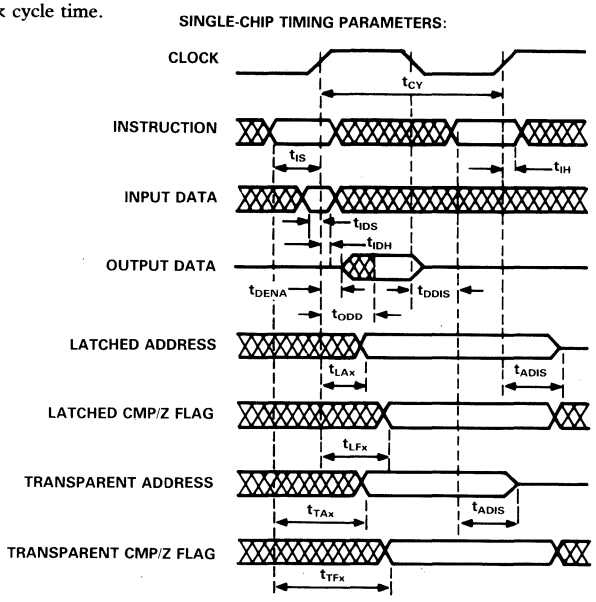


Figure 8. Timing Diagram

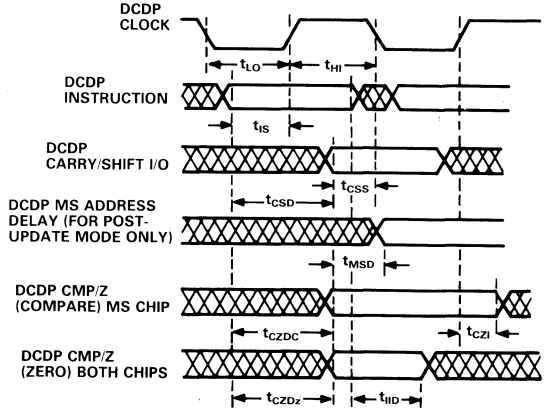


Figure 9. Supplemental Parameters for Double-Chip/Double Precision Operation

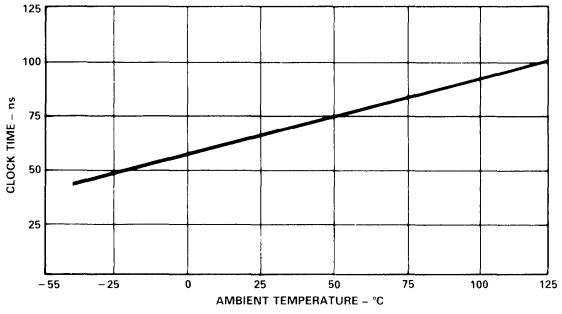


Figure 10. Clock Cycle Time vs. Temperature

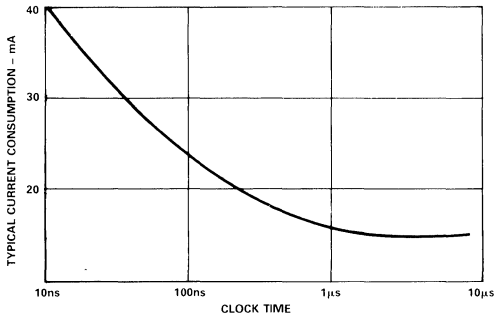


Figure 11. Typical I_{DD} vs. Frequency of Operation

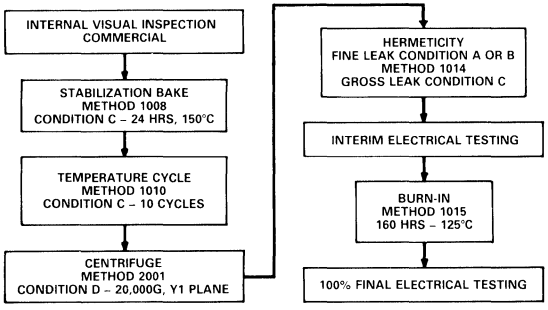


Figure 12. PLUS Processing Environmental Flow

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

Floating-Point Components

Contents

Since the introduction of our first floating-point chips in 1984, Analog Devices has been an innovative leader in supplying fast, floating-point processors. We now produce three floating-point chipsets (each consisting of a multiplier and an ALU), with a fourth chipset (the ADSP-3212 & ADSP-3222) being introduced as this catalog appears. All parts implement the IEEE Standard 754 for Binary Floating-Point Arithmetic; the next generation also supports DEC formats F and G. All deliver the highest performance in throughput and latency with the advantages of CMOS processing. Our introduction of a fast multiported register file (the ADSP-3128) provides local high-speed memory to enable the system to maximize computational throughput and provide operand transfer flexibility. The register file and floating-point chips are supported by our Word-Slice family for microcode processors; see Chapter 3.

The floating-point processors provide tradeoffs in performance, price and precision. All attain high pipelined throughput while minimizing latency with only one internal pipeline register. The key advantages of each particular chipset are summarized below and in the Selection Guide on the following 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 Floating-Point 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 Floating-Point ALUs
The ADSP-3221 is a three-port ALU attaining throughput of 10 MFLOPS single-precision, 10 MFLOPS double-precision and 10 MIPS fixed-point. The ADSP-3221, pin-compatible with the ADSP-3220, also has the capability to compute (completely on-chip) the IEEE exact division and square root functions. The ADSP-3220 ALU, also with a three-port structure, attains 10 MFLOPS/MIPS throughput for all three data formats but does not provide division and square root functions.

ADSP-3201/ADSP-3202 SINGLE-PRECISION FLOATING-POINT PRODUCTS
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-3128 MULTIPORT REGISTER FILE
This 128×16 or 64×32 register file provides high-speed local storage for the floating-point components, 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 all our 1.5μm floating-point parts. One bidirectional port provides an interface with another processor or external system. The ADSP-3128 contains on-chip the latches for a multitude of system interfacing requirements without external glue logic. On-chip multiplexers automatically sequence double-precision data transfers.

ADSP-3212 MULTIPLIER & ADSP-3222 ALU FLOATING-POINT CHIPSET
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 advanced CMOS processing 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. At press time, these products had just been introduced; consult your Sales Office for the current data sheet and specifications.

Selection Guide

FLOATING POINT COMPONENTS

Part	Grade	Number of Ports	Pipelined Throughput (ns) Single Precision	Double Precision	32-Bit Fixed	Latency (ns) Single Precision	Double Precision	32 Bit Fixed	IEEE Exact Divide (μs) Single Precision	Double Precision	IEEE Exact Square Root (μs) Single Precision	Double Precision
	L	3	50	200	50	140	315	140				
ADSP-3211	K	3	100	400	100	240	590	240				
Multiplier	J	3	125	500	125	300	738	300				
	U	3	70	280	70	190	400	190				
	T	3	125	500	125	300	738	300				
	S	3	150	600	150	360	885	360				
	L	2	60	240	60	190	370	190				
ADSP-3210	K	2	100	400	100	290	590	290				
Multiplier	J	2	125	500	125	363	738	363				
	U	2	75	300	75	238	463	238				
	T	2	125	500	125	363	738	363				
	S	2	150	600	150	435	885	435				
	K	3	100	400	100	240	290	240	1.6	3	2.9	5.8
ADSP-3221	J	3	125	500	125	300	363	300	2	3.75	3.63	7.25
ALU	T	3	125	500	125	300	363	300	2	3.75	3.63	7.25
	S	3	150	600	150	360	435	360	2.4	4.5	4.35	8.7
	K	3	100	400	100	240	290	240				
ADSP-3220	J	3	125	500	125	300	363	300				
ALU	T	3	125	500	125	300	363	300				
	S	3	150	600	150	360	435	360				
ADSP-3201	K	3	100		100	240		240				
Single-	J	3	125		125	300		300				
Precision	T	3	125		125	300		300				
Multiplier	S	3	150		150	360		360				
ADSP-3202	K	3	100		100	240		240	1.6		2.9	
Single-	J	3	125		125	300		300	2		3.63	
Precision	T	3	125		125	300		300	2		3.63	
ALU	S	3	150		150	360		360	2.4		4.35	

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 Max 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

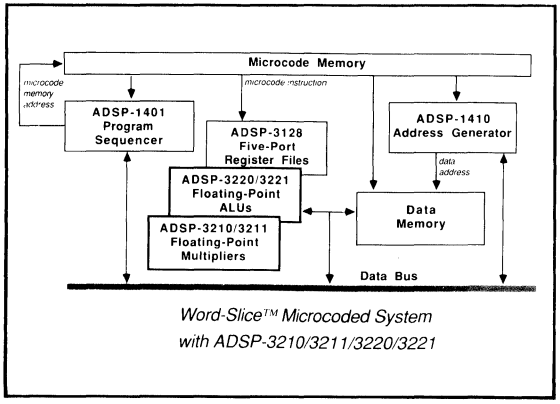
Array Processors

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 Floating-Point Multiplier adds a second input port and doubles the number of input registers to eight. It executes all ADSP-3210

Word-Slice is a trademark of Analog Devices, Inc.

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 with optional high-reliability processing ("PLUS" parts, see Figure 37) or 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.

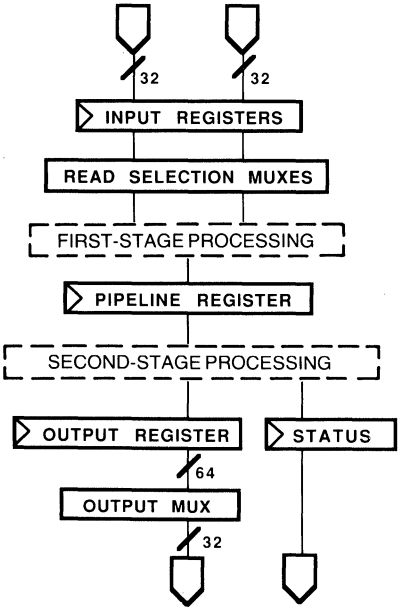


Figure 1. ADSP-3210/3211/3220/3221 Generic Architecture

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)	

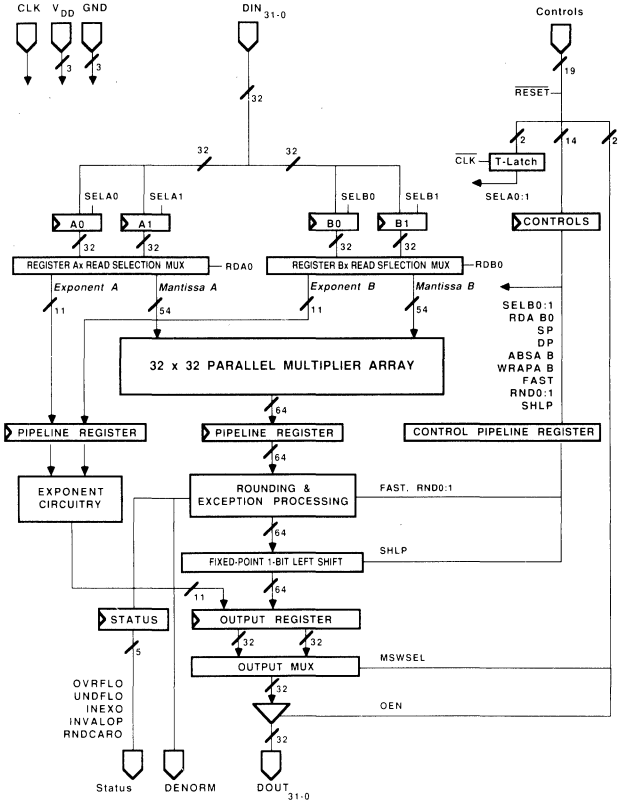


Figure 3. ADSP-3210 Functional Block Diagram

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)	

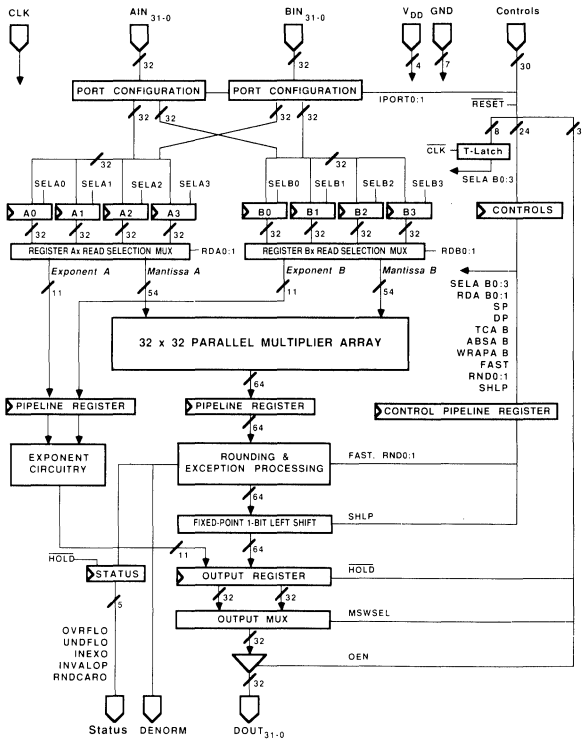


Figure 4. ADSP-3211 Functional Block Diagram

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)	

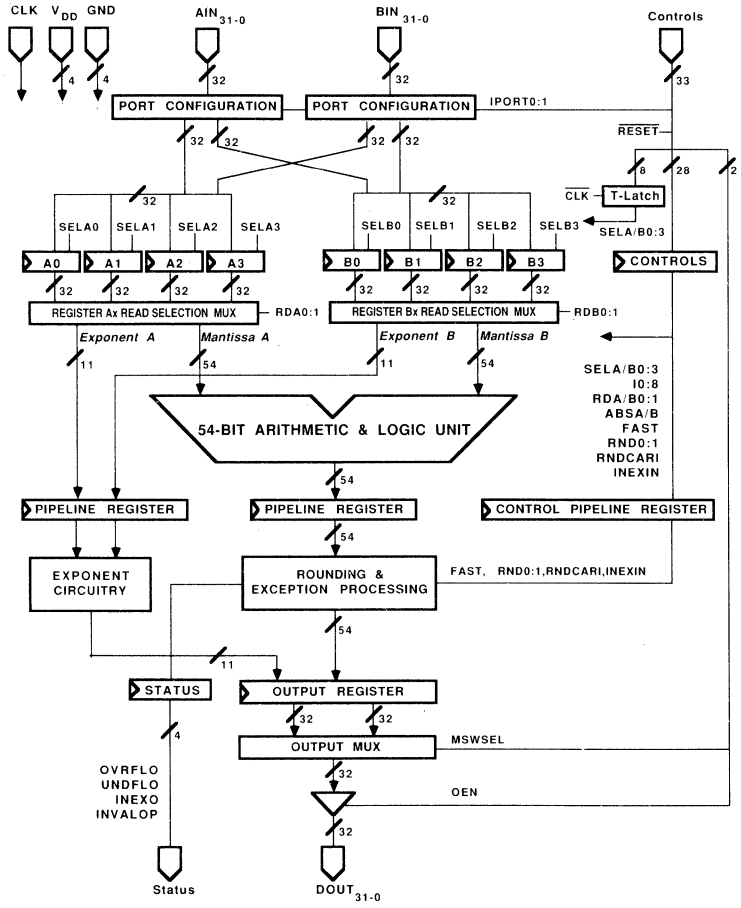


Figure 5. ADSP-3220/3221 Functional Block Diagram

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

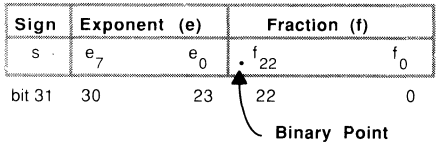


Figure 6. Single-Precision Floating-Point Format

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

The IEEE Standard also provides for several special data types. In the single-precision floating-point format, an exponent value of 255 (all ones) with a 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.)

Double-Precision Floating-Point Data Format

IEEE Standard 754 specifies a 64-bit double-precision floating point format:

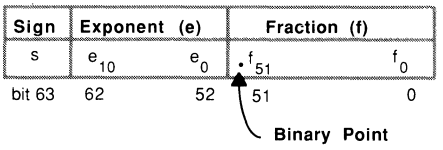


Figure 7. Double-Precision Floating-Point Format

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.

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

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.

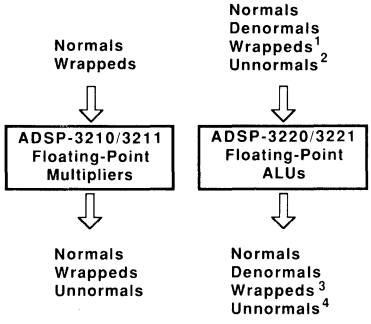
The ranges for the various double-precision data types are summarized in Table IV.

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 from floating-point to fixed-point. For these operations, the destination 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.) No register contents or status flags are affected by $\overline{\text{RESET}}$.

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 operands at the clock rate, but a single input port could not accept operands at that rate.

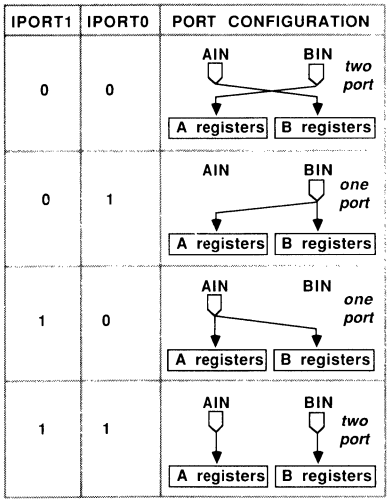


Figure 15. ADSP-3211/3220/3221 Input Port Configurations

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:

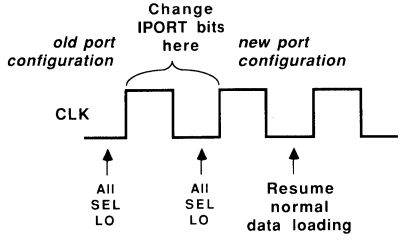


Figure 16. Timing Requirements for Changing the ADSP-3211/3220/3221 Input Port Configurations

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:

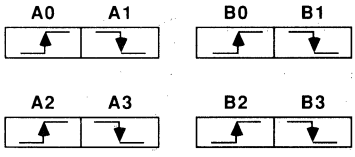


Figure 18. ADSP-3211/3220/3221 Clock Edge for Data Loading – Two-Port Configuration

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:

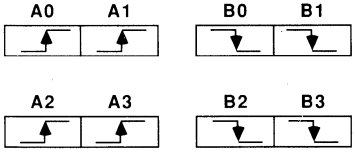


Figure 19. ADSP-3211/3220/3221 Clock Edge for Data Loading – One-Port Configuration

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:

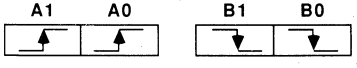


Figure 20. ADSP-3210 Clock Edge for Data Loading

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:

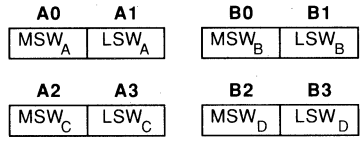


Figure 21. ADSP-3211/3220/3221 Operand Storage for Double-Precision Operations

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.

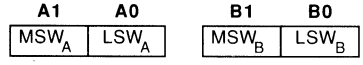


Figure 22. ADSP-3210 Operand Storage for Double-Precision Operations

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	RDB0	SP & Fixed: B register selected	DP: B registers selected
0	A1	illegal state	0	B1	illegal state
1	A0	A1, A0	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.

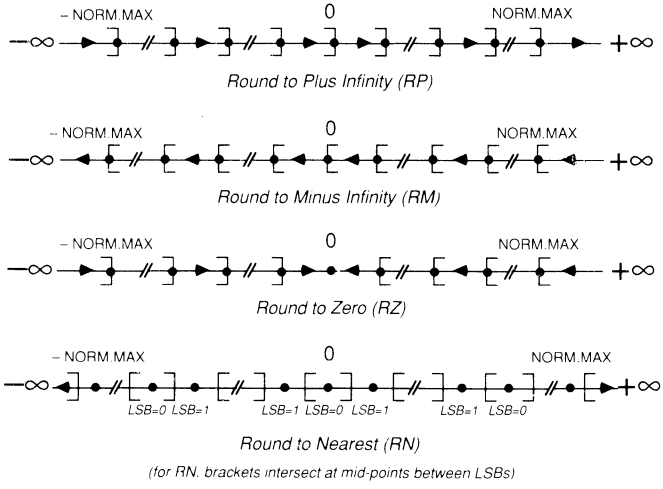


Figure 28. IEEE Rounding Modes

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.

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.
DITRNEA	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-3220/3221 ALUs, the sign of a NAN result when one input is a NAN will be the sign of the NAN operand. The sign of the NAN result for two NAN inputs (except division) will be the sign of the NAN with the larger magnitude fraction. If the fractions are equal, then the sign will be computed in the same way as for additions on signed zeros:

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

In this notation, the first line refers to either $+\text{ZERO}+\text{ZERO}$ or $-\text{ZERO}-\text{ZERO}$. The second and third lines refer to $+\text{ZERO}-\text{ZERO}$ or $-\text{ZERO}+\text{ZERO}$. Some ALU operations with two INF inputs can cause INVALOP and generate NANs.

The assignment of sign to the NAN is also 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)

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.

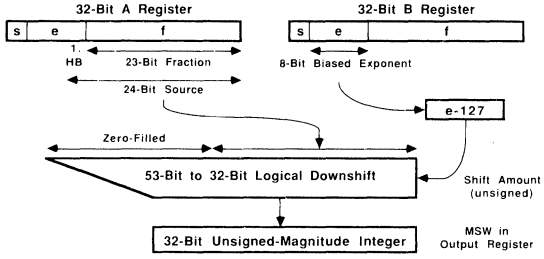


Figure 29. ADSP-3220/3221 SITRN Instruction

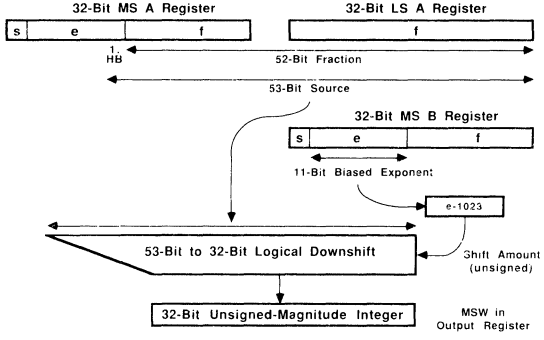


Figure 30. ADSP-3220/3221 DITRN Instruction

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.

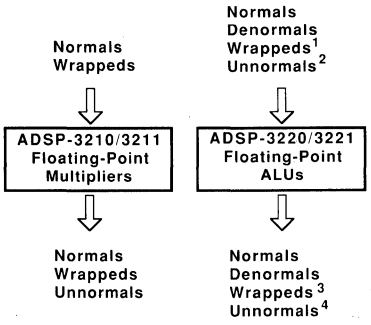
The asynchronous MSWSEL control's delay is t_{ENO}. The maximum specification for t_{ENO} is the delay which guarantees valid data. The minimum specification for t_{ENO} is the earliest time after the MSWSEL control is changed that data can change.

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	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	1	0 . . . 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.

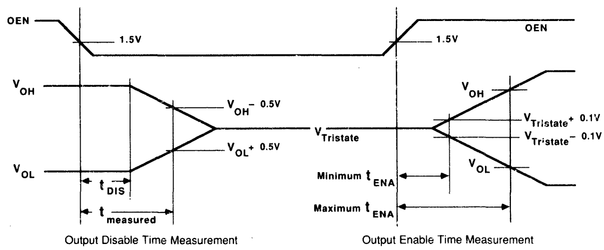
NOTE: Heavy line indicates rounding boundary in source.

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	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

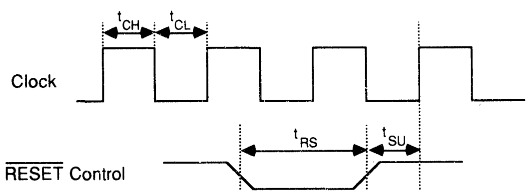


Figure T2. ADSP-3210/3211/3220/3221 Reset Timing

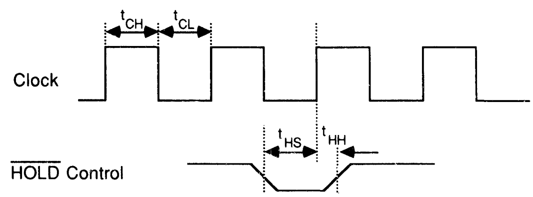
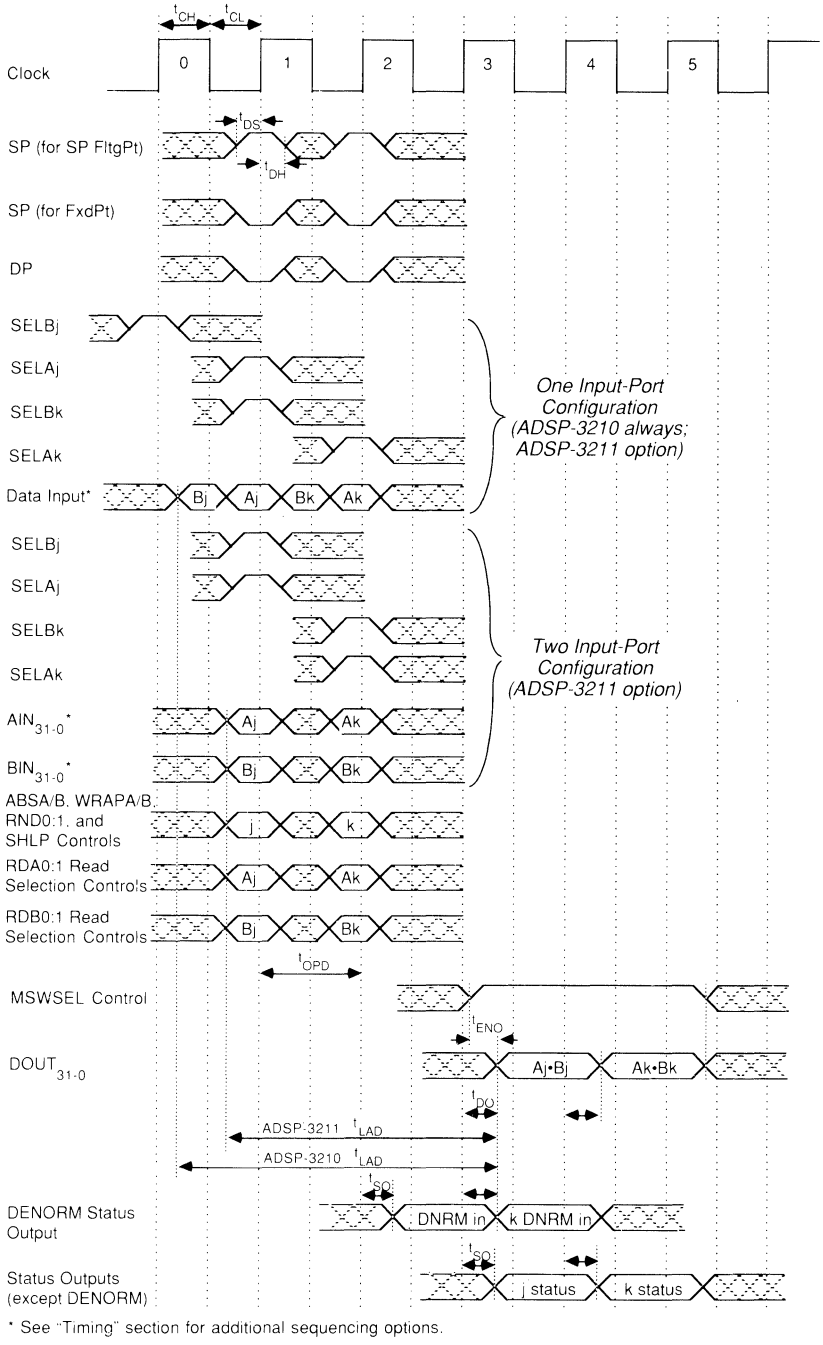
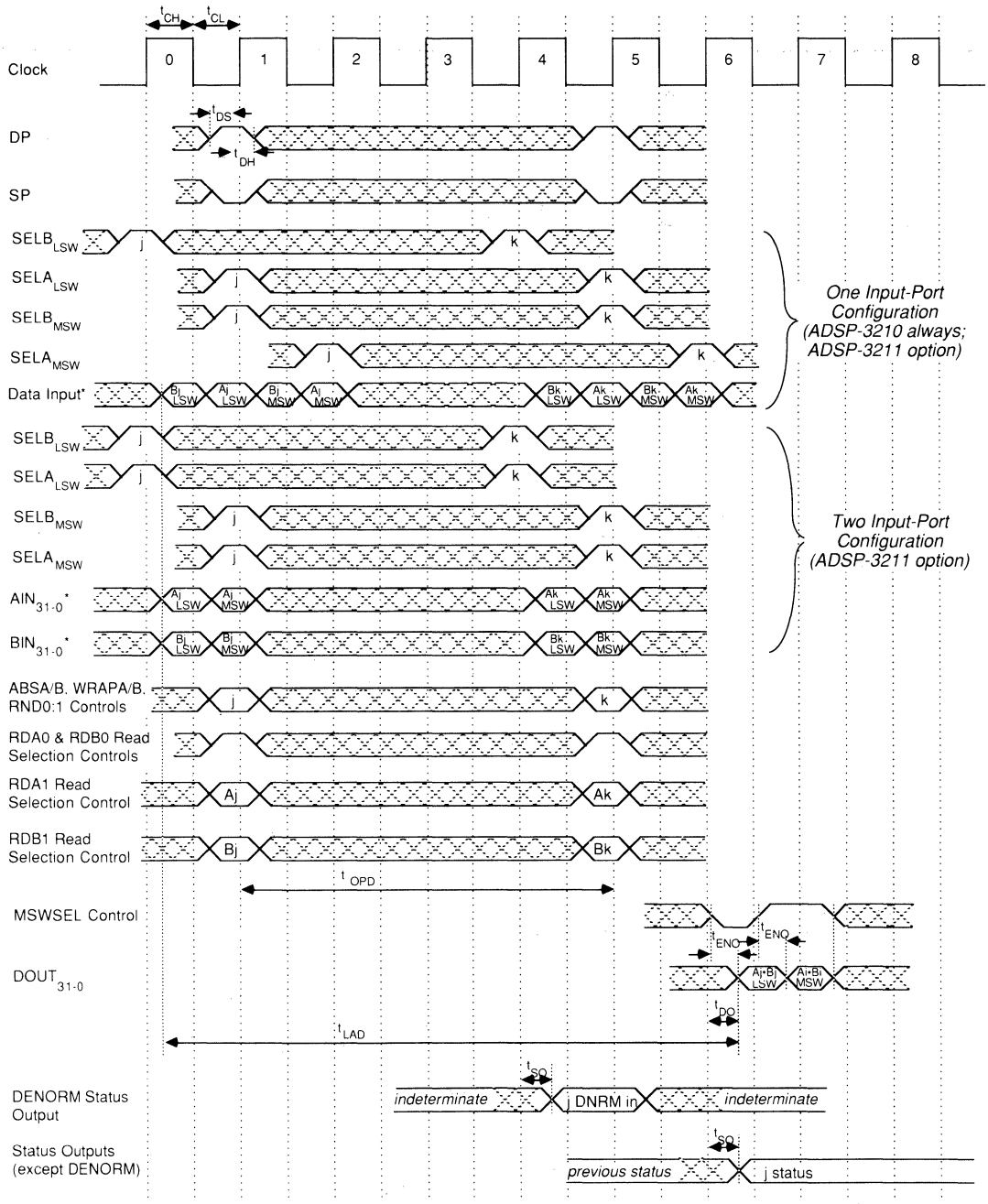


Figure T3. ADSP-3211 Multiplier Output Register Hold Timing



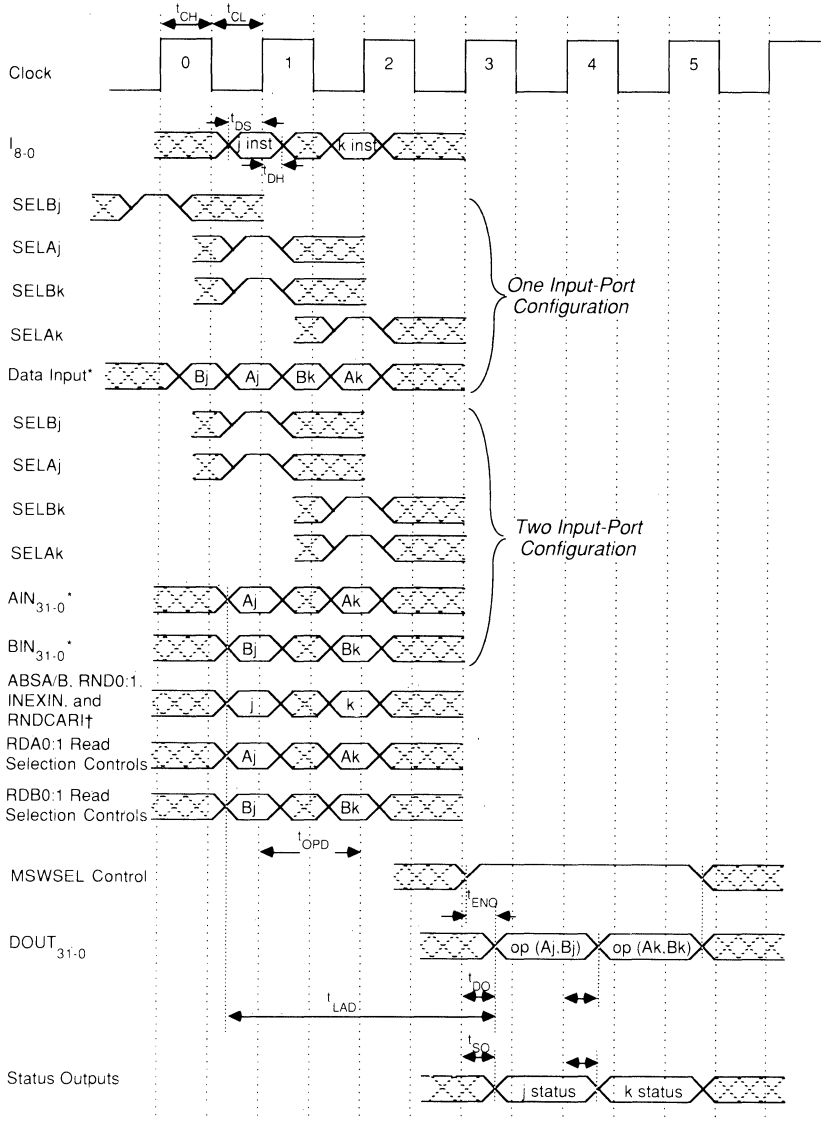
* See "Timing" section for additional sequencing options.

Figure T4. ADSP-3210/3211 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplications



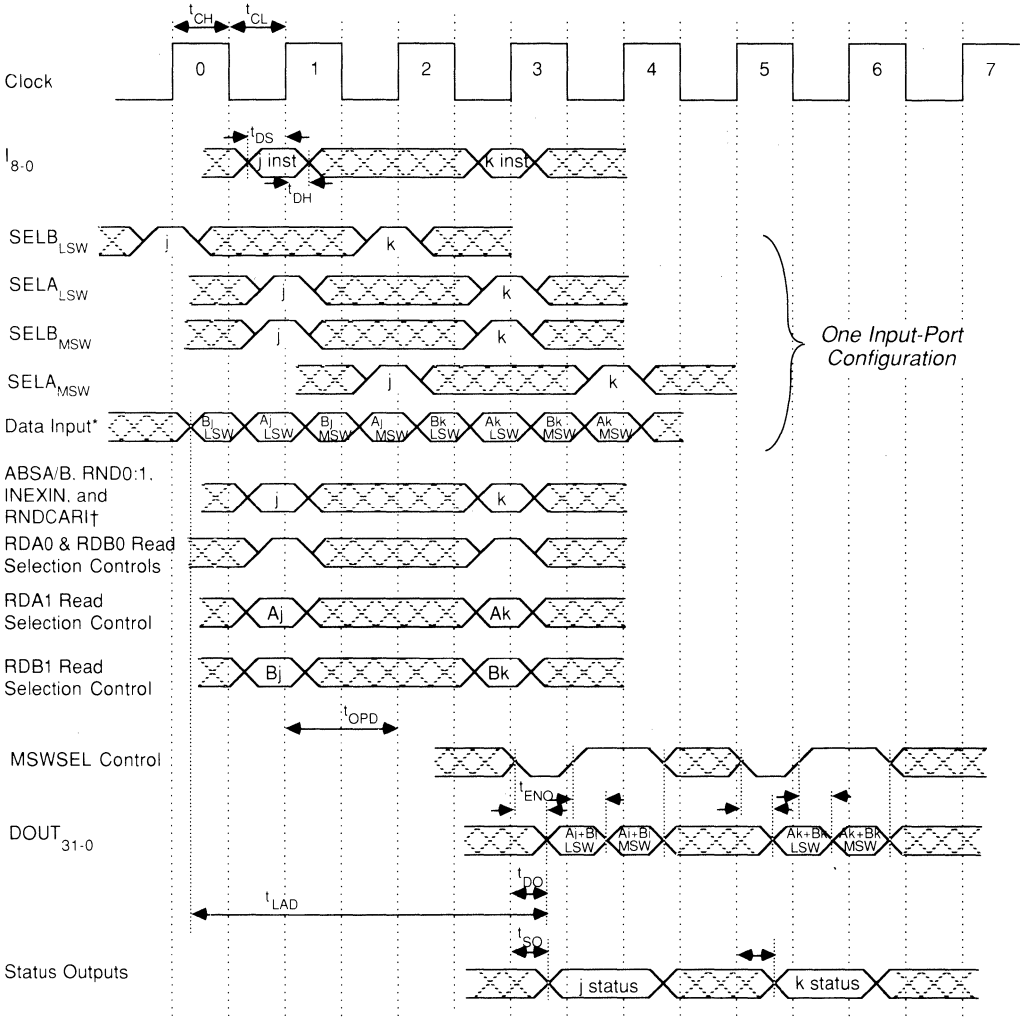
* See "Timing" section for additional sequencing options.

Figure T5. ADSP-3210/3211 64-Bit Double-Precision Floating-Point Multiplications



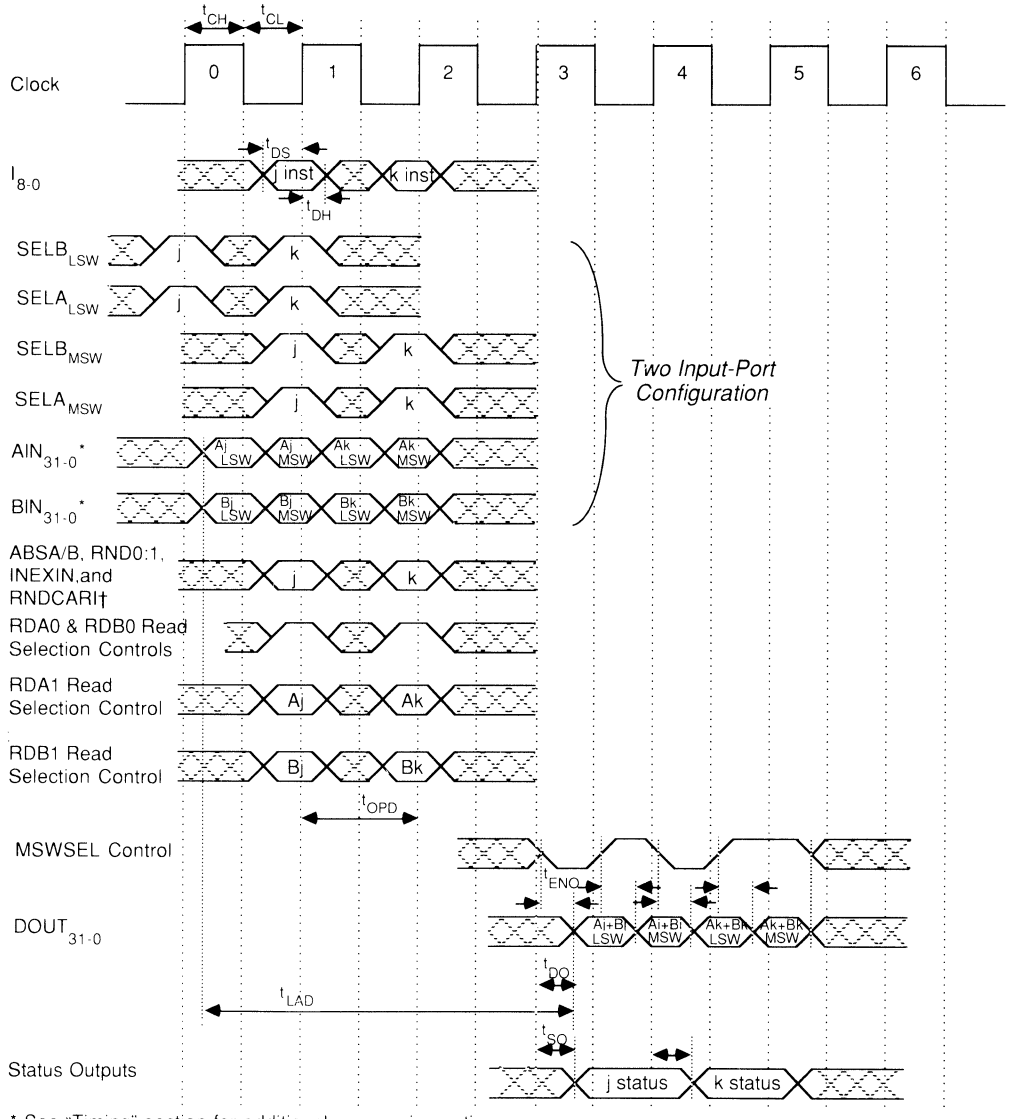
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T6. ADSP-3220/3221 32-Bit Single-Precision Floating-Point Logical, and Fixed-Point ALU Operations



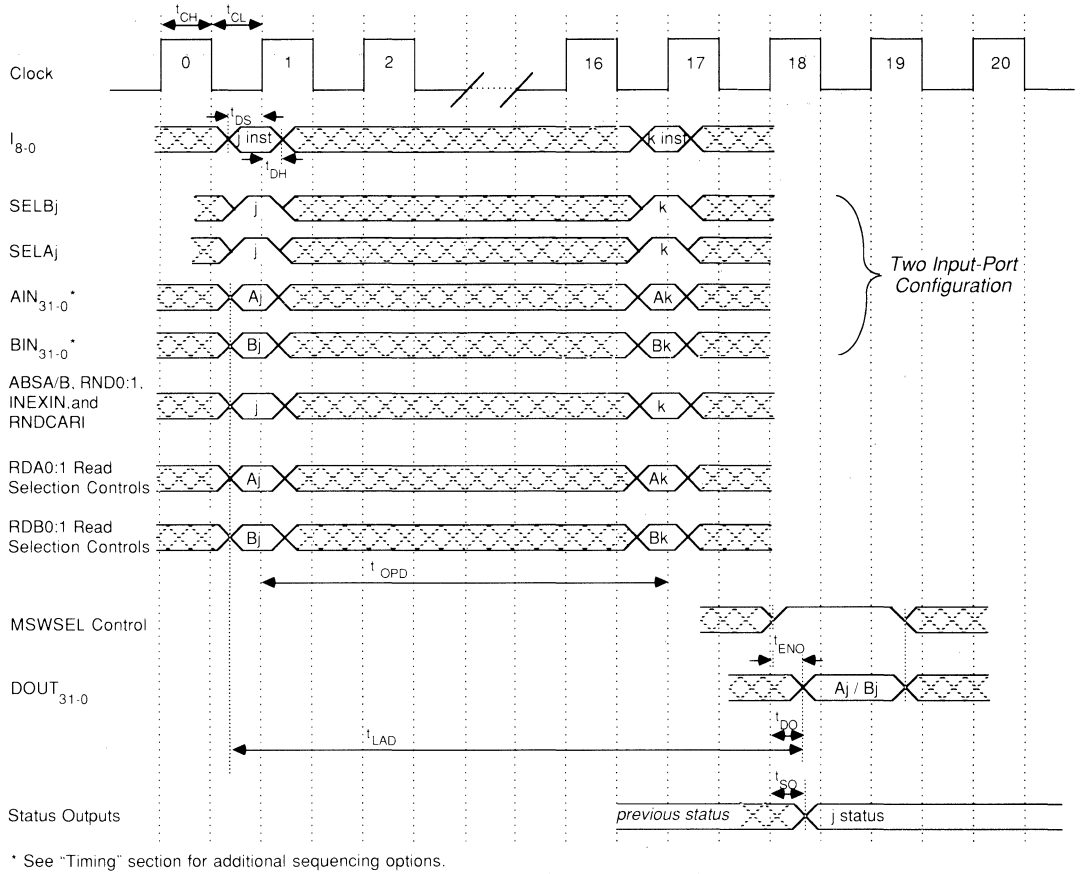
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T7. ADSP-3220/3221 64-Bit Double-Precision Floating-Point ALU Operations – One-Port Configuration



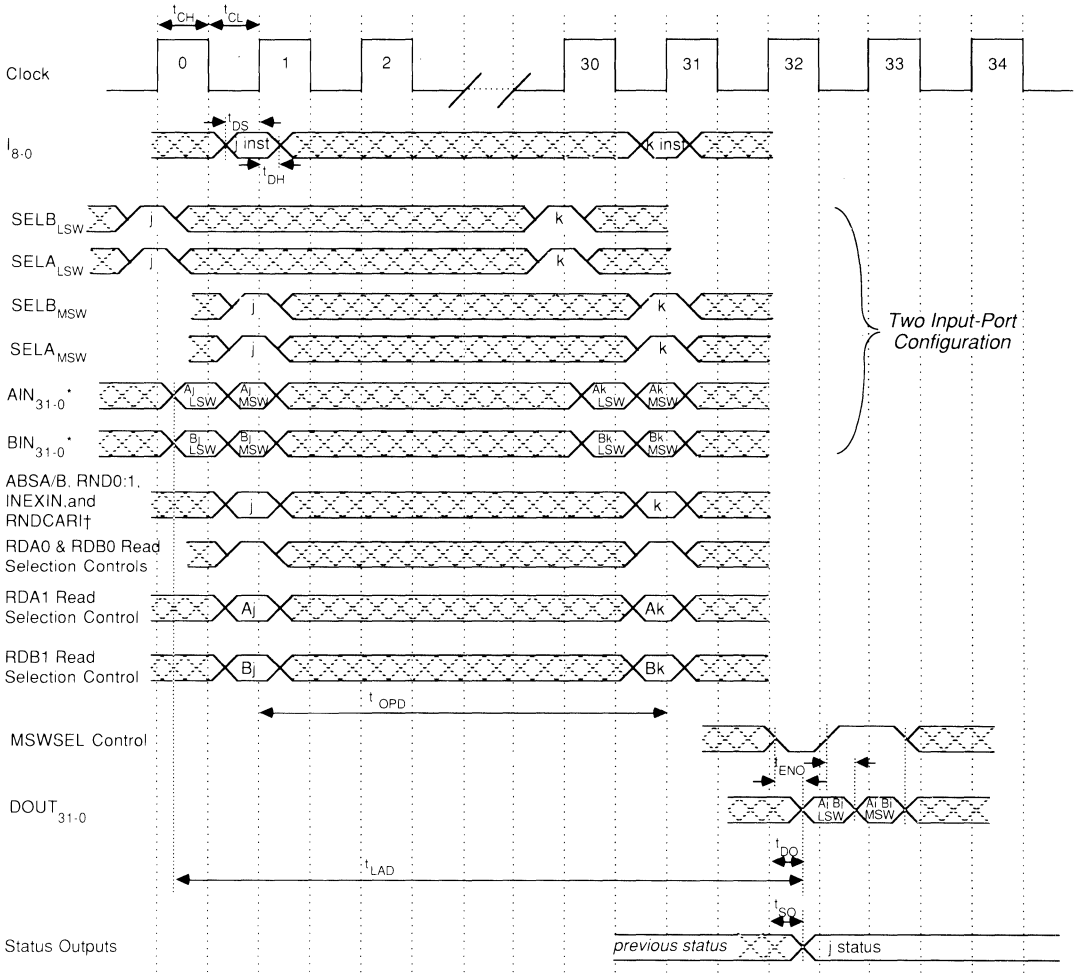
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T8. ADSP-3220/3221 64-Bit Double-Precision Floating-Point ALU Operations – Two-Port Configuration



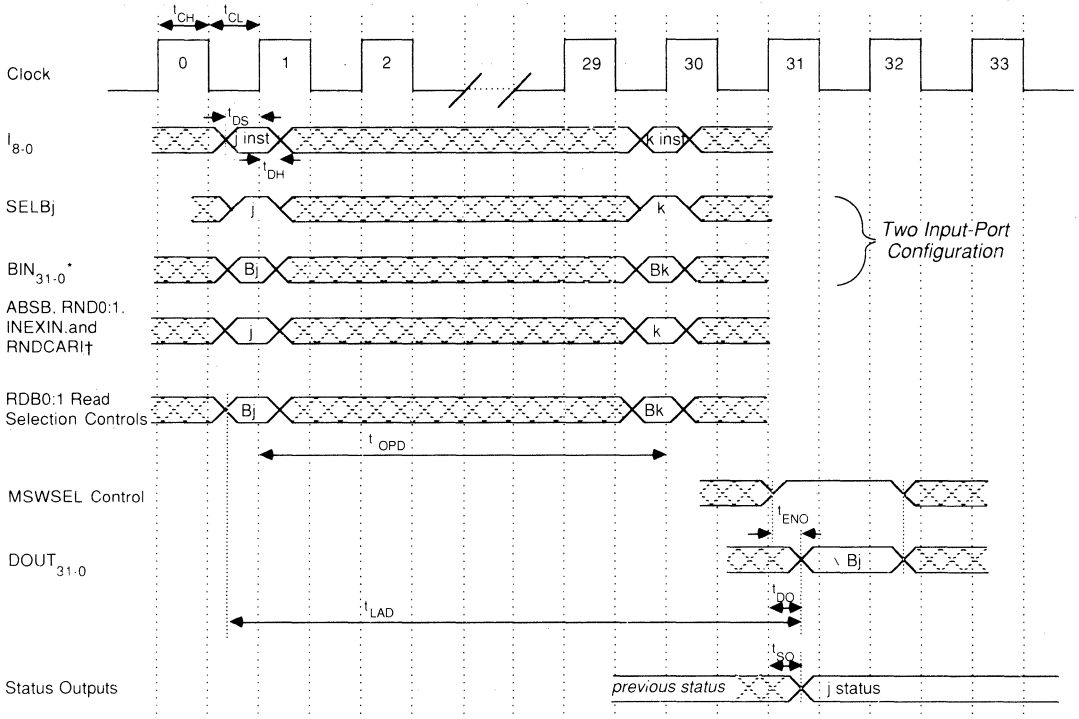
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T9. ADSP-3221 32-Bit Single-Precision Floating-Point Division – Two Input Port Configuration



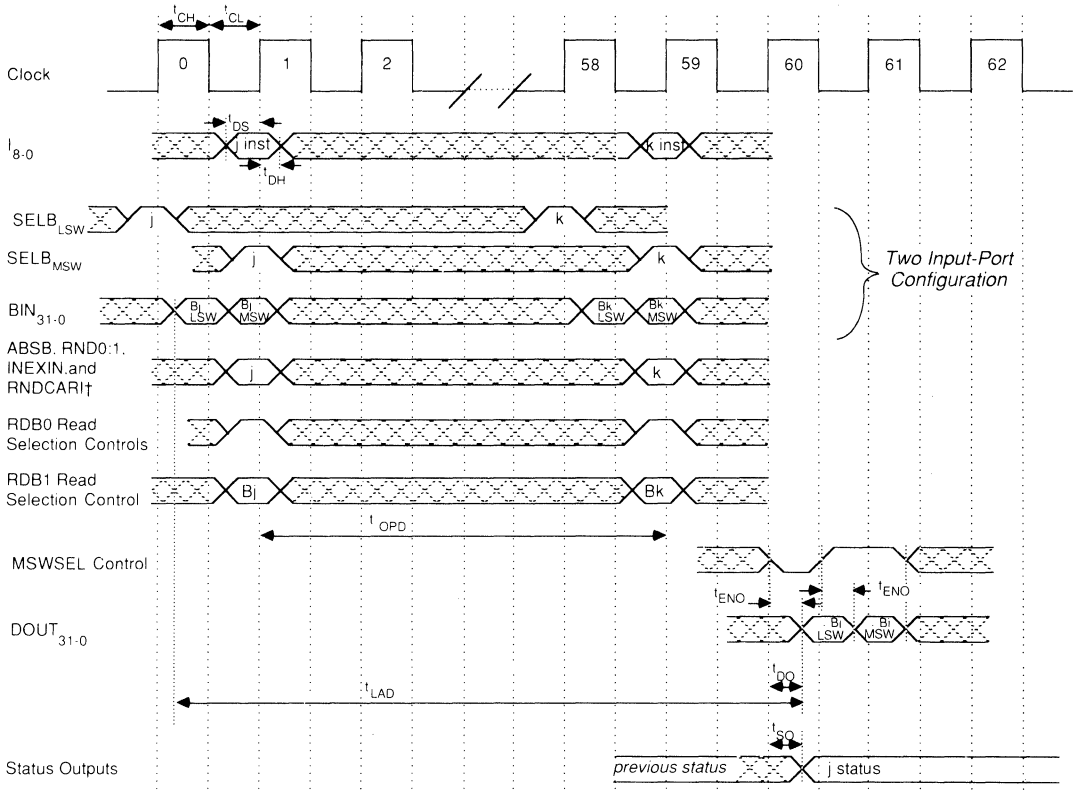
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

Figure T10. ADSP-3221 64-Bit Double-Precision Floating-Point Division – Two Input-Port Configuration



* 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

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3210/3211/3220/3221 J K, and L Grades Min	Max	S, T, and U Grades[2] 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

Parameter		Test Conditions	ADSP-3210/3211/3220/3221 J, K, and L Grades Min	Max	S, T, and U Grades[2] Min	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 Grades 0 to 70°C Min	Max	K Grades 0 to 70°C Min	Max	S Grades[2] −55°C to +125°C Min	Max	T Grades[2] −55°C to +125°C Min	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	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

Parameter		ADSP-3210/3211/3220/3221 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[2] −55°C to +125°C Min	S Grades[2] −55°C to +125°C Max	T Grades[2] −55°C to +125°C Min	T Grades[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		18		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 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_{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 Data Book. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 37.
[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.

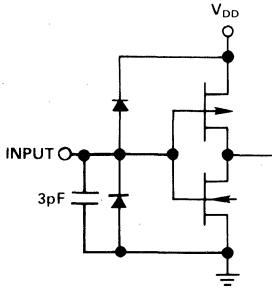


Figure 34. Equivalent Input Circuits

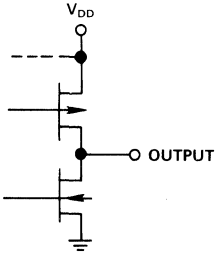


Figure 35. Equivalent Output Circuits

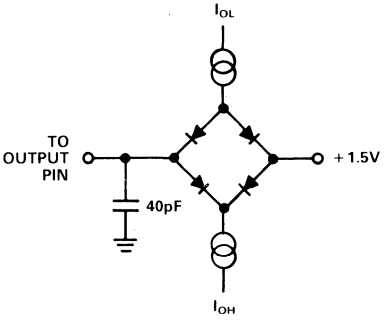


Figure 36. Normal Load for ac Measurements

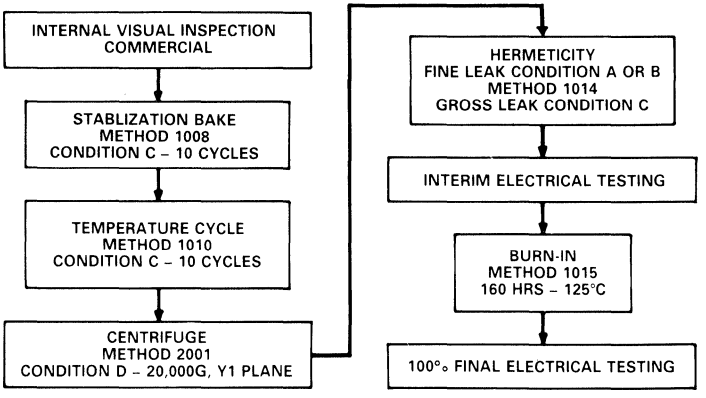


Figure 37. PLUS Processing Flow

ORDERING INFORMATION

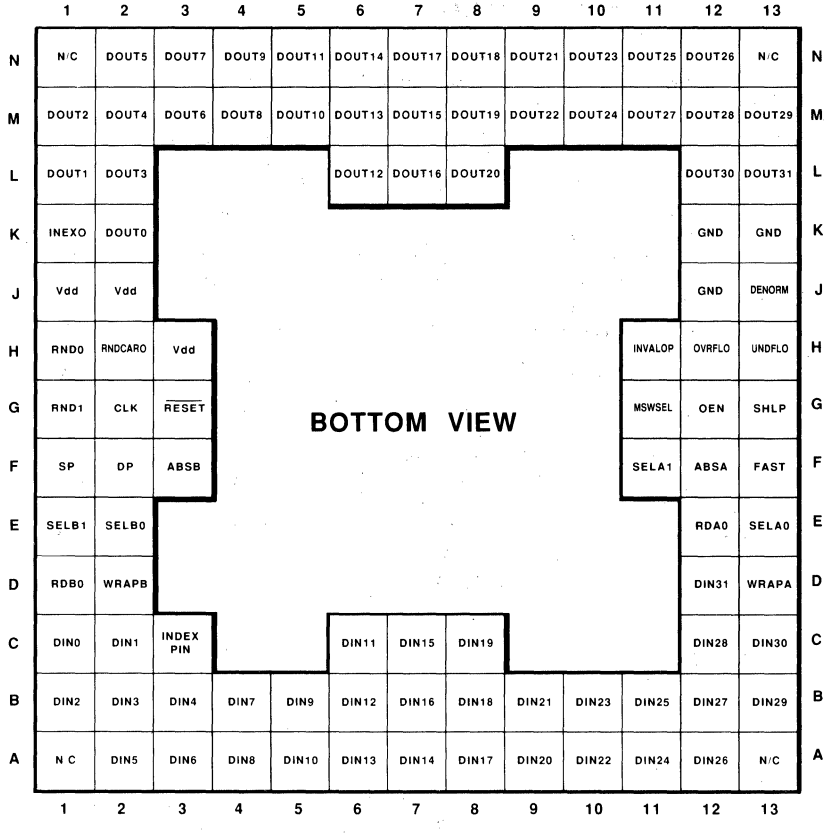
Part Number	Temperature Range	Package	Package Outline
ADSP-3210JG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210KG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210LG	0 to +70°C	100-Pin Grid Array	G-100A
ADSP-3210SG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210TG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210UG	−55°C to +125°C	100-Pin Grid Array	G-100A
ADSP-3210SG/+	−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/+	−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/+	−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/+	−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.

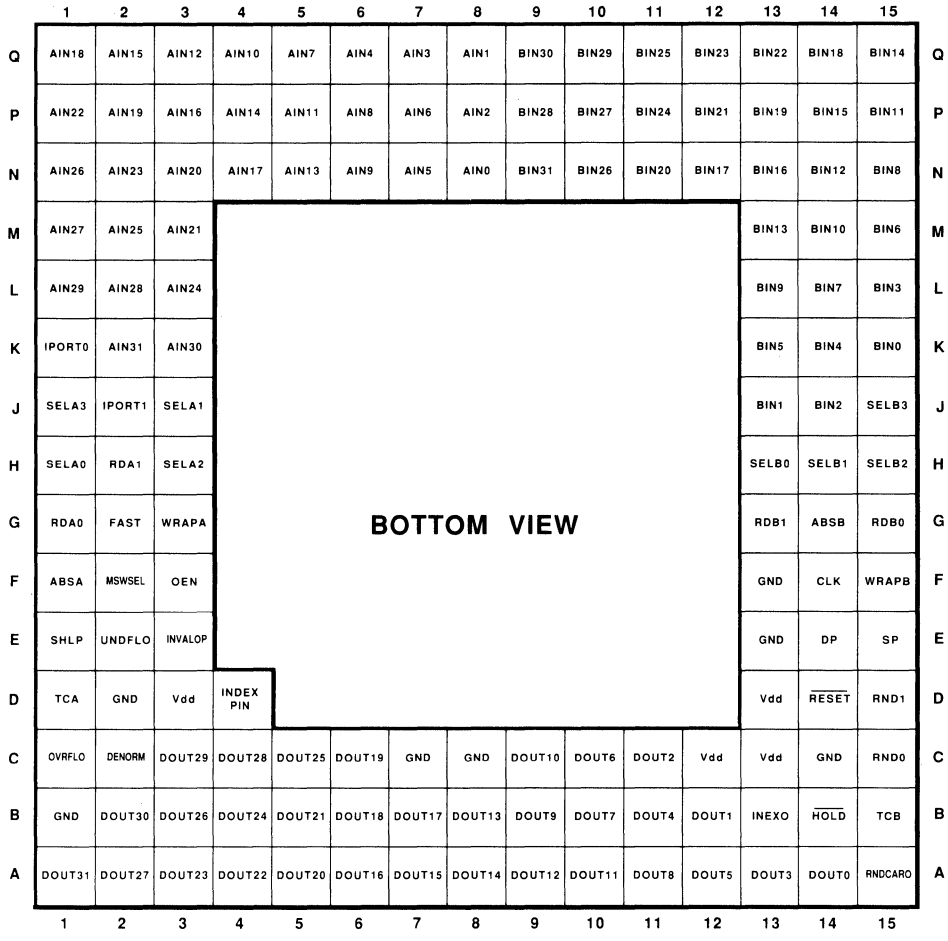
ESD SENSITIVITY

Each chip in the ADSP-3210/3211/3220/3221 chipset features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C these chips have been classified as Category A devices.

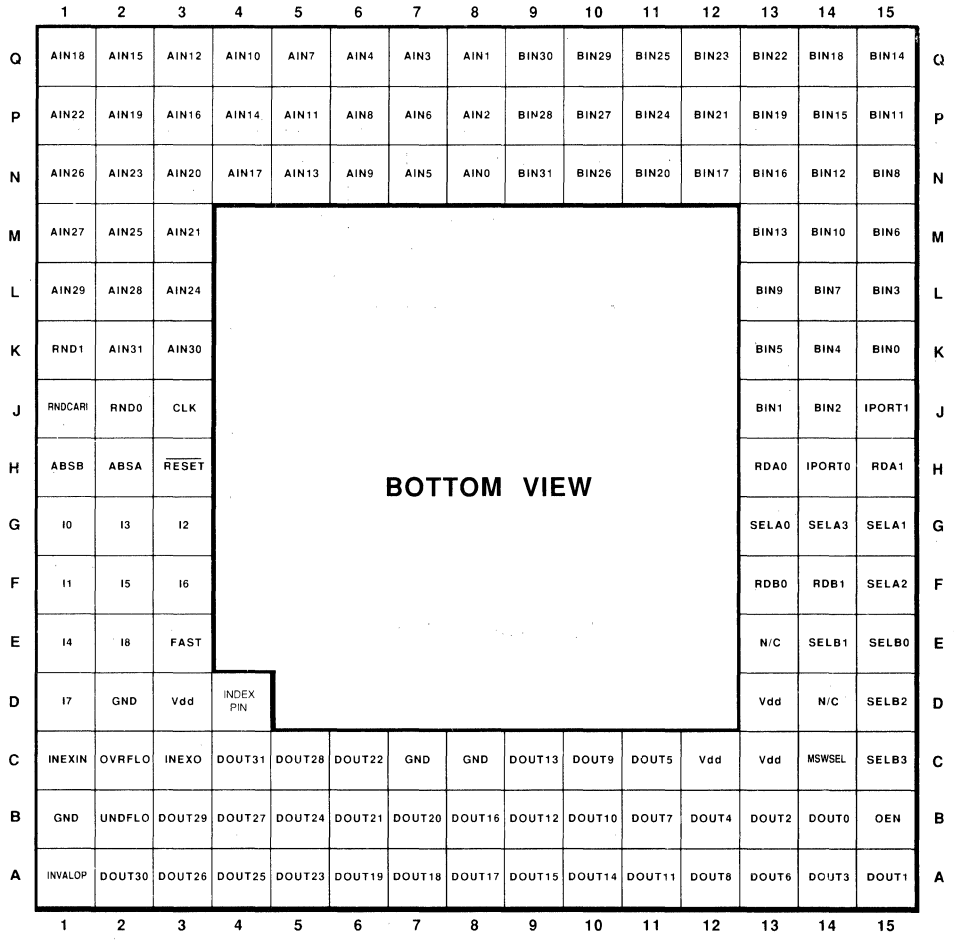
Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should 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

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 Max Power Dissipation per Chip with 1.5μm CMOS Technology
144-Lead Pin Grid Array
Available Specified to MIL-STD-883, Class B

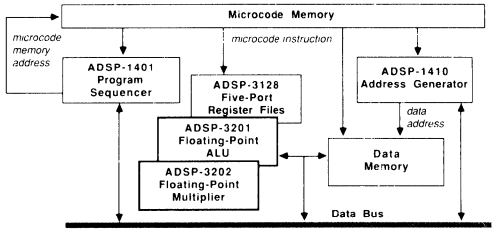
APPLICATIONS

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

GENERAL DESCRIPTION

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

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



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

4

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

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

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

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

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

The ADSP-3201/3202 are available for both commercial and extended temperature ranges. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts, see Figure 29) or 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.

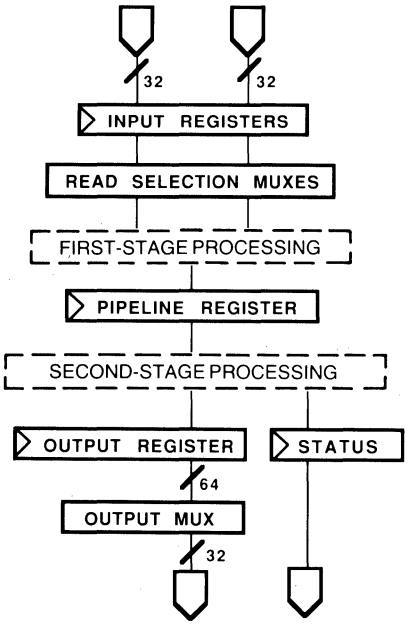


Figure 1. ADSP-3201/3202 Generic Architecture

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.

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)	

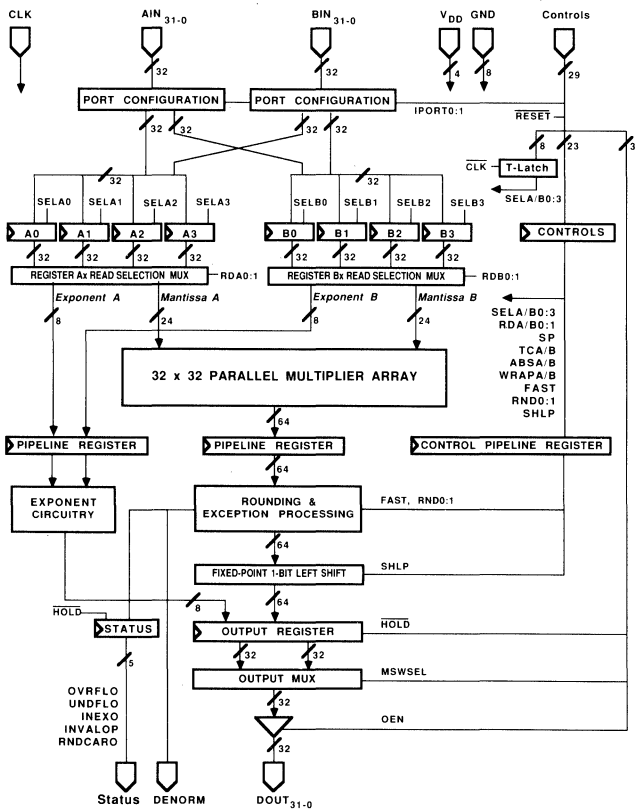


Figure 3. ADSP-3201 Functional Block Diagram

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)	

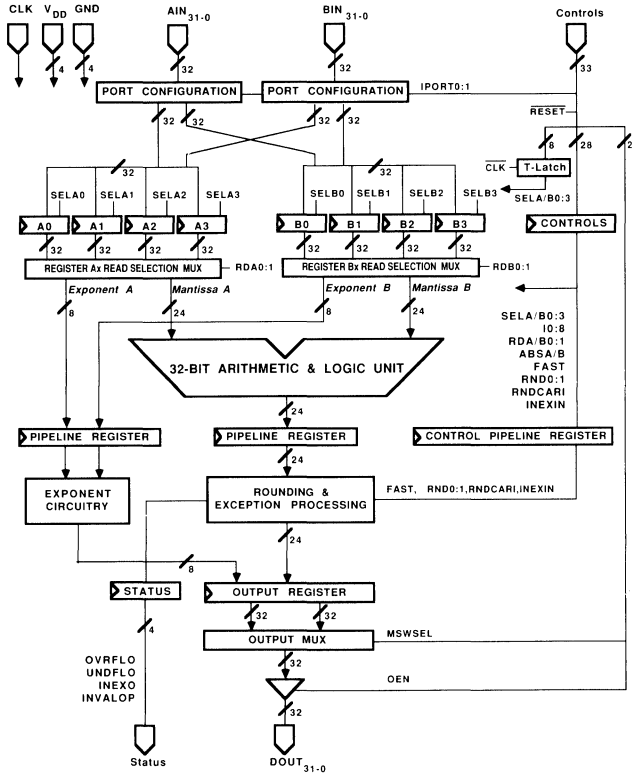


Figure 4. ADSP-3202 Functional Block Diagram

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,

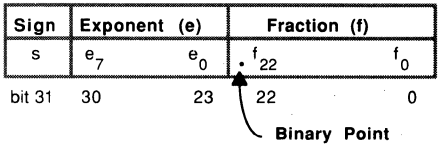


Figure 5. Single-Precision Floating-Point Format

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

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

The IEEE Standard requires the support of denormalized data formats and operations. A denormalized number, or "denormal," is a number with a magnitude less than the minimum normalized ("normal") number in the IEEE format. Denormals have a zero exponent and a nonzero fraction. Denormals have no hidden "one" bit. (Equivalently, the hidden bit of a denormal is zero.) The unbiased (true) value of a denormal's exponent is −126 in the single-precision format, i.e., one minus the exponent bias. Note that because denormals are not required to have a significant leading one bit, the precision of a denormal's significand can be as little as one bit for the minimum representable denormal.

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

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

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

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

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

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

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

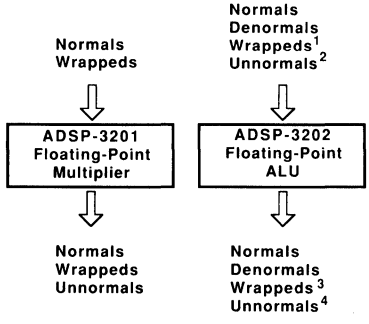
Table I. Single-Precision Floating-Point Data Types and Interpretations

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 from floating-point to fixed-point. For these operations, the destination 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.) No register contents or status flags are affected by $\overline{\text{RESET}}$.

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.

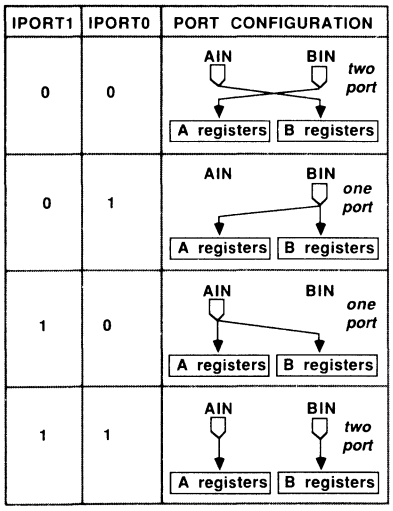


Figure 13. ADSP-3201/3202 Input Port Configurations

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:

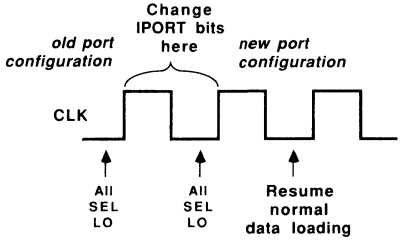


Figure 14. Timing Requirements for Changing the ADSP-3201/3202 Input Port Configurations

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:

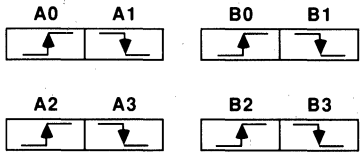


Figure 16. ADSP-3201/3202 Clock Edge for Data Loading – Two Port Configuration

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:

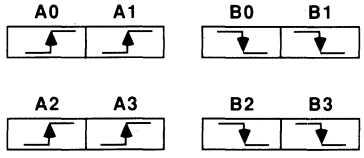


Figure 17. ADSP-3201/3202 Clock Edge for Data Loading – One Port Configuration

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.

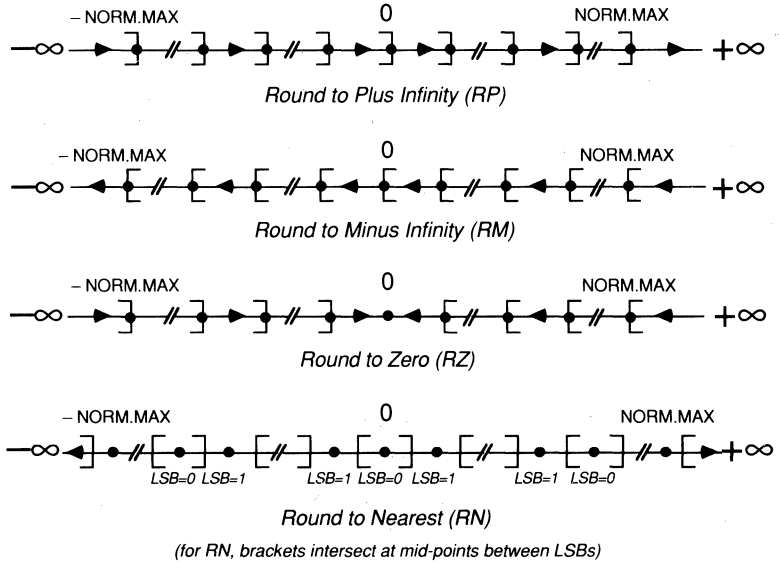


Figure 21. IEEE Rounding Modes

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.

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 (DNRM.MIN)≥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.
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-3202 ALU, the sign of a NAN result when one input is a NAN will be the sign of the NAN operand. The sign of the NAN result for two NAN inputs (except division) will be the sign of the NAN with the larger magnitude fraction. If the fractions are equal, then the sign will be computed in the same way as for additions on signed zeros:

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

In this notation, the first line refers to either $+\text{ZERO}+\text{ZERO}$ or $-\text{ZERO}-\text{ZERO}$. The second and third lines refer to $+\text{ZERO}-\text{ZERO}$ or $-\text{ZERO}+\text{ZERO}$. Some ALU operations with two INF inputs can cause INVALOP and generate NANs. The assignment of sign to the NAN is also 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)

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.

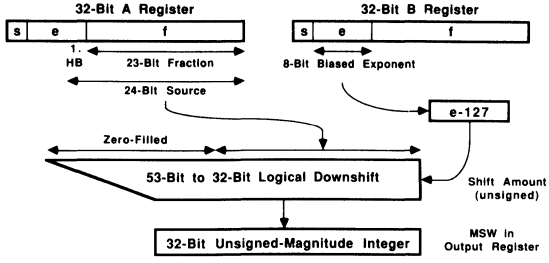


Figure 22. ADSP-3202 SITRN Instruction

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.

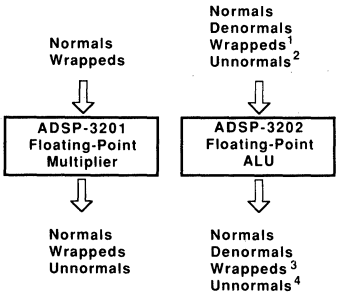
Note that for all operations (Figures T4 through T7) a new operation can begin the cycle before output results and status flags (other than DENORM) results from the previous operation are driven off chip. This feature leads to improved pipeline throughput.

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 "Rounc 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	1	0 . . .	0	0	0	0	0	0	0	1	all	
1	1	1 . . . 1	1	2**	30		1	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	1	all	
1	1	0 . . . 0	0	2**	31		1	0	0	0	0	0	0	0	0	0 . . .	0	0	0	0	0	0	0	0	all	
1	1	0 . . . 0	1	2**	31		U	U	U	U	U	U	U	U	U	U . . .	U	U	U	U	U	U	U	U	all	
1	1	0 . . . 0	0	2**	128	−INF	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP
1	1	X . . . X	X	2**	128	−NAN	1	1	1	1	1	1	1	1	1	1 . . .	1	1	1	1	1	1	1	1	all	INVALOP

*"U" denotes an undefined result.

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

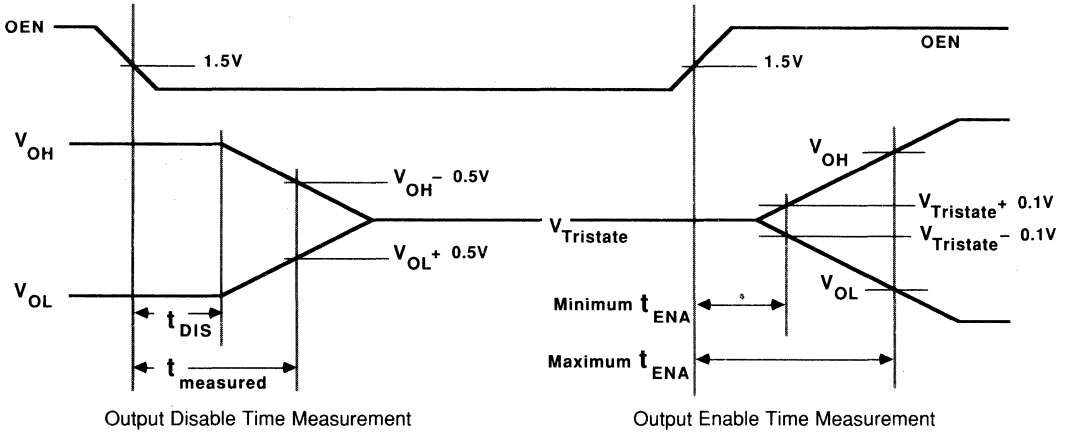


Figure T1. ADSP-3201/3202 Three-State Disable and Enable Timing

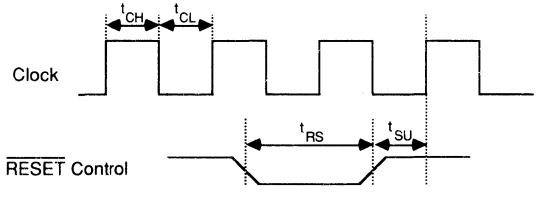


Figure T2. ADSP-3201/3202 Reset Timing

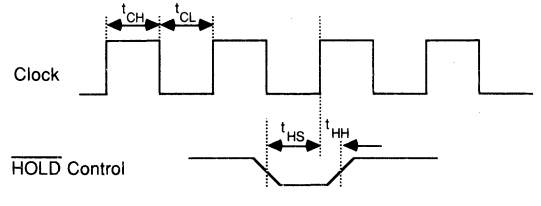


Figure T3. ADSP-3201 Multiplier Output Register Hold Timing

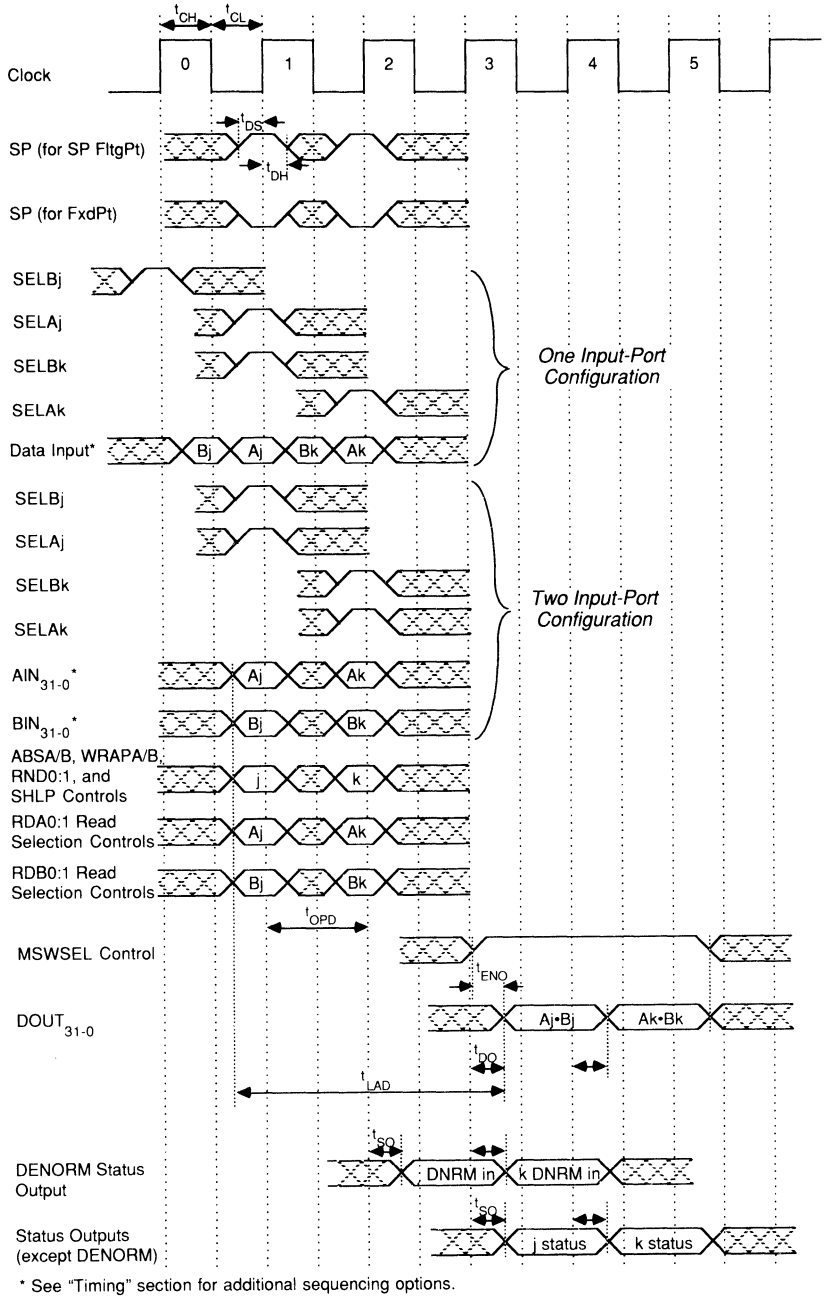
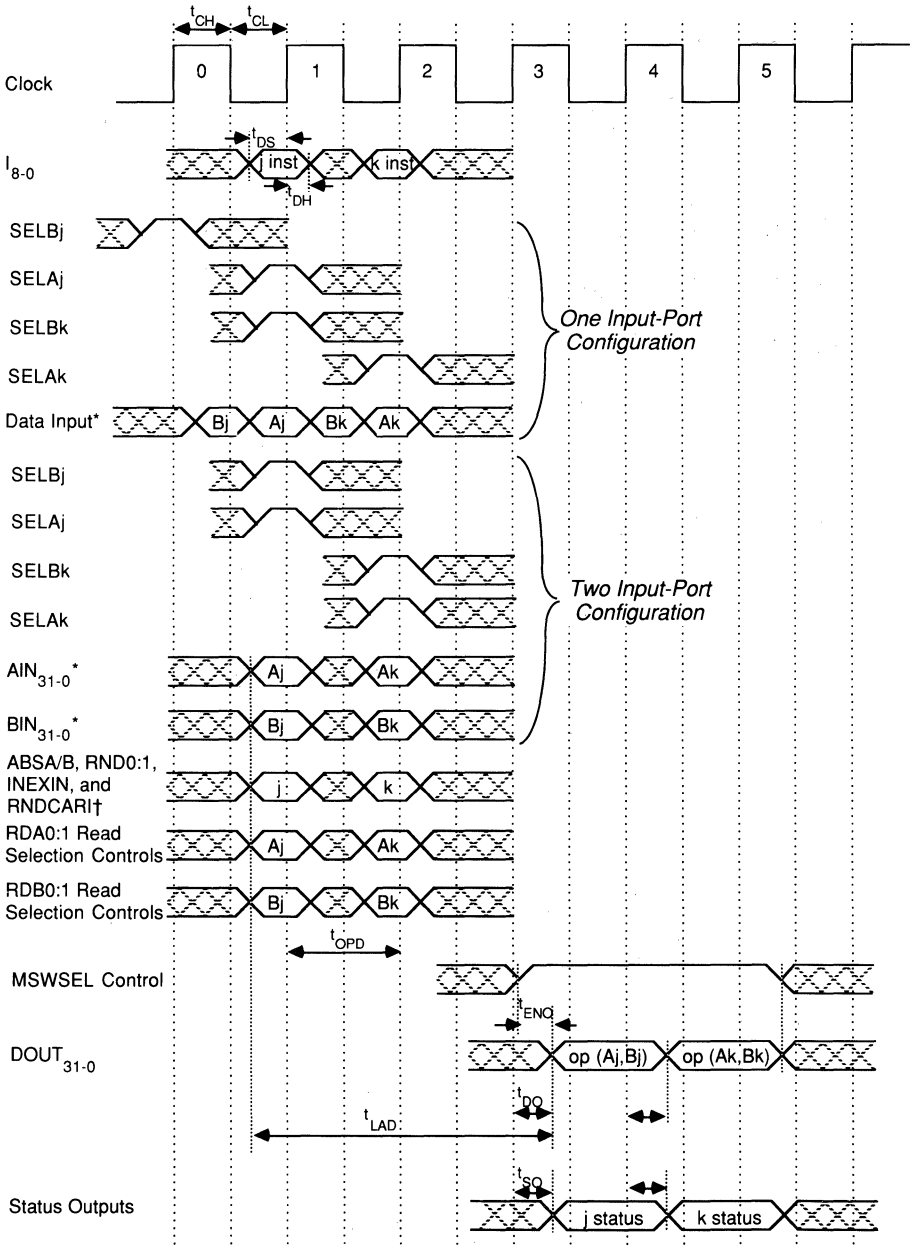
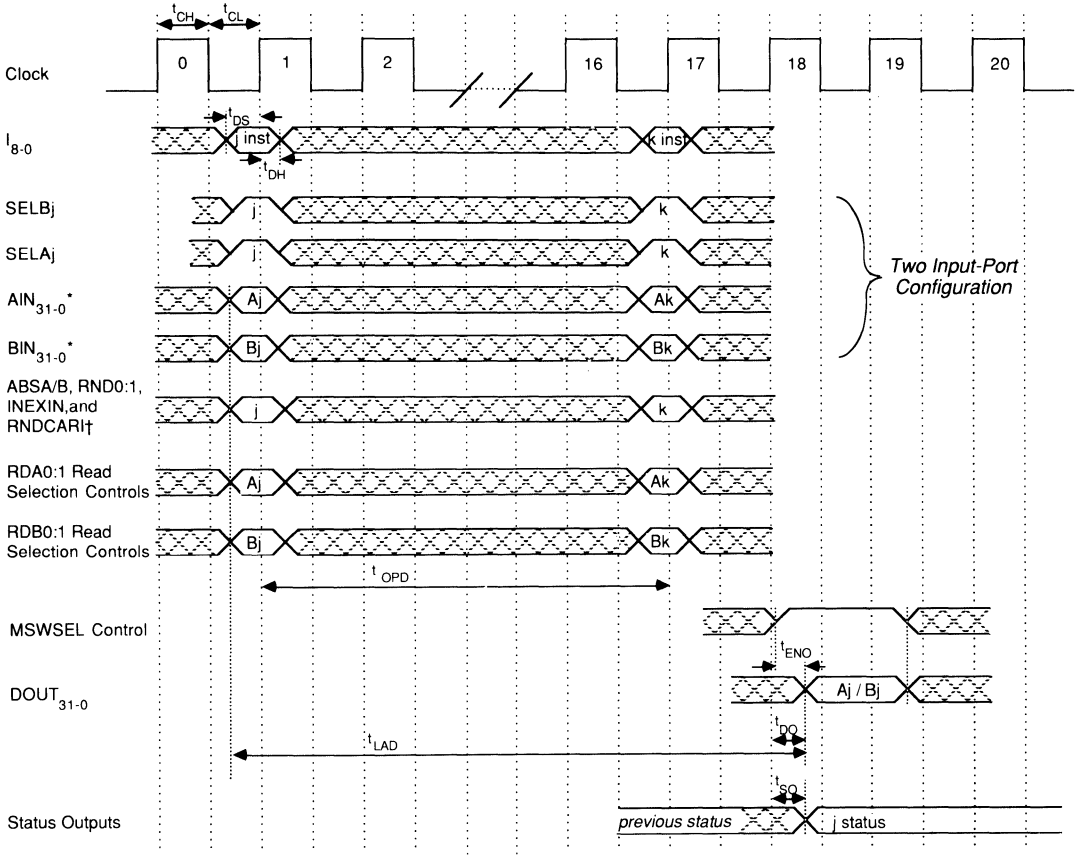


Figure T4. ADSP-3201 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplications



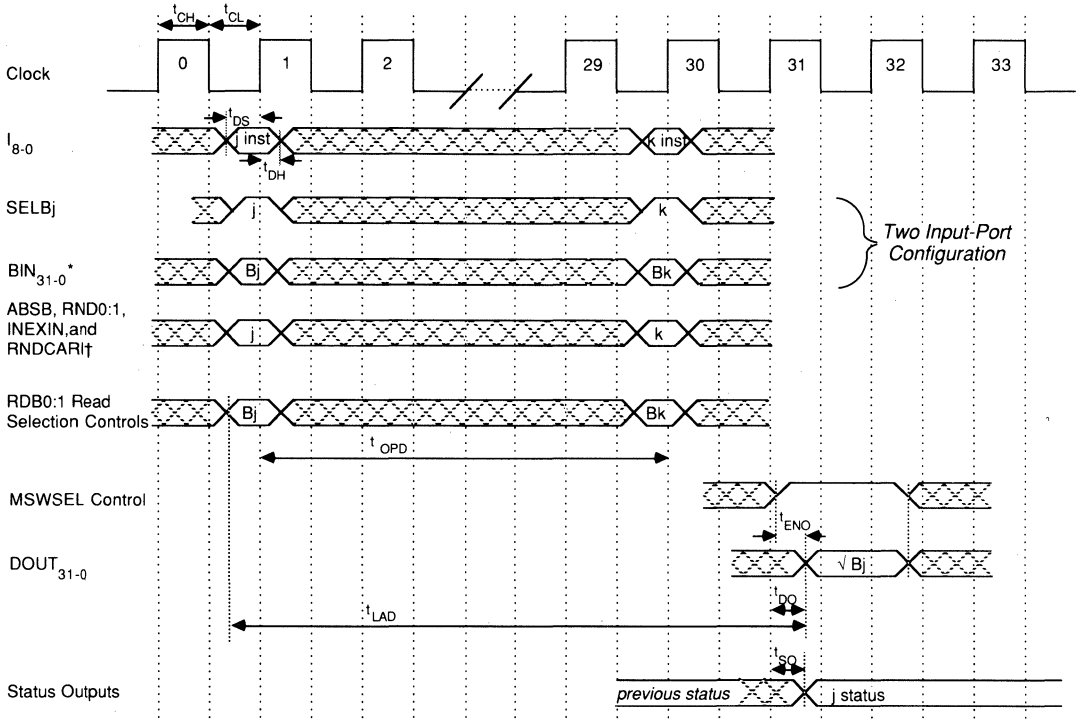
* See "Timing" section for additional sequencing options.
† RNDCARI and INEXIN should be LO except for unwrap, division, and square root operations.

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

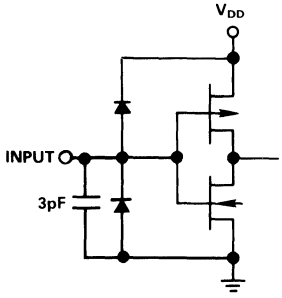


Figure 26. Equivalent Input Circuits

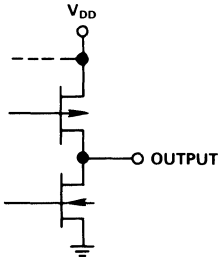


Figure 27. Equivalent Output Circuits

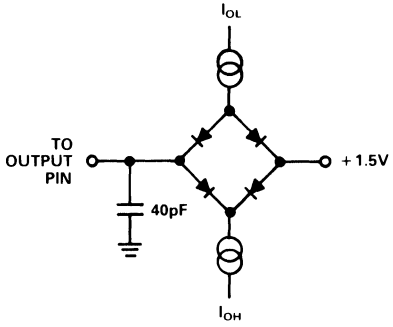


Figure 28. Normal Load for ac Measurements

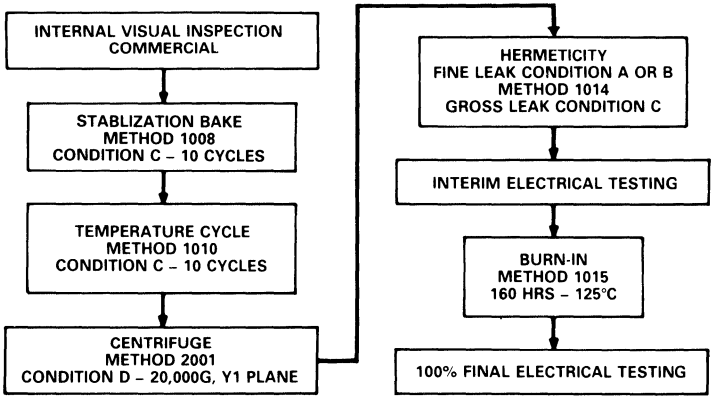


Figure 29. PLUS Processing Flow

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3201/3202 J and K Grades Min	Max	ADSP-3201/3202 S and T Grades[2] 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

Parameter		Test Conditions	ADSP-3201/3202 J and K Grades Min	Max	ADSP-3201/3202 S and T Grades[2] Min	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 Grades 0 to +70°C Min	Max	K Grades 0 to +70°C Min	Max	S Grades[2] −55°C to +125°C Min	Max	T Grades[2] −55°C to +125°C Min	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 Grades 0 to +70°C Min	Max	K Grades 0 to +70°C Min	Max	S Grades[2] −55°C to +125°C Min	Max	T Grades[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 Data Book. Alternatively, S and T grade parts are available with optional high-reliability "PLUS" processing as shown in Figure 29.
[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_{IHA} to data input V_{IH} or V_{IL} crossing points.

Specifications subject to change without notice.

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 (10 Sec) . . . . . +300°C

ESD SENSITIVITY

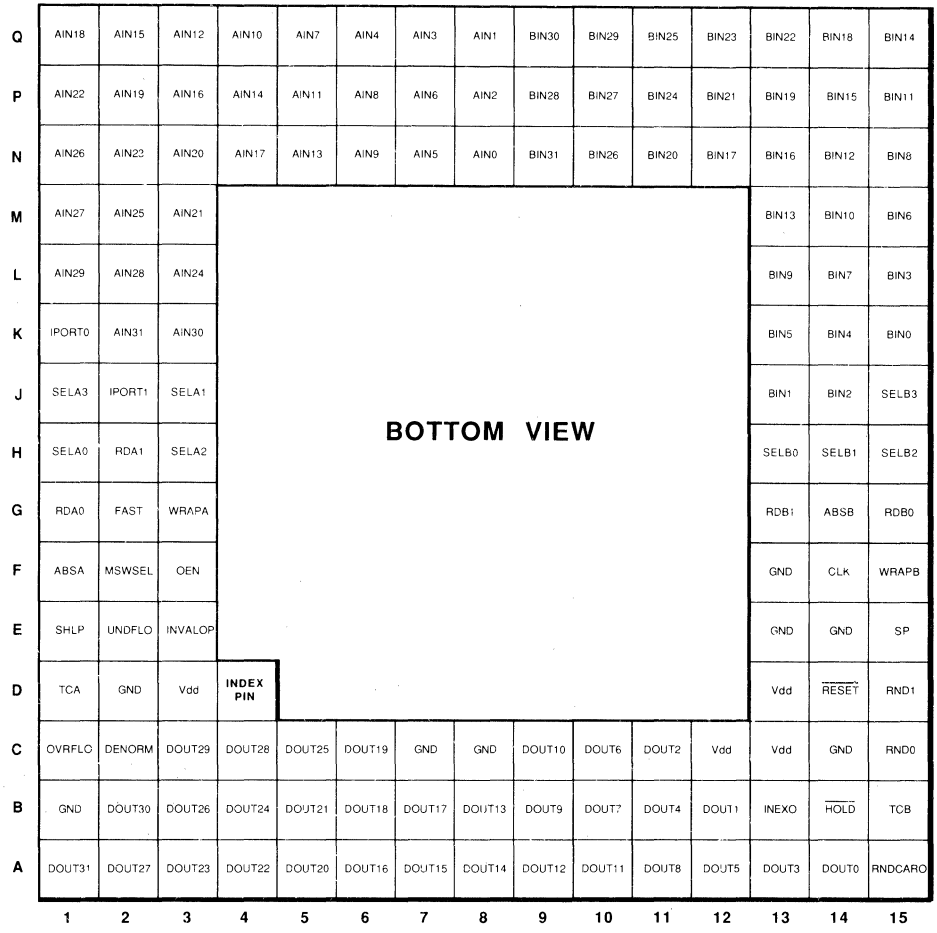
The ADSP-3201/3202 feature input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-3201/3202 have been classified as Category A 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' ESP Prevention Manual.

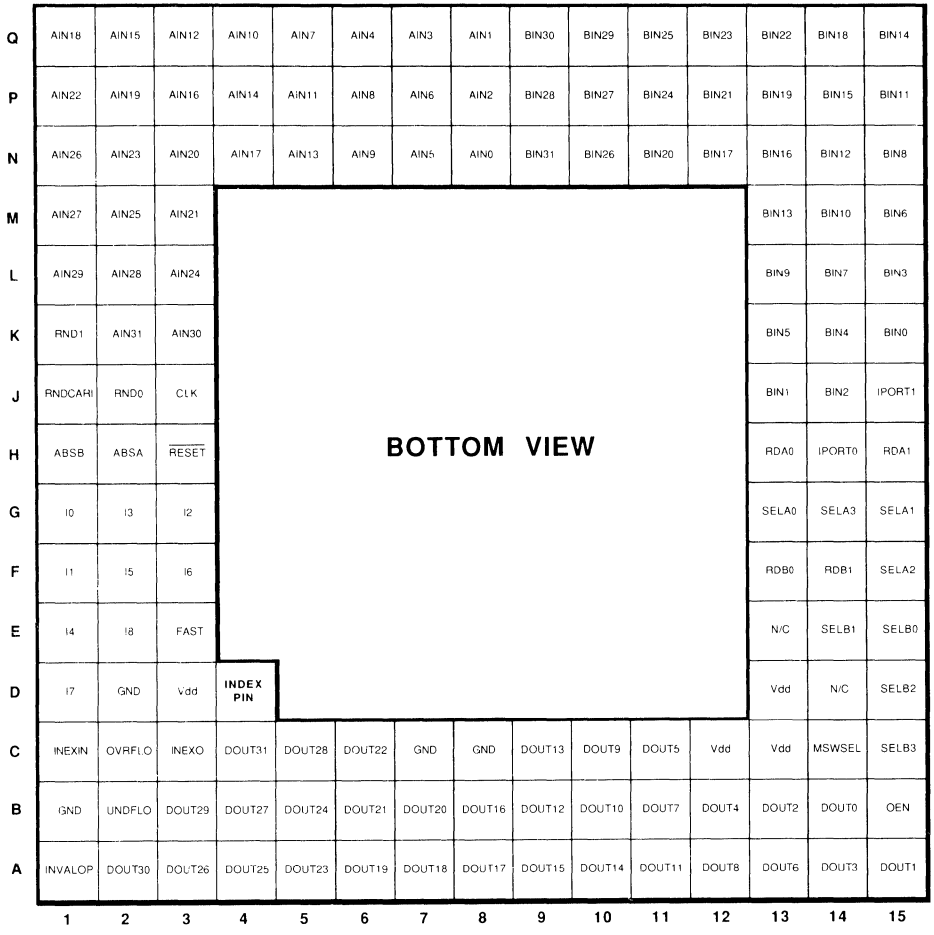
ORDERING INFORMATION

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/+	−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/+	−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



ADSP-3202 ALU Pinouts

64-Bit IEEE Floating-Point Chipset Multiplier/Divider and ALU

ADSP-3212/ADSP-3222

FEATURES

Complete 40 MFLOPS Chipset Implementing Floating-Point Arithmetic: 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 All Multiplications and Standard ALU Operations
Exact Division at 300ns Single-Precision and 600ns Double-Precision Rates
Low Latency for Scalar Operations
- **130ns for 32-Bit Multiplication or Standard ALU Operations**
- **155ns for 64-Bit Multiplication or Standard ALU Operations**

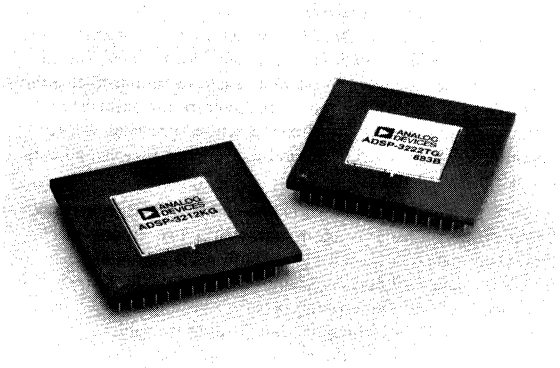
Exact Square Root ALU Instruction
1W Max Power Dissipation per Chip with 1.0μm CMOS Technology
144-Lead Pin Grid Array
Available Specified to MIL-STD-883, Class B
Pin Compatible Upgrades from ADSP-3211/ADSP-3221

APPLICATIONS

High-Performance Digital Signal Processing
Engineering Workstations
Floating-Point Accelerators
Array Processors
Mini-Supercomputers
RISC Processors

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.



A similar chipset, the ADSP-3213/ADSP-3223, provides the same features as the ADSP-3212/ADSP-3222 using DEC floating-point formats (F and G) instead of the IEEE Standard.

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.

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 compatible with those of 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 is always 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, 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/ADSP-3222 also have HOLD controls that prevent the updating of the output data and status registers.

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 and the ADSP-3221. It also includes a 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 1W 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 their 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 with optional high-reliability processing ("PLUS" parts) or processed fully to MIL-STD-883, Class B. The ADSP-3212 and ADSP-3222 are packaged in a ceramic 144-lead pin grid array.

ANALOG DEVICES

Multiport Register File

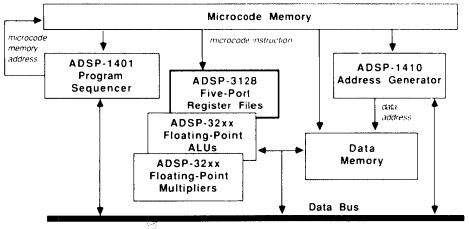
ADSP-3128

FEATURES

128 × 16 or 64 × 32 Register File Organization
Five Ports
  Two Input
  Two Output
  One Bidirectional
Cascadable Horizontally and Vertically
50ns Cycle Time from Single 1 × Clock
22ns 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 Flow-Through Control at Each Write Port
Write Inhibit Control on Each Write Port
Correctly Pipelined Bank Select and Port Select
Register-to-Register Transfers
Three-State Outputs
Fully Static Operation
1.75W Power Dissipation in Low-Power TTL-Compatible CMOS
144-Pin Grid Array

APPLICATIONS

High-Speed Temporary Data Storage in
  Digital Signal Processing
  Numeric Processing
  Graphics
Floating-Point and Fixed-Point



Word-Slice™ Floating-Point Microcoded System using ADSP-3128 Multiport Register Files

GENERAL DESCRIPTION

The ADSP-3128 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-3128's Functional Block Diagram.) The ADSP-3128 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-3128 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-3128 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 its single clock's 1 × cycle rate.

To accommodate critical system timing requirements, the ADSP-3128 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 six 16-bit data transfer operations per cycle. The input and output latches transfer data to and from the RAM using effectively 16-bit internal buses. The bidirectional Edata-Port can be directly controlled to either (a) write, (b) read, (c) both write and read in a single cycle, or (d) perform two reads in a single cycle. Normal operation allows up to three 16-bit writes in clock HI and three 16-bit reads in clock LO per cycle. However, additional reads can be made in clock HI in lieu of writes, allowing up to six 16-bit reads per cycle. Register-to-register transfers are made via the bidirectional Edata-Port (which can be accomplished in two sequential clock phases).

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

Word-Slice is a trademark of Analog Devices, Inc.

Input or a Late Input mode. With Early Input, the Least Significant Word (LSW) is presented to the input data latches in clock HI and the Most Significant Word (MSW) in clock LO. With Late Input, the LSW is presented to the input latches in clock LO and the MSW 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-3128'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-3128 provides an independent Write-Flow-Through control that allows data to pass through the Register File without a pipeline delay. Data is written to RAM and read out in the same (extended) clock LO phase. Each input port also 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 a condition not known until early in the cycle of an attempted write.

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 LSW can be output in clock LO and the MSW in clock HI of the next cycle. With Late Read, the LSW can be output in clock HI and the MSW 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 defined to latch addresses in one of two ways. Either (a) the write address is latched to an address latch on clock HI, or (b) the address latch is transparent. The 7-bit *read address* latches can be defined to latch addresses in one of two different ways. Either (a) the read address is registered to an address latch on the clock's rising edge, or (b) the address latch is 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 enables or disables individual ports consistent with their pipelines.

Bank Select enables or disables an entire ADSP-3128 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-3128 is fabricated in double-metal 1.5μm CMOS. Each chip consumes 1.75W maximum, significantly less than comparable bipolar solutions. The differential between the chip-set's junction temperature and the ambient temperature stays small because of this low power dissipation. Thus, the ADSP-3128 can be safely specified for operation at environmental temperatures over its extended temperature range (−55°C to +125°C ambient).

The ADSP-3128 is available for both commercial and extended temperature ranges. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts, see Figure 26) or processed fully to MIL-STD-883, Class B. The ADSP-3128 is packaged in a ceramic 144-lead pin grid array.

ADSP-3128 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 (and Cdata-Port for Single-Precision Extra Read)
$Badr_{6-0}$	Address Port for Bdata-Port Writes (and Ddata-Port for Single-Precision Extra Read)
$Cadr_{6-0}$	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)
Awft	Write Flow-Through from Adata-Port (Asynchronous)
Bwft	Write Flow-Through from Bdata-Port (Asynchronous)
Ewft	Write Flow-Through from Edata-Port (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	Read Port (Cdata-Port, Ddata-Port, and Edata-Port) Fully Transparent Control for Extra Reads in Single-Precision Mode (Registered)
Eio	Edata-Port Input and Output (In the Same Cycle) in Single-Precision Mode; 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 (Four Lines)
V_{DD}	+5V Power Supply (Three Lines)

TABLE OF CONTENTS

FUNCTIONAL DESCRIPTION

The ADSP-3128 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 time-multiplexed to allow the presentation of six addresses to the RAM per cycle. Hence, up to a total of six reads from and writes to the RAM are possible per cycle. Because of the abundance of data paths, many of which can transfer data twice per cycle, many combinations of six reads and writes are possible.

Three addresses are presented to RAM in clock HI from the Aadr, Badr, and Eadr address latches. Normally in clock HI, 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. Normally in clock LO, these are RAM read addresses. Three simultaneous reads from the same RAM location are possible for normal, clock LO reads. The EadrPort 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.

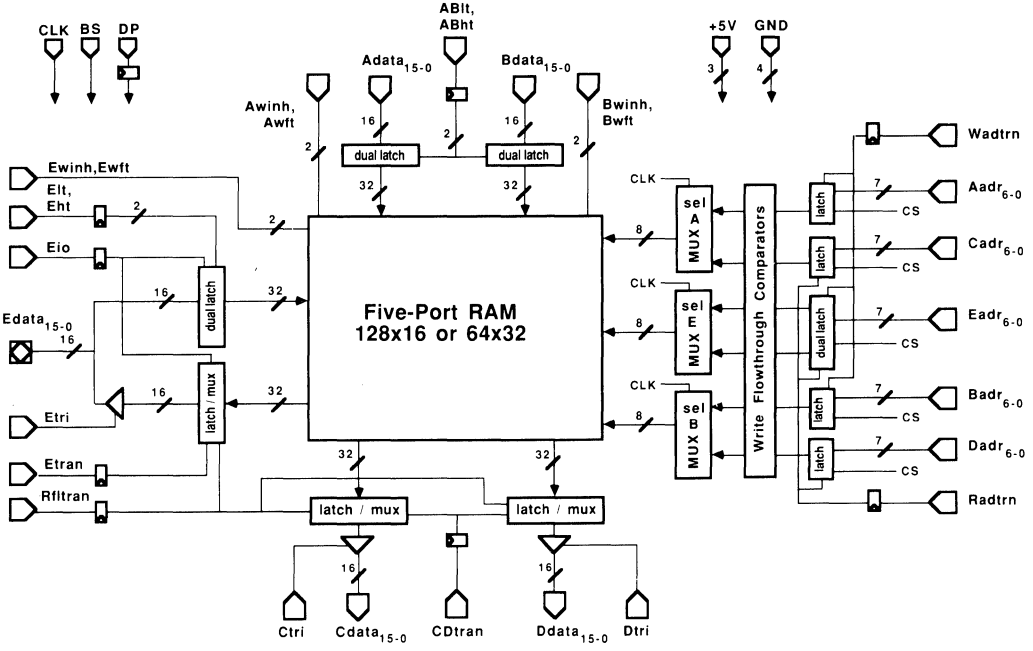


Figure 1. ADSP-3128 Multiport Register File Functional Block Diagram

Physically, the RAM is *always* reading from all three presented address locations to the data latches. For this reason, clock HI addresses can also be used for "extra" reads in clock HI from the RAM to the output latches. In Single-Precision mode, this feature can be enabled by making the output latches transparent for the entire clock cycle (Rfltran HI) to read two 16-bit words per cycle through the Cdata-, Ddata-, and/or Edata-port. The Aadr specifies the word to be read through the Cdata-Port in clock HI, and the Badr, for the Ddata-Port. The Eadr clock HI address latch is selected for any clock HI read to the Edata-Port output latch. (Some restrictions apply to the addresses used in extra reads. See "SP Normal and Extra Reads" below.) If a write to RAM occurs in the clock HI of an extra read, the data written will also be the data read. Thus, in Single-Precision mode, data can be passed through the register file (with write) in the same (extended) clock HI.

Writes to the RAM normally 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. Write data, however, may not be available in a user's system until late in the clock cycle. The Write-Flow-Through controls – Awft, Bwft, and Ewft – allow a write to RAM and a read from the same RAM location to occur in the same clock LO for low-latency data pass-through. Write Flow-Through Comparators monitor the addresses and detect legal Write-Flow-Through operations. Because of these Comparators, the ADSP-3128 can use both a clock HI write address and a clock LO read address to accomplish a Write Flow-Through in a particular (extended) clock LO phase, in spite of the address muxes.

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 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 22).

In Double-Precision mode, the Register File is 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; the only exception is Write Flow-Through which allows a clock LO 32-bit write to RAM (and read from RAM). Multiplexers between the latches and the 16-bit data ports alternately select Least Significant (LSW) and Most Significant (MSW) 16-bit words. Note that when ADSP-3128 Register Files are configured in horizontal pairs for Double-Precision operation, the LSWs from the pair will make up half the external 64-bit double-precision word and the MSWs the other half.

In Single-Precision mode, the input latches can be configured to latch input data at clock HI, to register input data on the falling clock edge, to be made transparent, or to hold the most recent data. The output latches can be configured to register data from the RAM on the rising clock edge, to be transparent clock LO and latched clock HI, or to be fully transparent through both phases (for extra reads). The bidirectional Edata-Port can be configured to do either one read, one write, two reads, or both a read and a 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. Early and Late Inputs are distinguished by a one clock phase difference between when the LSW and MSW 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 clock phase difference between when the LSW and MSW 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 MSW to RAM or a direct read of the LSW from RAM, respectively.

The write address latches can be made transparent or latched at clock HI. The read address latches can be made transparent or registered at clock HI. 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.

The output drivers at the Cdata-, Ddata-, and Edata-ports are dual NMOS in structure rather than true CMOS. Their steady-state HI voltage level will be between three and four volts, rather than the V_{DD} characteristic of CMOS drivers. These drivers are faster HI to LO because of shorter voltage swings; the HI to LO transition time is the same as the LO to HI transition. Shorter voltage swings into a given capacitance reduce the peak current drain. Because their peak current drain is reduced, the ADSP-3128 exhibits improved noise immunity to ground spikes and reduced power consumption.

CONTROLS

The ADSP-3128 Register File has 21 control lines. Their functional consequences 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 the relevant timing diagrams.

CYCLE RATE

The maximum cycle rate for the ADSP-3128 is t_{CLK}, which presumes that all read data is registered. For Double-Precision reads, only Late Reads meet this requirement. The cycle rate for the Register File will have to be extended if any reads are transparent, for Write Flow-Through, for Single-Precision Write-Plus-Extra-Read, or for most Register-to-Register Transfers.

The cycle rate to encompass Write Flow-Through when all read data is registered is t_{CLKFT}. Clock LO will also have to be a minimum of t_{LOWR}.

If all addresses are latched or registered, all write data is latched, and *any* reads are transparent (DP Early Reads), then the maximum cycle rate for the ADSP-3128 will be t_{CLKS} to accommodate the output delays from transparent reads. (In Single-Precision mode, the clock *can* be run faster, with output data allowed to slide into the next clock phase. The user, however, would have to clock the output data externally by the next falling clock edge.) The cycle rate to encompass Write Flow-Through or SP Write-Plus-Extra-Read when all addresses are latched or registered, all write data is latched, and *any* reads are transparent is t_{CLKSFT}. Clock LO will also have to be a minimum of t_{LOWR} for Write Flow-Through and clock HI will have to be a minimum of t_{HIER} for Write-Plus-Extra-Read.

BS	DP	AB & Elt	AB & Eht	A & B & Einh	A & B & Ewft	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	0	X	X	0	**0**	Normal write to RAM from the A or B or Edata input latches during clock HI
1	0	X	X	X	**1**	Flow-through write (transparent) to RAM from the A or B or Edata input latches during clock LO when write/read addresses are equal
1	1	**0**	**0**	X	X	Early Input to A & B or Edata input latches: register LSW on falling edge to input latches and latch MSW to input latches in clock HI
1	1	**0**	**1**	X	0	Late Input to A & B or Edata input latches: latch LSW to input latches in clock HI and make input latches transparent for MSW in clock HI
1	1	0	0	X	1	Undefined
1	1	**1**	X	X	X	Hold most recent data at A & B or Edata input latches for the next cycle
1	1	1	1→0	X	X	Edata Slow Input: register LSW to Edata input latch on next falling edge (Eht only)
1	1	1	0→1	X	X	Edata Slow Input: register MSW to Edata input latch on next falling edge (Eht only)

Table I. ADSP-3128 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	**1**	**1**	**X**	**0**	C & D or Edata RAM output latches are fully transparent for both phases. If Awinh/Bwinh/Ewinh = 1, an additional read(s) can be performed at C/D/Edata, respectively
1	0	X	X	X	**0**	Edata-Port is configured for one read, one write, or two reads per cycle
1	0	X	X	X	**1**	Edata-Port is configured for both a read and a write every cycle; read Eadr is registered on falling edge and write Eadr is registered on rising edge
1	1	**0**	0	X	**0**	Configured for Late Read at C&D or Edata-Port: register LSW & MSW from RAM to output latches on rising edge; output LSW in clock HI, output MSW on next clock LO
1	1	**1**	0	X	**0**	Configured for Early Read at C&D or Edata-Port: output LSW from RAM through transparent output latches in clock LO; latch MSW 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 LSW at clock HI
1	1	**1**	0	X	**1**	Configured for Edata Slow Read: hold RAM read data at Edata output latch; output MSW at clock HI
1	**1**	X	**1**	X	X	Undefined

Table II. ADSP-3128 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

*Note: Eio = 1 overrides Wadtrn and Radtrn at the Eadr-Port (only). That is, when Eio = 1, Eadr write addresses are registered on rising edges and read addresses are registered on falling edges, regardless of the state of Wadrn and Radtrn. The other four address ports are unaffected by Eio and always behave as described in this table.

Table III. ADSP-3128 Summary of Address Control Modes

If at least one address *or* one data write is transparent *and* at least one data read is transparent (DP Early Read), then the ADSP-3128's cycle time will have to be extended to t_{CLKA} to accommodate transparent setup times and output delays. If any Write Flow-Throughs or SP Write-Plus-Extra-Reads are attempted, the cycle time will be t_{CLKAFT} when at least one address *or* one data write is transparent *and* at least one data read is transparent. Clock LO will also have to be a minimum of t_{LOWR} for Write Flow-Through and clock HI will have to be a minimum of t_{HIER} for Write-Plus-Extra-Read.

	All reads clocked	All addresses latched or registered, all writes latched, and at least one read transparent	At least one address or write transparent, and at least one read transparent
No Write Flow-Throughs or SP Write-Plus-Extra Reads	t_{CLK}	t_{CLKS}	t_{CLKA}
Some Write Flow-Throughs or SP Write-Plus-Extra Reads	t_{CLKFT}	t_{CLKSFT}	t_{CLKAFT}

Table IV. ADSP-3128 Cycle Times

Register-to-register transfers are accomplished by driving the source through the Edata-Port and reading that data back through the Edata-Port's input latch. Therefore the ADSP-3128's cycle rate must allow for both data output delay and input setup time.

Single-Precision register-to-register transfers in Single I/O Per Cycle Mode (Eio = 0) can be accomplished at the fastest specified clock rate, t_{CLK}, under the following conditions: clocked read address, transparent read, and transparent write. This works because $(t_{ODRT} + t_{DST}) < t_{CLK}$. In general, for Single-Precision *transparent inputs* (when Eio = 0), the maximum clock cycle time must be equal to or greater than the sum of the chosen data output delay, $t_{OD?T}$, and t_{DST}, the setup time for transparent writes. (See Figure 10.)

If Single-Precision inputs in Single I/O Per Cycle Mode (Eio = 0) are *latched*, similar requirements apply to clock LO, since both data output delay and input setup must be completed in that clock phase. That is, for latched inputs (when Eio = 0), clock LO must be equal to or greater than the sum of the chosen (transparent) data output delay, $t_{OD?T}$, and t_{DSR}, the setup time for the Edata-Port's input latch. (See Figure 9.)

If Single-Precision inputs in Double I/O Per Cycle Mode (Eio = 1) are *latched*, clock LO must be long enough so that clock LO $\geq (t_{ODRT} + t_{DSR})$. (See Figure 11.) If Single-Precision inputs in Double I/O Per Cycle Mode (Eio = 1) are *transparent*, clock HI must be long enough so that clock HI $\geq (t_{ODC} + t_{DST})$. (See Figure 12.)

The minimum cycle rate to accommodate Double-Precision register-to-register transfers depends on the chosen timing mode. For Early Read/Late Write, there are two requirements: clock LO $\geq$ (transparent output delay $+ t_{DSR}$) and clock HI $\geq (t_{ODC} + t_{DST})$. (See Figure 19.) For Late Read/Early Write the requirements are the same for both clock LO and clock HI: clock LO $\geq (t_{ODC} + t_{DSR})$ and clock HI $\geq (t_{ODC} + t_{DSR})$. (See Figure 20.) That in turn implies a minimum clock rate $\geq 2 \cdot (t_{ODC} + t_{DSR})$.

ADDRESS LATCHES FOR BOTH SINGLE- AND DOUBLE-PRECISION MODES

The three read (clock HI) address latches hold the seven bits required for Register File addressing and Port Select, and also Bank Select. Radtrn controls whether these three latches are transparent or latched. When Radtrn is HI, addresses presented at the read 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.

The three write (clock LO) address latches hold the seven bits required for Register File addressing and Port Select, and also Bank Select. Wadtrn controls whether these three latches are transparent or latched. When Wadtrn is HI, addresses presented at the read address ports are transferred directly to the RAM with no pipeline delay. When Wadtrn is LO, addresses presented at the read 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. (See Figures 3-7, 9-17, 19, and 20.) 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. To a limited degree, transparent read addresses can be setup in the phase *before* the actual RAM read while maintaining the specified relationship between address valid and data valid (t_{ODTT}). The maximum usable setup for transparent addresses is $-t_{ASTE}$ prior to the RAM read phase (note that t_{ASTE} is negative). Setting up the read address any earlier than that will not get the read data out any sooner; the earliest possible data out will be (t_{ODTT} plus t_{ASTE}) after the clock edge beginning the RAM read phase. The transparent read address must still 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.

	min clock LO	min clock HI	min clock period
SP Single I/O Per Cycle – Transparent Inputs			$t_{OD??} + t_{DST}$
SP Single I/O Per Cycle – Latched Inputs	$t_{OD?T} + t_{DSR}$		
SP Double I/O Per Cycle – Transparent Inputs		$t_{ODC} + t_{DST}$	
SP Double I/O Per Cycle – Latched Inputs	$t_{ODRT} + t_{DSR}$		
DP Early Read Late Write	$t_{OD?T} + t_{DSR}$	$t_{ODC} + t_{DST}$	
DP Late Read Early Write	$t_{ODC} + t_{DSR}$	$t_{ODC} + t_{DSR}$	

Table V. ADSP-3128 Register-to-Register Cycle Time Requirements

Up to six 16-bit data transfers per cycle are possible in Single-Precision mode. These six transfers can be comprised of three writes and three reads, two writes and four reads, one write and five reads, or six reads.

SP NORMAL AND EXTRA 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-3128. 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 normal 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 will be valid t_{ODTT} after a valid read address when the output latches are transparent. As described above, the transparent address can be setup $-t_{ASTE}$ before the RAM read phase and still maintain the t_{ODTT} output delay. 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.

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. See "Cycle Rate" above for clock rate implications of various read modes.

Note that in all four cases 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 four Single-Precision normal 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.

Extra reads through the Cdata-Port, Ddata-Port, and/or Edata-Port can be accomplished by making the output latches fully transparent with registered control Rfltran HI. The latches must already be transparent (CDtran/Etran HI); Rfltran HI prevents the transparent latches from holding data in the second phase after output as described for normal reads. The latches remain transparent throughout the clock cycle, allowing two 16-bit reads in the same cycle. If CDtran or Etran is LO, Rfltrn will have no effect on the relevant ports.

Extra reads are unusual in that they occur in clock HI (phase β in Figure 4), normally a write phase. Single-Precision extra reads are the only exception to the "read in clock LO" general rule. Extra reads require extra addresses. They must be supplied at the write address ports. The Aadr "write" address will determine the data read out of the Cdata-Port as an extra read. Badr will determine the Ddata-Port extra read. Aadr and Badr can be either transparent or latched. For Edata-Port extra reads, the Eadr must be transparent to enable two addresses to reach the RAM in a single cycle. Note that the timing parameters for extra reads are the same as for normal reads. The key difference is that data is only held valid until t_{ODH} after the next clock edge, not throughout the entire next phase.

Concurrent extra reads through any two or three of the three output ports are allowed. However, only one extra read from a given set of RAM locations, (n, $n \pm 64$), is possible during a given clock HI. If more than one, simultaneous extra read is attempted from (n, $n \pm 64$), only the result at the highest priority port will be valid. Results at all other ports are undefined. The ports for extra reads are prioritized: Edata-Port (Eadr) first, Cdata-Port (Aadr) second, and Ddata-Port (Badr) third.

Each write data port has an independent Write Inhibit Control (Awinh, Bwinh, and Ewinh). If the write data ports are not inhibited (as shown in Figure 5), the data at the write data latches will be written to the RAM in phase β. An extra read performed in the same phase will read out the very same data written to RAM (Write-Plus-Extra-Read). So single-precision data can be passed through the Register File in clock HI without a pipeline delay, subject to the constraint that Adata goes to the Cdata-Port and Bdata goes to the Ddata-Port. A Write-Plus-Extra-Read will cause a cycle time penalty. Clock HI will have to be extended to t_{HIER} or results will be indeterminate. See "Cycle Rate" above for clock rate implications of SP Write-Plus-Extra-Reads.

Attempting a write-plus-extra-read with the Edata-Port will cause a contention, since the ADSP-3128 and the external device would both be driving. So Ewinh must be HI for an Edata-Port extra read. See "SP Write Flow-Through"and "DP Write Flow-Through" sections below for a more general way of passing data through the ADSP-3128 in the same clock phase, clock LO.

SP NORMAL 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-3128. 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 Write 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 not be enabled until at least t_{ARBE} after the rising clock edge and no later than t_{WEN} before the falling edge. (The design target for t_{ARBE} is zero; see "Specifications" below.) If a transparent write address is used, there is the additional requirement that the address be valid t_{ATBE} before A/B/Ewinh goes LO; otherwise a write might be attempted at the wrong address, i.e., a wrong write.

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/Binh controls normally HI, i.e., write inhibited, and bring them LO every time you actually want to write. Alternatively, for simplicity, the A/Binh controls can be wired LO (write enable) and dummy writes be performed to an unused RAM location in every clock HI (if t_{ARBE} is zero; see "Specifications" below). Write addresses must always be stable, however, whenever the Write Inhibit controls are LO. (Do not hardwire Ewinh.)

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 WRITE FLOW-THROUGH

Write Flow-Through allows a low-latency write to RAM and read from the same RAM location in clock LO (Figure 6). With Write Flow-Through, the designer can pass data through the RAM late in the clock cycle without suffering a pipeline delay. (If write data is available early in the clock cycle, a normal write followed by a normal read can pass data through the ADSP-3128 in one cycle.) Write Flow-Through is the only exception to the "write in clock HI" general rule. In Figure 6, both the write to RAM and the read from RAM occur in phase γ, a clock LO.

Write Flow-Through is enabled at the respective write data ports by asynchronous controls Awft, Bwft, and Ewft. The Write Flow-Through controls override the Write Inhibit controls and always force a write whenever write and read addresses match, independent of the levels on the Write Inhibit controls. (If no read address matches the Write-Flow-Through write address, the write will not take place.) The Write Flow-Through Controls must be asserted HI t_{WFT} before the clock's rising edge. By the next clock LO they must be LO.

Only transparent write addresses can be used in Write Flow-Through mode. (Write addresses latched at the previous rising edge would no longer be valid by clock LO.) Read addresses can be either transparent or registered. Transparent addresses are also required to be valid t_{AWFT} before Write Flow-Through for the data port in question is asserted as shown in Figure 6.

The data setup and hold requirements are the same whether the input data latches are configured for transparent or latched write data (since their behavior in clock LO is the same either way). Write data must be setup t_{DSTFT} before the rising edge if the read is transparent and t_{DSCFT} if the read is registered. The output delay for a transparent read will be t_{ODTFT} from the write data valid or t_{ODWFT} from the Write Flow-Through control's rising edge, whichever is greater. The output delay for a registered read will be t_{ODCFT} from the rising edge.

For clock-on-falling write data, the data setup time is t_{DSN}, the same as shown in Figure 5. The output delay for a transparent read will be t_{ODTFT} *from the falling clock edge* or t_{ODWFT} from the Write Flow-Through control's rising edge, whichever is greater. The output delay for a registered read will be t_{ODCFT} from the rising edge.

Two independent Write Flow-Through operations can occur in the same phase LO. (There aren't enough ports for three.) There are no restrictions on the relationships between addresses, even when attempting two concurrent Write Flow-Throughs.

The write ports are prioritized for Write Flow-Through in exactly the same way as for normal writes 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 LO phase, the data presented at the higher priority enabled write data port will be the data written to RAM.

Using Write Flow-Through will cause a cycle time penalty. Clock LO will have to be extended to t_{LOFT} for any phase in which a Write Flow-Through is attempted. See "Cycle Rate" above for clock rate implications of Write Flow-Through.

SP BIDIRECTIONAL EDATA-PORT

The Edata-Port will behave like any write port if treated as such. Alternatively, it will also behave like any read port if treated as such, including supporting extra reads. 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. In Single-Precision mode, the Edata-Port can also function as a bidirectional port both reading and writing in every clock cycle (Figure 7).

The control that configures the Edata-Port for bidirectional operation is the registered Eio control. It must be asserted in conjunction with the DP control as shown in Figure 7 for proper interpretation. (Eio has a different meaning in Double-Precision mode.) It takes effect in the cycle immediately after it is asserted.

When in Bidirectional mode, all Eadr-Port latches go to registered mode, *regardless of the state of Radtrn and Wadtrn* (Table III). All other address ports are unaffected by Eio. Address setup and hold times are the usual t_{ASR} and t_{AH} for registered and latched addresses.

Three combinations of Edata-Port input and output latch settings are possible: transparent write and transparent read, clock-on-falling write and transparent read, and latched write and registered read. Note carefully in Figure 7 that because of the pipelining differences, only in the first case do data and addresses track in the same order. All reads from the RAM are in phase γ and all writes to the RAM are in phase δ which follows. But in the

latter two cases the phase δ write data is presented before the phase γ read data is output.

Setup times and output delays are standard. Note that minimum output delays are specified for both t_{ODRT} and t_{ODC}, indicating when the device writing to the Edata-Port must be no longer driving to avoid contention. The Edata-Port will be in a high-impedance state t_{EDIS} after the clock edge after the read data became valid. That is, when the Edata-Port is configured for bidirectional operation (Eio HI), it will automatically go into a high-impedance state to allow writes without forcing the user to manipulate Etri. This disable time, t_{EDIS}, is measured as shown for t_{ENA} in Figure 3.

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 as of the next falling edge. Figure 8 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 and writing the same data to multiple RAM locations.

SP REGISTER-TO-REGISTER TRANSFERS

Register-to-register transfers are accomplished with the Multiport Register File by reading data from the Edata-Port output latch and writing that very data to the Edata-Port input latch. The input Edata-Port latch can hold the data in question as just described, allowing a large number of possible ways to accomplish a register-to-register transfer. Figures 9, 10, 11 and 12 illustrate those register-to-register transfers that *don't* involve data holds. A read in clock LO is followed by a write in the next clock HI. Figures 9 and 10 represent two methods with Eio LO; Figures 11 and 12, one method with Eio HI. All three make RAM reads in phase γ and RAM writes in phase δ.

To get the data from Edata-Port output latch to the input latch, the data must be driven through the Edata-Port. That is, Etri must be LO as shown in Figures 9 through 12. What actually shows up at the Edata-Port is described at the bottom of Figures 9 through 12 and follows standard timing. Note that the read data will only be held valid through a full phase if both DP and Eio are LO.

With Eio LO (Edata-Port not bidirectional), there are several ways to accomplish the transfer. The Edata input latch can be either transparent or in latched mode. With a registered read address (Eadr), the write address (Eadr) can be either latched or transparent. If the read address (Eadr) is transparent, however, the write address (Eadr) must be transparent also. The combination of a transparent E read address and latched E write address is impossible, since both would have to be valid in the same phase γ.

With Eio HI (Edata-Port bidirectional), the read and write Eadrs will be registered, since Eio HI forces that addressing mode.

Since the data driven off-chip must be re-input to the ADSP-3128 in a register-to-register transfer, the clock period, clock LO, and/or clock HI, as relevant, must be long enough to allow for data output delays and input setup times. This requirement imposes some restrictions on the maximum speed at which the Register File can be operated. See "Clock Rate" above for a detailed description of timing issues related to register-to-register transfers. These timing requirements are summarized in Table V.

SP BANK SELECT

Bank Select is treated in exactly the same way in both Single-Precision and Double-Precision modes (Figure 21). 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 21. 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-3128s that produce a single "word" and applying the low-order seven bits to the address ports in all rows. In other words, the register file space can be expanded in exactly the way a designer normally builds up a random access memory.

The only restriction on this method of 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-3128s. (Port Select removes this restriction for Double-Precision mode). In Single-Precision mode, the user can select/deselect individual *ports* 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 21). 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. Since the Write Inhibit controls for the three write data ports are independent, if any is enabled in phase α, BS will have to be LO in phase α to disable all writes. Also, if there is any possibility of a Write Flow-Through in phase γ, BS will have to be LO throughout phase γ.

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 is redefined as the 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-3128 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 Slow Input and Slow Read modes, however, the user does have 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-3128 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-3128's 32-bit word is referenced as an "LSW," the other half as an "MSW". Note that normally a user would put together the LSWs from two ADSP-3128s to create the 32-bit Least Significant Word of a 64-bit double-precision floating-point number. Similarly, the floating-point number's Most Significant Word would be constituted from the MSWs of two ADSP-3128s.

What is called an "LSW" 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. For interfacing with Analog Devices' ADSP-3210 Floating-Point Multiplier, this is the more meaningful name for the first datum transferred. But it is nothing more than a semantic convention; what is called here an "LSW" can be the more significant data in a user's system. 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 13 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 13), 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 LSW 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 MSW 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 setup $-t_{ASTE}$ before the RAM read phase and still maintain the t_{ODTT} output delay for the LSW, as in Single-Precision mode.

Late Reads are generated when CDtran and/or Etran are LO. As with Early Reads, both the LSW and MSW are read from the RAM to the 32-bit output latches in phase γ. In the case of Late Read, the LSW is held at the output latch until the next phase, when it is driven off chip with delay t_{ODC}. The MSW follows in the phase after that with the same delay characteristic of registered reads. See "Cycle Rate" above for clock rate implications of various read modes.

Each read port has its own asynchronous three-state control: Ctri, Dtri, and Etri. See Figure 3 for enable and disable timing.

DP NORMAL WRITES

Double-Precision mode must be asserted as shown in Figure 14 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 14), 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 14 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 LSW and MSW are written to the 32-bit input latches before they are both written to RAM in phase δ. Both LSW and MSW 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 LSW and MSW into the Register File one-half cycle later for a phase δ write to RAM. The LSW is latched with setup time t_{DSR}. The MSW, however, is transparently written to RAM in phase δ. Note that the setup requirement on the MSW 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 14. Writes should not be enabled until at least t_{ARBE} after the rising clock edge and no later than t_{WEN} before the falling edge. (The design target for t_{ARBE} is zero; see "Specifications" below.) If a transparent write address is used, there is the additional requirement that the address be valid t_{ATBE} before A/B/Ewinh goes LO; otherwise a write might be attempted to the wrong address.

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/B/Ewinh controls can be wired LO (write enable) and dummy writes be performed to an unused RAM location in every clock HI (if t_{ARBE} is zero; see "Specifications" below). Write addresses must always be stable, however, whenever the Write Inhibit controls are LO.

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 WRITE FLOW-THROUGH

Write Flow-Through is an available option in Double-Precision. It allows the user to pass double-precision data through the Register File with minimal latency (three clock phases). See Figure 15. Write Flow-Through allows both a 32-bit write from latch to RAM and a 32-bit read from RAM to latch in the same clock LO. (If write data is available one phase earlier than shown in Figure 15, a normal Late Write followed by a normal Early Read can pass data through the ADSP-3128 in four clock phases.) Write Flow-Through is the only exception to the "write in clock HI" general rule. In Figure 15, both the write to RAM and the read from RAM occur in phase γ.

Only Early Writes can be used for Write Flow-Through. Since the write actually occurs in clock LO, Late Write is too late.

Write Flow-Through is enabled at the respective write data ports by asynchronous controls Awft, Bwft, and Ewft. The Write Flow-Through controls override the Write Inhibit controls and always force a write whenever write and read addresses match, independent of the levels on the Write Inhibit controls. (If no read address matches the Write-Flow-Through write address, the write will not take place.) The Write Flow-Through Controls must be asserted HI t_{WFT} before the clock's rising edge. By the next clock LO they must be LO.

Only transparent write addresses can be used in Write Flow-Through mode. (Write addresses latched at the previous rising edge would no longer be valid by clock LO.) Read addresses can be either transparent or registered. Transparent addresses are also required to be valid t_{AWFT} before Write Flow-Through for the data port in question is asserted as shown in Figure 15.

The LSW must be setup t_{DSR} before the falling edge, as for a normal Early Write. For the MSW, however, the timing characteristic of Write Flow-Through becomes effective. The MSW must be setup t_{DSTFT} before the rising edge if the read is Early and t_{DSCFT} if the read is Late. The output delay for the LSW for an Early Read will be t_{ODWFT} from the Write Flow-Through control's rising edge. The output delay for the LSW for a Late Read will be t_{ODCFT} from the rising edge. MSWs read out are registered with delay t_{ODC} for both Early and Late Reads.

Two independent Write Flow-Through operations can occur in the same phase LO. There are no restrictions on address locations for two concurrent Write Flow-Throughs in Double-Precision.

The write ports are prioritized for Write Flow-Through in exactly the same way as for normal writes 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 LO phase, the data presented at the higher priority enabled write data port will be the data written to RAM.

Using Write Flow-Through will cause a cycle time penalty. Clock LO will have to be extended to t_{LOFT} for any phase in which a Write Flow-Through is attempted. See "Cycle Rate" above for clock rate implications of Write Flow-Through.

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-3128 provides a mode for loading the LSW and MSW into the input latches over multiple ADSP-3128 clock cycles (Figure 16). Also a mode is provided for multiplexing LSW and MSW read data from the output latches over multiple clock cycles (Figure 17).

For a Slow Input (Figure 16), the input latches are updated when there is a *transition* in Eht from one clock rising edge to the next clock rising edge. Elt must be concurrently HI (Hold mode). Data is setup to the input latches with setup time t_{DSR}. When Eht is LO in the cycle after it was HI, the data will be written to the LSW position in the Edata input latch at the next falling edge and held there. When Eht is HI in the cycle after it was LO, the data will be written to the MSW position in the Edata input latch at the next falling edge and held there. A write to RAM can be enabled (with Ewinh LO) at the next clock HI from either latched or transparent Eadr. Because it is the *transition* in Eht that determines whether data is loaded to the LSW or MSW position in the input latch, the user needs to program Eht for setup in both phase α and phase γ.

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 15, 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 17. When configured for Slow Read, the ADSP-3128's registered Etran control becomes a direct controller of the Edata-Port's Double-Precision output multiplexer. When Etran is LO, the LSW read from RAM in phase γ will be driven through the Edata-Port (if enabled with Etri). When Etran is HI, the MSW 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 INPUT LATCHES AND HOLD

Data input to the input latches can be held at those latches with the ABlt, ABht, Elt, and Eht controls (Table I). These controls are always registered on the rising edge and become effective as of the next falling edge. Figure 18 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/Elt HI, data at the write data port is ignored and will be ignored until the next falling edge after either ABlt/Elt goes LO or Eht makes a transition. (If Eht makes a transition, the ADSP-3128 will make a Slow Input.) The hold feature allows the input latches to be used for temporary data storage.

DP REGISTER-TO-REGISTER TRANSFERS

Register-to-register transfers are accomplished with the Multiport Register File by reading data from the Edata-Port output latch and writing that very data to the Edata-Port input latch. The input Edata-Port latch can hold the data in question as just described, allowing a large number of possible ways to accomplish a register-to-register transfer. Figures 19 and 20 illustrate those register-to-register transfers that *don't* involve data holds.

Only two combinations of Double-Precision reads and writes can be used for Register-to-Register transfers: Early Read with Late Write (Figure 19) and Late Read with Early Write (Figure 20). Early Read with Late Write offers the lower latency.

To get the data from Edata-Port output latch to the input latch, the data must be driven through the Edata-Port. That is, Etri must be LO as shown in Figures 19 and 20. What actually shows up at the Edata-Port is described at the bottom of Figures 19 and 20 and follows standard timing.

Since the data driven off-chip must be re-input to the ADSP-3128 in a register-to-register transfer, clock LO, and clock HI must each be long enough to allow for data output delays and input setup times. This requirement imposes some restrictions on the maximum speed at which the Register File can be operated. See "Clock Rate" above for a detailed description of timing issues related to register-to-register transfers. These timing requirements are summarized in Table V.

DP BANK SELECT AND PORT SELECT

Bank Select is treated in exactly the same way in both Single-Precision and Double Precision modes (Figure 21). 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 22 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_{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 and Port Select to track all read and write pipelines as shown in Figures 21 and 22. When LO, writes will be disabled and output ports put in high-impedance.

With either Bank Select or Port 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 six bits for addressing, decoding the high-order bits to select a horizontal row of ADSP-3128s with Bank Select or individual ports with Port Select that produce a single "word" and applying the low-order six bits to the address ports of all ADSP-3128s. In other words, the register file space can be expanded in exactly the way a designer normally builds up a random access memory.

The only restriction on this method of 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-3128s.

Port Select removes this restriction for Double-Precision mode (only). Like Bank Select, the Port Select controls track the ADSP-3128'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 21 and 22). If writes are currently enabled, BS or A/B/Eadr$_6$ must be set up in phase α; 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. Also, if there is any possibility of a Write Flow-Through in phase γ, BS or A/B/Eadr$_6$ will have to be LO throughout phase γ.

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-3128 from Single-Precision to Double-Precision and conversely as long as these restrictions are observed.

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 ($n \pm 64$). The LSW of the double-precision word will be in n; the MSW in $n \pm 64$. By switching from Double- to Single-Precision, the user can independently access the LSW and the MSW.

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 21).

Any ADSP-3128 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-3128 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-3128 provides separate pins for driver GND and V_{DD}s and logic GND and V_{DD}s. For printed circuit boards, the ADSP-3128'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, the driver V_{DD} should be bypassed to the driver GND with 0.1μF ceramic and 20μF tantalum capacitors. The logic GND and V_{DD} need a

separate 0.1μF bypass. 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.

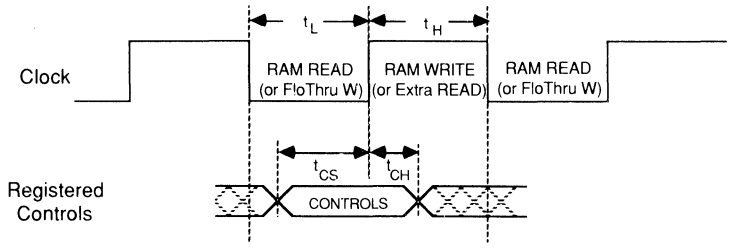
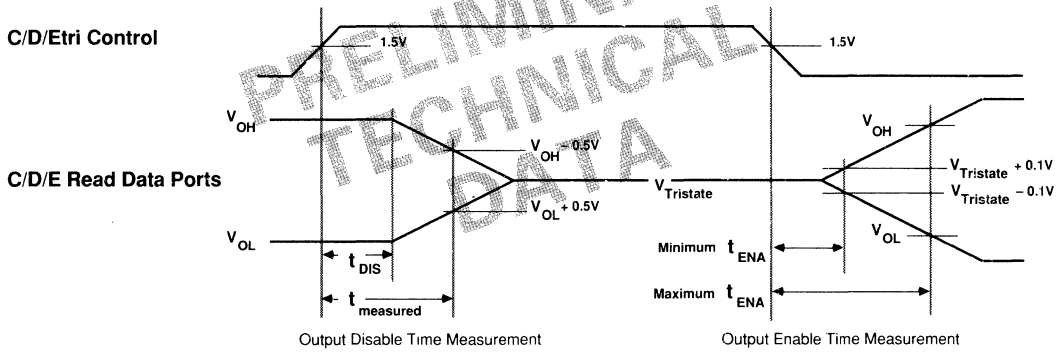


Figure 2. Register Controls Timing



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.

Figure 3. Three-State Disable and Enable Timing

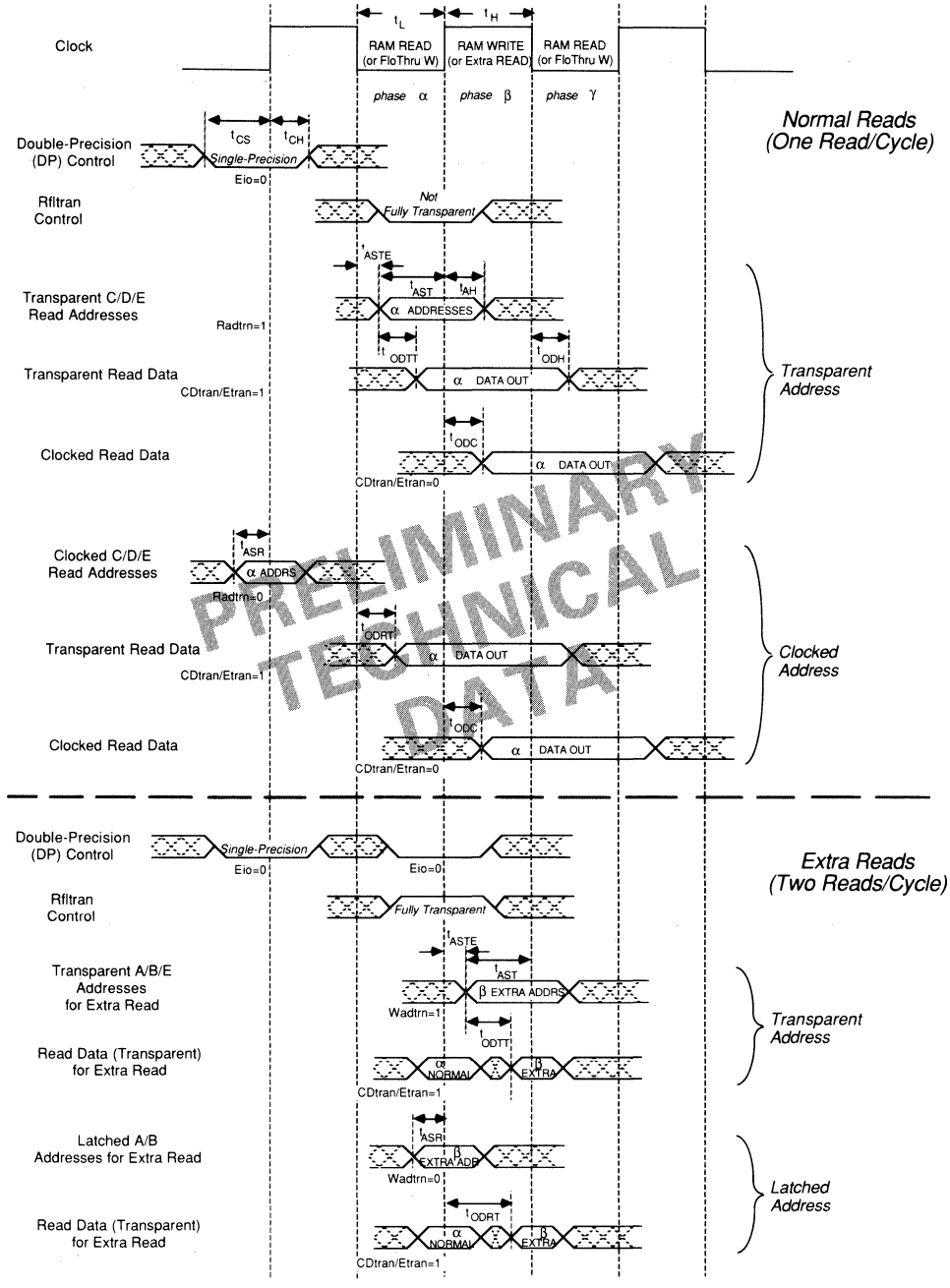


Figure 4. Single-Precision Read Output Timing

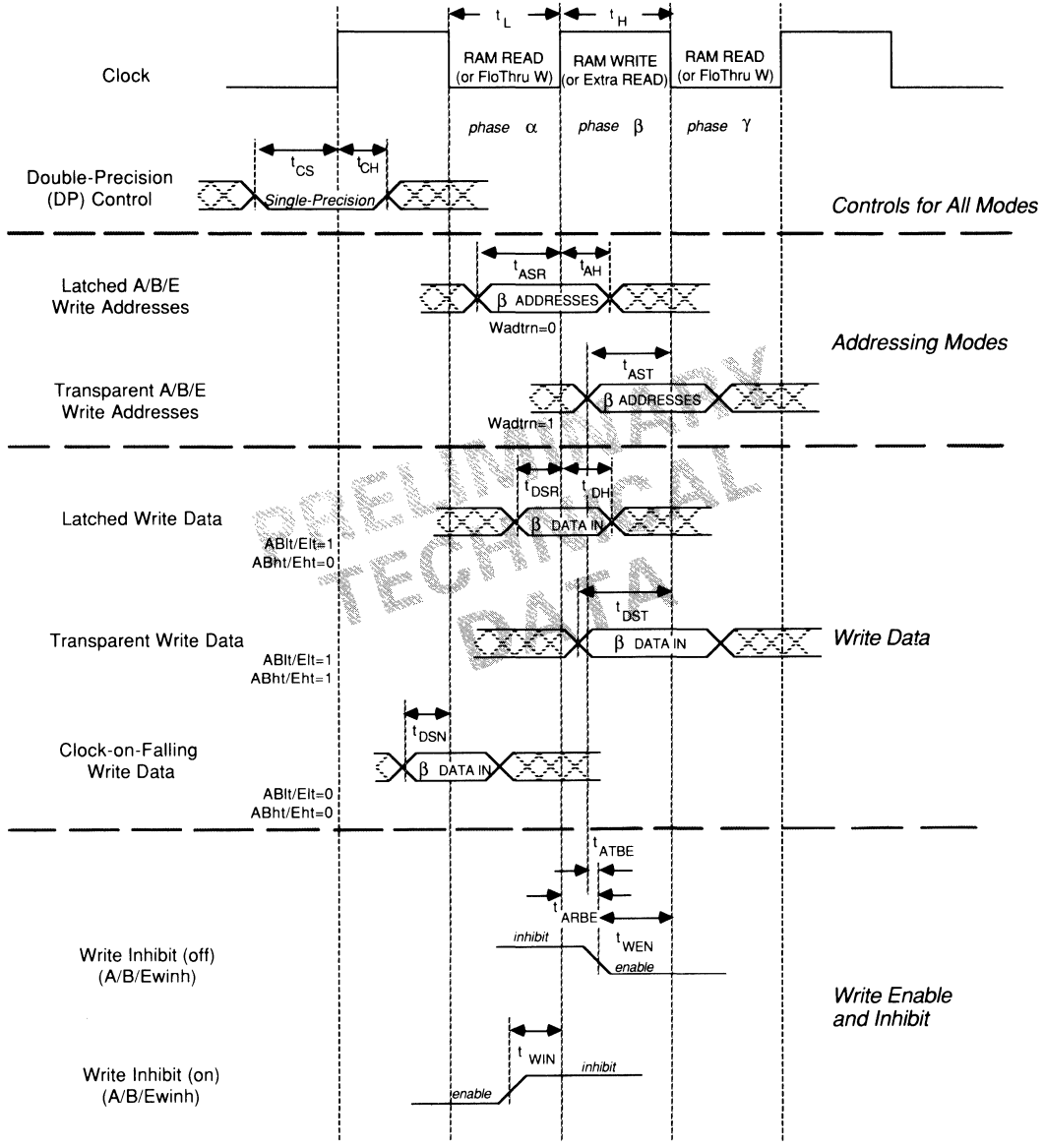
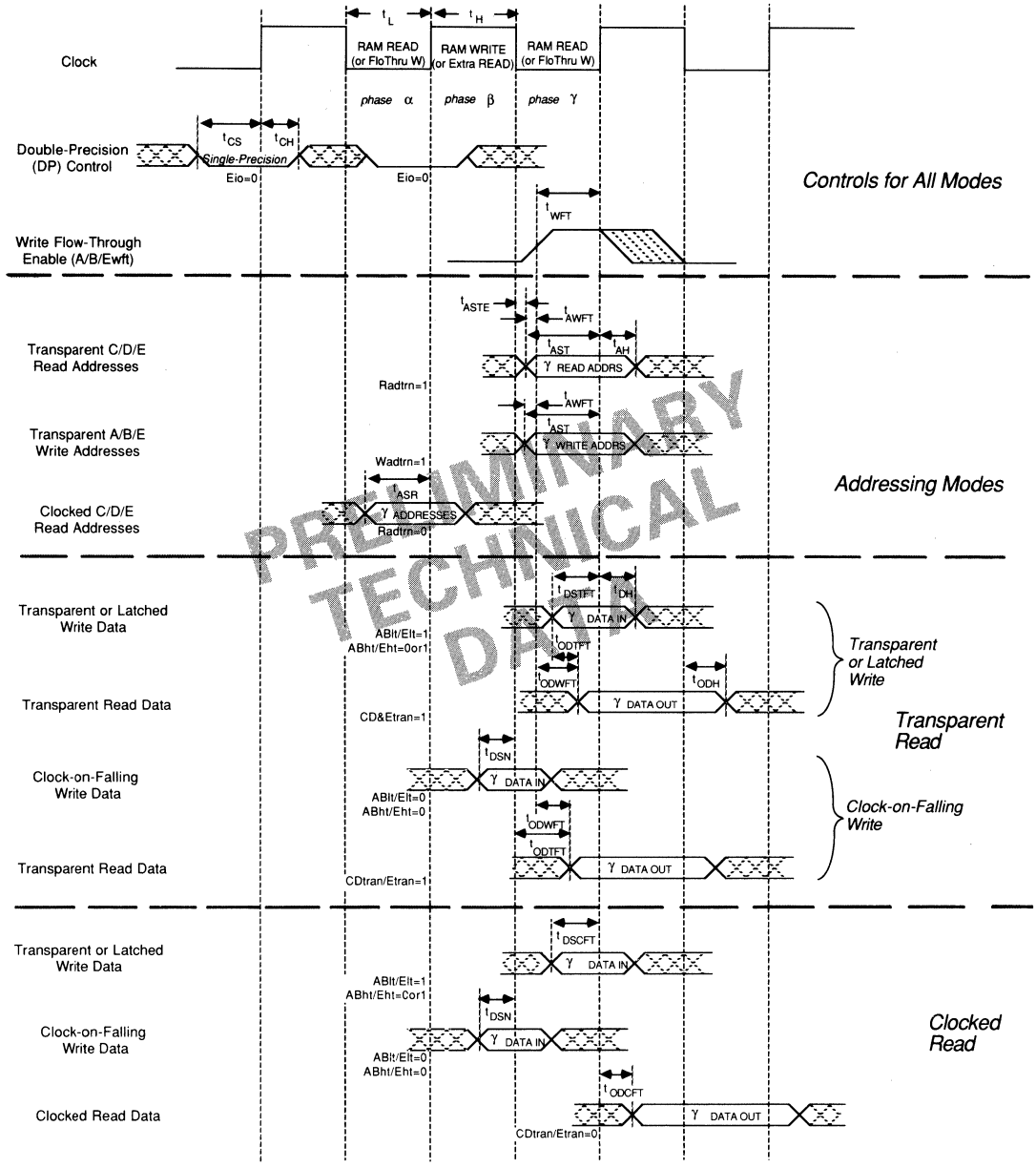
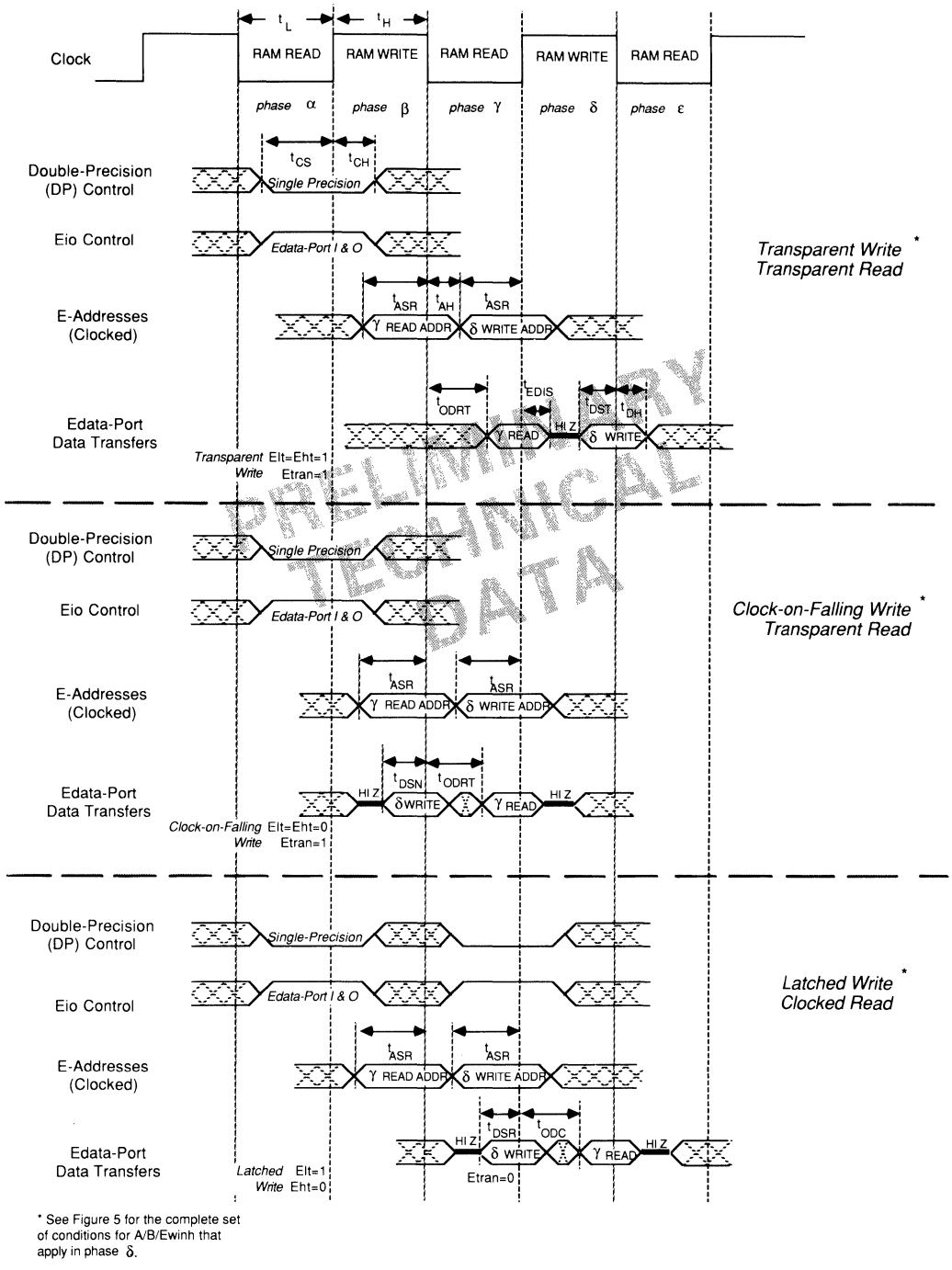


Figure 5. Single-Precision Normal Write Input Timing



* If Eio=1, data outputs from the EdataPort will cross only a single clock edge, as shown in Figure 7.

Figure 6. Single-Precision Write Flowthrough Timing



* See Figure 5 for the complete set of conditions for A/B/Ewinh that apply in phase δ.

Figure 7. Single-Precision Timing for Bidirectional Edata-Port

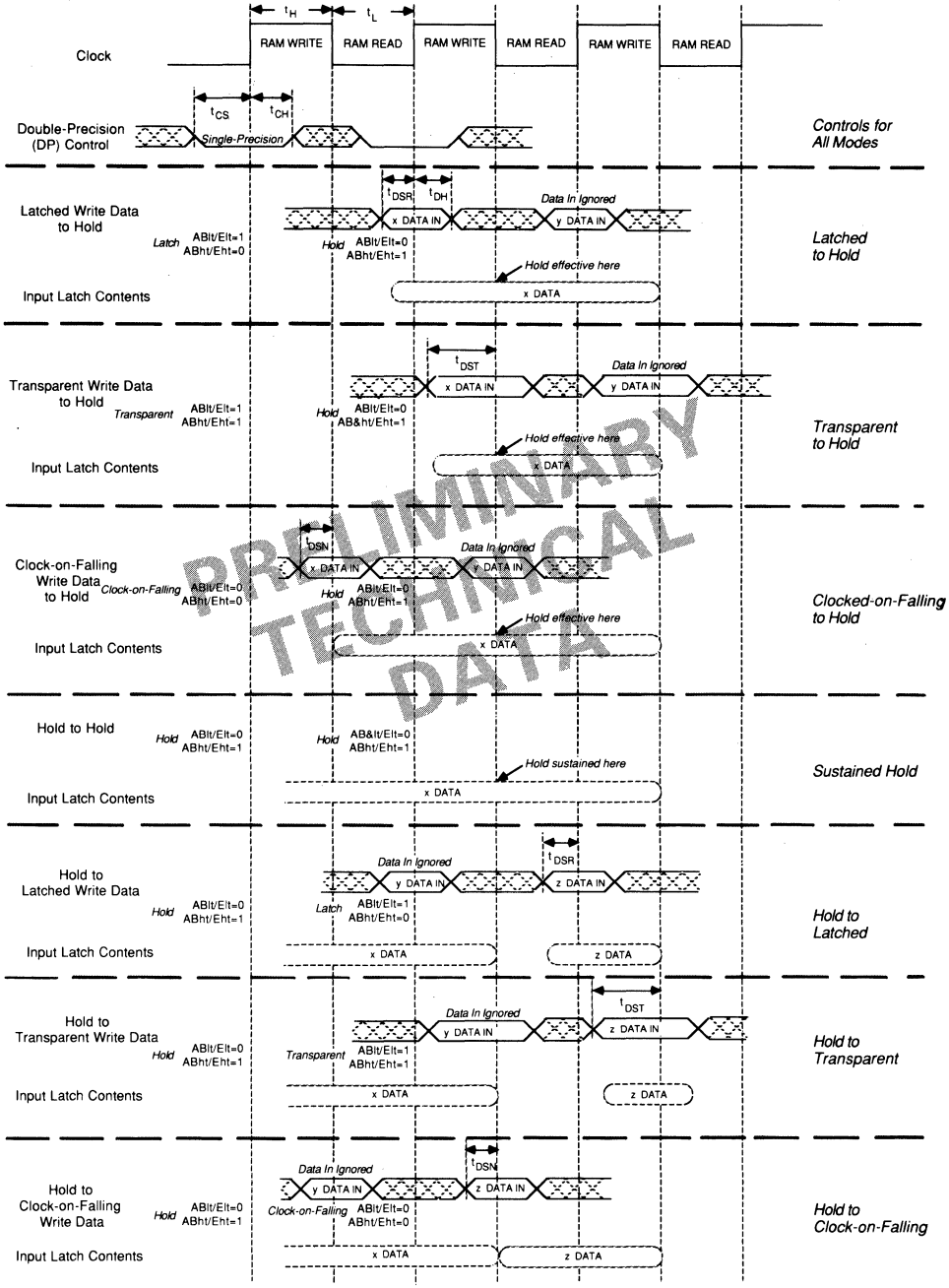
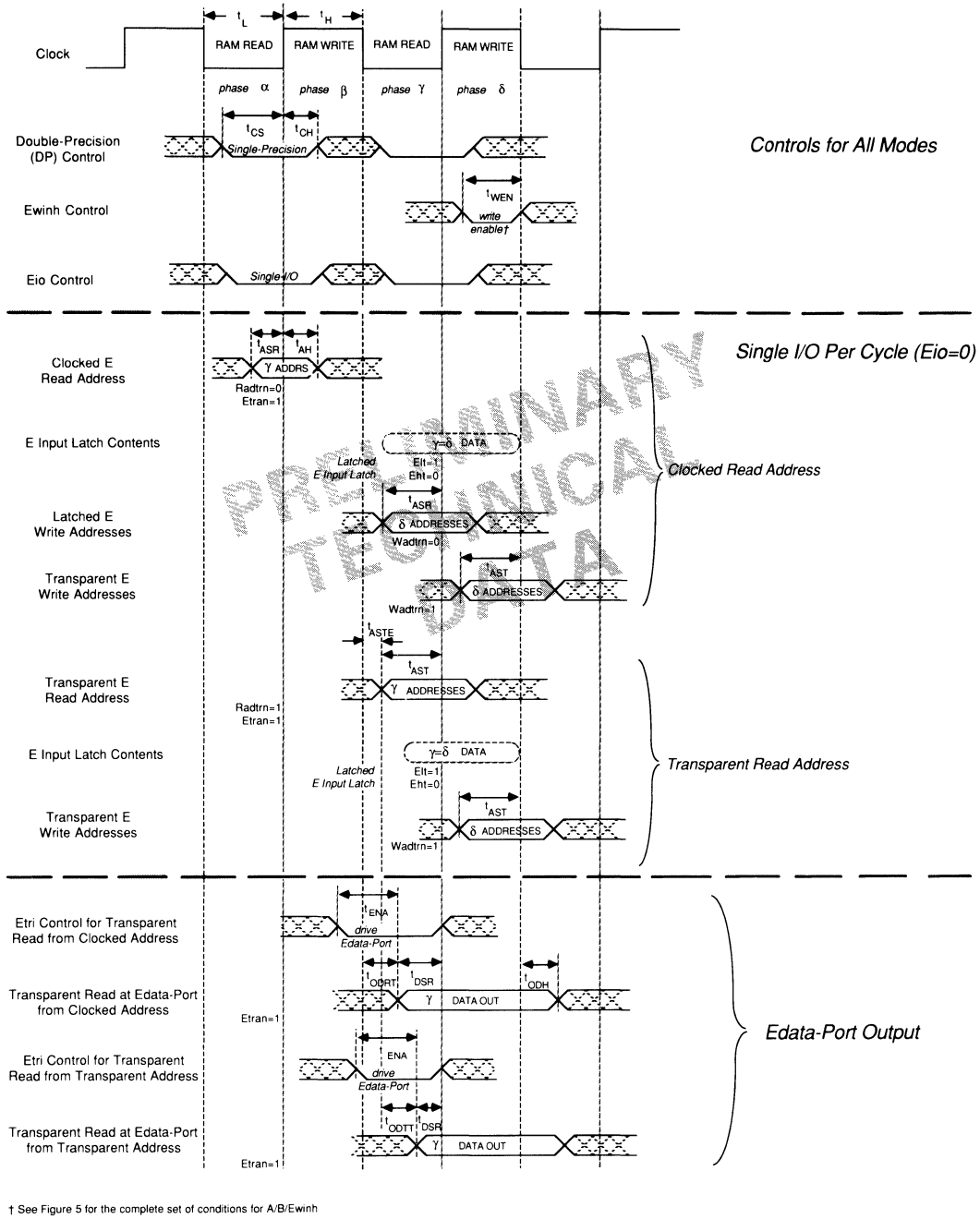
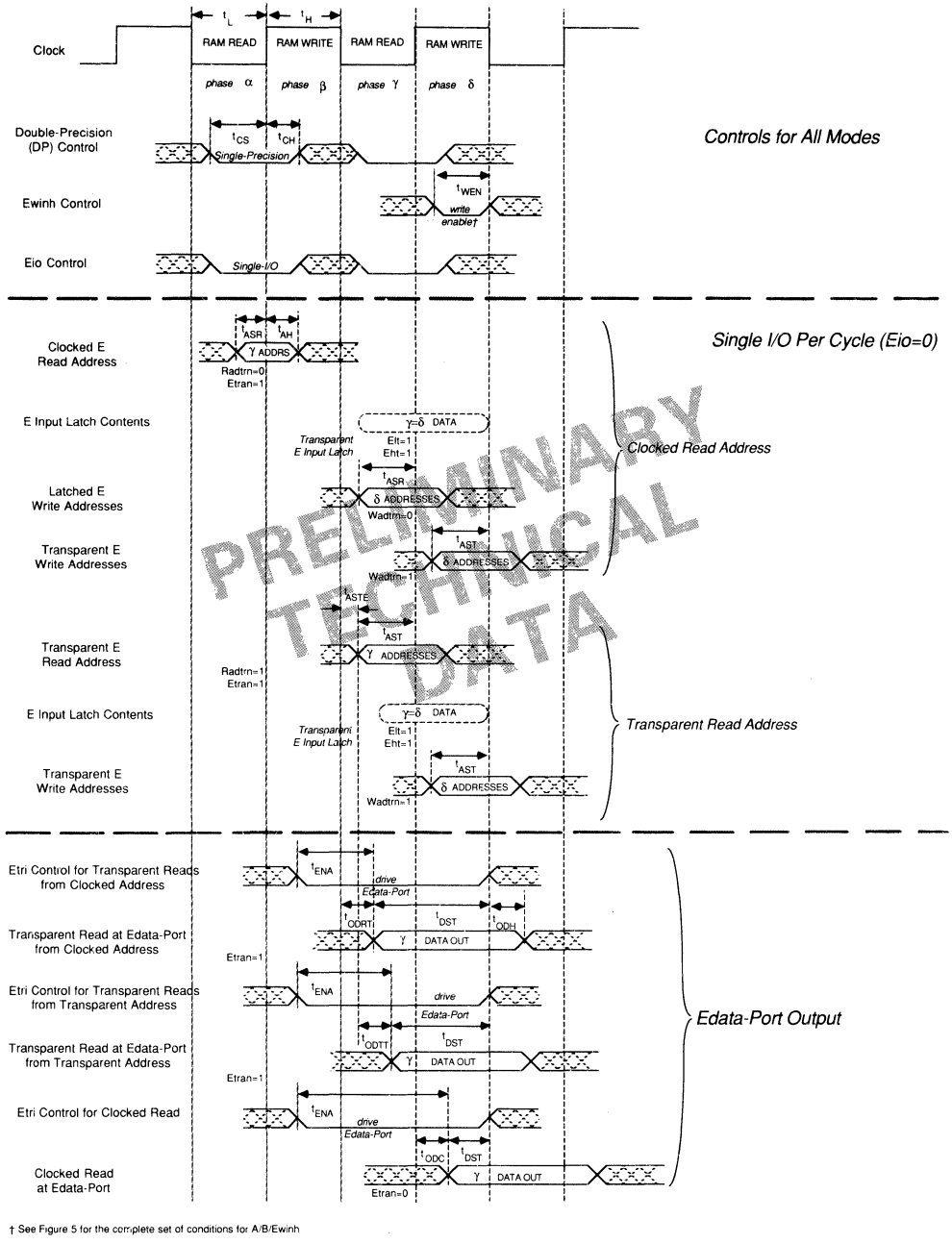


Figure 8. Single-Precision Write to Input Latches and Hold Timing



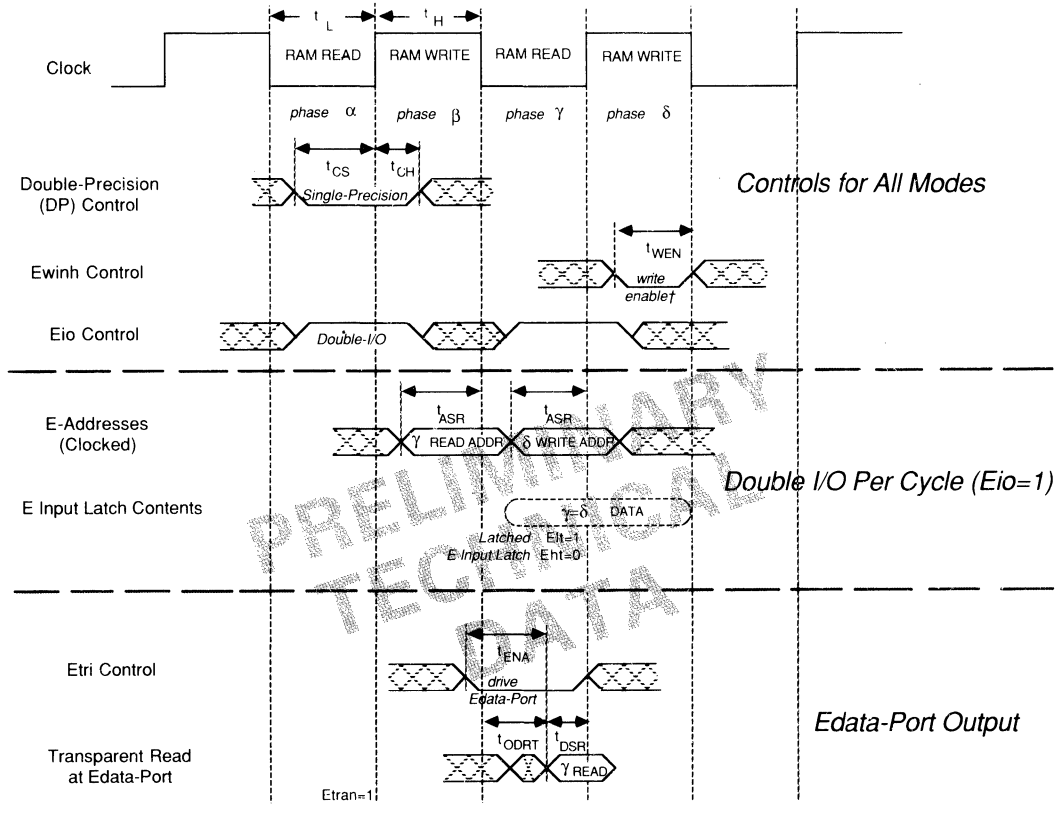
† See Figure 5 for the complete set of conditions for A/B/Ewinh

Figure 9. Single-Precision Timing for Register-to-Register Transfer via Edata-Port – Edata-Port in Single I/O per Cycle Mode (Eio = 0) and Latched Inputs



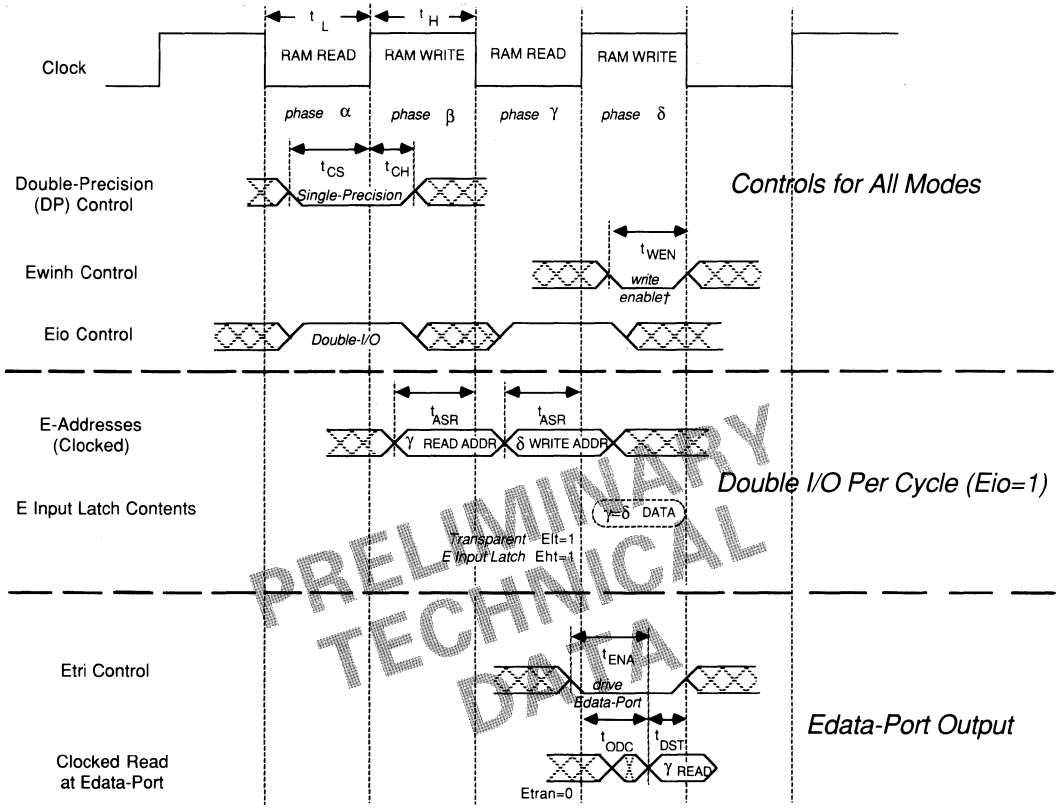
† See Figure 5 for the complete set of conditions for A/B/Ewinh

Figure 10. Single-Precision Timing for Register-to-Register Transfer via Edata-Port – Edata-Port in Single I/O per Cycle Mode (Eio = 0) and Transparent Inputs



† See Figure 5 for the complete set of conditions for A/B/Ewinh

Figure 11. Single-Precision Timing for Register-to-Register Transfer via Edata-Port – Edata-Port in Double I/O per Cycle Mode (Eio = 1) and Latched Inputs



† See Figure 5 for the complete set of conditions for A/B/Ewinh

Figure 12. Single-Precision Timing for Register-to-Register Transfer via Edata-Port – Edata-Port in Double I/O per Cycle Mode (Eio = 1) and Transparent Inputs

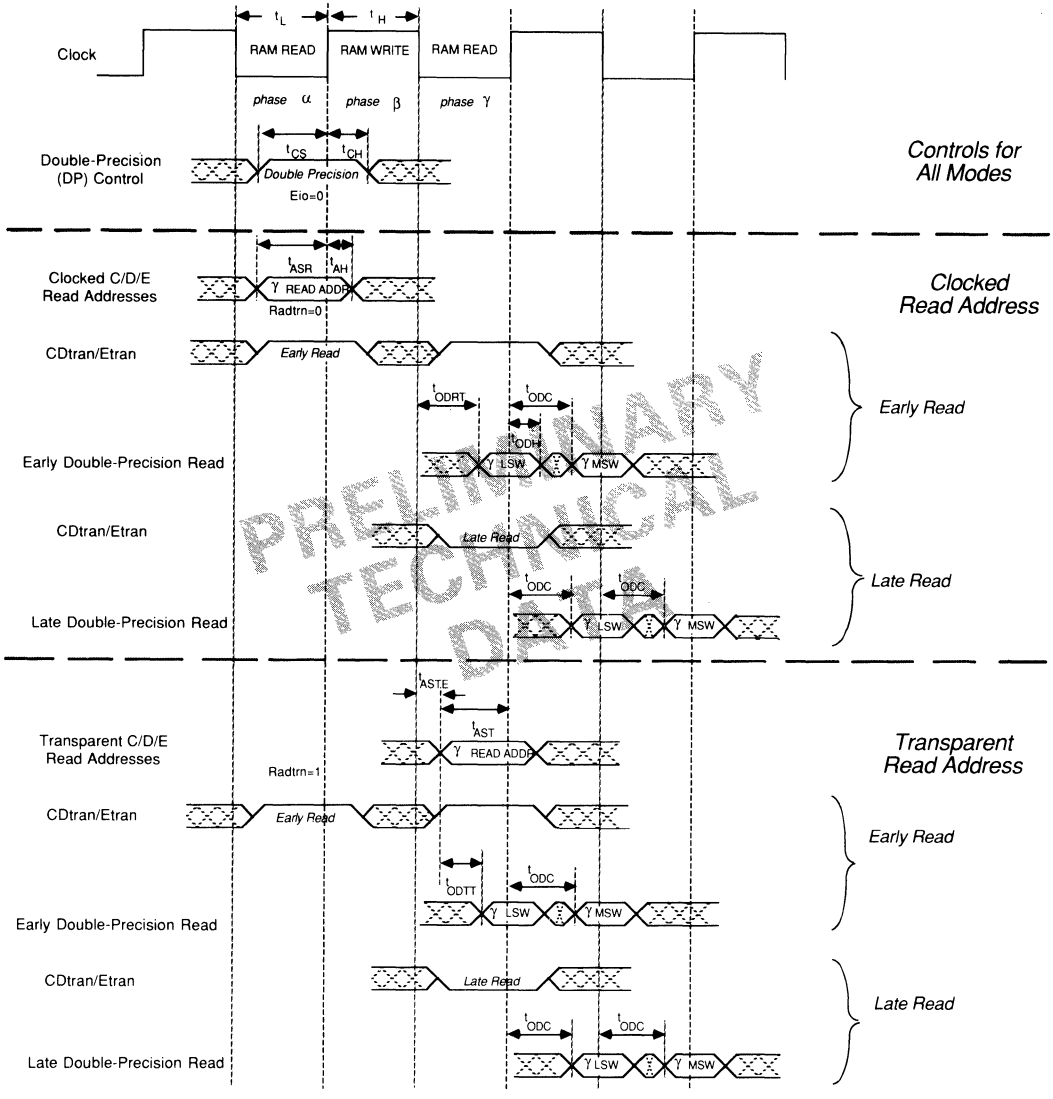


Figure 13. Double-Precision Read Output Timing.

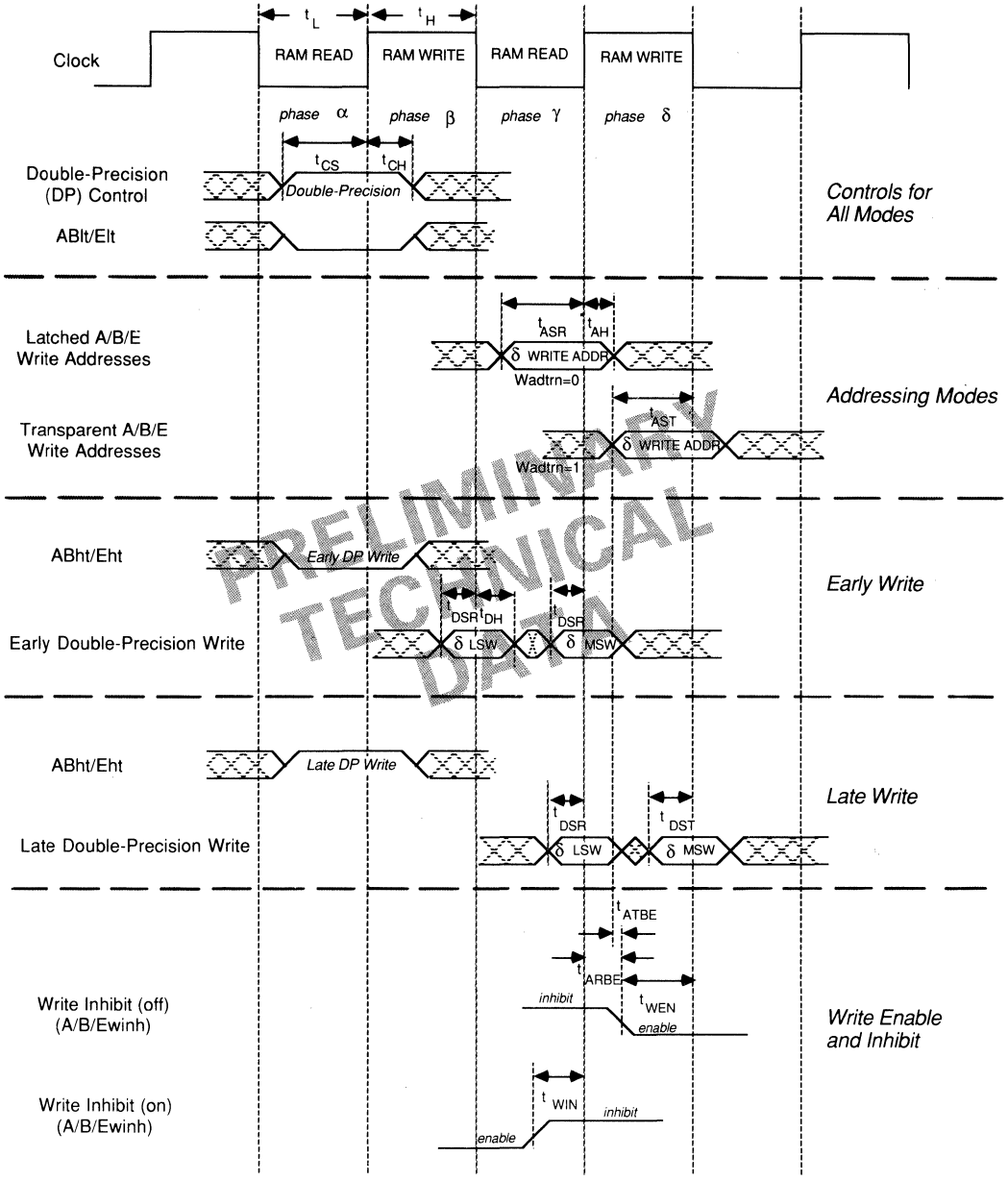


Figure 14. Double-Precision Normal Write Input Timing

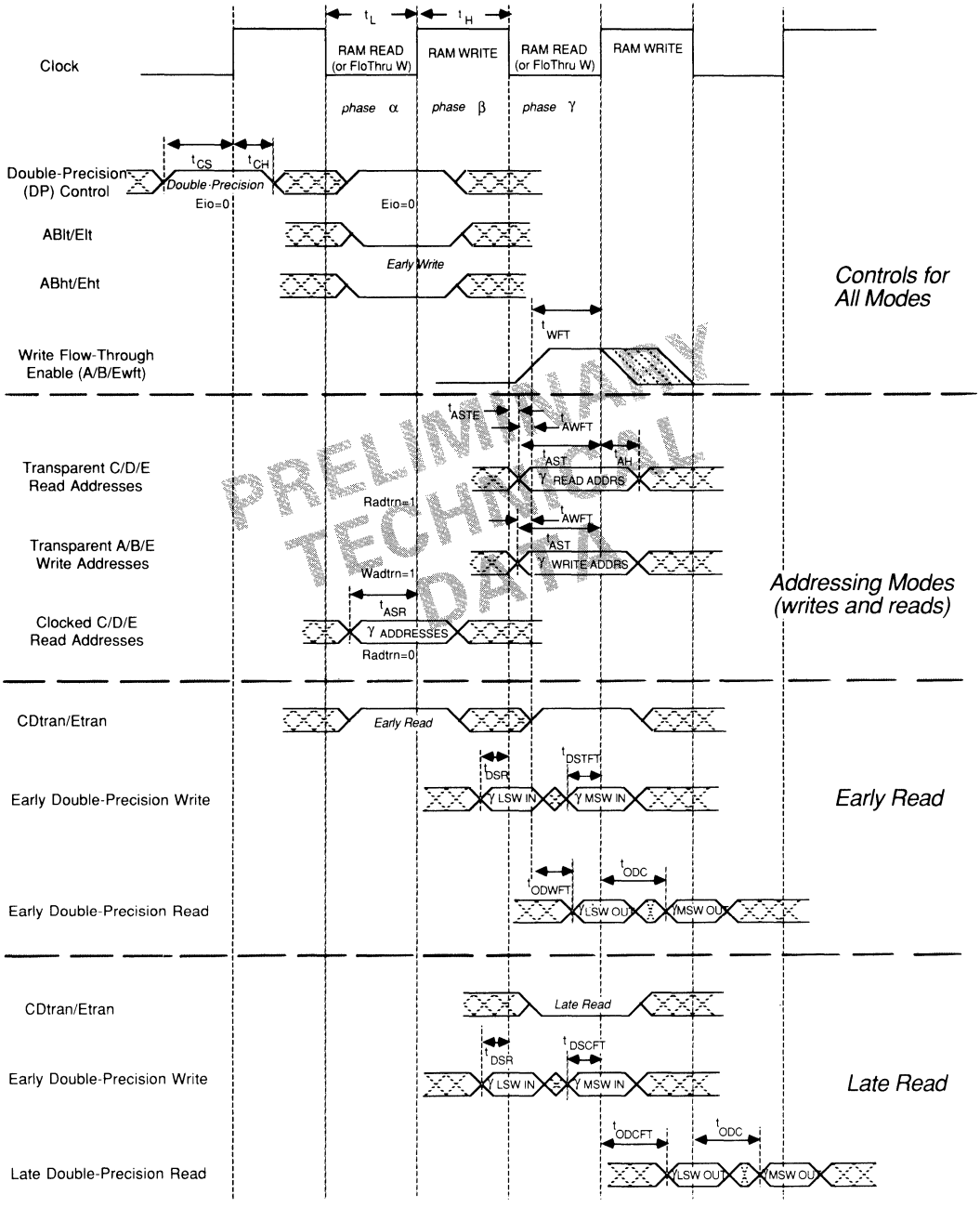


Figure 15. Double-Precision Write Flow-Through Time (Early DP Write Only)

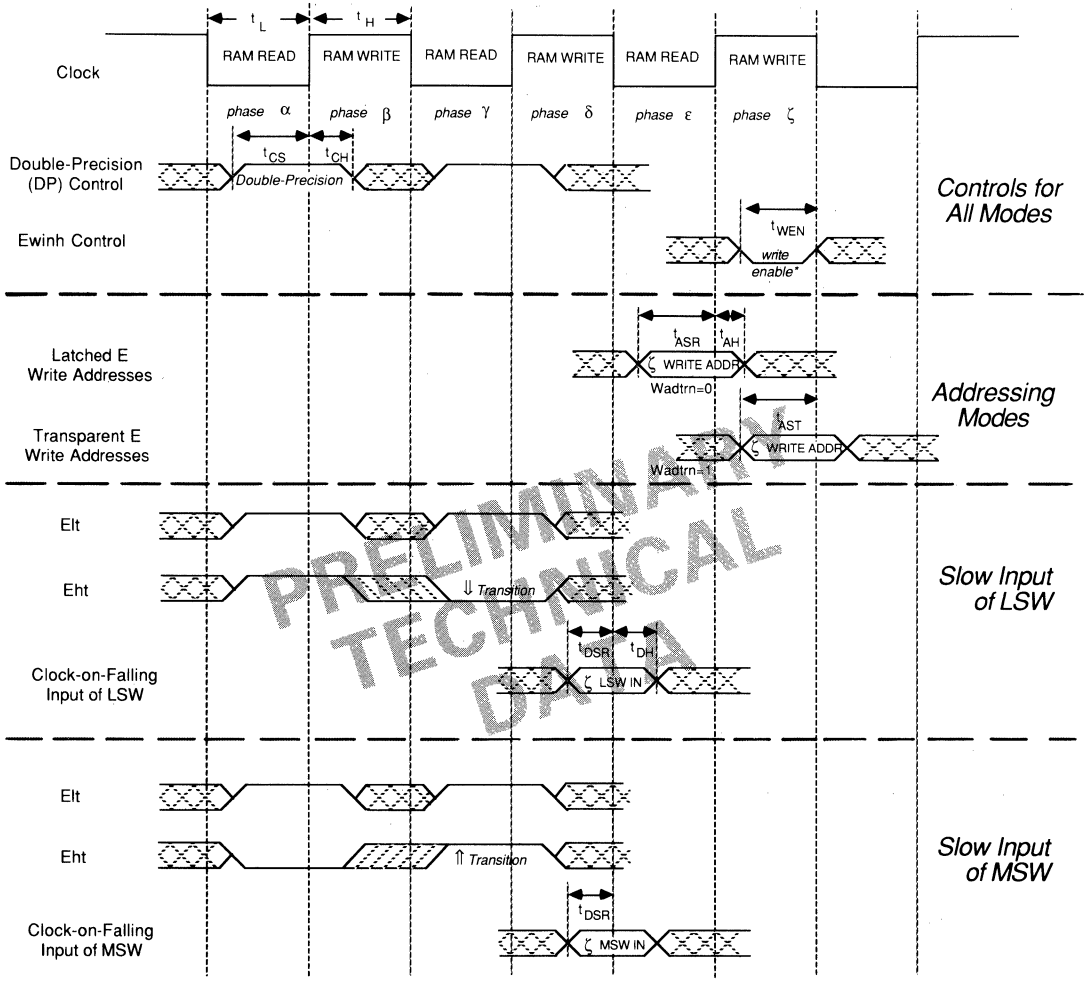


Figure 16. Double-Precision Slow Input Timing

*See Figure 14 for the complete set of conditions for A/B/Ewinh.

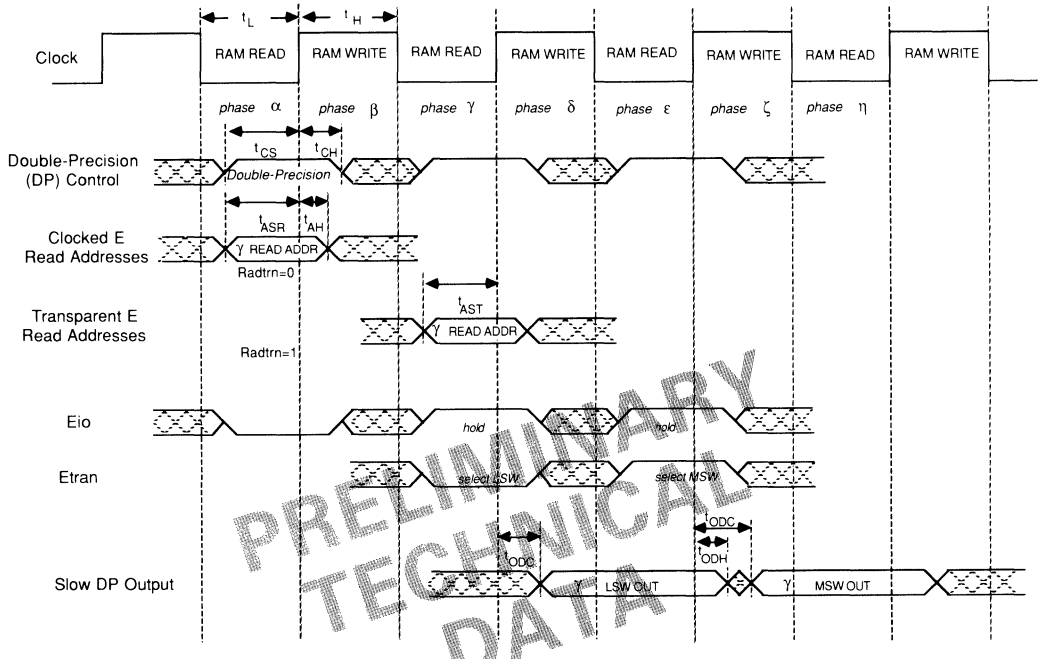


Figure 17. Double-Precision Slow Output Timing

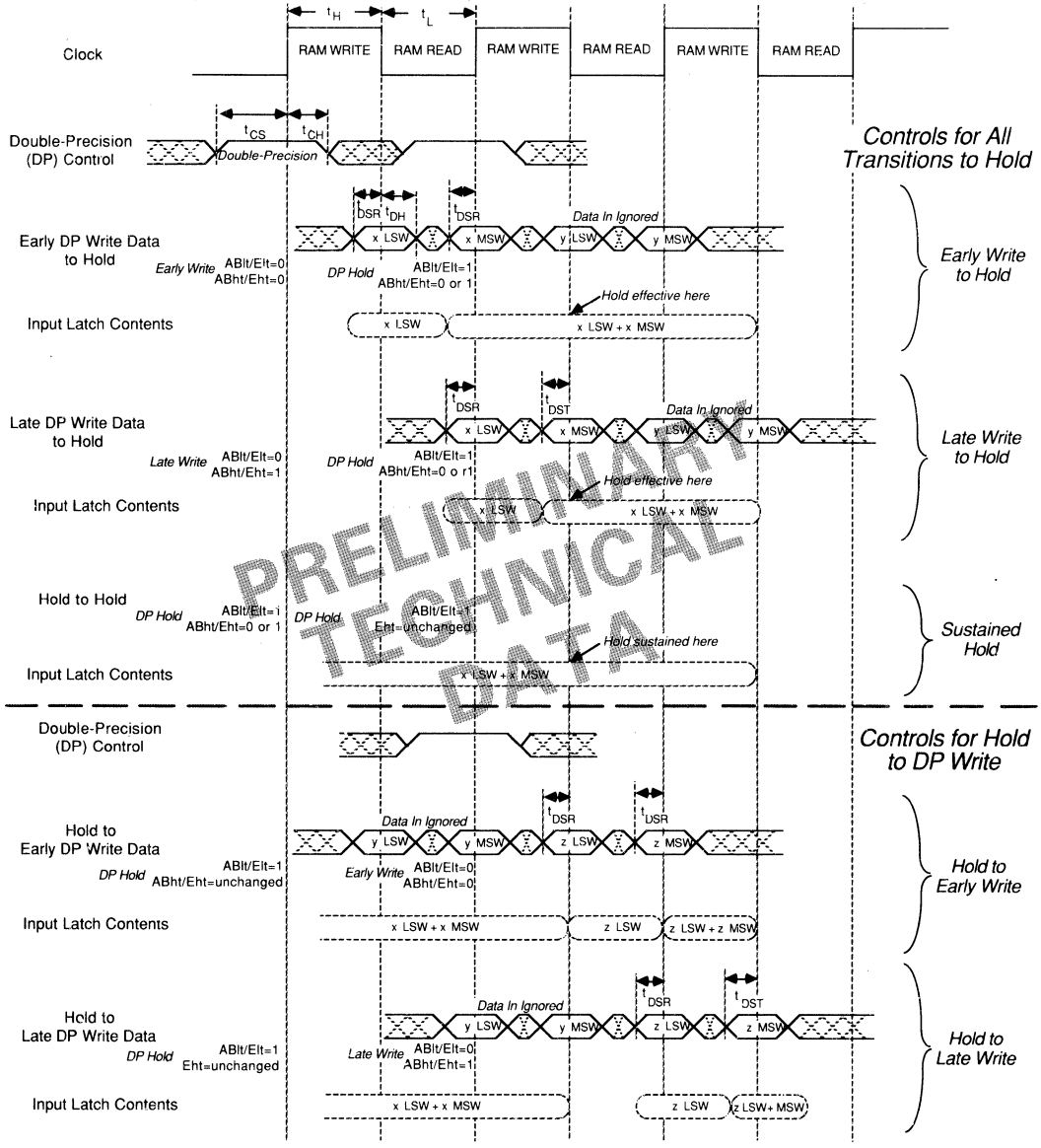


Figure 18. Double-Precision Write to Input Latches and Hold Timing

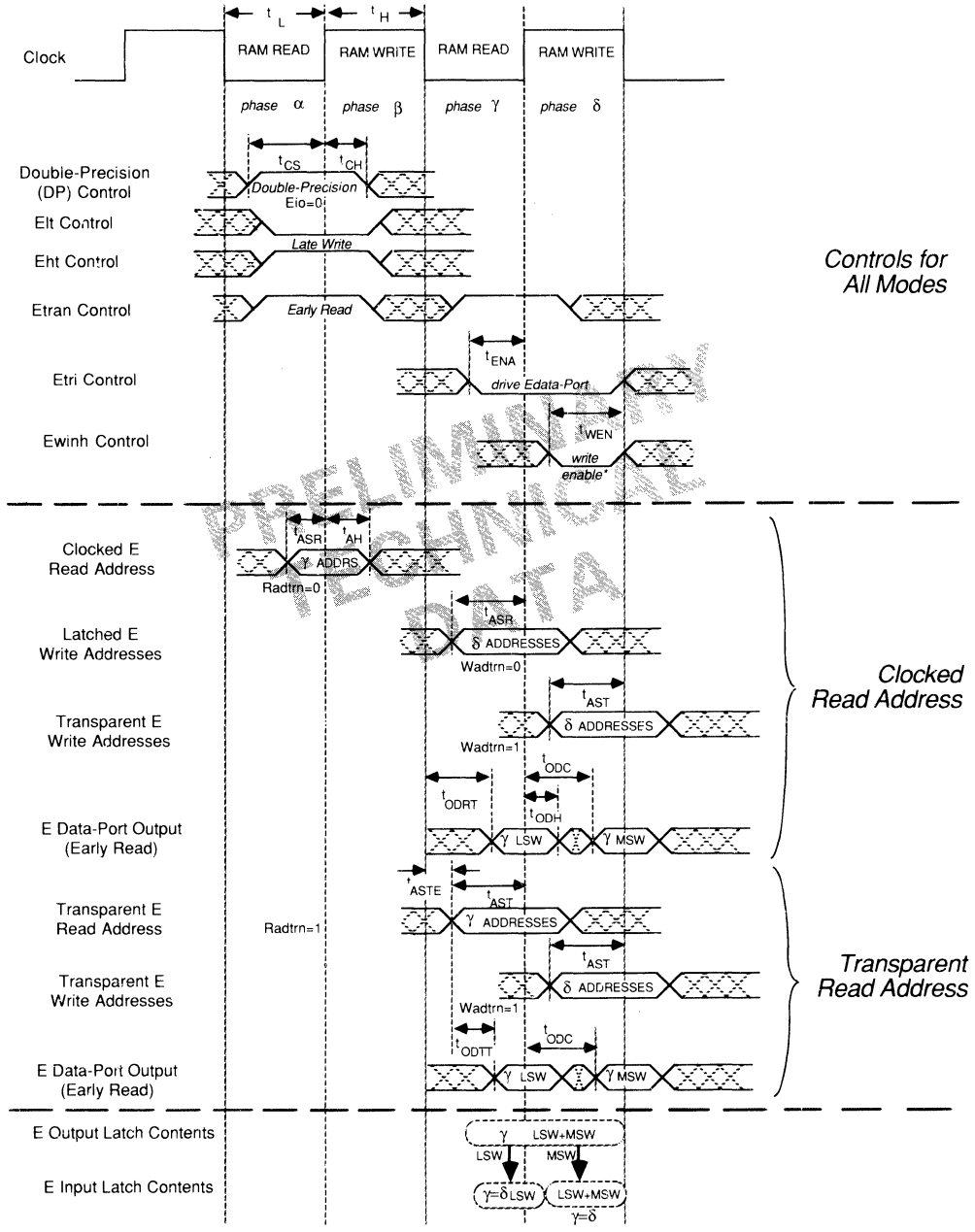


Figure 19. Double-Precision Timing for Register-to-Register Transfer via Edata-Port, Early Read/Late Write Mode

*See Figure 14 for the complete set of conditions for A/B/Ewinh.

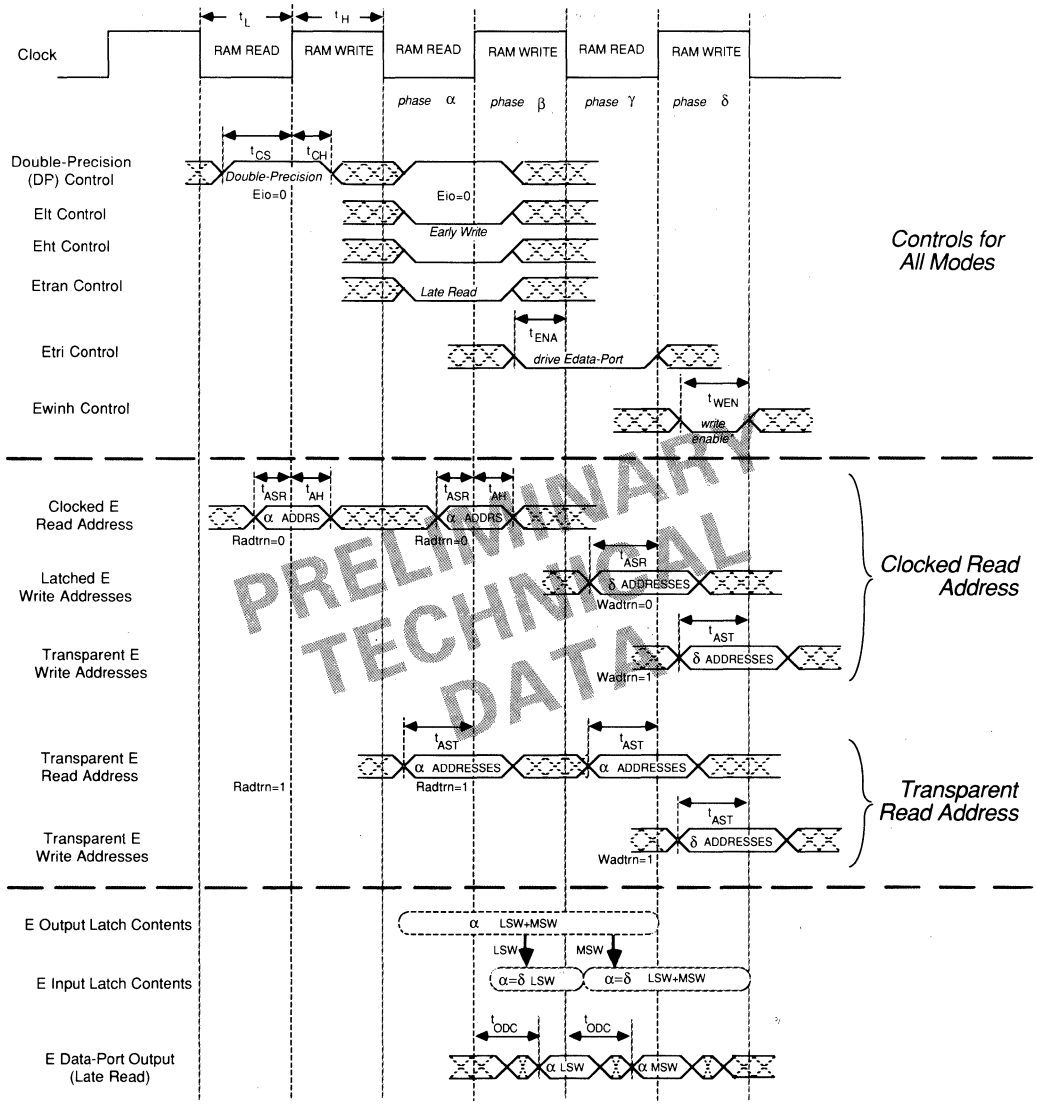


Figure 20. Double-Precision Timing for Register-to-Register Transfer via Edata-Port, Late Read/Early Write Mode

*See Figure 14 for the complete set of conditions for A/B/Ewinh.

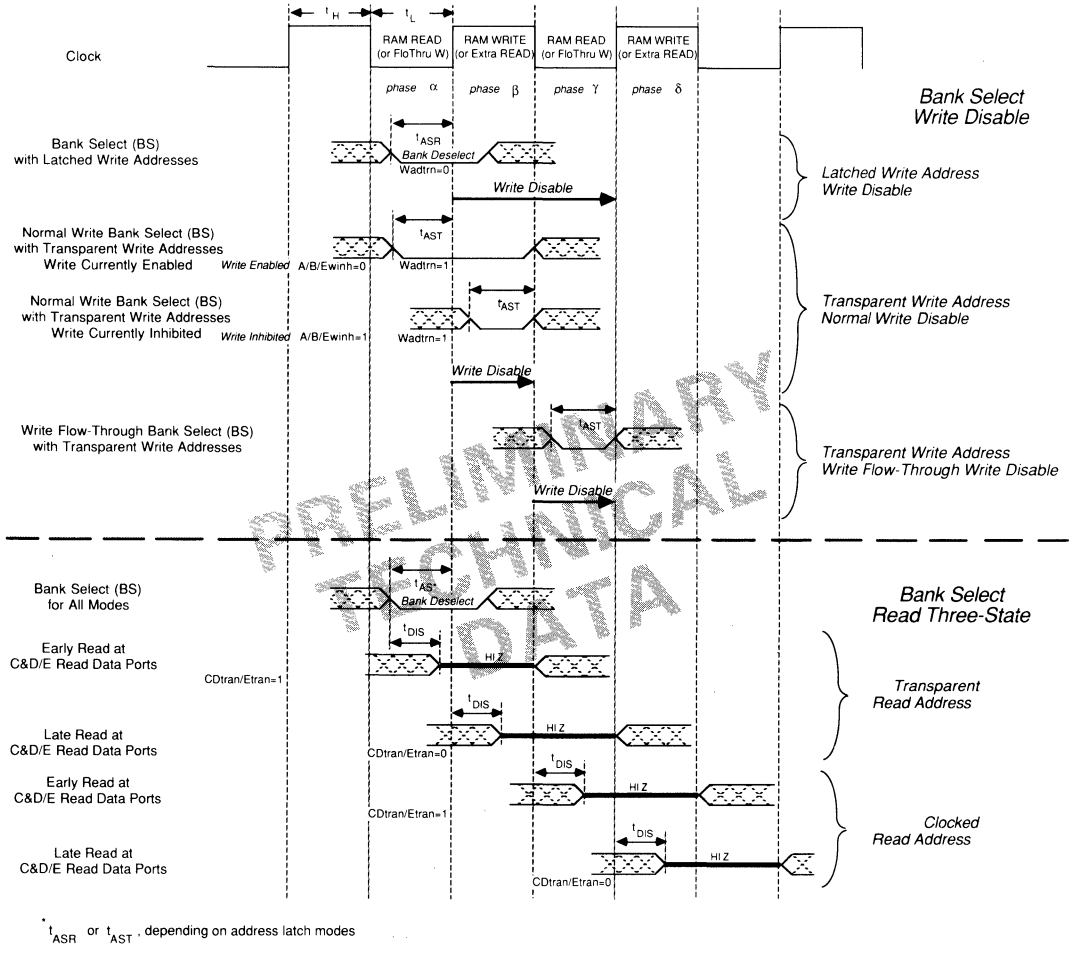


Figure 21. Chip Select Timing

4

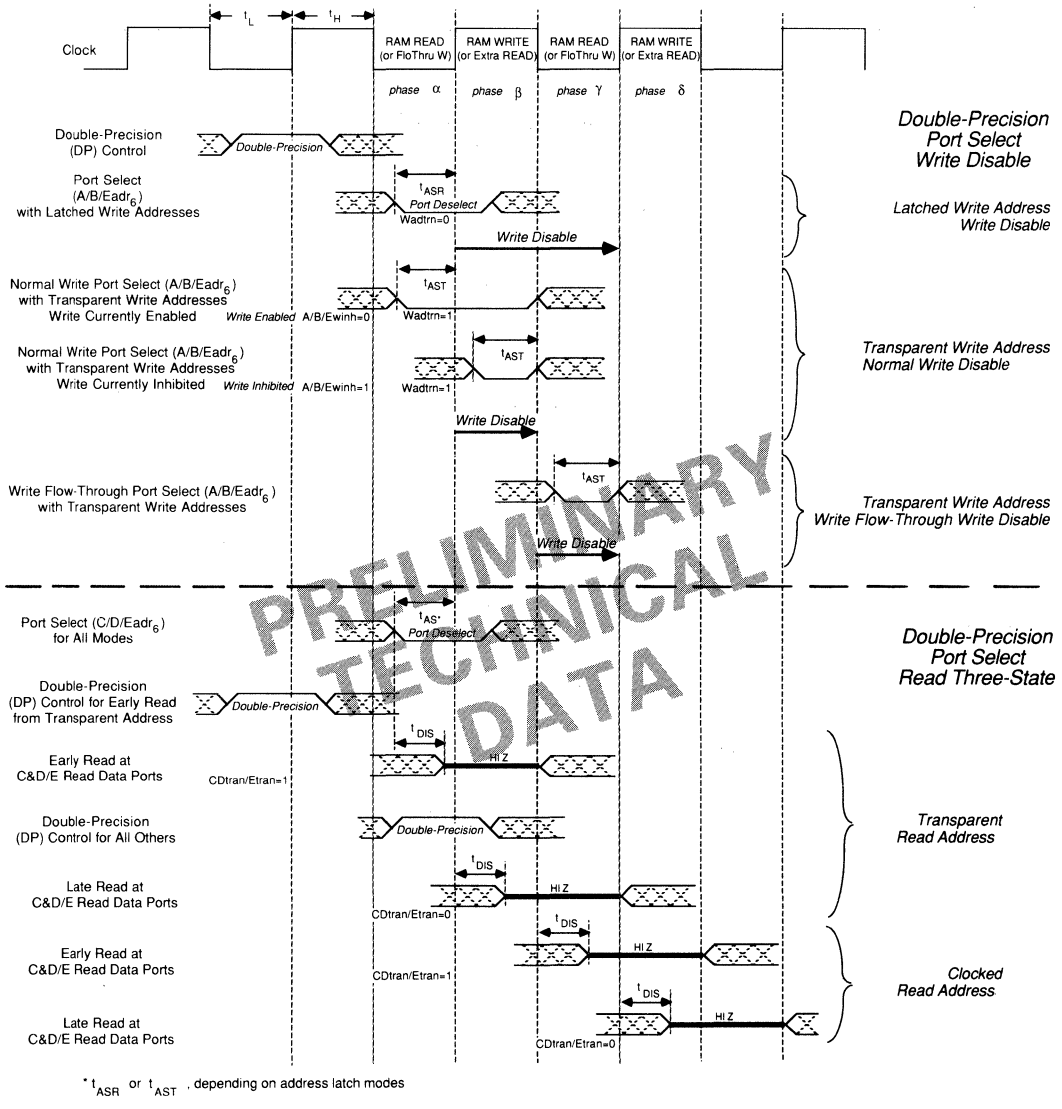
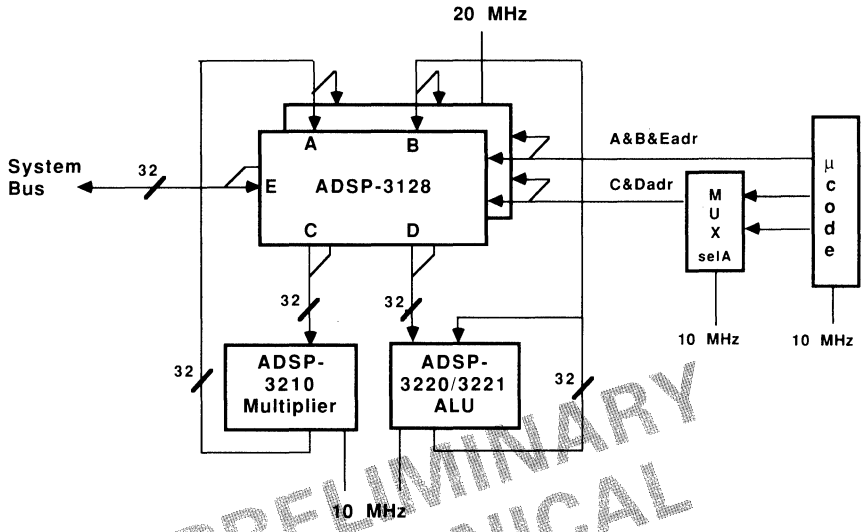
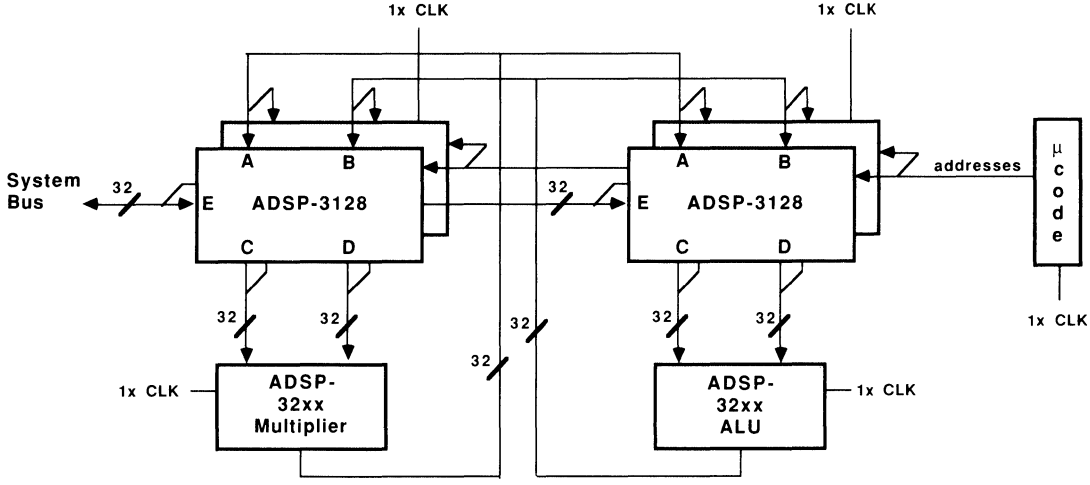


Figure 22. Port Select Timing



Muxing the read addresses allows two reads (per 10 MHz) for loading the input ports of both the ADSP-3210 and ADSP-3220/3221 with two 32-bit words per 3210/3220/3221 cycle (10MHz) while still using 10 MHz 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 23. ADSP-3128 Single-Precision Applications with ADSP-3210/ADSP-3220/ADSP-3221. Microcode Operates at 10MHz



Double-Precision mode allows transfers of both MSW and LSW in a single cycle while still using µcode at the same cycle rate. Pairing pairs of ADSP-3128s 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-3128s.

Figure 24. ADSP-3128 Seven-Port Double-Precision Application with ADSP-3211 and ADSP-3220/ADSP-3221. Microcode Operates at 10MHz

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-3128 J and K Grades Min	Max	ADSP-3128 S and T Grades[2] 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

Parameter		Test Conditions	ADSP-3128 J and K Grades Min	Max	ADSP-3128 S and T Grades[2] Min	Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.0		V
V_{IHA}	High-Level Input Voltage, CLK and All Asynchronous Inputs	@ V_{DD} = max	2.4		2.6		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					μA
I_{DD}	Supply Current	@ max clock rate; TTL inputs					mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 0V; High Z Output					μA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V					mA

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-3128JG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3128KG	0 to +70°C	144-Pin Grid Array	G-144A
ADSP-3128SG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3128TG	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3128SG/+	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3128TG/+	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3128SG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A
ADSP-3128TG/883B	−55°C to +125°C	144-Pin Grid Array	G-144A

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

SWITCHING CHARACTERISTICS

		ADSP-3128								
		J Grades 0 to 70°C		K Grades 0 to 70°C		S Grades[2] −55°C to +125°C		T Grades[2] −55°C to +125°C		
		Min	Max	Min	Max	Min	Max	Min	Max	Unit
t_{CLK}	Clock Period – Clocked Reads									ns
t_{CLKFT}	WFT Clock Period – Clked Reads									ns
t_{CLKS}	Clock Period – Trans Reads									ns
t_{CLKSFT}	WFT Clock Period – Transparent Reads									ns
t_{CLKA}	Clock Period – Transparent I/O									ns
t_{CLKAFT}	WFT Clock Period – Transparent I/O									ns
t_L	Clock LO Period									ns
t_H	Clock HI Period									ns
t_{HIER}	Clock HI Period – Write plus Extra Read									ns
t_{LOWR}	Clock LO Period – Write Flow-Through									ns
t_{CS}	Control Setup									ns
t_{CH}	Control Hold									ns
t_{AST}	Transparent Address Setup									ns
t_{ASTE}	Trans Address Max Usable Setup									ns
t_{ASR}	Registered Address Setup									ns
t_{AH}	Address Hold									ns
t_{ENA}	Three-State Enable Delay									ns
t_{DIS}	Three-State Disable Delay									ns
t_{EDIS}	Three-State Eport Auto Disable									ns
t_{ODTT}	Trans Adr-to-Trans Output Delay									ns
t_{ODC}	Clk-to-Data Output Delay									ns
t_{ODRT}	Clkd Adr-to-Trans Output Delay									ns
t_{ODH}	Output Data Hold									ns
t_{DSR}	Latched Data Setup									ns
t_{DST}	Transparent Data Setup									ns
t_{DSN}	Clock-on-Falling Data Setup									ns
t_{DH}	Input Data Hold									ns
t_{WEN}	Write Inhibit-to-Enable Setup									ns
t_{WIN}	Write Enable-to-Inhibit Setup									ns
t_{ATBE}	Trans Adr to Write Enable									ns
t_{ARBE}	Clock to Write Enable									ns
t_{WFT}	Write Flow-Through Setup									ns
t_{AWFT}	Trans Adr to WFT									ns
t_{DSTFT}	WFT Data Setup – Trans Read									ns
t_{DSCFT}	WFT Data Setup – Clkd Read									ns
t_{ODCFT}	WFT Clk-to-Data Output Delay									ns
t_{ODTFT}	WFT Address to Trans Data									ns
t_{ODWFT}	WFT to Trans Data									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 3 and t_{EDIS} in Figure 7. 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 3 and t_{EDIS} in Figure 7 and 2) t_{DS} and t_{DH} which are measured from clock V_{IHA} to data input V_{IH} or V_{IL} crossing points.

[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-3128 can be found in Analog Devices Military Databook. S and T grade parts are also available with optional high-reliability "PLUS" processing as shown in Figure 26. Regular S and T grade parts are tested at + 125°C.

Specifications subject to change without notice.

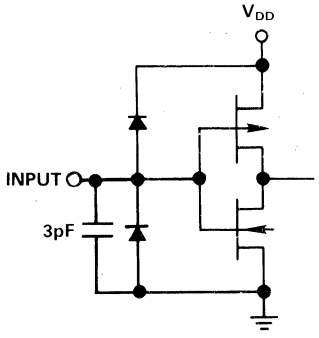


Figure 25. Equivalent Input Circuits

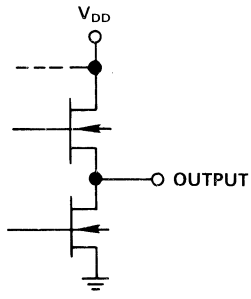


Figure 26. Equivalent Output Circuits

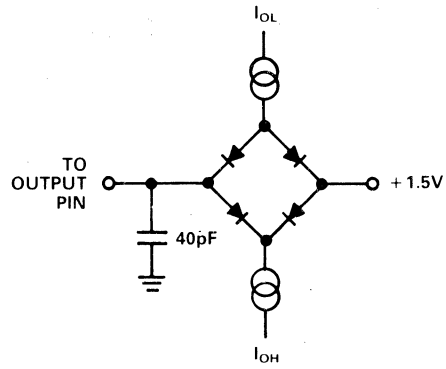


Figure 27. Normal Load for ac Measurements

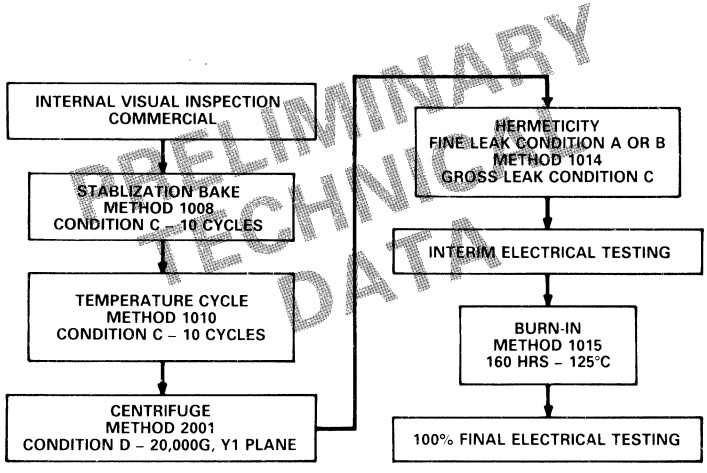


Figure 28. PLUS Processing Flow

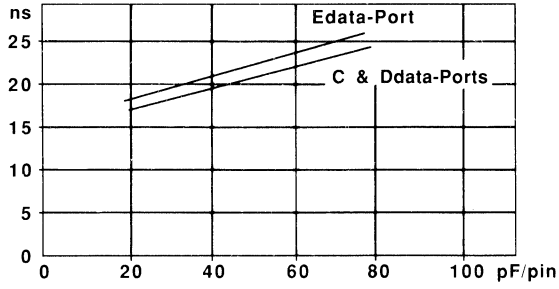


Figure 29. Clocked Output Delay as a Function of Load Capacitance

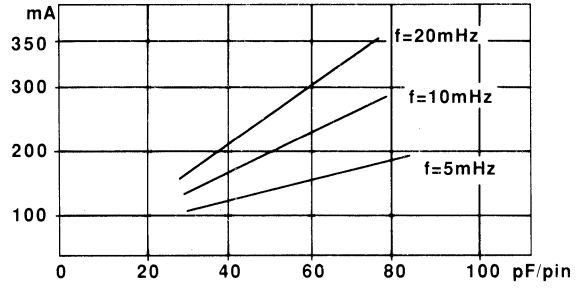
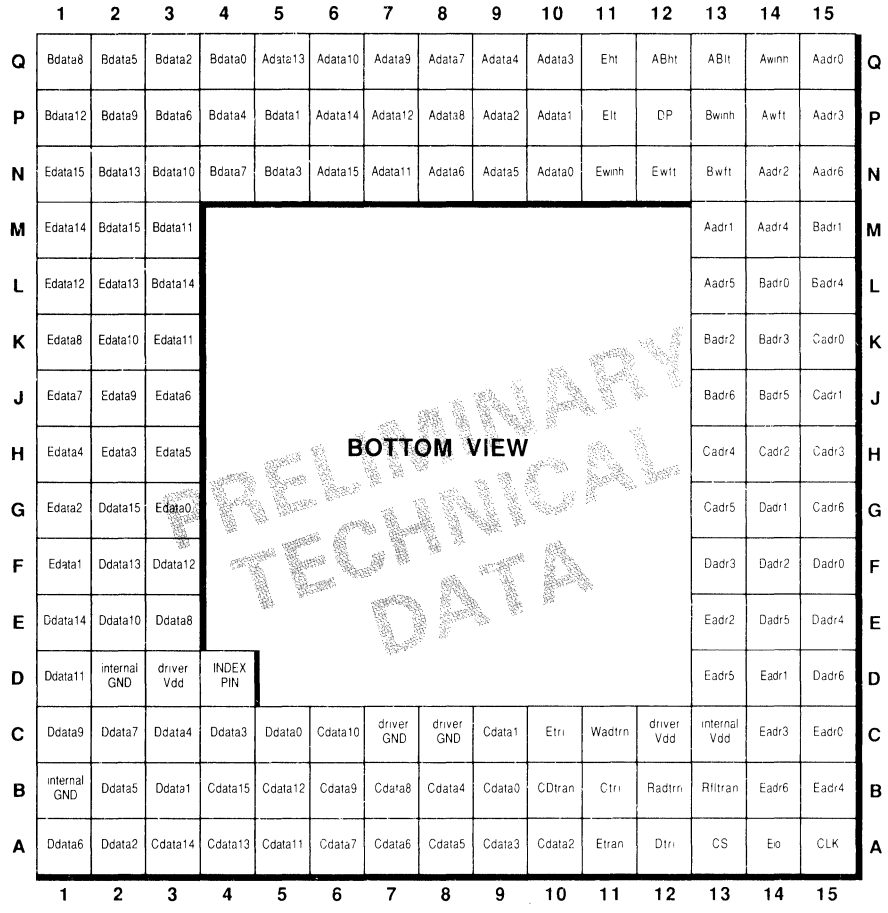


Figure 30. Typical Current Consumption as a Function of Load Capacitance, All Outputs Switching



ADSP-3128 Pin Configuration

Fixed-Point Components

Contents

5

Introduction

GENERAL INFORMATION

In 1982, 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 have significantly increased the speed over the original 5 micron specifications while remaining pin compatible.

Our standard production process is currently the 1.5 micron process. All 1.5 micron fixed-point multipliers and multiplier/accumulators formerly issued in 5 micron are now identified by an "A" suffix on their four digit code, as in ADSP-1016A. New parts, such as the ADSP-1101, do not use the "A" suffix.

The specifications in this databook for multipliers and multiplier/accumulators supersede the specifications in all previous publications, including the Analog Devices Databook, the 1986 Supplement and 1986 and earlier data sheets. This publication takes precedence over prior publications in the event of conflicting information.

Contact your local sales office for information concerning new, faster versions of these fixed-point components.

BIPOLAR ADDITIONS TO THE PRODUCT LINE

The ADSP-7018 and ADSP-8018 are the latest additions to our multiplier family. Fabricated using a scaled bipolar process, these devices offer the ultimate in performance.

Selection Guides

FIXED POINT MULTIPLIERS

Word Size	Model No.	Multiplication[1] Time, ns Clocked Comm	Clocked MIL	Unclocked Comm	Unclocked MIL	I_{DD}[2] Comm	I_{DD}[2] MIL	Twos Comp	Data Formats Unsign Mag.	Mixed Mode	No. of Pins	Package Options[3]	Process	Logic Type
8×8	ADSP-1080A	J=45 K=33	S=55 T=45	N/A	N/A	45	55	X			40	D, N, E[4]	CMOS	TTL
8×8	ADSP-1081A	J=45 K=33	S=55 T=45	N/A	N/A	45	55		X		40	D, N, E[4]	CMOS	TTL
12×12	ADSP-1012A	J=75 K=50	S=90 T=60	J=105 K=80	S=125 T=95	60	70	X	X	X	64 68	D, N G, E	CMOS	TTL
16×16	ADSP-1016A	J=85 K=70	S=95 T=80	J=105 K=90	S=120 T=105	45	55	X	X	X	64 68	D, N G, E	CMOS	TTL
16×16	ADSP-7018	J=19	Note 5	J=25	Note 5	625	Note 5	X	X	X	108	G	Bipolar	TTL
16×16	ADSP-8018	J=15	Note 5	J=15.5	Note 5	825	Note 5	X	X	X	108	G	Bipolar	ECL
24×24	ADSP-1024A	J=120 K=95	S=150 T=120	N/A	N/A	70	90	X			84	G	CMOS	TTL[6]

NOTES
[1]ns max @ T_A = +70°C Comm., T_A = +125°C MIL.
[2]mA max, f_{CLK} = max, V_{DD} = +5V @ T_A = +70°C Comm., +125°C MIL.
[3]D = ceramic DIP, N = plastic DIP, E = leadless chip carrier, G = pin grid array.
[4]Contact factory.
[5]MIL versions available.
[6]TTL levels of 0.8V and +2.2V.

MULTIPLIER/ACCUMULATORS

Word Size	Model No.	Multiplication[1] Accumulate Time Comm	Multiplication[1] Accumulate Time MIL	Accum. Size	No. of Accum.	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 E, G
16×16	ADSP-1010A	J= 85 K=75	S=100 T=90	35	1	80	100	64 68	D, N E, G
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, E, P

NOTES
[1]ns max @ T_A = +70°C Comm., T_A = +125°C MIL.
[2]mA max, f_{CLK} = max, V_{DD} = +5V @ T_A = +70°C Comm., +125°C MIL.
[3]D = ceramic DIP, N = plastic DIP, E = leadless chip carrier, G = pin grid array, P = PLCC.

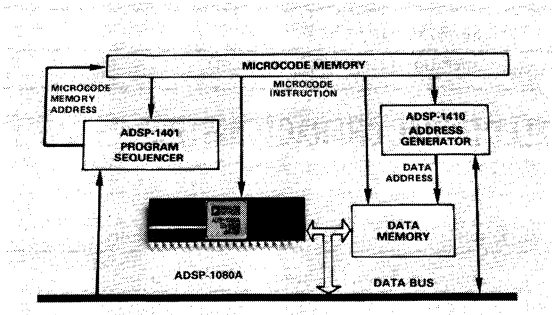
ANALOG DEVICES

8 × 8-Bit 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.

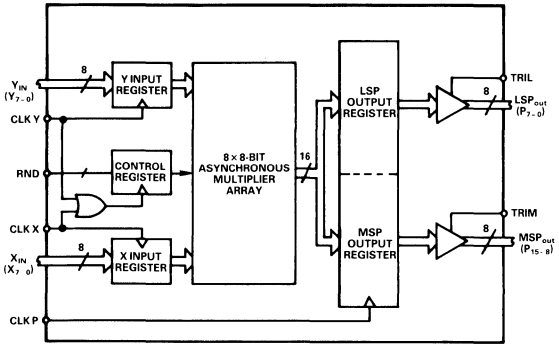
The ADSP-1080A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, and an 8-bit Least Significant Product (LSP) port. Input data is interpreted in twos-complement format. The ADSP-1080A produces a 16-bit result whose twos-complement MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are diode-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. Extended temperature range parts are available with high-reliability processing ("PLUS" parts) (see Figure 5). 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	@ min of (20MHz, max clock rate); TTL Inputs		45		55	mA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 0V; TRIM & TRIL = 5V		400		500	μA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		30		35	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1080A K Grades 0 to +70°C Min	K Grades 0 to +70°C Max	J Grades 0 to +70°C Min	J Grades 0 to +70°C Max	S Grades[2] −55°C to +125°C Min	S Grades[2] −55°C to +125°C Max	T Grades[2] −55°C to +125°C Min	T Grades[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		25		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. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 5.

[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 1.

Specifications subject to change without notice.

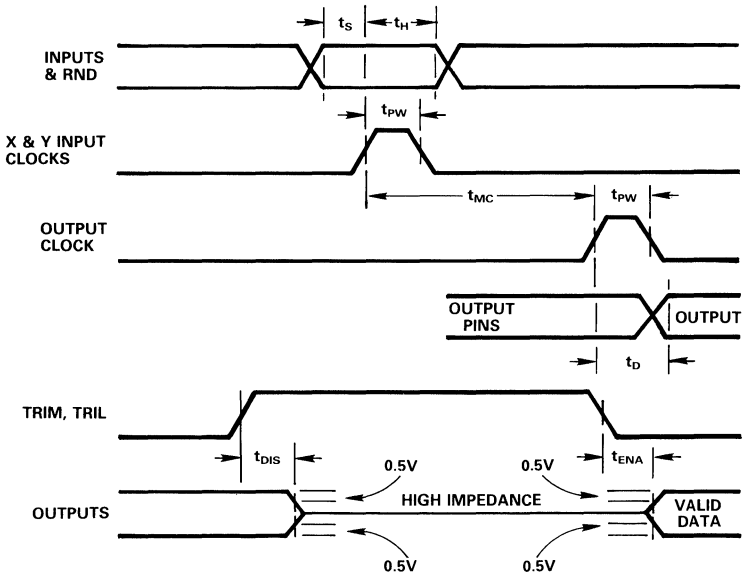


Figure 1. Timing Diagram

5

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 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 $+1/2$LSB 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

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

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.

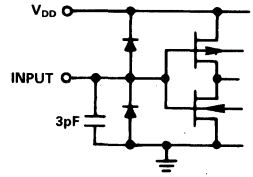


Figure 2. Equivalent Input Circuit

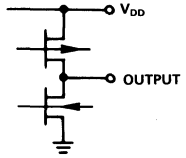


Figure 3. Equivalent Output Circuit

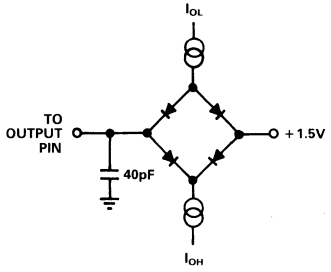


Figure 4. Normal Load for AC Measurements

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.

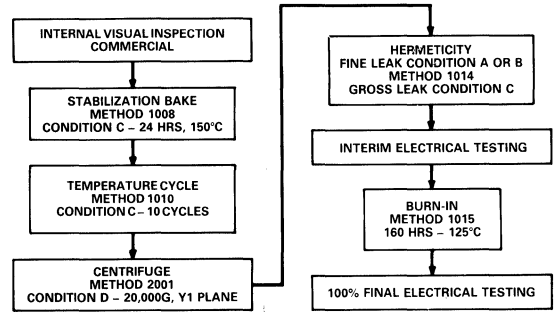


Figure 5. PLUS Processing Environmental Flow

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

CAUTION:

1. ESD sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts.
2. Do not insert this device into powered sockets. Remove power before insertion or removal.

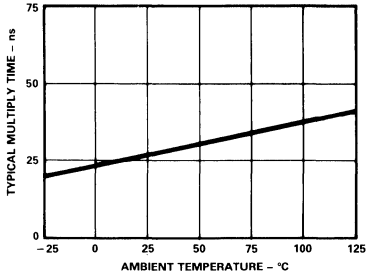


Figure 6. Approximate Clocked Multiply Time vs. Temperature

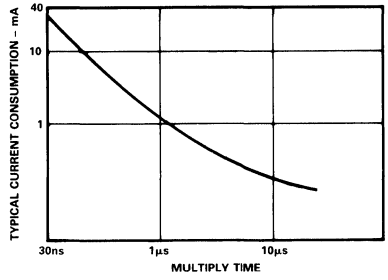


Figure 7. Typical I_{DD} vs. Frequency of Operation

ADSP-1080A PIN CONFIGURATION

DIP
D-40
N-40

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-40
ADSP-1080AKN	0 to + 70°C	40-Pin Plastic DIP	N-40
ADSP-1080AJD	0 to + 70°C	40-Pin Ceramic DIP	D-40
ADSP-1080AJN	0 to + 70°C	40-Pin Plastic DIP	N-40
ADSP-1080ATD/+	– 55°C to + 125°C	40-Pin Ceramic DIP	D-40
ADSP-1080ASD/+	– 55°C to + 125°C	40-Pin Ceramic DIP	D-40
ADSP-1080ATD/883B	– 55°C to + 125°C	40-Pin Ceramic DIP	D-40
ADSP-1080ASD/883B	– 55°C to + 125°C	40-Pin Ceramic DIP	D-40

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

ANALOG DEVICES

8 × 8-Bit Unsigned-Magnitude CMOS Multiplier

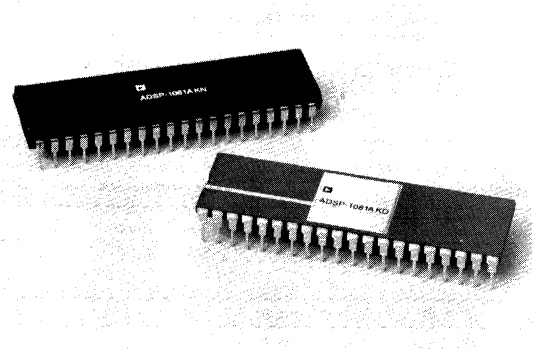
ADSP-1081A

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.

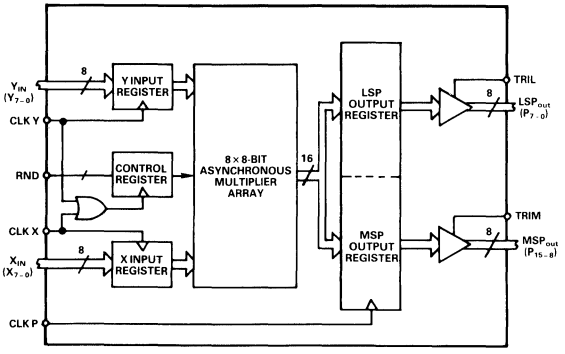
The ADSP-1081A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, and an 8-bit Least Significant Product (LSP) port. Input data is interpreted in unsigned-magnitude format. The ADSP-1081A produces a 16-bit result whose unsigned-magnitude MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are diode-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 5% 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. Extended temperature range parts are available with high-reliability processing ("PLUS" parts) (see Figure 5). 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.



ADSP-1081A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1081A 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-1081A 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_{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} = 0V; TRIM & TRIL = 5V		400		500	μA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		30		35	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1081A J Grades Min	J Grades Max	K Grades Min	K Grades Max	S Grades[2] Min	S Grades[2] Max	T Grades[2] Min	T Grades[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. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 5.

[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 1.

Specifications subject to change without notice.

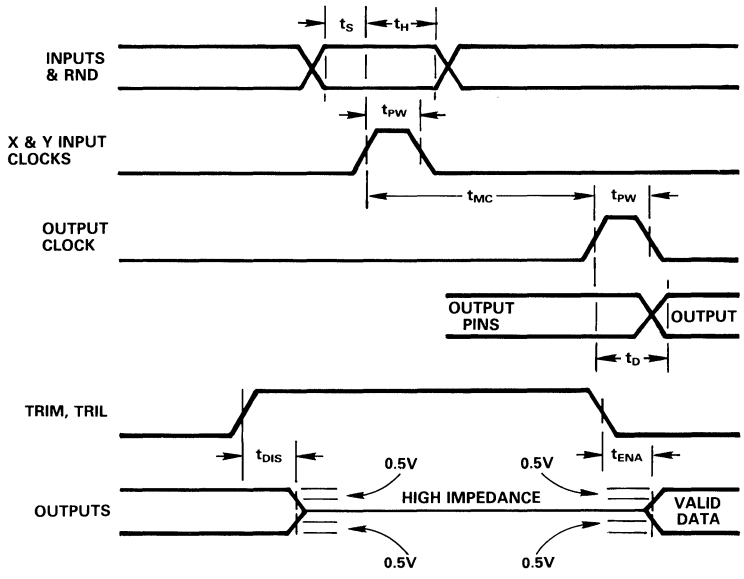


Figure 1. Timing Diagram

5

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 $-127/2$LSBs of the LSP, while rounding (RND HI) reduces the bias to only $+1/2$LSB 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.

X & Y INPUT DATA FORMAT								OUTPUT DATA FORMATS							
								MSP (P_{15-8})							
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
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}
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

OUTPUT DATA FORMATS															
LSP (P_{7-0})															
7	6	5	4	3	2	1	0								
2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}								
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0								

Table I. Data Formats for the ADSP-1081A

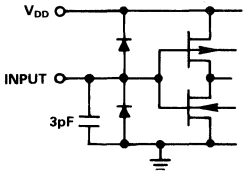


Figure 2. Equivalent Input Circuit

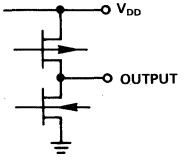


Figure 3. Equivalent Output Circuit

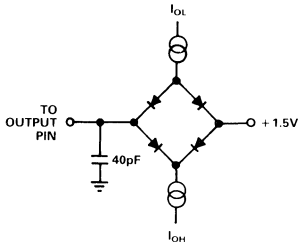


Figure 4. Normal Load for AC Measurements

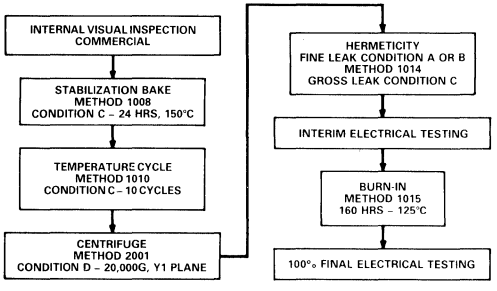


Figure 5. PLUS Processing Environmental Flow

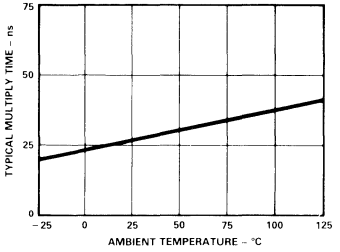


Figure 6. Approximate Clocked Multiply Time vs. Temperature

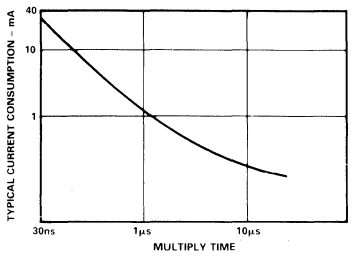


Figure 7. Typical I_{DD} vs. Frequency of Operation

ADSP-1081A PIN CONFIGURATION
DIP
D-40
N-40

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

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	Outline
ADSP-1081AKD	0 to +70°C	40-Pin Ceramic DIP	D-40
ADSP-1081AKN	0 to +70°C	40-Pin Plastic DIP	N-40
ADSP-1081AJD	0 to +70°C	40-Pin Ceramic DIP	D-40
ADSP-1081AJN	0 to +70°C	40-Pin Plastic DIP	N-40
ADSP-1081ATD/+	−55°C to +125°C	40-Pin Ceramic DIP	D-40
ADSP-1081ASD/+	−55°C to +125°C	40-Pin Ceramic DIP	D-40
ADSP-1081ATD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40
ADSP-1081ASD/883B	−55°C to +125°C	40-Pin Ceramic DIP	D-40

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

CAUTION:

1. ESD sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts.
2. Do not insert this device into powered sockets. Remove power before insertion or removal.

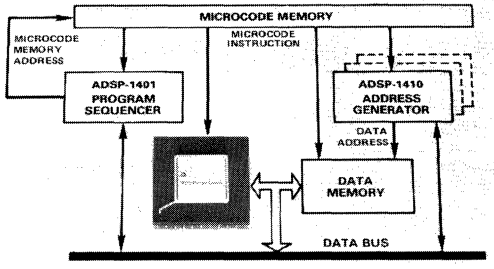
ANALOG DEVICES

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**



WORD-SLICE™ MICROCODED SYSTEM WITH ADSP-1012A

GENERAL DESCRIPTION

The ADSP-1012A is a high-speed, low-power 12 × 12-bit parallel multiplier fabricated in 1.5 micron CMOS.

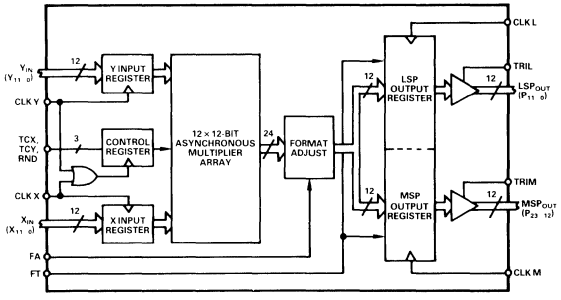
The ADSP-1012A has two 12-bit input ports, a 12-bit Most Significant Product (MSP) port, and a 12-bit Least Significant Product (LSP) port. Input data is interpreted in twos-complement, unsigned-magnitude, or mixed-mode formats. The ADSP-1012A produces a 24-bit result whose MSP can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP.

All input pins are diode-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).

The ADSP-1012A is available for both commercial and military temperature ranges. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts). (See Figure 7). 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} = −0.4mA	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 Grades 0 to +70° Min	J Grades 0 to +70° Max	K Grades 0 to +70° Min	K Grades 0 to +70° Max	S Grades[2] −55°C to +125°C Min	S Grades[2] −55°C to +125°C Max	T Grades[2] −55°C to +125°C Min	T Grades[2] −55°C to +125°C 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. Alternatively, S and T grade parts are available with high-temperature testing or with optional high-reliability "PLUS" processing as shown in Figure 7.

[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 1.

Specifications subject to change without notice.

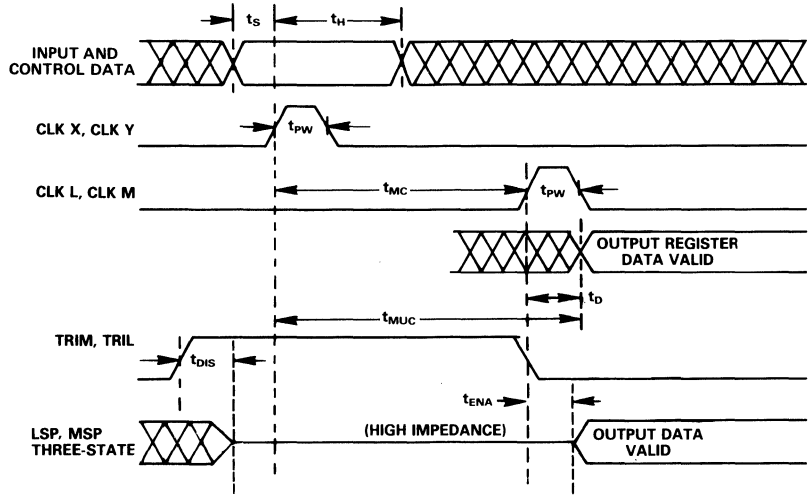


Figure 1. ADSP-1012A Timing Diagram

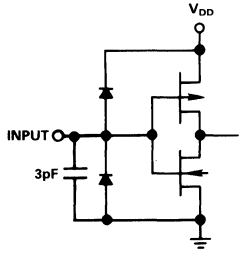


Figure 2. Equivalent Input Circuit

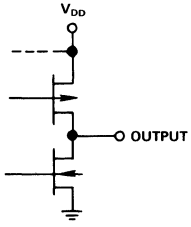


Figure 3. Equivalent Output Circuit

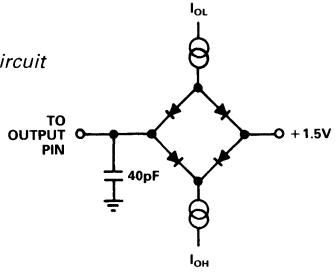


Figure 4. Normal Load for ac Measurements

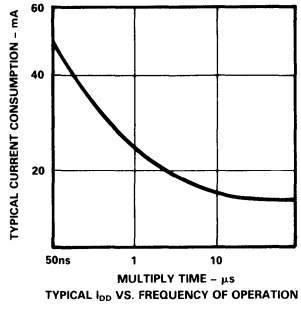


Figure 5. Typical I_{DD} vs. Frequency

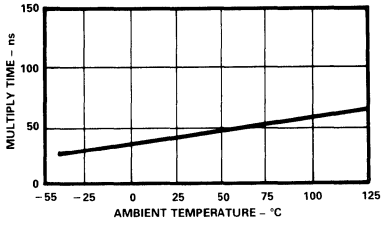


Figure 6. Approximate Worst Case Multiply Time vs. Temperature

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)		
sign -2^{11} | 2^{10} | ... | 2^0	UNSHIFTED (FA = 1)	
	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)		
sign -2^0 | 2^{-1} | ... | 2^{-11}	UNSHIFTED (FA = 1)	
	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)		
2^{11} | 2^{10} | ... | 2^0	UNSHIFTED (FA = 1)	
	2^{23} | 2^{22} | ... | 2^{12} | 2^{11} | 2^{10} | ... | 2^0	
UNSIGNED–MAGNITUDE FRACTIONAL (TCX, TCY = 0)		
2^{-1} | 2^{-2} | ... | 2^{-12}	UNSHIFTED (FA = 1)	
	2^{-1} | 2^{-2} | ... | 2^{-12} | 2^{-13} | 2^{-14} | ... | 2^{-24}	
MIXED–MODE INTEGER (TCX, TCY mixed)		
-2^{11} | 2^{10} | ... | 2^0 & 2^{11} | 2^{10} | ... | 2^0	UNSHIFTED (FA = 1)	
	sign -2^{23} | 2^{22} | ... | 2^{12} | 2^{11} | 2^{10} | ... | 2^0	
MIXED–MODE FRACTIONAL (TCX, TCY mixed)		
-2^0 | 2^{-1} | ... | 2^{-11} & 2^{-1} | 2^{-2} | ... | 2^{-12}	UNSHIFTED (FA = 1)	
	sign -2^0 | 2^{-1} | ... | 2^{-11} | 2^{-12} | 2^{-13} | ... | 2^{-23}	

Table I. ADSP 1012A Data Formats

	MSP							LSP						
	23	22	21	------	14	13	12	11	10	9	------	2	1	0
FA = 1	sign -2^1	2^0	2^{-1}	------	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	------	2^{-20}	2^{-21}	2^{-22}
DISCARD														
FA = 0	sign -2^0	2^{-1}	2^{-2}	------	2^{-9}	2^{-10}	2^{-11}	sign (-2^0)	2^{-12}	2^{-13}	------	2^{-20}	2^{-21}	2^{-22}

Table II. ADSP-1012A Format Adjust

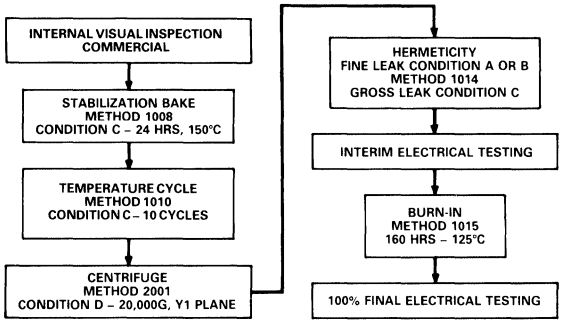


Figure 7. PLUS Processing Environmental Flow

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-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/+	−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/+	−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/+	−55°C to +125°C	68-Contact LCC	E-68A
ADSP-1012ATE/+	−55°C to +125°C	68-Contact LCC	E-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.

ESD SENSITIVITY

The ADSP-1012A features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1012A 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.

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

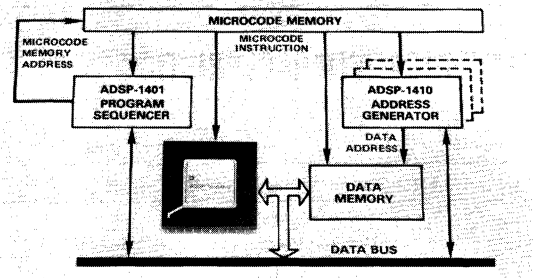
The ADSP-1016A is a high-speed low-power 16 × 16-bit parallel multiplier fabricated in 1.5 micron CMOS.

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 diode-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.

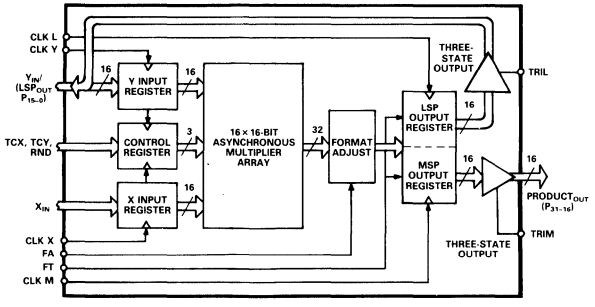
The ADSP-1016A is a pin-for-pin replacement for Analog Devices' ADSP-1016 and is also pin-for-pin compatible in a DIP package with TRW's MPY016HJ1. The ADSP-1016A's multiply time is more than twice as fast as the TRW device.

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. Extended temperature range parts are available with optional high-reliability processing ("PLUS") parts (see Figure 6). 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		45		55	mA
I_{DD}	Supply Current–Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1016A J Grades 0 to +70° Min	J Grades Max	K Grades Min	K Grades Max	S Grades[2] −55°C to +125°C Min	S Grades[2] Max	T Grades[2] Min	T Grades[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. Alternatively, S and T grade parts are available with high-temperature testing or with optional high-reliability "PLUS" processing as shown in Figure 6.

[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 1.

Specifications subject to change without notice.

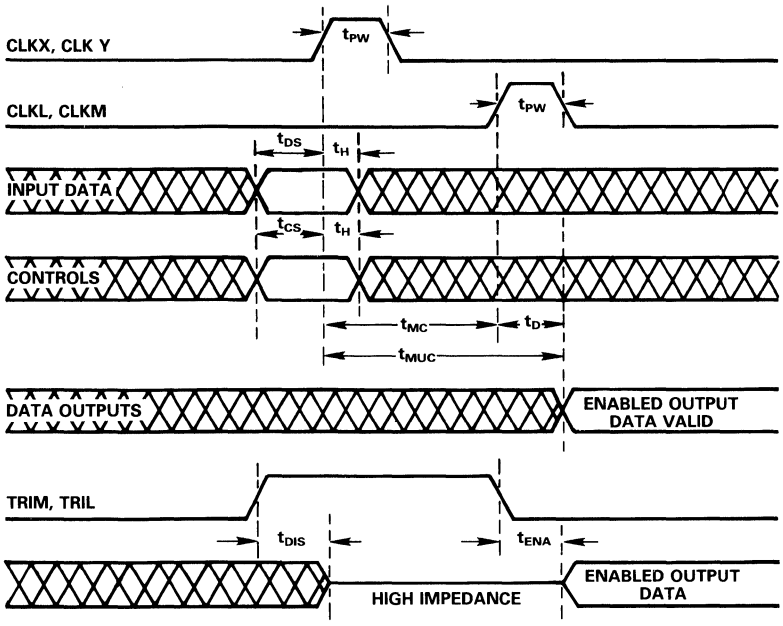


Figure 1. ADSP-1016A Timing Diagram

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 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 TXC 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 an 16-bit MSP and an 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 $+1/2$LSBs of the LSP.

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.

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.

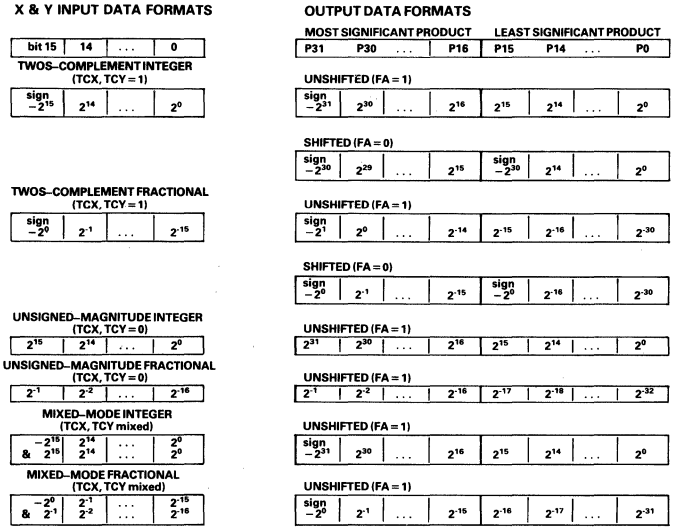


Table I. ADSP-1016A Data Formats

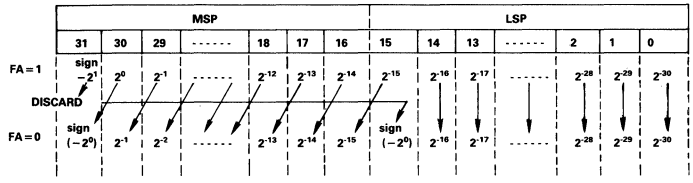
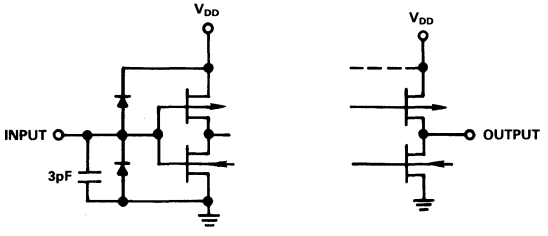


Table II. Format Adjust



a. Equivalent Input Circuit *b. Equivalent Output Circuit*

Figure 2.

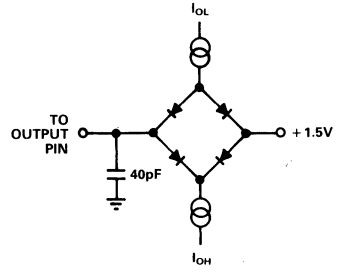


Figure 3. Normal Load for ac Measurements

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

ESD SENSITIVITY

The ADSP-1016A features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1016A 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.

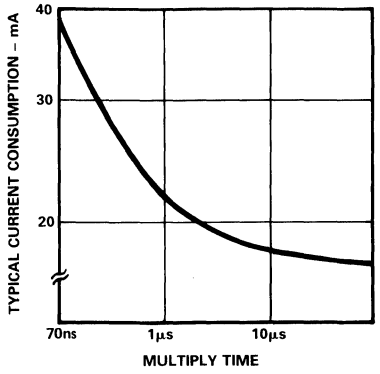


Figure 4. Typical Power Dissipation vs. Frequency

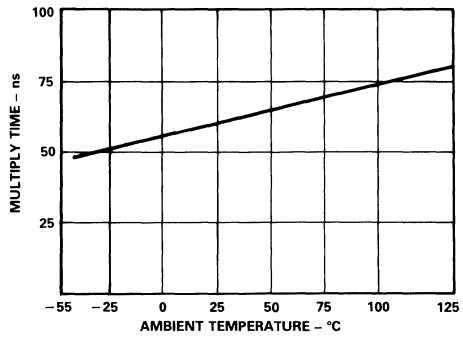


Figure 5. Approx. Multiply Time vs. Temperature

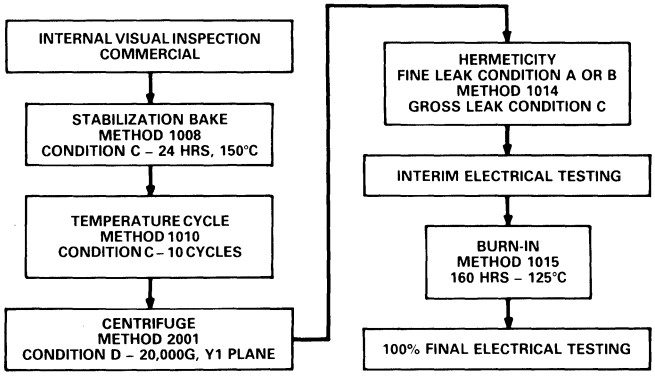


Figure 6. PLUS Processing Environmental Flow

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/+	−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/+	−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.

ADSP-1016A PIN CONFIGURATIONS

DIP
D-64A
N-64A

PIN	FUNCTION	PIN	FUNCTION
1	X4	33	P24
2	X3	34	P25
3	X2	35	P26
4	X1	36	P27
5	X0	37	P28
6	TRIL	38	P29
7	CLK L	39	P30
8	CLK Y	40	P31
9	P0, Y0	41	CLK M
10	P1, Y1	42	TRIM
11	P2, Y2	43	FA
12	P3, Y3	44	FT
13	P4, Y4	45	GND
14	P5, Y5	46	GND
15	P6, Y6	47	GND
16	P7, Y7	48	V_{DD}
17	P8, Y8	49	V_{DD}
18	P9, Y9	50	TCY
19	P10, Y10	51	TCX
20	P11, Y11	52	RND
21	P12, Y12	53	CLK X
22	P13, Y13	54	X15
23	P14, Y14	55	X14
24	P15, Y15	56	X13
25	P16	57	X12
26	P17	58	X11
27	P18	59	X10
28	P19	60	X9
29	P20	61	X8
30	P21	62	X7
31	P22	63	X6
32	P23	64	X5

PIN GRID ARRAY
G-68A

PIN	FUNCTION	PIN	FUNCTION
1	P0, Y0	35	CLK M
2	P1, Y1	36	TRIM
3	P2, Y2	37	FA
4	P3, Y3	38	FT
5	P4, Y4	39	GND
6	P5, Y5	40	GND
7	P6, Y6	41	GND
8	P7, Y7	42	V_{DD}
9	P8, Y8	43	V_{DD}
10	P9, Y9	44	TCY
11	P10, Y10	45	TCX
12	P11, Y11	46	RND
13	P12, Y12	47	CLK X
14	P13, Y13	48	X15
15	P14, Y14	49	X14
16	P15, Y15	50	X13
17	N/C	51	N/C
18	P16	52	X12
19	P17	53	X11
20	P18	54	X10
21	P19	55	X9
22	P20	56	X8
23	P21	57	X7
24	P22	58	X6
25	P23	59	X5
26	P24	60	X4
27	P25	61	X3
28	P26	62	X2
29	P27	63	X1
30	P28	64	X0
31	P29	65	TRIL
32	P30	66	CLK L
33	P31	67	CLK Y
34	N/C	68	N/C

LCC
E-68A

PIN	FUNCTION	PIN	FUNCTION
1	X4	35	P24
2	X3	36	P25
3	X2	37	P26
4	X1	38	P27
5	X0	39	P28
6	TRIL	40	P29
7	CLK L	41	P30
8	CLK Y	42	P31
9	N/C	43	N/C
10	P0, Y0	44	CLK M
11	P1, Y1	45	TRIM
12	P2, Y2	46	FA
13	P3, Y3	P47	FT
14	P4, Y4	48	GND
15	P5, Y5	49	GND
16	P6, Y6	50	GND
17	P7, Y7	51	V_{DD}
18	P8, Y8	52	V_{DD}
19	P9, Y9	53	TCY
20	P10, Y10	54	TCX
21	P11, Y11	55	RND
22	P12, Y12	56	CLK X
23	P13, Y13	57	X15
24	P14, Y14	58	X14
25	P15, Y15	59	X13
26	N/C	60	N/C
27	P16	61	X12
28	P17	62	X11
29	P18	63	X10
30	P19	64	X9
31	P20	65	X8
32	P21	66	X7
33	P22	67	X6
34	P23	68	X5

ANALOG DEVICES

8 × 8-Bit CMOS Multiplier/Accumulator

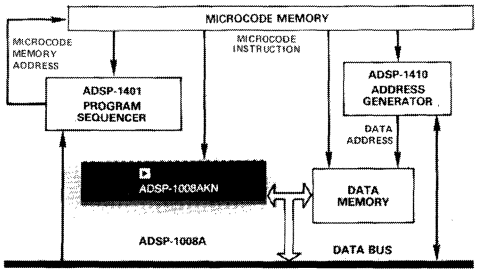
ADSP-1008A

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.

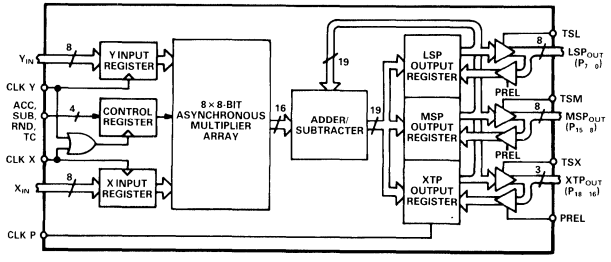
The ADSP-1008A has two 8-bit input ports, an 8-bit Most Significant Product (MSP) port, an 8-bit Least Significant Product (LSP) port and a 3-bit Extended Product (XTP) port. Inputs can be represented in either twos-complement or unsigned-magnitude formats. The ADSP-1008A produces a 16-bit product whose 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 diode-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. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts) (see Figure 6.) 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.



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 Grades 0 to +70°C Min	J Grades Max	K Grades 0 to +70°C Min	K Grades Max	S Grades −55°C to +125°C Min	S Grades Max	T Grades −55°C to +125°C Min	T Grades 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. Alternatively, S and T grade parts are available with optional high-reliability "PLUS" processing as shown in Figure 6.

[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 1.

Specifications subject to change without notice.

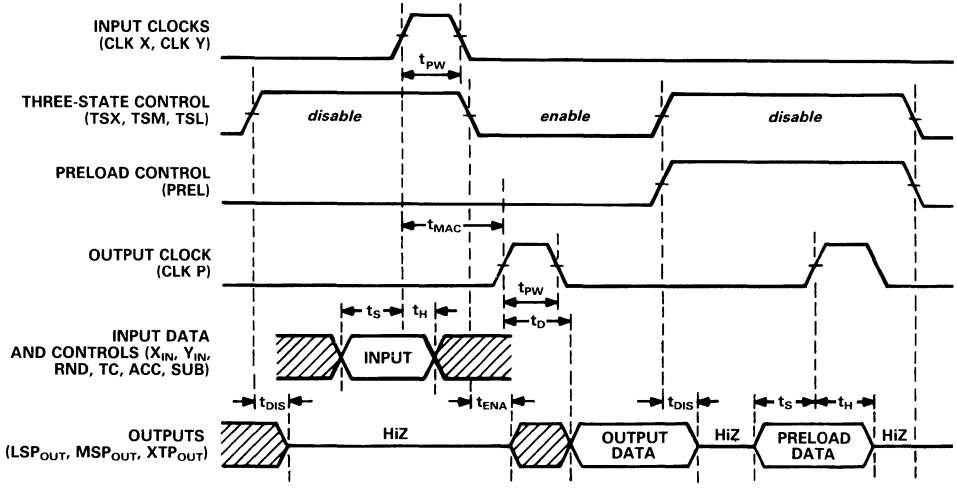


Figure 1. ADSP-1008A Timing Diagram

METHOD OF OPERATION

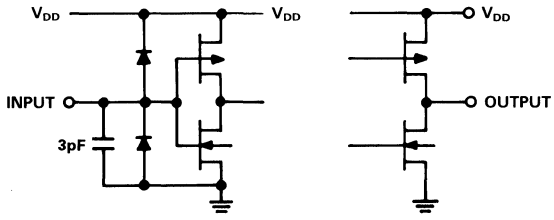
The X and Y input registers are positive-edge triggered D-type flip-flops. Input data is loaded to the X and Y registers with the rising edges of CLK X and CLK Y, respectively. The X and Y input data can be represented in either twos-complement or unsigned-magnitude formats. (Mixed-mode is not supported.)

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-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.



a. Equivalent Input Circuit *b. Equivalent Output Circuit*

Figure 2.

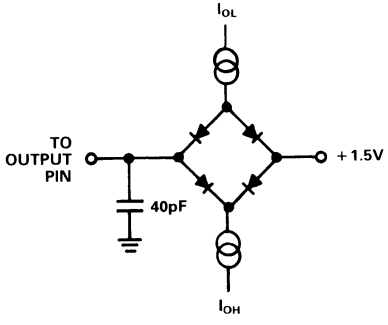


Figure 3. Normal Load for AC Measurements

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.

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	1	$\text{Accumulator}_t = X_t \cdot Y_t + \text{Accumulator}_{t-1}$
0	X	$\text{Accumulator}_t = X_t \cdot Y_t$

Table II. Function Truth Table

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

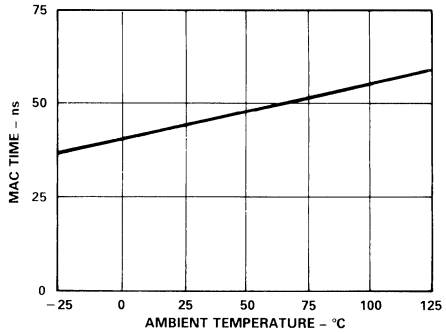


Figure 4. Typical MAC Time vs. Temperature

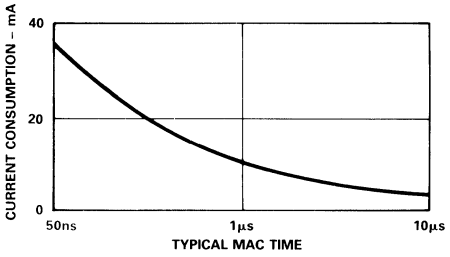


Figure 5. Typical I_{DD} vs. Frequency of Operation

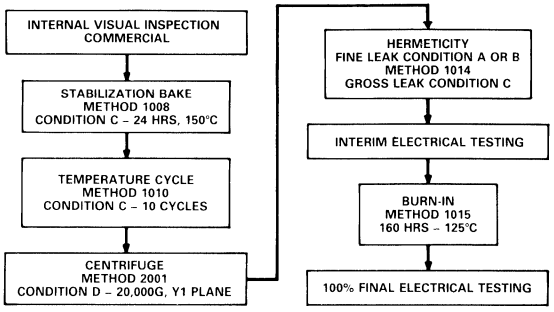


Figure 6. "PLUS" Processing

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/+	−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.

ADSP-1008A PIN CONFIGURATION

DIP
D-48A
N-48A

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	CLK X
7	CLK P	31	CLK Y
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

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

ESD SENSITIVITY

The ADSP-1008A features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1008A 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' ESP Prevention Manual.

ANALOG DEVICES

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**

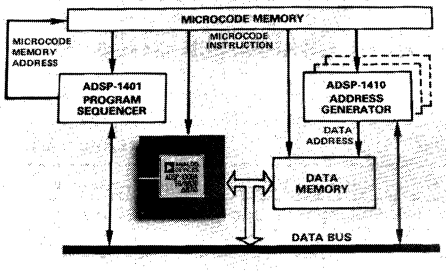
GENERAL DESCRIPTION

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 diode-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.

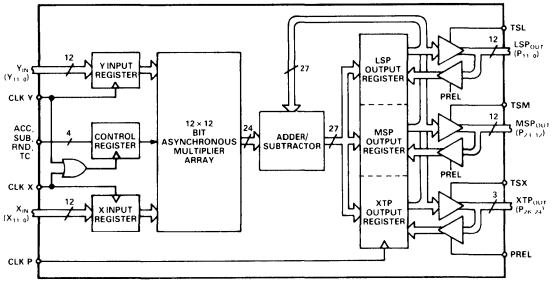
The ADSP-1009A is a pin-for-pin replacement for Analog Devices' ADSP-1009 and is also pin-for-pin compatible in a DIP package with TRW's TDC1009J1 and TMC2009J3. The ADSP-1009A's multiply/accumulate time is over twice as fast as either TRW device.



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. Extended temperature range parts are available with high-reliability processing ("PLUS" parts). (See Figure 7.) 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} = 0V; TSL, TSM, TSX = 5V		400		500	µA
I_{DD}	Supply Current-Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS[3]

Parameter		ADSP-1009A 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[2] −55°C to +125°C Min	S Grades[2] −55°C to +125°C Max	T Grades[2] −55°C to +125°C Min	T Grades[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. Alternatively, S and T grade parts are available with high-reliability "PLUS" processing as shown in Figure 7.

[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 1.

Specifications subject to change without notice.

CAUTION:

1. ESD sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts.
2. Do not insert this device into powered sockets. Remove power before insertion or removal.

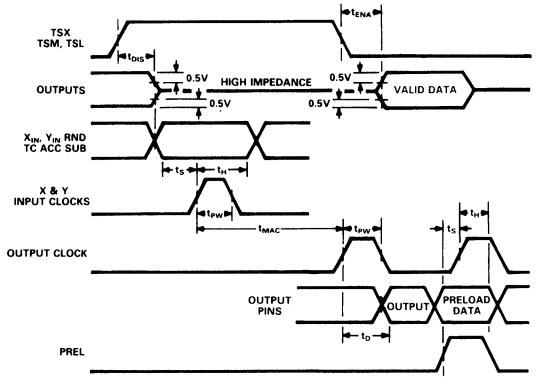


Figure 1. ADSP-1009A Timing Diagram

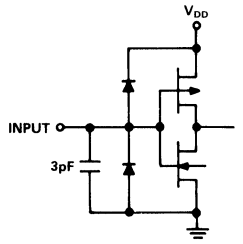


Figure 2. Equivalent Input Circuits

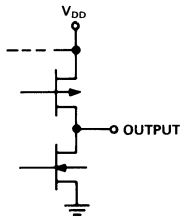


Figure 3. Equivalent Output Circuits

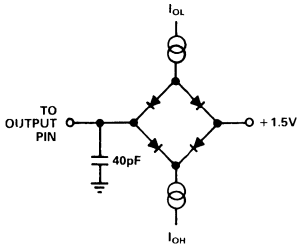


Figure 4. Normal Load for AC Measurements

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 $+1/2$LSBs 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.)

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

X & Y INPUT DATA FORMATS							OUTPUT DATA FORMATS																		
							XTP			MSP						LSP									
11	10	9	---	2	1	0	26	25	24	23	22	21	---	14	13	12	11	10	9	---	2	1	0
INTEGER TWOS COMPLEMENT (TC = 1)																									
sign							sign																		
(-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}
FRACTIONAL TWOS COMPLEMENT (TC = 1)																									
sign							sign																		
(-2^{0})	2^{-1}	2^{-2}	---	2^{-9}	2^{-10}	2^{-11}	-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 1. Data Formats

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.

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

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

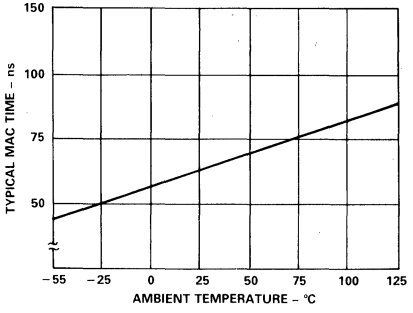


Figure 5. Approx. Multiply Time vs. Temperature

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.)

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

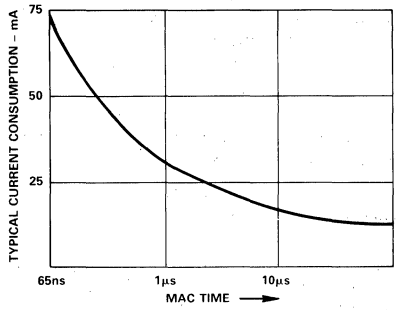


Figure 6. Typical I_{DD} vs. Frequency of Operation

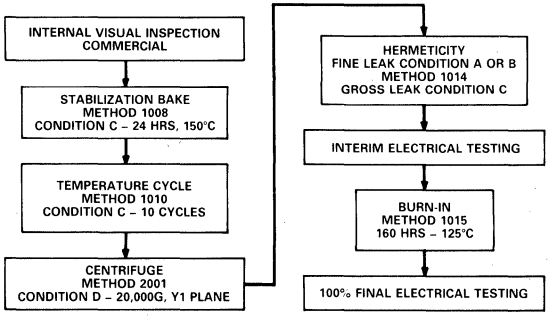


Figure 7. PLUS Processing Environmental Flow

ADSP-1009A 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	CLK P	49	Y0
16	PREL	50	CLK Y
17	N/C	51	N/C
18	TSM	52	CLK X
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.

16 × 16-Bit CMOS Multiplier/Accumulator

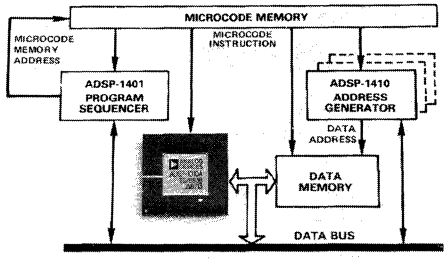
ADSP-1010A

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.

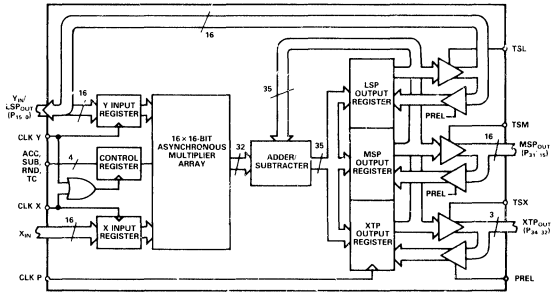
The ADSP-1010A has two 16-bit input ports, a 16-bit Most Significant Product (MSP) port, a 16-bit Least Significant Product (LSP) port, and a 3-bit Extended Product (XTP) port. The LSP output port is a bidirectional port shared with the Y input port. Inputs can be represented in either twos-complement 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 diode-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. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts) (see Figure 7.). 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 Min	ADSP-1010A S and T Grades 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 Grades Min	J Grades Max	K Grades Min	K Grades Max	−55°C to +125°C S Grades Min	S Grades Max	T Grades Min	T Grades 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. Alternatively, S and T grade parts are available with high-temperature testing or with optional high-reliability "PLUS" processing as shown in Figure 7.

[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 1.

Specifications subject to change without notice.

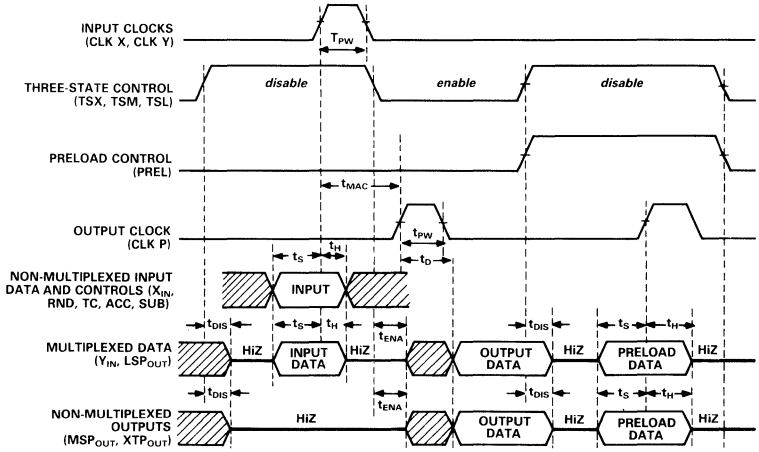


Figure 1. ADSP-1010A Timing Diagram

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.)

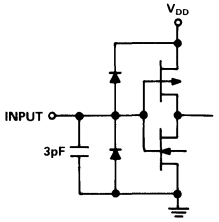


Figure 2. Equivalent Input Circuits

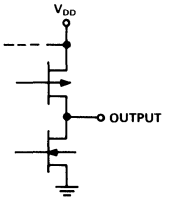


Figure 3. Equivalent Output Circuits

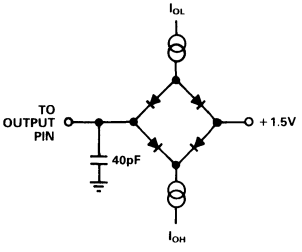


Figure 4. Normal Load for AC Measurements

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.)

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 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.

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

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

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.)

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 input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1010A 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.

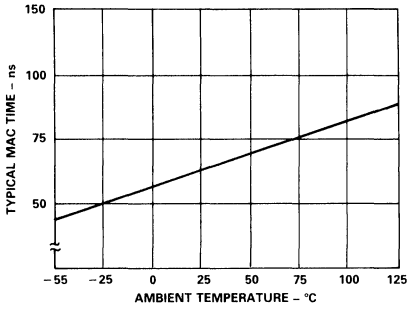


Figure 5. Approx. Multiply Time vs. Temperature

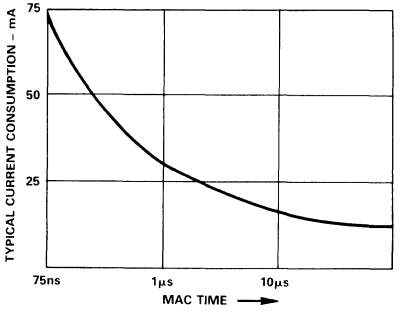


Figure 6. Typical I_{DD} vs. Frequency of Operation

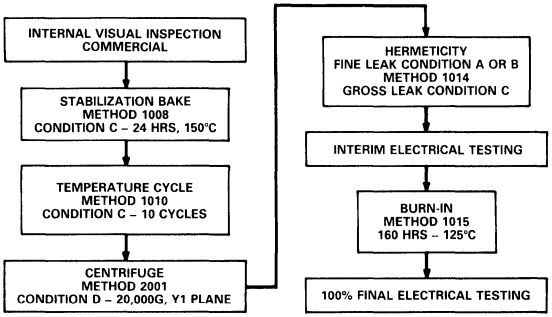


Figure 7. PLUS Processing Environmental Flow

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 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ATD	–55 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ASD/+	–55 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ATD/+	–55 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ASD/883B	–55 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ATD/883B	–55 to +125°C	64-Pin Ceramic DIP	D-64A
ADSP-1010ASG	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ATG	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ASG/+	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ATG/+	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ASG/883B	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ATG/883B	–55 to +125°C	68-Pin Grid Array	G-68A
ADSP-1010ASE/+	–55 to +125°C	68-Contact LCC	E-68A
ADSP-1010ATE/+	–55 to +125°C	68-Contact LCC	E-68A
ADSP-1010ASE/883B	–55 to +125°C	68-Contact LCC	E-68A
ADSP-1010ATE/883B	–55 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

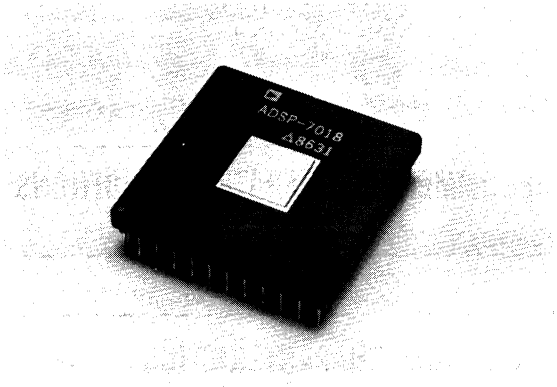
ANALOG DEVICES

TTL 16 × 16-Bit Multiplier

ADSP-7018

FEATURES
16 × 16-bit Parallel Multiplication
19ns Max Multiplication Time
TTL Compatible
4.0W Max Power Dissipation
Independent Input and Output Latches Which Can Be Made Transparent
Unclocked, Single-, Double-, and Triple-Clock Operation
Twos Complement, Unsigned Magnitude, and Mixed Mode
Format Adjust and Rounding
Parallel Data Load and Passthrough from Input Port
Full 32-Bit Product Output Port
16-Bit Output Multiplexer
Independent Three-State Control for LSP and MSP
Sign, Overflow, and Zero Status Flags
Pin-Compatible with B2018
1.20″ Square 108-Pin Grid Array

APPLICATIONS
Digital Signal Processing
Array Processors
Super-Minicomputers
General-Purpose Computing



GENERAL DESCRIPTION

The ADSP-7018 is a high-speed 16 × 16-bit parallel TTL multiplier, fabricated in TTL-compatible bipolar ECL. The ADSP-7018 has two 16-bit input ports and a fielded 32-bit output port. 32-bit products are available in parallel, as independently enabled 16-bit Least Significant Product (LSP) and 16-bit Most Significant Product (MSP) portions, or LSP and MSP multiplexed to a single 16-bit port. Each 16-bit field of the product latch has its own three-state output control.

Inputs can be represented in either twos-complement, unsigned-magnitude, or mixed-mode formats. The input, instruction, status, and output latches are all latches which can be enabled to be transparent. Input operands and output results are stored in individually enabled latches with separate clocks for input and output. For unclocked flow-through operation, the latches can be made transparent by holding clocks and enables active. Tying clocks together as a single clock causes input and output latches to operate in complementary fashion, simplifying synchronous operation. The two clocks are fully independent, allowing double clock operation. With both clock lines held active, the three enable controls can function as three independent clock lines for the two data input ports and the output port, respectively.

The ADSP-7018 produces a 32-bit product that may be format-adjusted for consistent signed fractional output format and for maximum precision in a 16-bit MSP. The MSP of the format-adjusted 32-bit product can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP. Three status flags indicate the presence of a zero, negative, or overflowed result. The input operands can be loaded and passed directly through without multiplication, but with format adjustment, rounding, and setting of status flags.

The ADSP-7018 is a pin-for-pin replacement for Bipolar Integrated Technology's B2018 TTL 16 × 16 Multiplier.

The ADSP-7018 is available for commercial temperature ranges in a compact, hermetically-sealed 108-pin grid array.

5

SPECIFICATIONS[1]

ABSOLUTE MAXIMUM RATINGS

Parameter		ADSP-7018 Min	ADSP-7018 Max	Unit
V_{CC}	Supply Voltage (GND = 0)	−0.5	+7.0	V
V_{IN}	Input Voltage (GND = 0)	−0.5	V_{CC} + 0.5	V
I_O	Output Source Current			
	Continuous		30	mA
	Surge		100	mA
T_{ST}	Storage Temperature (ambient)	−55	+150	°C
T_J	Operating Junction Temperature		+165	°C

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-7018 Min	ADSP-7018 Max	Unit
V_{CC}	Supply Voltage	4.75	5.25	V
T_{AMB}	Operating Temperature (ambient)[3]	0	+70	°C

ELECTRICAL CHARACTERISTICS

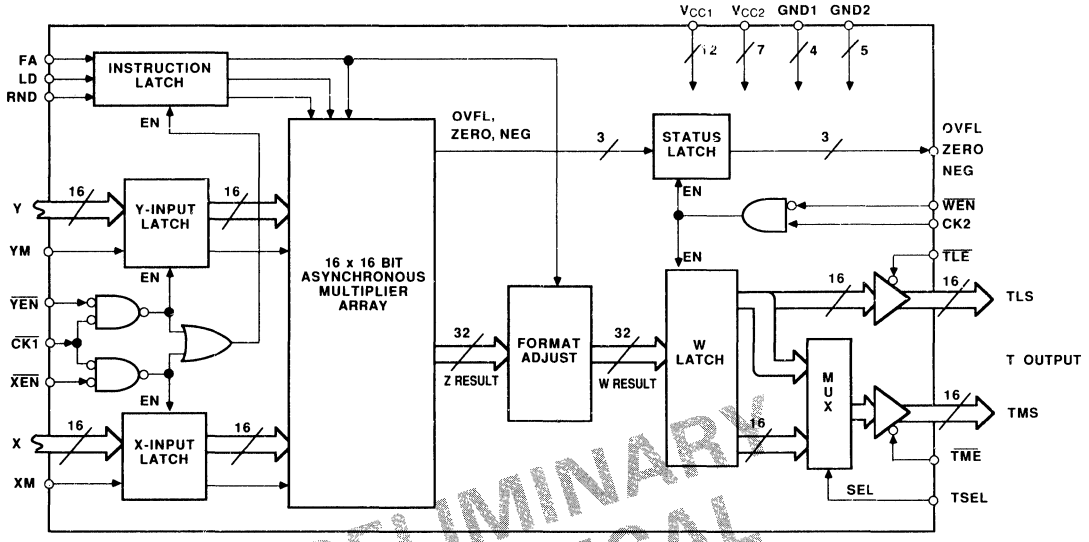
Parameter		Test Conditions	ADSP-7018 Min	ADSP-7018 Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{CC} = max	2.0		V
V_{IL}	Low-Level Input Voltage	@ V_{CC} = min		0.8	V
V_{OH}	High-Level Output Voltage	@ V_{CC} = min & I_{OH} = max	2.4		V
V_{OL}	Low-Level Output Voltage	@ V_{CC} = min & I_{OL} = max		0.4	V
I_{IH}	High-Level Input Current, All Inputs	@ V_{CC} = max & V_{IN} = 2.4V		200	μA
I_{IL}	Low-Level Input Current, All Inputs	@ V_{CC} = max & V_{IN} = 0.4V		200	μA
I_{OH}	High-Level Output Current			−400	μA
I_{OL}	Low-Level Output Current			4.0	mA
I_{OZ}	Three-State Leakage Current	@ V_{CC} = max; High Z; V_{IN} = 0 or V_{CC} = max		40	μA
I_{CC}	Supply Current	@ V_{CC} = max; max clock rate; TTL inputs		800	mA

SWITCHING CHARACTERISTICS[2]

Parameter		ADSP-7018J Min	ADSP-7018J Max	Unit
t_S	Input Setup Time	4		ns
t_H	Input Hold Time	3		ns
t_C	Clock and Latch Enable Pulse Duration	5.5		ns
t_{DCM}	Data to Clock Multiply Time		14	ns
t_{CCH}	Clock to Clock Hold Time	−1.5		ns
t_{CCM}	Clock to Clock Multiply Time		19	ns
t_{COD}	Clocked Output Delay		12	ns
t_{DDM}	Data to Data Multiply Time		25	ns
t_{CDM}	Clock to Data Multiply Time		26	ns
t_{PHZ}	Output Disable Delay High to HI-Z		7	ns
t_{PLZ}	Output Disable Delay Low to HI-Z		7	ns
t_{PZH}	Output Enable Delay HI-Z to High		12	ns
t_{PZL}	Output Enable Delay HI-Z to Low		11	ns

NOTES
[1]All min and max specifications are over power-supply and temperature range indicated.
[2]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_{ENO} and t_{DAO} which are as indicated in Figure 1.
[3]500 linear feet per minute ambient air flow.

Specifications subject to change without notice.



Functional Block Diagram

PIN DESCRIPTION

DATA	
X_{15-0}	X Operand Input
Y_{15-0}	Y Operand Input
TMS_{15-0}	MSP of 32-Bit T Output
TLS_{15-0}	LSP of 32-Bit T Output
CONTROL	
$\overline{XEN}$	Enable X-Input Latch
$\overline{YEN}$	Enable Y-Input Latch
$\overline{WEN}$	Enable W Latch
XM	X-Input Data Format
YM	Y-Input Data Format
FA	Format Adjust
LD	Load Concatentated X and Y Operands
RND	Round MSP
$\overline{TME}$	Enable TMS Output (Active LO)
$\overline{TLE}$	Enable TLS Output (Active LO)
TSEL	Select MSP or LSP to TMS Port
STATUS	
ZERO	Zero
OVFL	Overflow
NEG	Negative
CLOCKS	
$\overline{CK1}$	Input Latches Clock
CK2	Output Latches Clock
POWER	
V_{CC1}	Positive Supply Voltage to Internal Logic
V_{CC2}	Positive Supply Voltage to Output Circuits
GND1	Negative Supply Voltage to Internal Logic
GND2	Negative Supply Voltage to Output Circuits

METHOD OF OPERATION

The X-Input Latch holds its most recent contents unless $\overline{CLK1}$ is LO (logic 0) when enabled (made transparent) by $\overline{XEN}$ also LO. Similarly, the Y-Input Latch holds its most recent contents unless $\overline{CLK1}$ is LO when enabled by $\overline{YEN}$ also LO. The state of the $\overline{CK1}$ input clock has no effect on each input latch unless the input enable lines $\overline{XEN}$ and $\overline{YEN}$ are LO, respectively. When these enable lines are LO, the respective latches will pass data when the $\overline{CK1}$ input is LO and hold data when the $\overline{CK1}$ input is HI (logic 1).

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 levels on the XM line for X input data and HI levels on the YM line for Y input data. Unsigned-magnitude X and Y inputs are indicated by LO levels on the XM and YM 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. XM and YM are latched into the X-Input and Y-Input Latches with the input operand data.

The Instruction Latch holds its most recent contents unless $\overline{CLK1}$ is LO when enabled by *either* $\overline{XEN}$ *or* $\overline{YEN}$ also LO. When $\overline{CK1}$ is HI, the contents of the Instruction Latch will be held. If both $\overline{XEN}$ and $\overline{YEN}$ are HI, the contents of the Instruction Latch will also be held. Round (RND), format adjust (FA), and load (LD) are the three instruction bits that pass through or are held at the Instruction Latch.

The ADSP-7018's W output is fielded into a 16-bit most significant product (MSP) and a 16-bit least significant product (LSP). When the RND is HI, the MSP of the W result 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 of $-(2^{16}-1)/2$ LSBs of the LSP, while rounding (RND HI) reduces the bias to $+1/2$ LSBs of the LSP.

The FA control format adjusts the Z output from the multiplier array. When FA is HI, the full 32-bit Z result is passed unmodified to the W latch (Tables II and III). All possible products can be represented without overflow, including twos-complement fractional $(-1.0) \times (-1.0)$. The binary point in the MSP of a twos-complement fractional product will not align with the binary point in the input operands. Thus, the numeric weighting of data within a system using this format (with FA HI) will not be uniform. Also, the two highest-order bits in a twos-complement product are normally redundant. An additional bit of precision in the MSP could be obtained by eliminating one of these redundant bits.

Format adjust (FA LO) shifts the Z result left by one bit position and right-fills a binary zero (Table III). Thus, format adjust eliminates redundant high-order bits in twos-complement products. The MSPs of fractional twos-complement products also receive proper weighting relative to input data. 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). This special condition is monitored by the OVFL flag so that it does not go undetected.

The Load (LD) instruction determines whether the multiplier array is active or put into a pass-through mode. When LD is LO, the multiplier array is active and produces the product of the X and Y input values as Z result. When LD is HI, the multiplier array is put in the pass-through mode and the X and Y input values are concatenated to form a Z result with the X input in the most significant position (Table II). XM determines the sign mode of the complete 32-bit Z result. ZERO, OVFL and NEG are set depending on the value of this Z result. RND and FA can also be used in this mode. Load allows any 32-bit number loaded through the input ports to be rounded, tested for zero and sign, and to have fractional binary points aligned while testing for overflow. It is also valuable for system testing by providing a direct path through the multiplier and greater access to latches within the ADSP-7018.

Output flags ZERO, OVFL, and NEG are generated from the intermediate Z result. When not rounding (RND LO), ZERO is set when the full 32-bit result is zero. When rounding (RND HI), ZERO is set when the format-adjusted MSP in the W result will be zero. That is, when rounding, ZERO is set for $Z_{31\text{-}16}$ equals zero when FA HI but ZERO is set for $Z_{31\text{-}15}$ equals zero when FA LO. Zero will not be set if an overflow condition exists.

OVFL is set only for twos-complement multiplications of full-scale negative times full-scale negative when format adjusting (FA LO).

NEG is set only when a twos-complement or mixed-mode Z result is negative. A loaded, previously overflowed result (MSB of 1 followed by all zeros) will set NEG as well as OVFL, if loaded with XM HI.

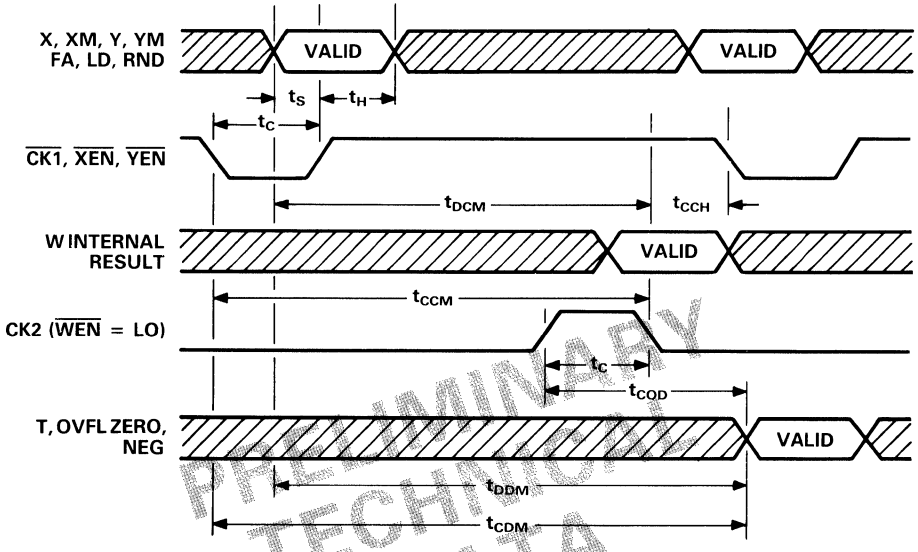
The output latches for data and flags ZERO, OVFL and NEG hold their most recent contents unless CLK2 is HI when enabled by $\overline{WEN}$ LO. With $\overline{WEN}$ LO, the output latches are transparent when CK2 is HI and data is held when CK2 is LO. The ADSP-7018 can be made a totally asynchronous device by holding all enables active ($\overline{XEN}$, $\overline{YEN}$, and $\overline{WEN}$ all LO) and $\overline{CLK1}$ LO and CLK2 HI. Note that the clock definitions are complementary; if $\overline{CLK1}$ and CLK2 are tied together and enable lines asserted, input latches will be transparent when output latches are latched and vice versa. $\overline{CLK1}$ and CLK2 are fully independent, allowing double-clock operation. For triple-clock operation, the two clock lines can be held active which causes the three latch enable controls to behave like independent clock lines.

The T output is fielded into two 16-bit portions, TMS and TLS, each with its own three-state controls. A HI on the asynchronous $\overline{TME}$ or $\overline{TLE}$ lines disables the corresponding TMS or TLS output driver to a high-impedance state. Conversely, a LO on $\overline{TME}$ or $\overline{TLE}$ enables the corresponding output driver, driving the output bus.

TSEL controls the multiplexer feeding the TMS field (Table IV). When TSEL is LO, the MSP from the W Latch will be passed to the TMS output. When TSEL is HI, the LSP will be routed to TMS. This multiplexer and its associated TSEL control simplifies operation of the ADSP-7018 in 16-bit bus systems.

TTL 16 × 16-BIT FIXED-POINT MULTIPLIER

CLOCKED AND UNCLOCKED TIMING



NOTE: FOR FLOWTHROUGH OPERATION, THE LATCHES MUST BE MADE TRANSPARENT.

OUTPUT ENABLE

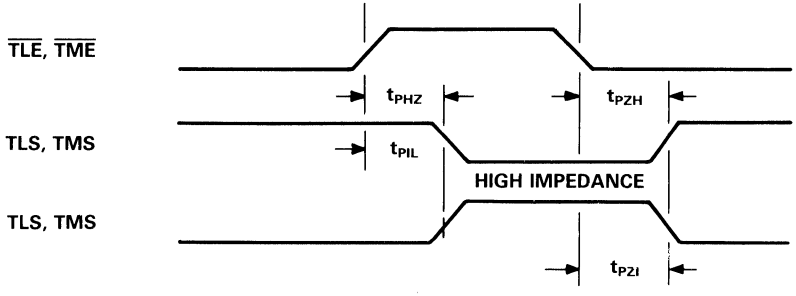


Figure 1. ADSP-7018 Timing Diagram

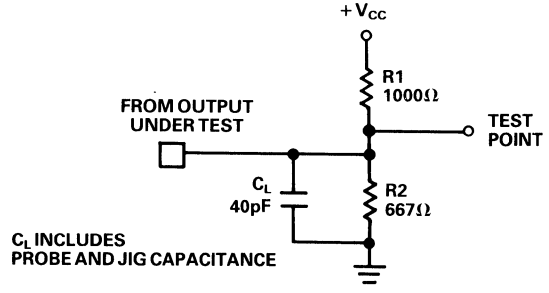


Figure 2. ADSP-7018 Load Test Circuit

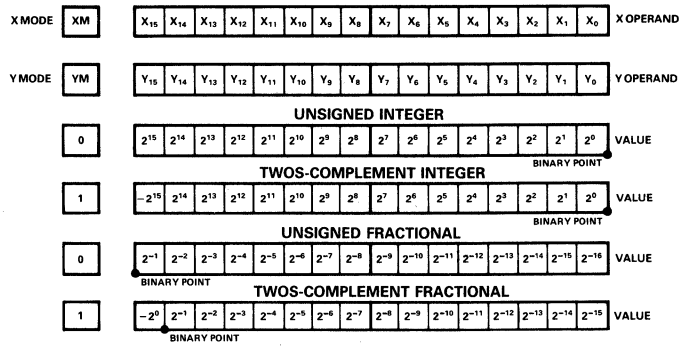


Table I. ADSP-7018 Input Data Formats

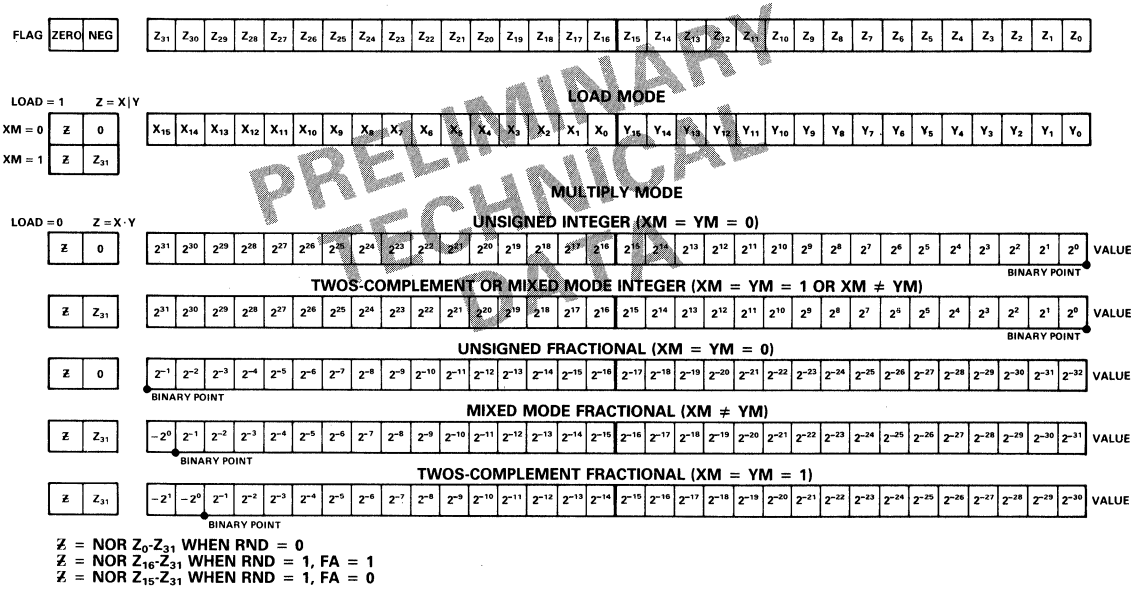


Table II. ADSP-7018 Z Result Formats, ZERO and NEG Flags, and Load Operation

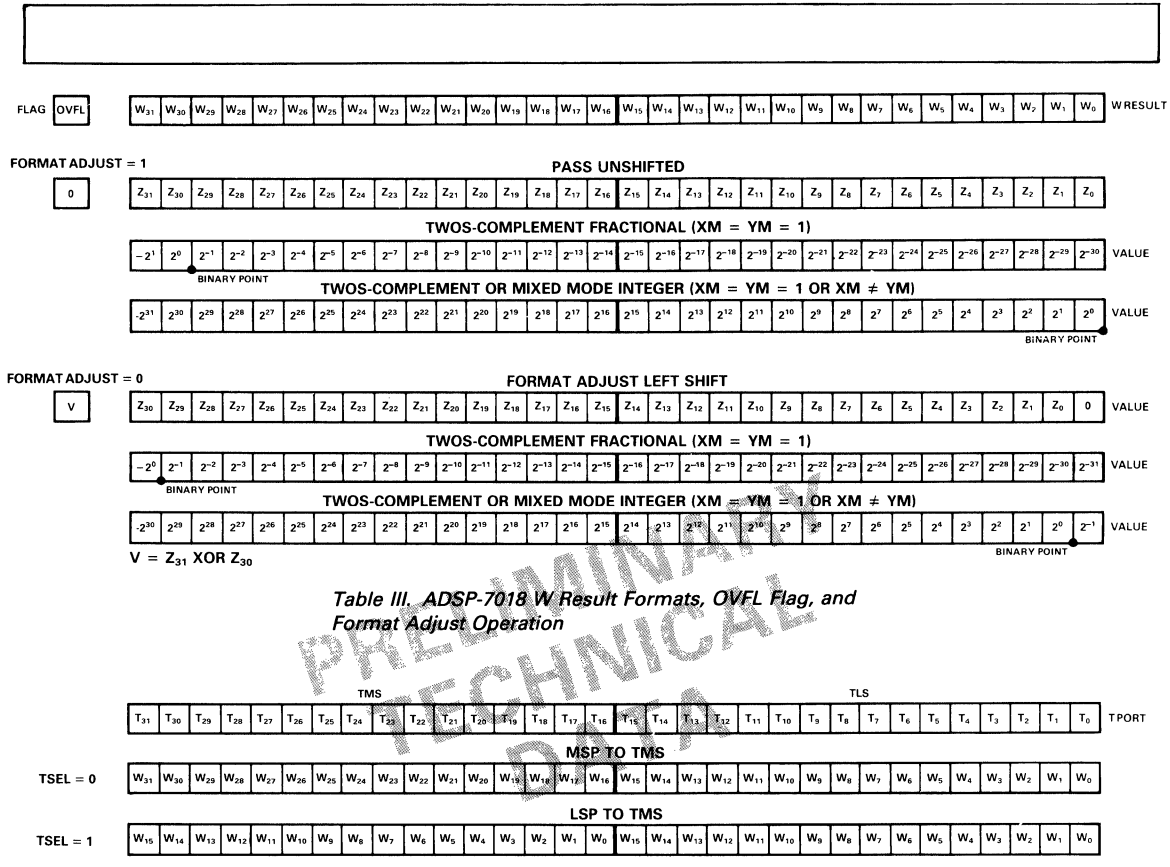


Table III. ADSP-7018 W Result Formats, OVFL Flag, and Format Adjust Operation

Table IV. ADSP-7018 Output Multiplexer

ORDERING INFORMATION

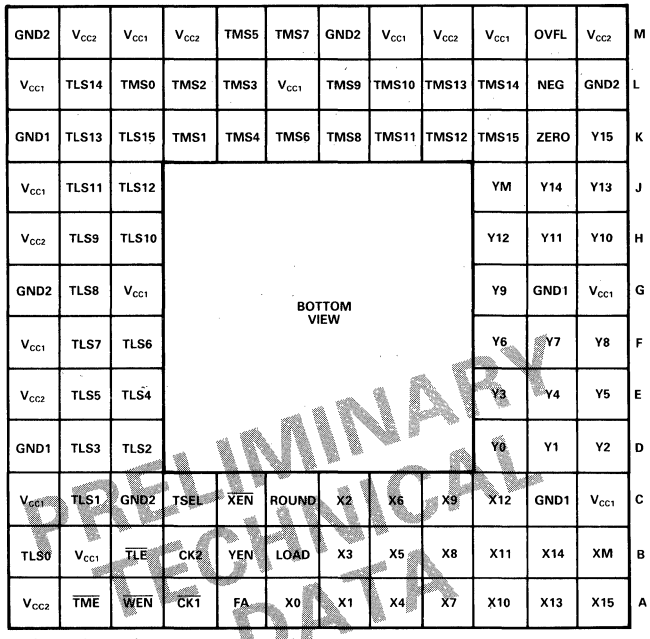
Part Number	Temperature Range	Package	Package Outline
ADSP-7018JG	0 to +70°C	108-Pin Grid Array	G-108A

ESD SENSITIVITY

The ADSP-7018 features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-7018 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.

ADSP-7018 PIN CONFIGURATION



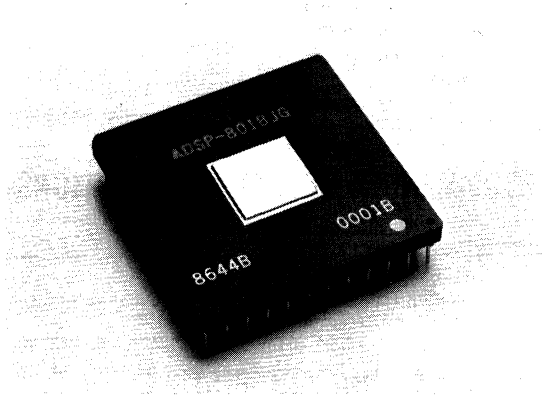
NOTES
1. ALL V_{CC1} PINS MUST BE TIED TO ALL V_{CC2} PINS.
2. ALL GND1 PINS MUST BE TIED TO ALL GND2 PINS.

ANALOG DEVICES

ECL 16 × 16-Bit Multiplier

ADSP-8018

FEATURES
16 × 16-bit Parallel Multiplication
15ns Max Multiplication Time
ECL 10KH Compatible
4.125W Max Power Dissipation
Independent Input and Output Latches which Can Be Made Transparent
Unclocked, Single-, Double-, and Triple-Clock Operation
Twos Complement, Unsigned Magnitude, and Mixed Mode
Format Adjust and Rounding
Parallel Data Load and Passthrough from Input Port
Full 32-Bit Product Output Port
Sign, Overflow, and Zero Status Flags
Pin-Compatible with B3018
1.20″ Square 108-Pin Grid Array



APPLICATIONS
Digital Signal Processing
Array Processors
Super-Minicomputers
General-Purpose Computing

GENERAL DESCRIPTION

The ADSP-8018 is a high-speed 16 × 16-bit parallel ECL multiplier. The ADSP-8018 has two 16-bit input ports and a fielded 32-bit output port. These 32-bit products are available in parallel, as independently enabled 16-bit Least Significant Product (LSP) and 16-bit Most Significant Product (MSP) portions, or LSP and MSP multiplexed to a single 16-bit port. Each 16-bit field of the product latch has its own output enable control.

Inputs can be represented in either twos-complement, unsigned-magnitude, or mixed-mode formats. The input, instruction, status, and output latches are all latches which can be enabled to be transparent. Input operands and output results are stored in individually enabled latches with separate clocks for input and output. For unclocked flow-through operation, the latches can be made transparent by holding clocks and enables active. Tying clocks together as a single clock causes input and output latches to operate in complementary fashion, simplifying synchronous operation. The two clocks are fully independent, allowing double clock operation. With both clock lines held active, the three enable controls can function as three independent clock lines for the two data input ports and the output port, respectively.

The ADSP-8018 produces a 32-bit product that may be format-adjusted for consistent signed fractional output format and for maximum precision in a 16-bit MSP. The MSP of the format-adjusted 32-bit product can be rounded with a control which causes a 1 to be added to the Most Significant Bit (MSB) of the LSP. Three status flags indicate the presence of a zero, negative, or overflowed result. The input operands can be loaded and passed directly through without multiplication, but with format adjustment, rounding, and setting of status flags.

The ADSP-8018 is a pin-for-pin replacement for Bipolar Integrated Technology's B3018 ECL 16 × 16 Multiplier.

The ADSP-8018 is available for commercial temperature ranges in a compact, hermetically-sealed 108-pin grid array.

5

SPECIFICATIONS[1]

ABSOLUTE MAXIMUM RATINGS

Parameter		ADSP-8018 Min	Max	Unit
V_{EE}	Supply Voltage (V_{CC} = 0)	−8.0	0	V
V_{IN}	Input Voltage (V_{CC} = 0)	V_{EE}	0	V
I_O	Output Source Current			
	Continuous		30	mA
	Surge		100	mA
T_{ST}	Storage Temperature (Ambient)	−55	+150	°C
T_J	Operating Junction Temperature		+165	°C

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-8018 Min	Nom	Max	Unit
V_{EE}	Supply Voltage (V_{CC} = 0)	−5.46	−5.20	−4.94	V
T_{AMB}	Operating Temperature (Ambient)[3]	0		+70	°C
R_L	Output Termination to −2.0V dc		50		Ω

ELECTRICAL CHARACTERISTICS

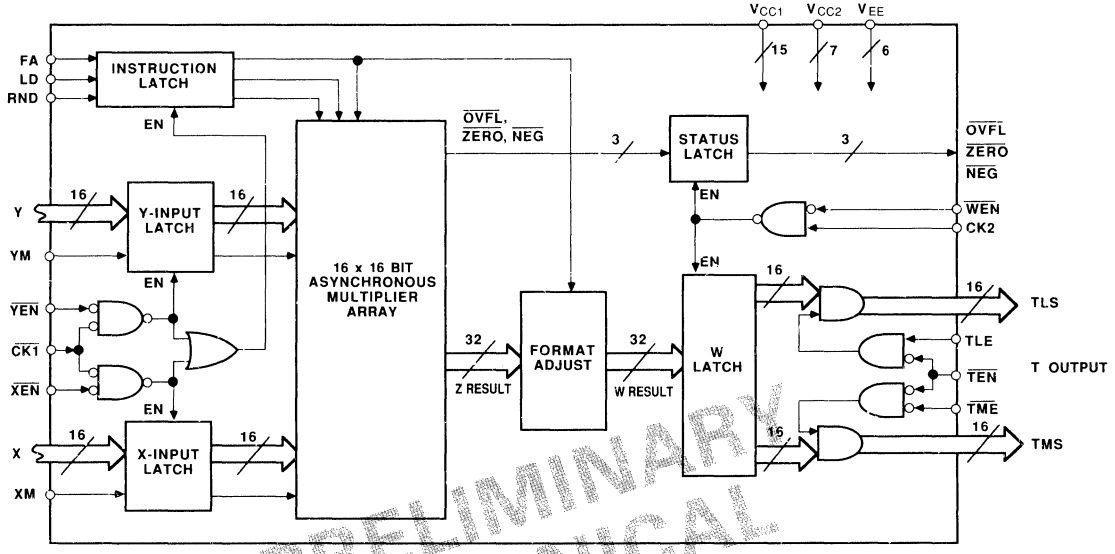
Parameter		Test Conditions	ADSP-8018 0°C Min	0°C Max	25°C Min	25°C Max	70°C Min	70°C Max	Unit
V_{IH}	High-Level Input Voltage	@V_{EE} = max	−1.17		−1.13		−1.07		V
V_{IL}	Low-Level Input Voltage	@V_{EE} = min		−1.48		−1.48		−1.45	V
V_{OH}	High-Level Output Voltage (Terminated)	@V_{EE} = max	−1.02		−0.98		−0.92		V
V_{OL}	Low-Level Output Voltage (Terminated)	@V_{EE} = max		−1.63		−1.63		−1.60	V

Parameter		Test Conditions	Min	Max	Unit
I_{IH}	High-Level Input Current, All Inputs	@ V_{EE} = min & V_{IN} = −800mV		300	μA
I_{IL}	Low-Level Input Current, All Inputs	@V_{EE} = min & V_{IN} = −1.8V		300	μA
I_{EE}	Supply Current	@V_{EE} = min; ECL inputs		825	mA

SWITCHING CHARACTERISTICS[2]

Parameter		ADSP-8018J Min	Max	Unit
t_S	Input Setup Time	0.8		ns
t_H	Input Hold Time	3.0		ns
t_C	Clock and Latch Enable Pulse Duration	3.5		ns
t_{DCM}	Data to Clock Multiply Time		12	ns
t_{CCH}	Clock to Clock Hold Time	−1.5		ns
t_{CCM}	Clock to Clock Multiply Time		15	ns
t_{COD}	Clocked Output Delay		5.5	ns
t_{DDM}	Data to Data Multiply Time		15.5	ns
t_{CDM}	Clock to Data Multiply Time		18	ns
t_{DAO}	Output Disable Delay		5.5	ns
t_{ENO}	Output Enable Delay		5.5	ns

NOTES
[1]All min and max specifications are over power-supply and temperature range indicated.
[2]Input levels are −1.8V and −800mV. Rise times are 2ns typical. Input and output timing reference levels are −1.3V.
[3]500 linear feet per minute ambient air flow. See also Figure 3.
Specifications subject to change without notice.



Functional Block Diagram

PIN DESCRIPTION

DATA	
X_{15-0}	X Operand Input
Y_{15-0}	Y Operand Input
TMS_{15-0}	MSP of 32-Bit T Output
TLS_{15-0}	LSP of 32-Bit T Output
CONTROL	
$\overline{XEN}$	Enable X-Input Latch
$\overline{YEN}$	Enable Y-Input Latch
$\overline{WEN}$	Enable W Latch
XM	X-Input Data Format
YM	Y-Input Data Format
FA	Format Adjust
LD	Load Concatentated X and Y Operands
RND	Round MSP
$\overline{TME}$	Enable TMS Output (Active LO) When $\overline{TEN}$ LO
TLE	Enable TLS Output (Active HI) When $\overline{TEN}$ LO
$\overline{TEN}$	Enable Both TMS and TLS Ouput (Active LO) When $\overline{TME}$ LO and TLE HI
STATUS	
$\overline{ZERO}$	Zero
$\overline{OVFL}$	Overflow
$\overline{NEG}$	Negative
CLOCKS	
$\overline{CK1}$	Input Latches Clock
CK2	Output Latches Clock
POWER	
V_{CC1}	Most Positive Supply Voltage to Internal Logic (Usually GND)
V_{CC2}	Most Positive Supply Voltage to Output Circuits (Usually GND)
V_{EE}	Most Negative Supply Voltage

METHOD OF OPERATION

The X-Input Latch holds its most recent contents unless $\overline{CLKI}$ is LO (logic 0) when enabled (made transparent) by $\overline{XEN}$ also LO. Similarly, the Y-Input Latch holds its most recent contents unless $\overline{CLKI}$ is LO when enabled by $\overline{YEN}$ also LO. The state of the $\overline{CKI}$ input clock has no effect on each input latch unless the input enable lines $\overline{XEN}$ and $\overline{YEN}$ are LO, respectively. When these enable lines are LO, the respective latches will pass data when the $\overline{CKI}$ input is LO and hold data when the $\overline{CKI}$ input is HI (logic 1).

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 levels on the XM line for X input data and HI levels on the YM line for Y input data. Unsigned-magnitude X and Y inputs are indicated by LO levels on the XM and YM 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. XM and YM are latched into the X-Input and Y-Input Latches with the input operand data.

The Instruction Latch holds its most recent contents unless $\overline{CLKI}$ is LO when enabled by *either* $\overline{XEN}$ *or* $\overline{YEN}$ also LO. When $\overline{CKI}$ is HI, the contents of the Instruction Latch will be held. If both $\overline{XEN}$ and $\overline{YEN}$ are HI, the contents of the Instruction Latch will also be held. Round (RND), format adjust (FA), and load (LD) are the three instruction bits that pass through or are held at the Instruction Latch.

The ADSP-8018's W output is fielded into a 16-bit most significant product (MSP) and a 16-bit least significant product (LSP). When the RND is HI, the MSP of the W result 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 of $-(2^{16}-1)/2$ LSBs of the LSP, while rounding (RND HI) reduces the bias to $+1/2$ LSBs of the LSP.

The FA control format adjusts the Z output from the multiplier array. When FA is HI, the full 32-bit Z result is passed unmodified to the W latch (Tables II and III). All possible products can be represented without overflow, including twos-complement fractional $(-1.0) \times (-1.0)$. The binary point in the MSP of a twos-complement fractional product will not align with the binary point in the input operands. Thus, the numeric weighting of data within a system using this format (with FA HI) will not be uniform. Also, the two highest-order bits in a twos-complement product are normally redundant. An additional bit of precision in the MSP could be obtained by eliminating one of these redundant bits.

Format adjust (FA LO) shifts the Z result left by one bit position and right-fills a binary zero (Table III). Thus, format adjust eliminates redundant high-order bits in twos-complement products. The MSPs of fractional twos-complement products also receive proper weighting relative to input data. 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). This special condition is monitored by the OVFL flag so that it does not go undetected.

The Load (LD) instruction determines whether the multiplier array is active or put into a pass-through mode. When LD is LO, the multiplier array is active and produces the product of the X and Y input values as Z result. When LD is HI, the multiplier array is put in the pass-through mode and the X and Y input values are concatenated to form a Z result with the X input in the most significant position (Table II). XM determines the sign mode of the complete 32-bit Z result. ZERO, OVFL and NEG are set depending on the value of this Z result. RND and FA can also be used in this mode. Load allows any 32-bit number loaded through the input ports to be rounded, tested for zero and sign, and to have fractional binary points aligned while testing for overflow. It is also valuable for system testing by providing a direct path through the multiplier and greater access to latches within the ADSP-8018.

Output flags $\overline{ZERO}$, $\overline{OVFL}$, and $\overline{NEG}$ are generated from the intermediate Z result. When not rounding (RND LO), $\overline{ZERO}$ is driven LO when the full 32-bit result is zero. When rounding (RND HI), $\overline{ZERO}$ is driven LO when the format-adjusted MSP in the W result will be zero. That is, when rounding, $\overline{ZERO}$ is driven LO for $Z_{31:16}$ equals zero when FA HI but $\overline{ZERO}$ is driven LO for $Z_{31:15}$ equals zero when FA LO. Zero will not be driven if an overflow condition exists.

$\overline{OVFL}$ is driven LO only for twos-complement multiplications of full-scale negative times full-scale negative when format adjusting (FA LO).

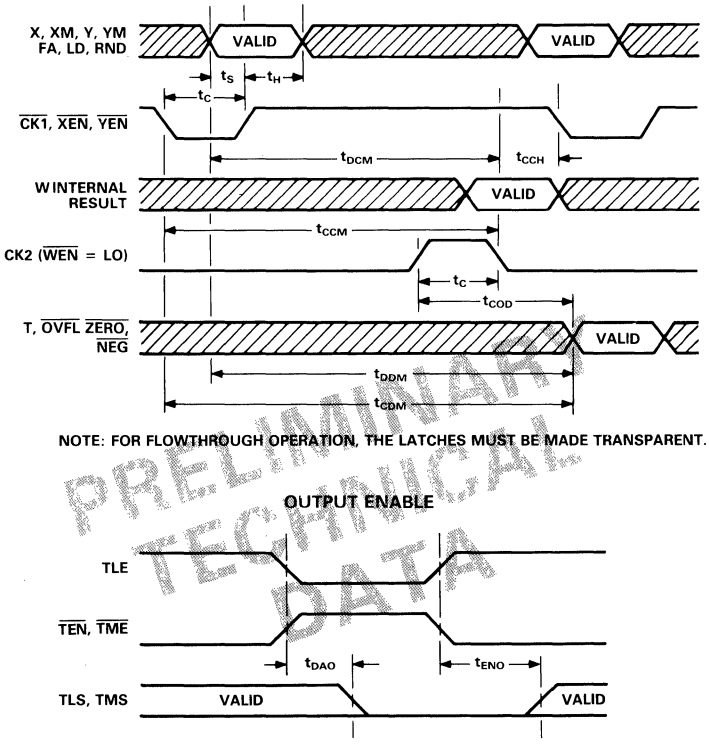
$\overline{NEG}$ is driven LO only when a twos-complement or mixed-mode Z result is negative. A loaded, previously overflowed result (MSB of 1 followed by all zeros) will drive $\overline{NEG}$ LO as well as $\overline{OVFL}$, if loaded with XM HI.

The output latches for data and flags $\overline{ZERO}$, $\overline{OVFL}$ and $\overline{NEG}$ hold their most recent contents unless CLK2 is HI when enabled by $\overline{WEN}$ LO. With $\overline{WEN}$ LO, the output latches are transparent when CK2 is HI and data is held when CK2 is LO. The ADSP-8018 can be made a totally asynchronous device by holding all enables active ($\overline{XEN}$, $\overline{YEN}$, and $\overline{WEN}$ all LO) and $\overline{CLKI}$ LO and CLK2 HI. Note that the clock definitions are complementary; if $\overline{CLKI}$ and CLK2 are tied together and enable lines asserted, input latches will be transparent when output latches are latched and vice versa. $\overline{CLKI}$ and CLK2 are fully independent, allowing double-clock operation. For triple-clock operation, the two clock lines can be held active, which causes the three latch enable controls to behave like independent clock lines.

The T output is fielded into two 16-bit portions, TMS and TLS. The three output enable controls, $\overline{TEN}$, $\overline{TME}$, and TLE, allow either independent control of TMS and TLS separately or control of all 32 bits together. For independent control of the 16-bit fields, $\overline{TEN}$ should be held LO, thereby enabling the other pair of output enable controls. $\overline{TME}$ LO then will enable the TLS output and TLE HI will enable the TMS output. For control of the entire 32-bit T output, $\overline{TME}$ should be held LO and TLE held HI. $\overline{TEN}$ LO then will enable the entire T output.

ECL 16 × 16-BIT FIXED-POINT MULTIPLIER

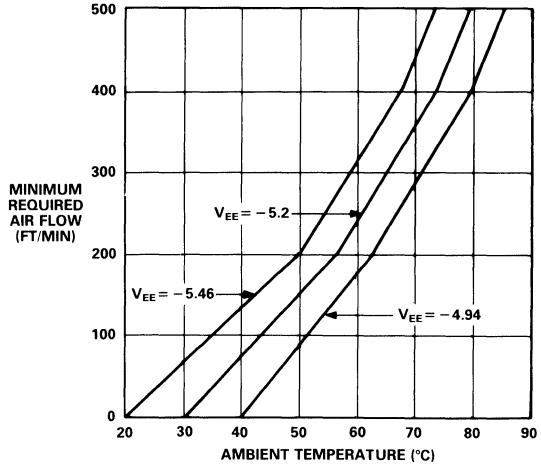
CLOCKED AND UNCLOCKED TIMING



NOTE: FOR FLOWTHROUGH OPERATION, THE LATCHES MUST BE MADE TRANSPARENT.

OUTPUT ENABLE

Figure 1. ADSP-8018 Timing Diagram



Air Flow Versus Ambient Temperature

This information applies to a product under development. Its characteristics and specifictions are subject to change without notice. Analog Devices assumes no obligation regarding future manufacture unless otherwise agreed to in writing.

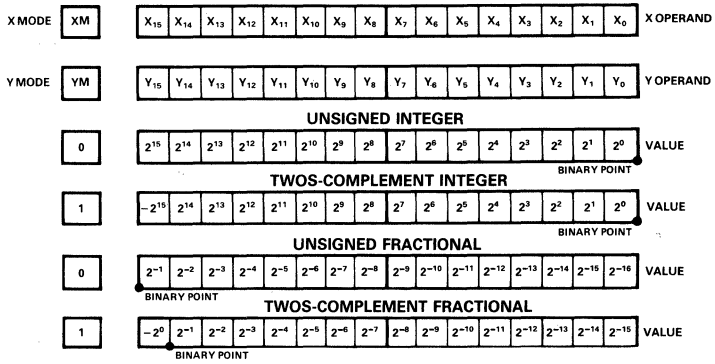
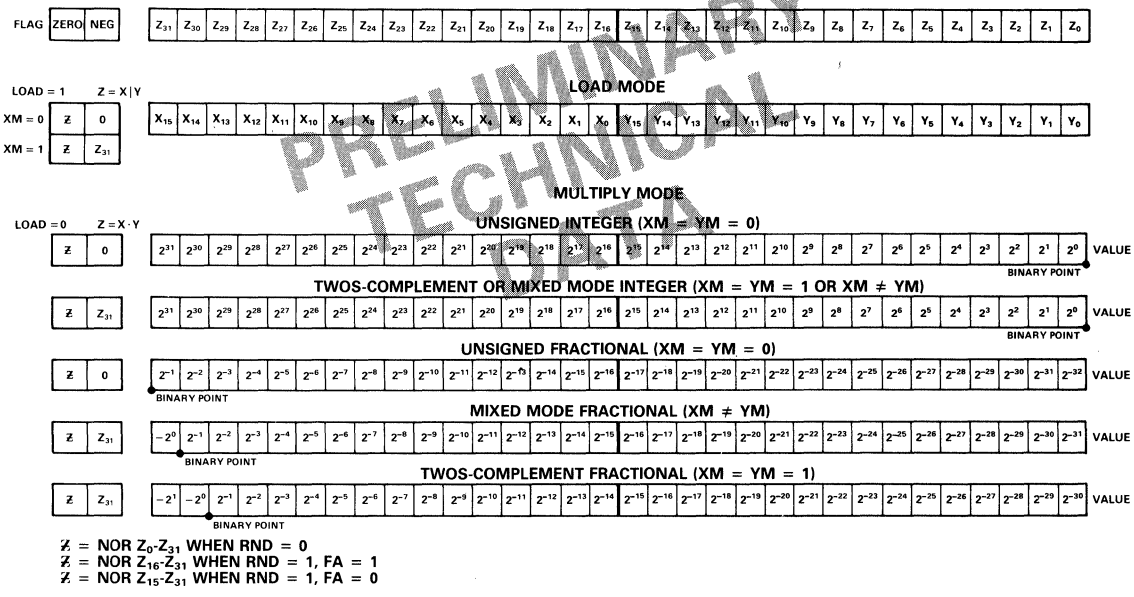


Table I. ADSP-8018 Input Data Formats



Z = NOR Z_0-Z_{31} WHEN RND = 0
Z = NOR Z_{16}-Z_{31} WHEN RND = 1, FA = 1
Z = NOR Z_{15}-Z_{31} WHEN RND = 1, FA = 0

Table II. ADSP-8018 Z Result Formats, ZERO and NEG Flags, and Load Operation

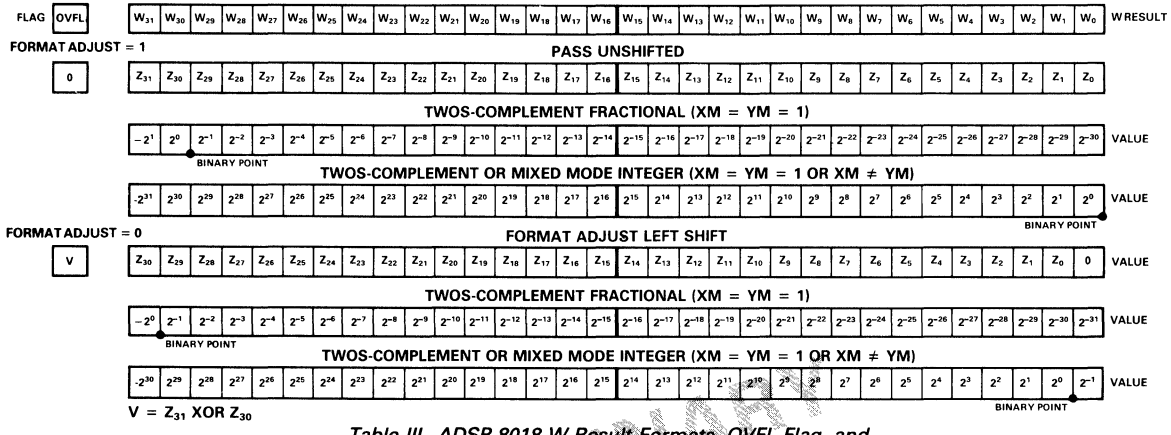


Table III. ADSP-8018 W Result Formats, OVFL Flag, and Format Adjust Operation

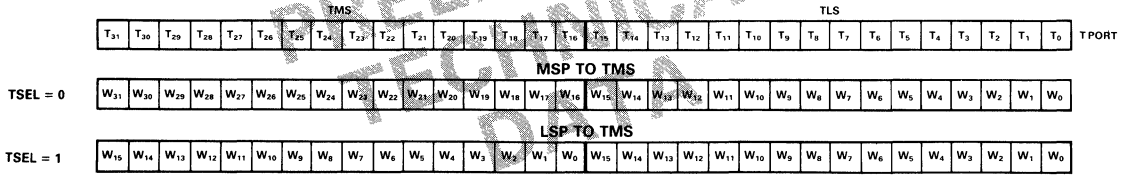


Table IV. ADSP-8018 Output Multiplexer

ESD SENSITIVITY

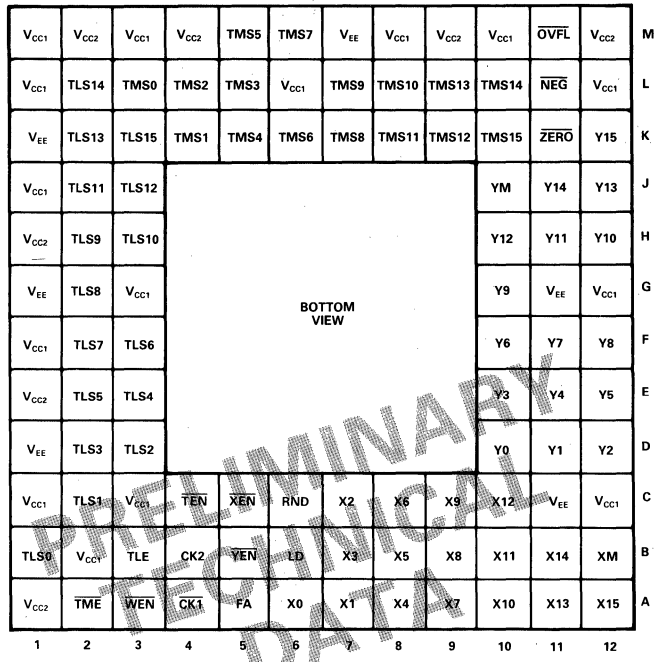
The ADSP-8018 features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-8018 has been classified 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.

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-8018JG	0 to +70°C	108-Pin Grid Array	G-108A

ADSP-8018 PIN CONFIGURATION



NOTE
ALL V_{CC1} PINS MUST BE TIED TO ALL V_{CC2} PINS.

This information applies to a product under development. Its characteristics and specifictions are subject to change without notice. Analog Devices assumes no obligation regarding future manufacture unless otherwise agreed to in writing.

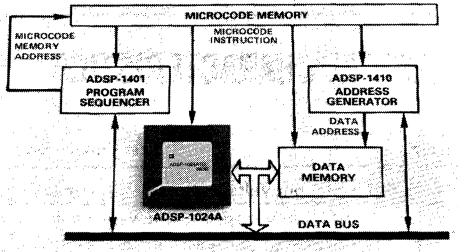
ANALOG DEVICES

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

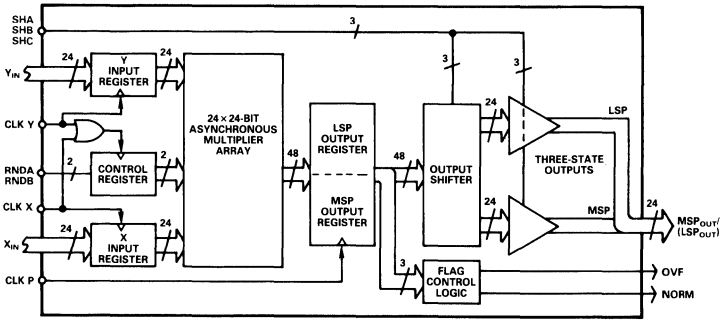
The ADSP-1024A is a high-speed, low-power 24 × 24-bit parallel multiplier fabricated in 1.5 micron CMOS. The ADSP-1024A is a pin-for-pin replacement for Analog Devices' ADSP-1024.

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 diode-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. Extended temperature range parts are available with optional high-reliability processing ("PLUS" parts) (see Figure 6). 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	ADSP-1024A J and K Grades Max	ADSP-1024A S and T Grades[2] Min	ADSP-1024A 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	ADSP-1024A J and K Grades Max	ADSP-1024A S and T Grades[2] Min	ADSP-1024A 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 Grades Min	J Grades Max	K Grades Min	K Grades Max	−55°C to +125°C S Grades Min	S Grades Max	T Grades Min	T Grades 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 1.

[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 Data Book. Alternatively, S and T grade parts are available with high temperature testing or with high-reliability "PLUS" processing as shown in Figure 6.

Specifications subject to change without notice.

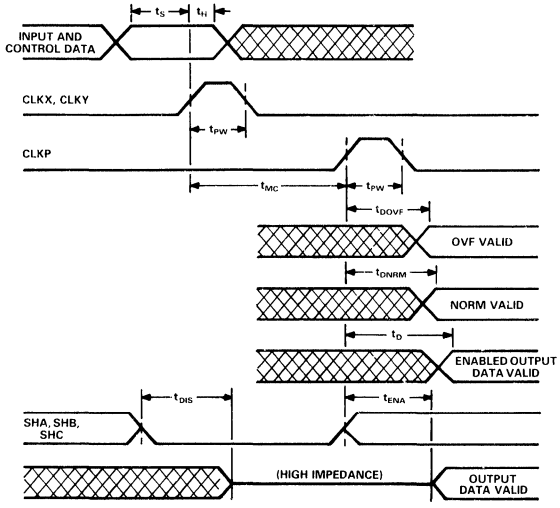


Figure 1. ADSP-1024A Timing Diagram

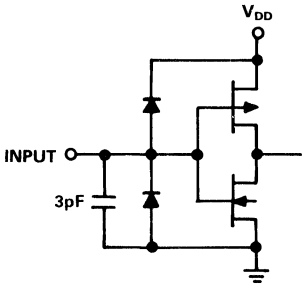


Figure 2. Equivalent Input Circuit

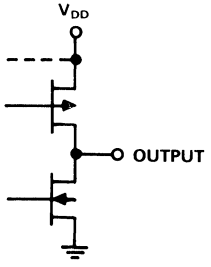


Figure 3. Equivalent Output Circuit

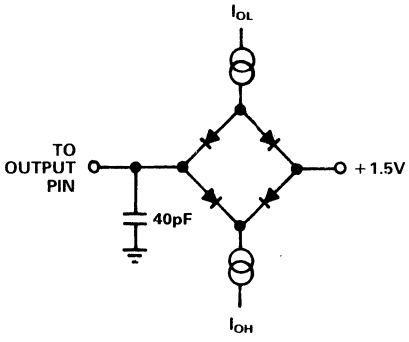


Figure 4. Normal Load for AC Measurements

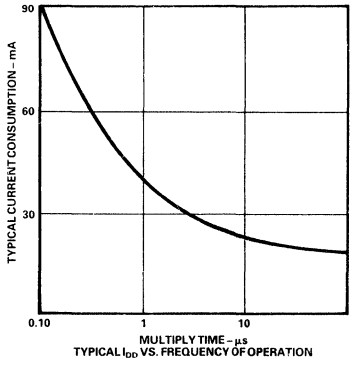


Figure 5. Typical Power Dissipation vs. Frequency

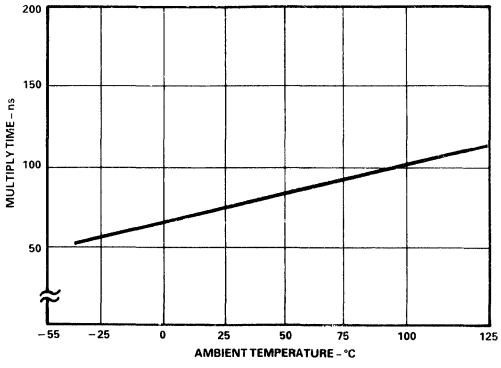


Figure 6. Approx. Worst Case Multiply Time vs. Temperature

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 (SH C, 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 (SH C, 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 (SH C, 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 (SH C, 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 (SH C, 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 (SH C, 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.

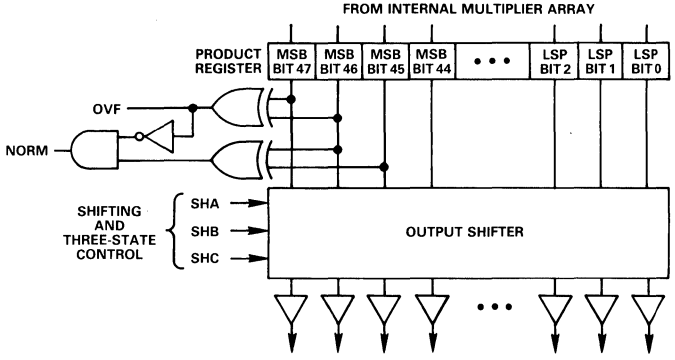


Figure 7. Flag and Shift Logic

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 8. Range of Values in Product Register and Their Effects on OVF and NORM

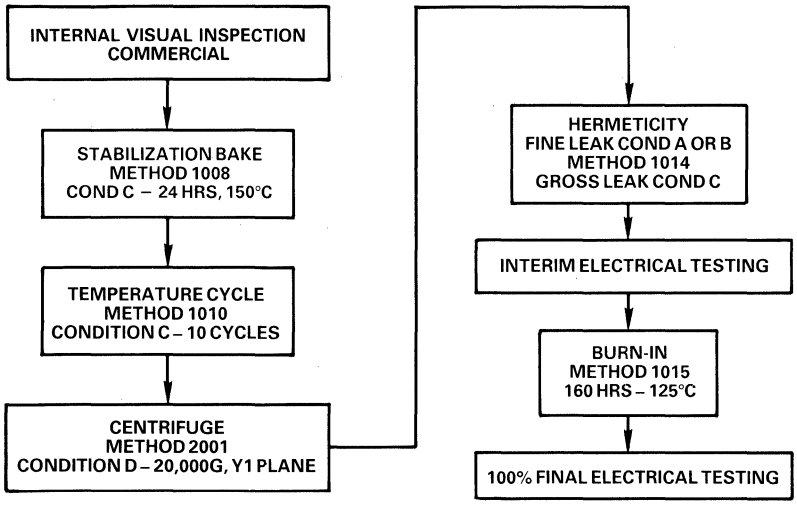


Figure 9. PLUS Processing Environmental Flow

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/+	−55°C to +125°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/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 input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1024A 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.

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.

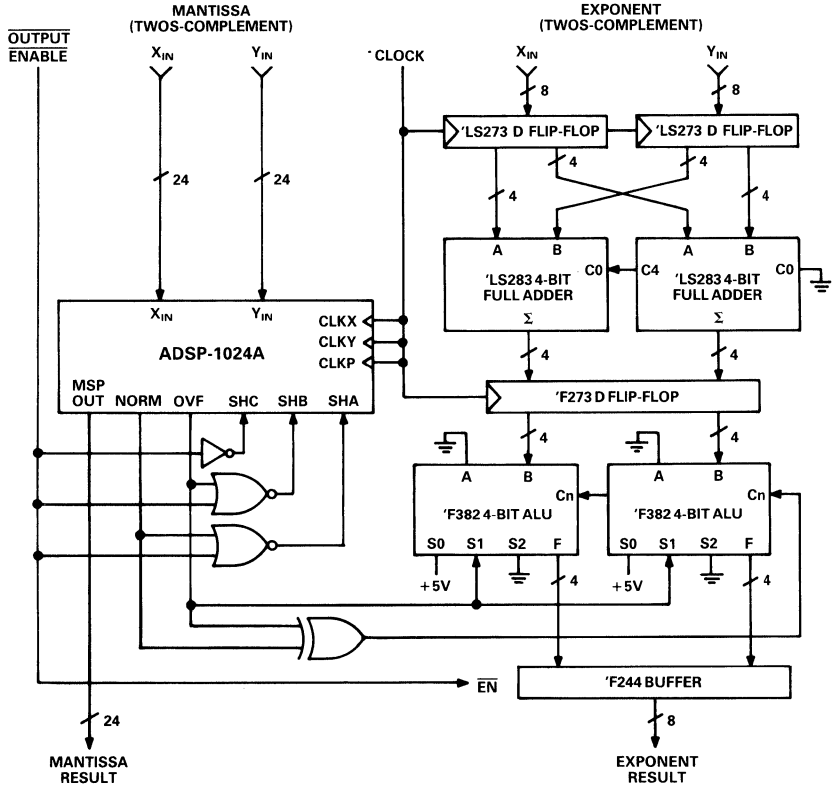


Figure 10. 32-Bit Floating-Point Multiplier Circuit

ADSP-1024A PIN CONFIGURATION

PIN GRID ARRAY
G-84A

FUNCTION	PIN NO.	FUNCTION	PIN NO.
X20	C6	P16, P40	J6
X21	C5	P15, P39	L6
X22	A5	P14, P38	L7
X23	B5	P13, P37	K7
CLKX	A6	P12, P36	J7
CLKY	A4	P11, P35	L8
RNDB	B4	P10, P34	K8
RNDA	A3	P9, P33	L9
Y0	A2	P8, P32	L10
Y1	B3	P7, P31	K9
Y2	A1	P6, P30	L11
Y3	B2	P5, P29	K10
Y4	C2	P4, P28	J10
Y5	B1	P3, P27	K11
Y6	C1	P2, P26	J11
Y7	D2	P1, P25	H10
Y8	D1	P0, P24	H11
Y9	F2	SHA	G9
Y10	E2	SHB	G10
Y11	E1	SHC	G11
V_{DD}	E3	CLKP	F10
Y12	F3	GND	F9
Y13	F1	X0	E9
Y14	G1	X1	E11
Y15	G2	X2	E10
Y16	G3	X3	F11
Y17	H1	X4	D11
Y18	H2	X5	D10
Y19	J1	X6	C11
Y20	K1	X7	B11
Y21	J2	X8	C10
Y22	L1	X9	A11
Y23	K2	X10	B10
OVF	K3	X11	B9
NORM	L2	X12	A10
P23, P47	L3	X13	A9
P22, P46	K4	X14	B8
P21, P45	L4	X15	A8
P20, P44	K6	X16	C7
P19, P43	K5	X17	B7
P18, P42	L5	X18	A7
P17, P41	J5	X19	B6

16 × 16-Bit CMOS Single Port Multiplier/Accumulator

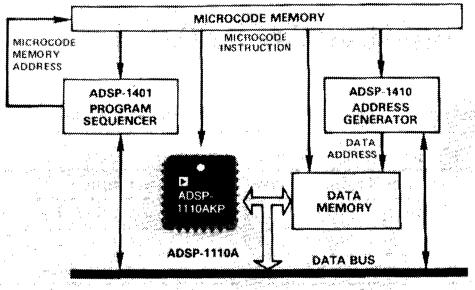
ADSP-1110A

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, Plastic Leaded Chip Carrier, or Leadless 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

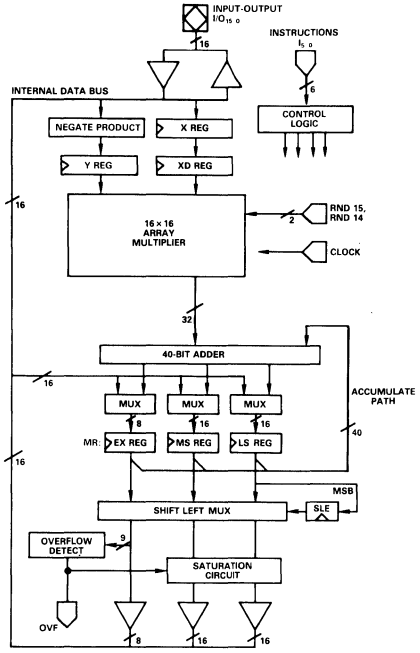
The ADSP-1110A is a high-speed, low-power single-port 16 × 16-bit multiplier/accumulator (MAC), with processing throughput comparable to existing three-port MACs. Its single-bus structure offers unique advantages: more compact packaging in a 28-pin package, simpler system interface to single-bus peripherals, 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

		ADSP-1110A J Grade 0 to +70°C Min	J Grade Max	K Grade 0 to +70°C Min	K Grade Max	S Grade[2] −55°C to +125°C Min	S Grade Max	T Grade[2] −55°C to +125°C Min	T Grade 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. Alternatively, S and T grade parts are available with optional high-reliability "PLUS" processing.

Specifications subject to change without notice.

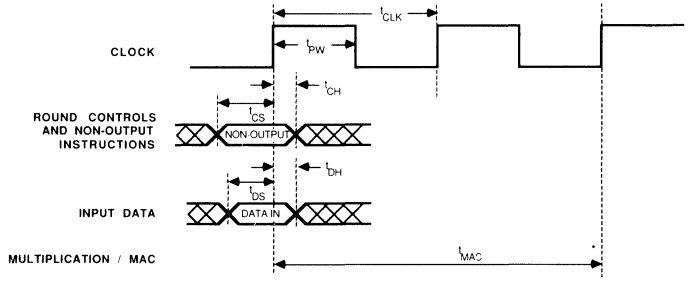


Figure 1a. ADSP-1110A Timing: Clocked (Synchronous) Operations All Non-Output Instructions

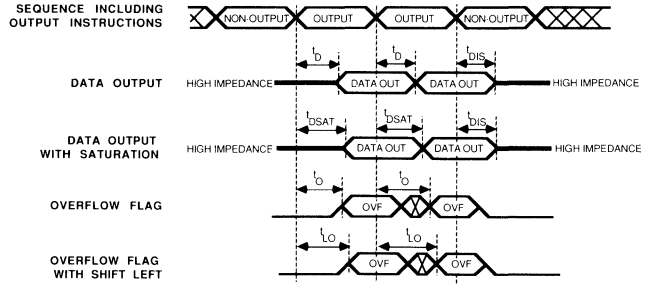


Figure 1b. ADSP-1110A Timing: Unclocked (Asynchronous) Operations All Output Instructions

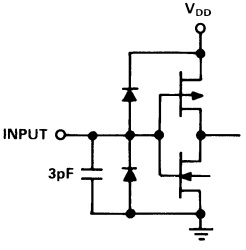


Figure 2a. Equivalent Input Circuits

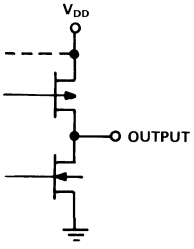


Figure 2b. Equivalent Output Circuits

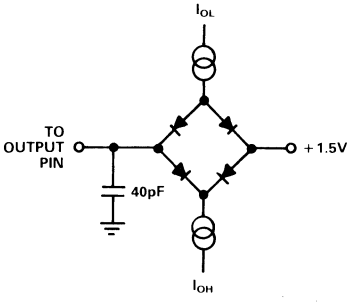


Figure 3. Normal Load Circuit for ac Measurements

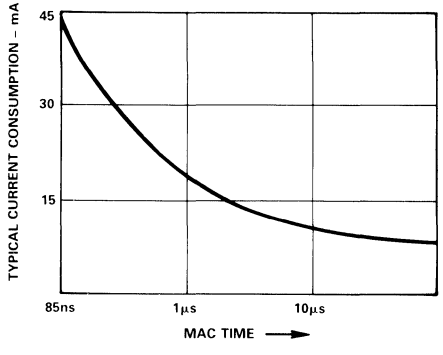


Figure 4. Typical Power Dissipation vs. Frequency

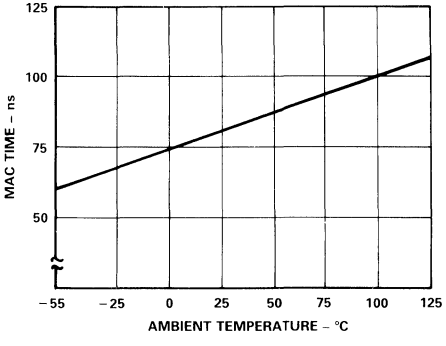


Figure 5. Typical Multiply Time vs. Temperature

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.

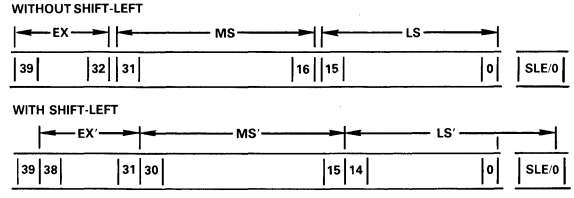


Figure 6. Effect of Left Shift on MR Outputs

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 ------ $2^{2\cdot}$	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.

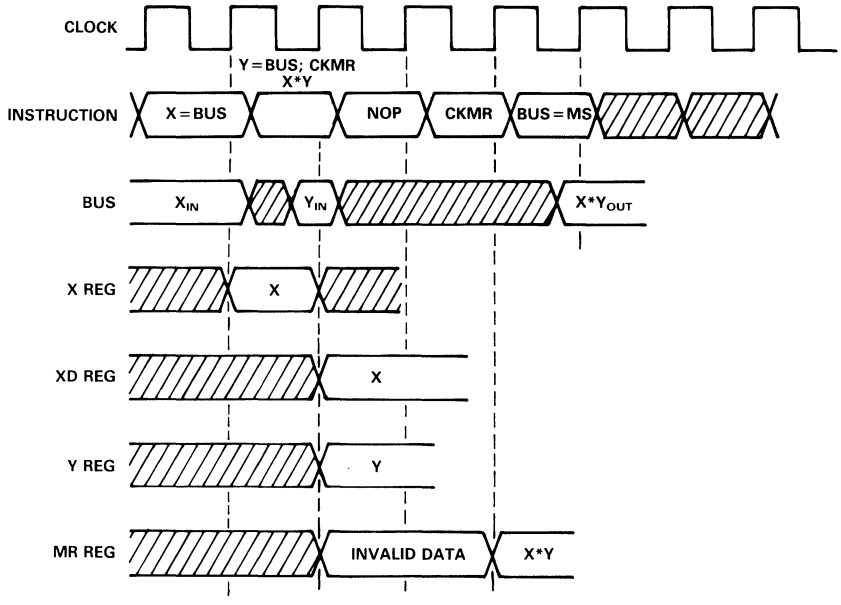


Figure 7. Multiply Operation Timing

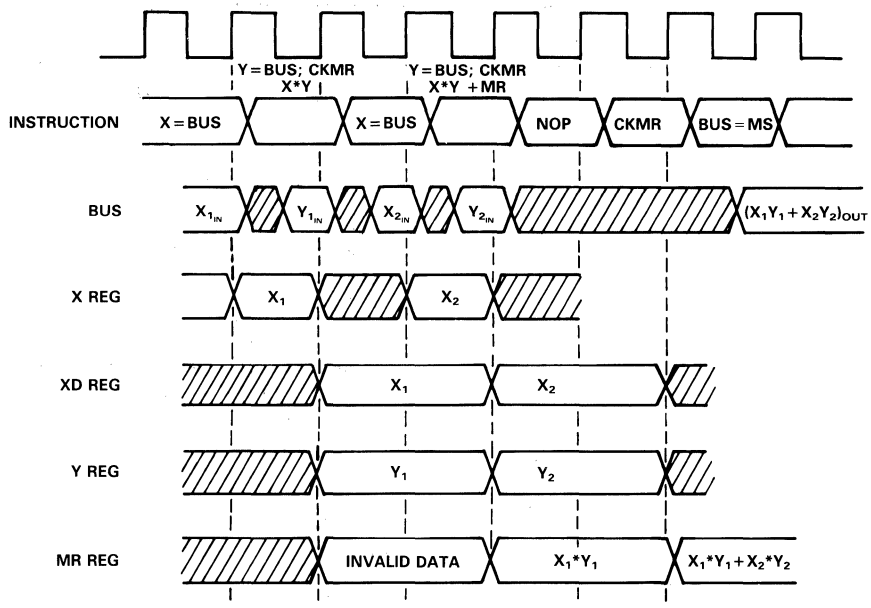


Figure 8. Multiply/Accumulate Operation Timing

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 IMPLEMENTATONS 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:

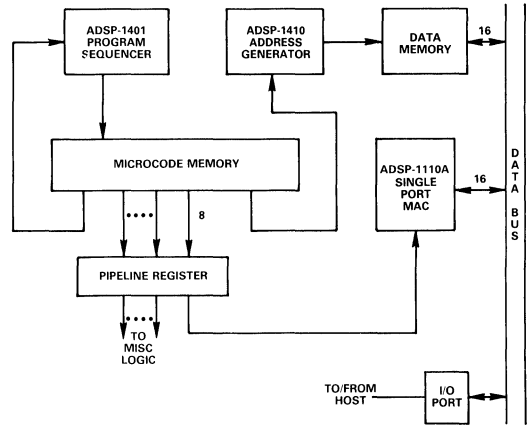


Figure 9.

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 trademark of Analog Devices, Inc.

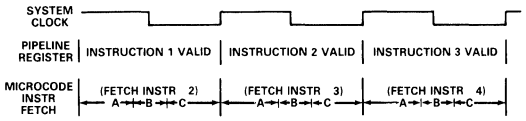


Figure 10.

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
CKRM	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 bits 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:

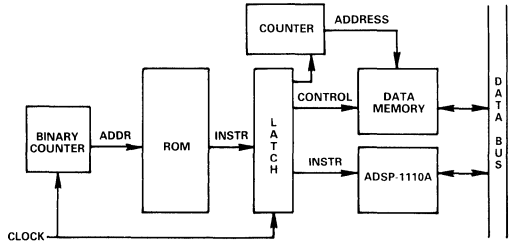


Figure 11.

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:

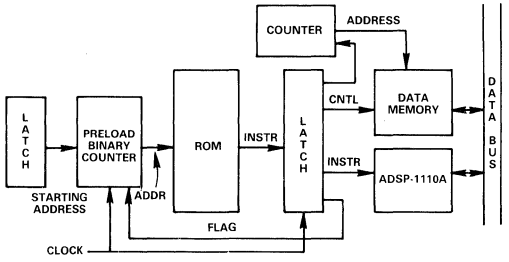


Figure 12.

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:

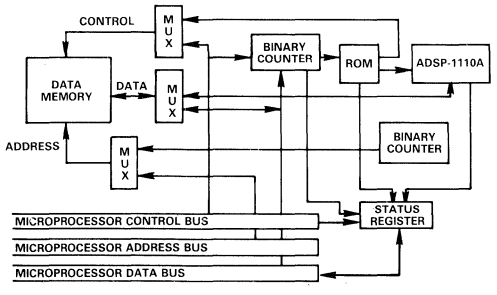


Figure 13.

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

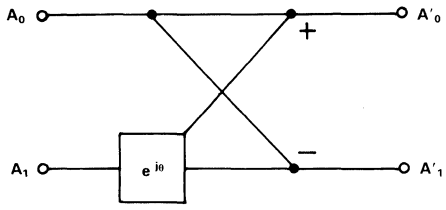


Figure 14. FFT "Butterfly" Diagram

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 butterly
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.

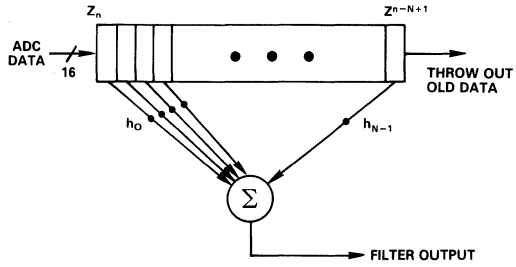


Figure 15. FIR Filter Design

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.

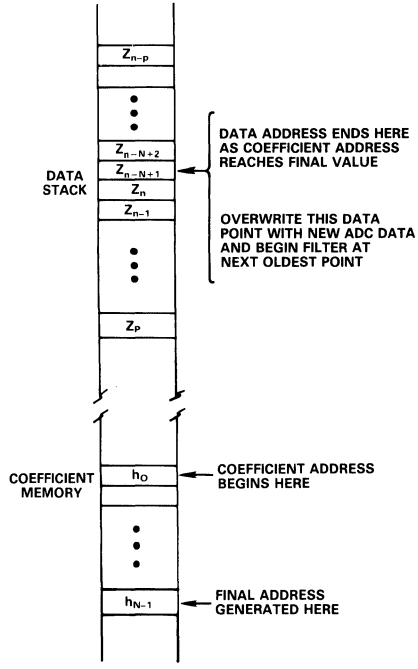


Figure 16. Memory Map

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.

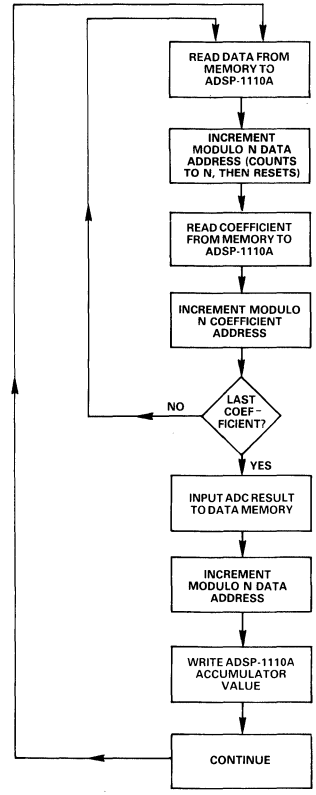


Figure 17. Controller Flow Chart for FIR Filter

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,

$$Y_0 = a_0X_0 + a_1X_{-1} + a_2X_{-2} - b_1Y_{-1} - b_2Y_{-2} \qquad (9)$$

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 MAC's

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.

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:

$$AP = \left[\left\{\sum_{i=1}^{N} (MSW_{xi})(MSW_{yi})\right\}\right] + \tag{12}$$

$$+ \left[\left\{\sum_{i=1}^{N} ((MSW_{xi})(MSW_{yi}) + (LSW_{xi})(MSW_{yi}))\right\}\right] \cdot 2^{-16}$$

$$+ \left\{\sum_{i=1}^{N} (LSW_{xi})(LSW_{yi})\right\} \cdot 2^{-32}$$

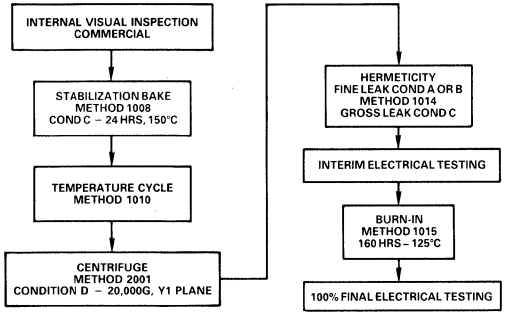
Computing the accumulated double-precision product AP requires the same basic sequence as in computing a single-precision MAC. Simply compute a summation of partial products, rather than the summation of products themselves.

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



PLUS Processing Environmental Flow

ADSP-1110A PIN CONFIGURATION ALL PACKAGES D-28, N-28A, P-28, E-28A

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-28
ADSP-1110AKD	0 to +70°C	28-Pin Ceramic DIP	D-28
ADSP-1110ASD	−55°C to +125°C	28-Pin Ceramic DIP	D-28
ADSP-1110ATD	−55°C to +125°C	28-Pin Ceramic DIP	D-28
ADSP-1110ASD/+	−55°C to +125°C	28-Pin Ceramic DIP	D-28
ADSP-1110ATD/+	−55°C to +125°C	28-Pin Ceramic DIP	D-28
ADSP-1110ASD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28
ADSP-1110ATD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28
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
ADSP-1110ASE/+	−55°C to +125°C	28-Contact Leadless Chip Carrier	E-28A
ADSP-1110ATE/+	−55°C to +125°C	28-Contact Leadless Chip Carrier	E-28A
ADSP-1110ASE/883B	−55°C to +125°C	28-Contact Leadless Chip Carrier	E-28A
ADSP-1110ATE/883B	−55°C to +125°C	28-Contact Leadless Chip Carrier	E-28A

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

ESD SENSITIVITY

The ADSP-1110A features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883B, the ADSP-1110A 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.

ANALOG DEVICES

Integer Arithmetic Unit

ADSP-1101

FEATURES

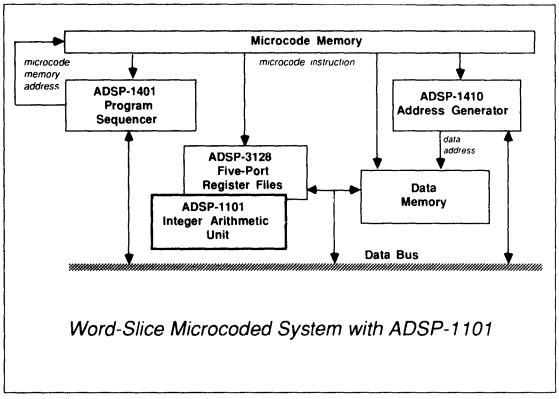
- **16x16-Bit Parallel Multiplication / 40-Bit Accumulation**
- **80ns Cycle Time**
- **Can Support 2.4ms 1024-Point Complex FFT with Block Floating-Point**
- **40-Bit Adder/Subtracter with Status Flags**
- **16-Bit Logic Unit**
- **Dual 40-Bit Accumulators with Status Flags**
- **Right/Left Shifts on 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 with optional high-reliability processing ("PLUS" parts, see Figure 18) or processed fully to MIL-STD-883, Class B. The ADSP-1101 is available packaged in either a ceramic 100-lead pin grid array or a 100-lead plastic leaded chip carrier.

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)

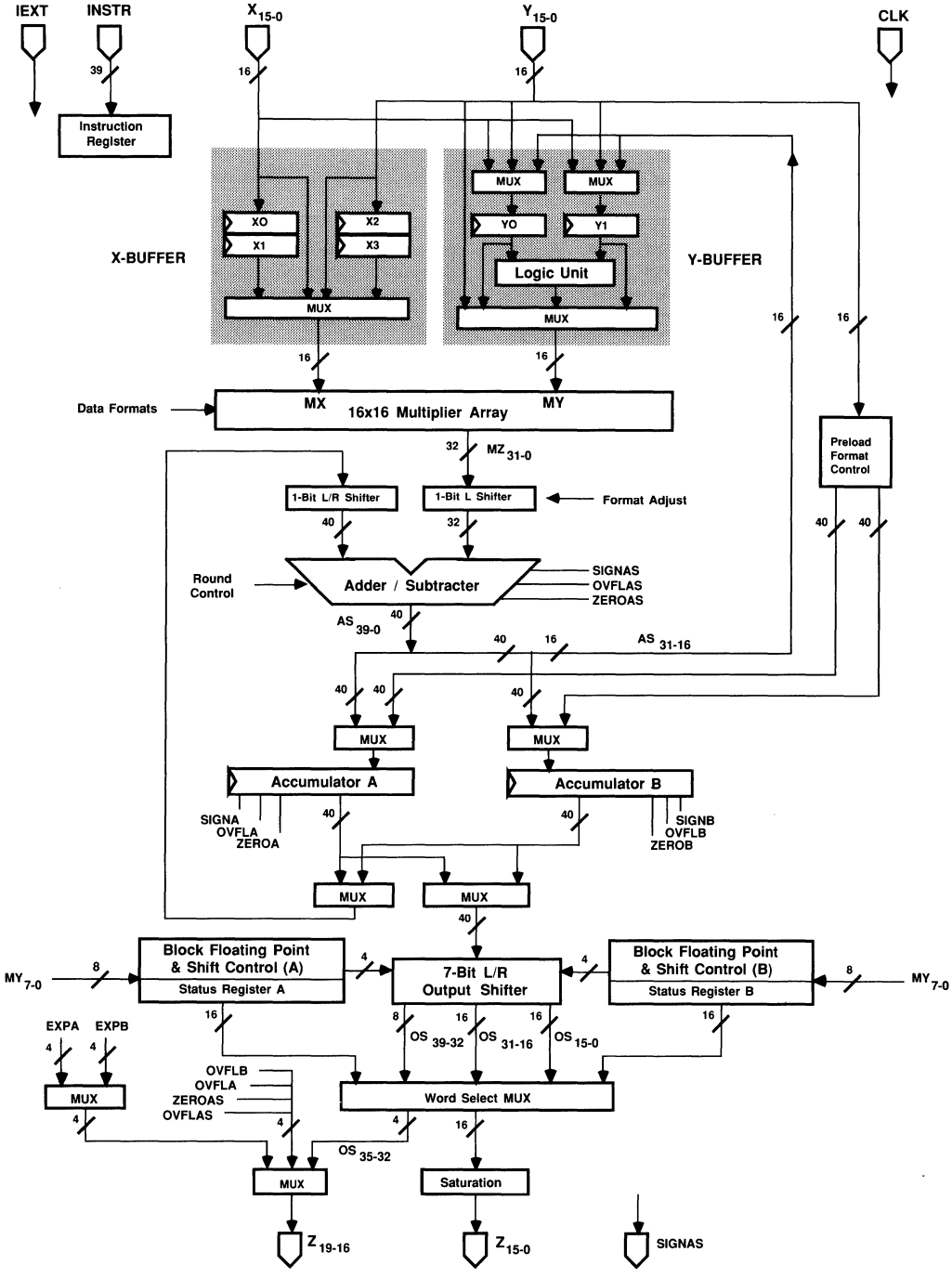


Figure 1. ADSP-1101 Functional Block Diagram

	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 Figure 20.) 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

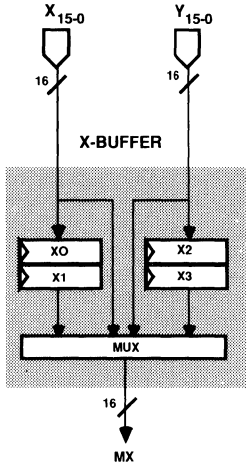


Figure 3 ADSP-1101 X-Buffer

	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.

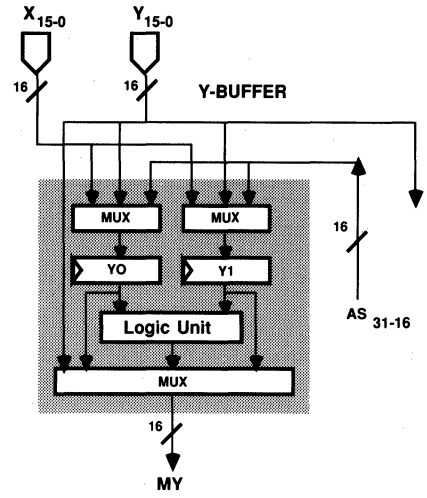


Figure 4. ADSP-1101 Y-Buffer

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 20).

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 $AS_{31\text{-}16}$ 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.

	Sign				
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 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.

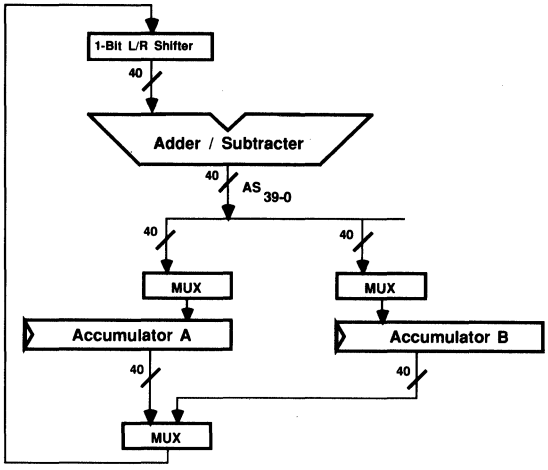


Figure 6. ADSP-1101 Accumulator Feedback Data Paths

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

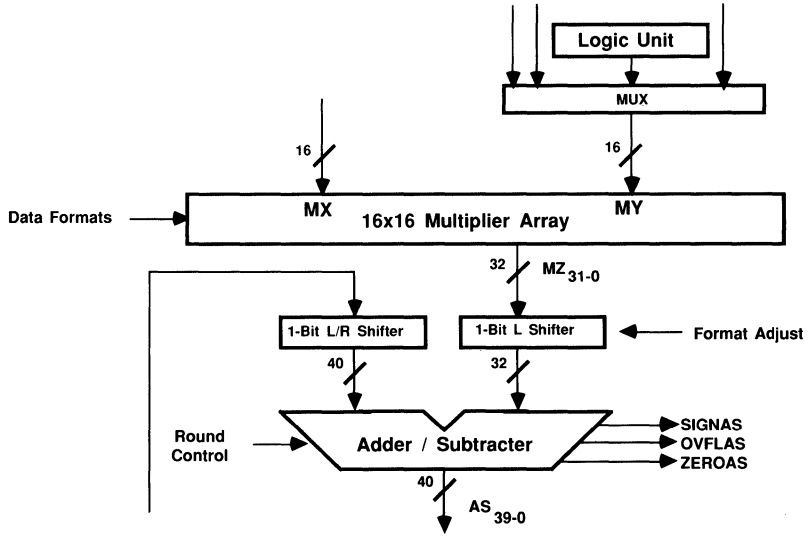


Figure 8. ADSP-1101 Logic Unit, Multiplier Array, and Adder/Subtracter

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 sign-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).

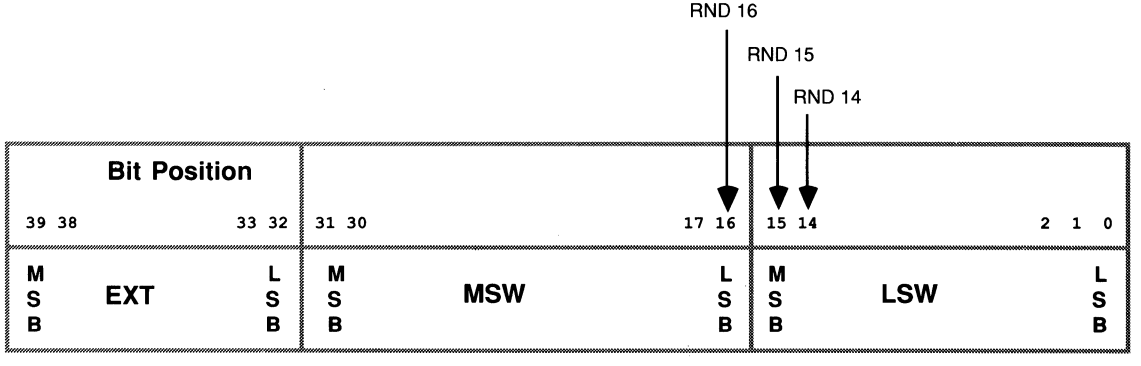
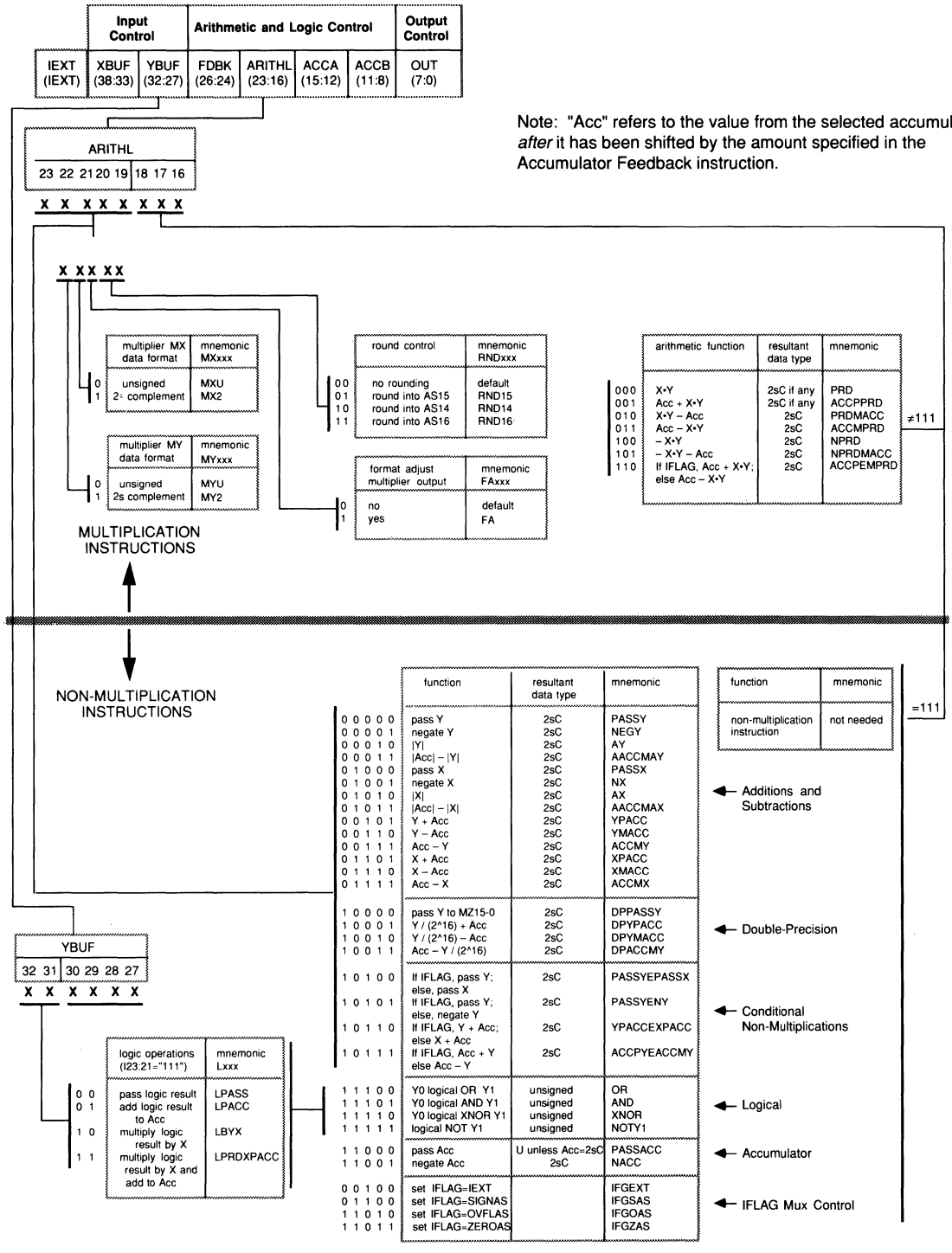


Figure 9. ADSP-1101 Rounding Positions



Note: "Acc" refers to the value from the selected accumulator *after* it has been shifted by the amount specified in the Accumulator Feedback instruction.

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.

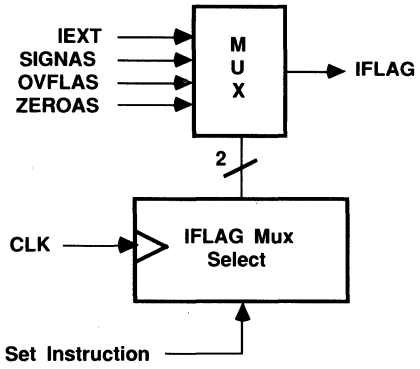


Figure 10. ADSP-1101 IFLAG Mux

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.

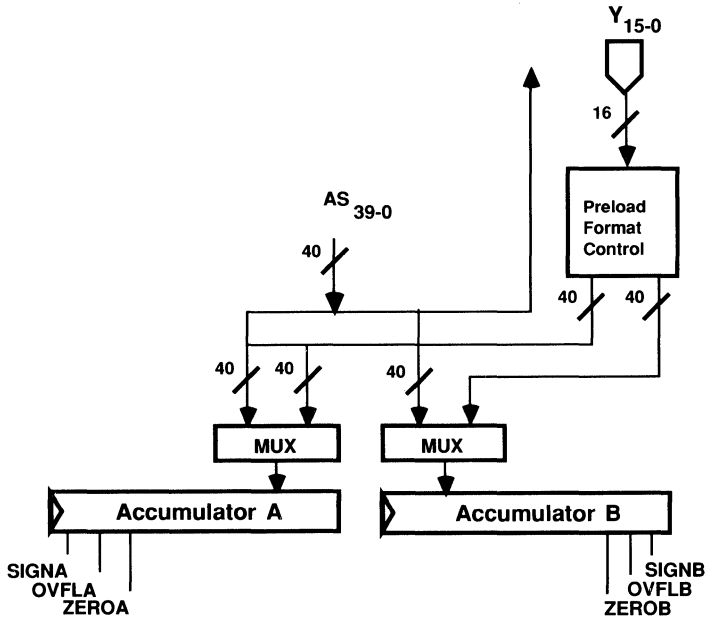
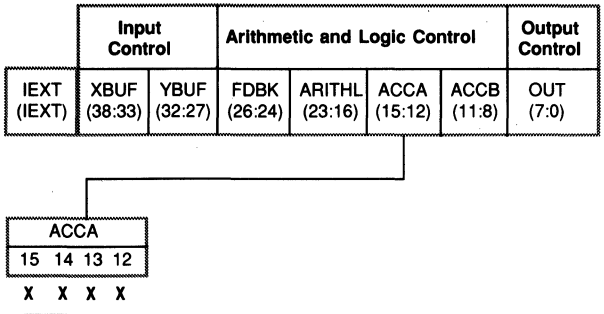
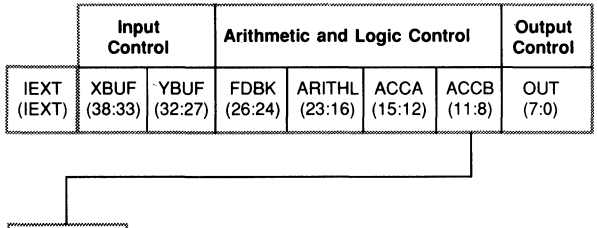


Figure 11. ADSP-1101 Accumulators



	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

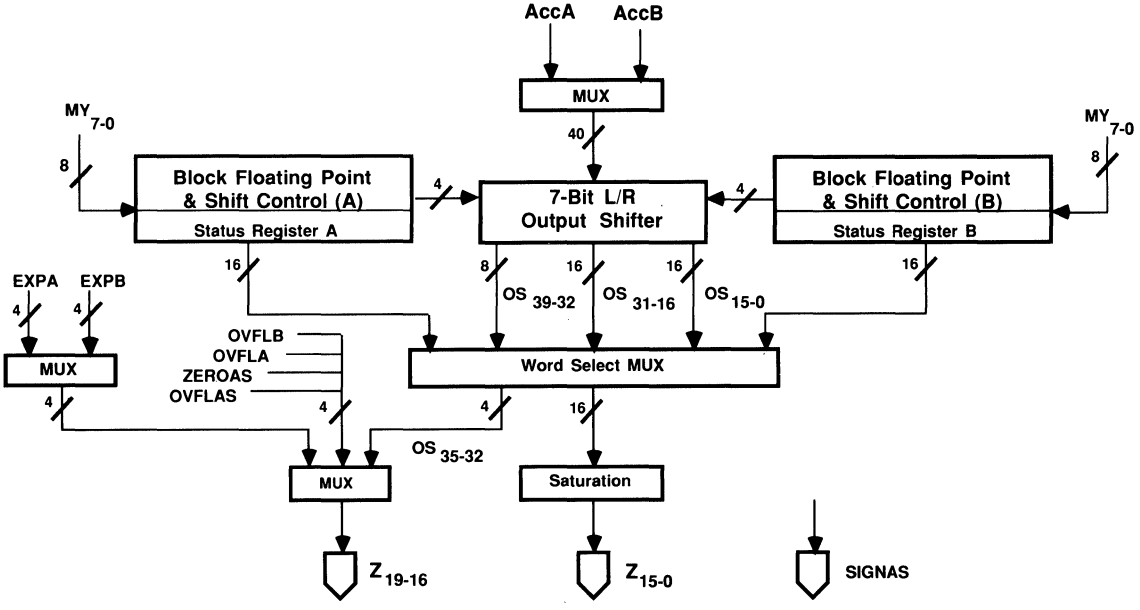


Figure 12. ADSP-1101 Shift Control, Block Floating-Point, Status Registers, Saturation, and Output Port

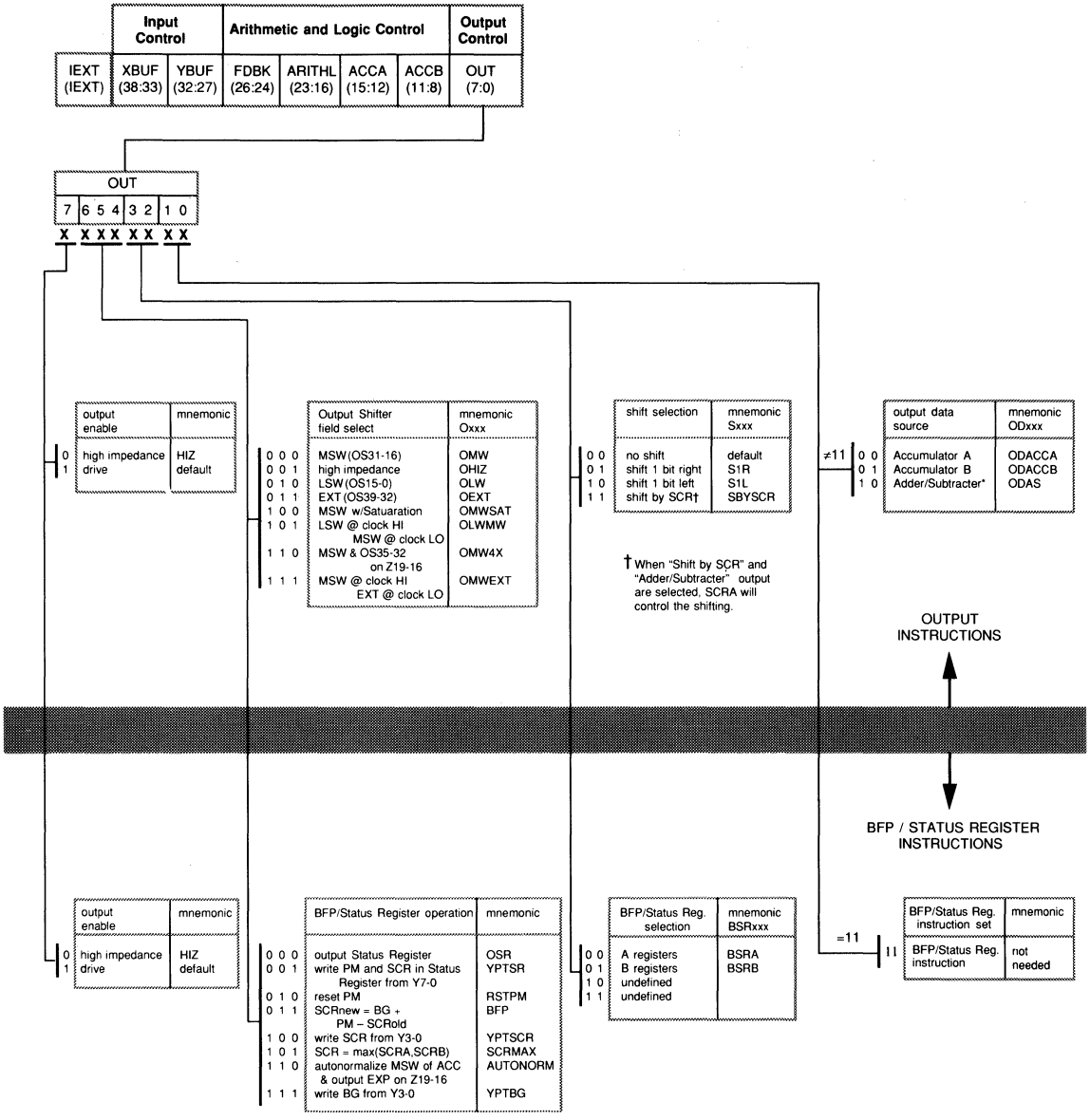


Table VIII. Output, Shift, Block Floating-Point, and Status Register Instructions

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.

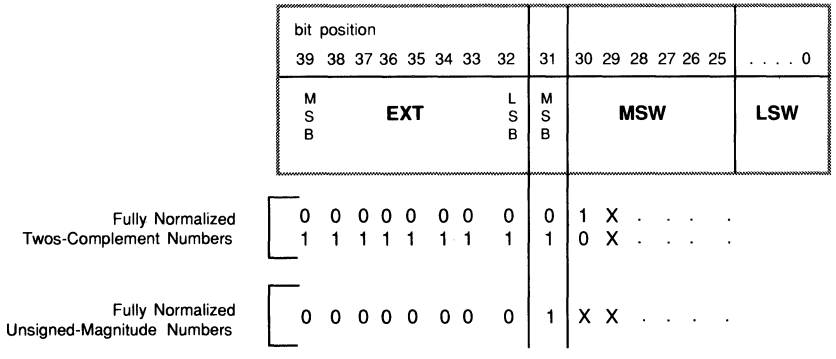


Figure 13. Fully Normalized Numbers

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.,

$$SCR \leftarrow BG + \text{Pass Magnitude of the input block.}$$

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):

$$SCR_{i+1} \leftarrow BG + (PM - SCR_i).$$

The reason this update works is that $(PM - SCR_i)$ 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:

$$SCR_{\text{normalization pass}} \leftarrow PM - SCR_{\text{final processing pass}}$$

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:

$$X_1 \leftarrow X_0 + SCRA_0$$

or in general,

$$X_{i+1} \leftarrow X_i + SCRA_i.$$

If all data in memory are kept scaled to the same block exponent, for an N-pass algorithm,

$$X_N \leftarrow X_0 + \Sigma\,(SCRA_i).$$

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

$$SCRA_{i+1} \leftarrow \max\,(SCRA_i, SCRB_i)$$

and

$$SCRB_{i+1} \leftarrow \max\,(SCRA_i, SCRB_i)$$

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 20). 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.

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 20. 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.

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
t_{FDAS}	Adder/Subtracter Flag Delay		110		100	not defined		not defined		ns
t_{FHAS}	Adder/Subtracter Flag Hold	10		10		not defined		not defined		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. Alternatively, S and T grade parts are available with optional high-reliability "PLUS" processing as shown in Figure 18.
[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 19 and 20, 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.

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/+	–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.

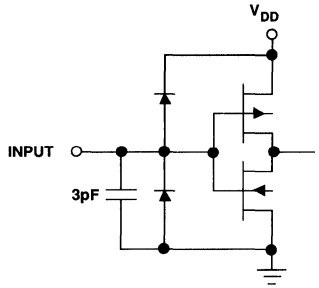


Figure 15. Equivalent Input Circuits

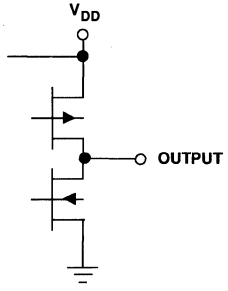


Figure 16. Equivalent Output Circuits

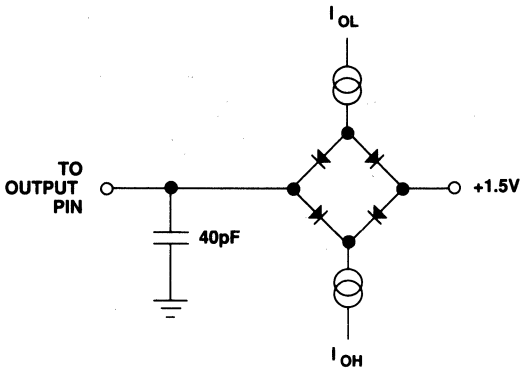


Figure 17. Normal Load for ac Measurements

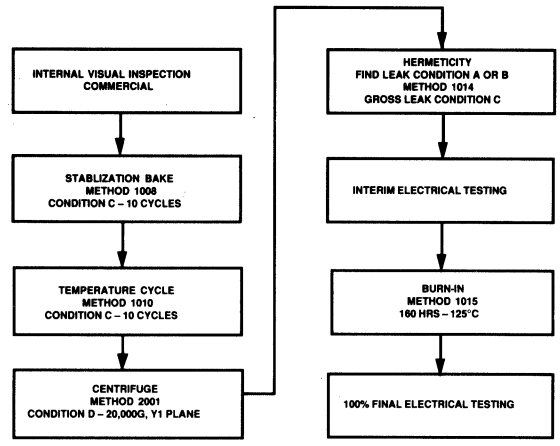


Figure 18. PLUS Processing Flow

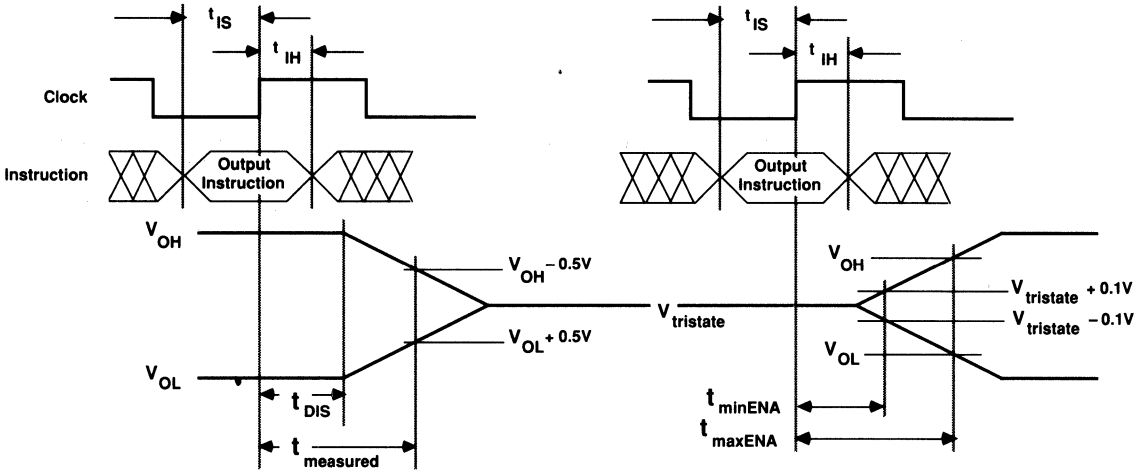


Figure 19. Output Disable and Enable Time Measurement

ESD SENSITIVITY

The ADSP-1101 features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high-energy discharges (Human Body Model) and fast, low-energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the ADSP-1101 has been classified as a Category A device.

WARNING

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.

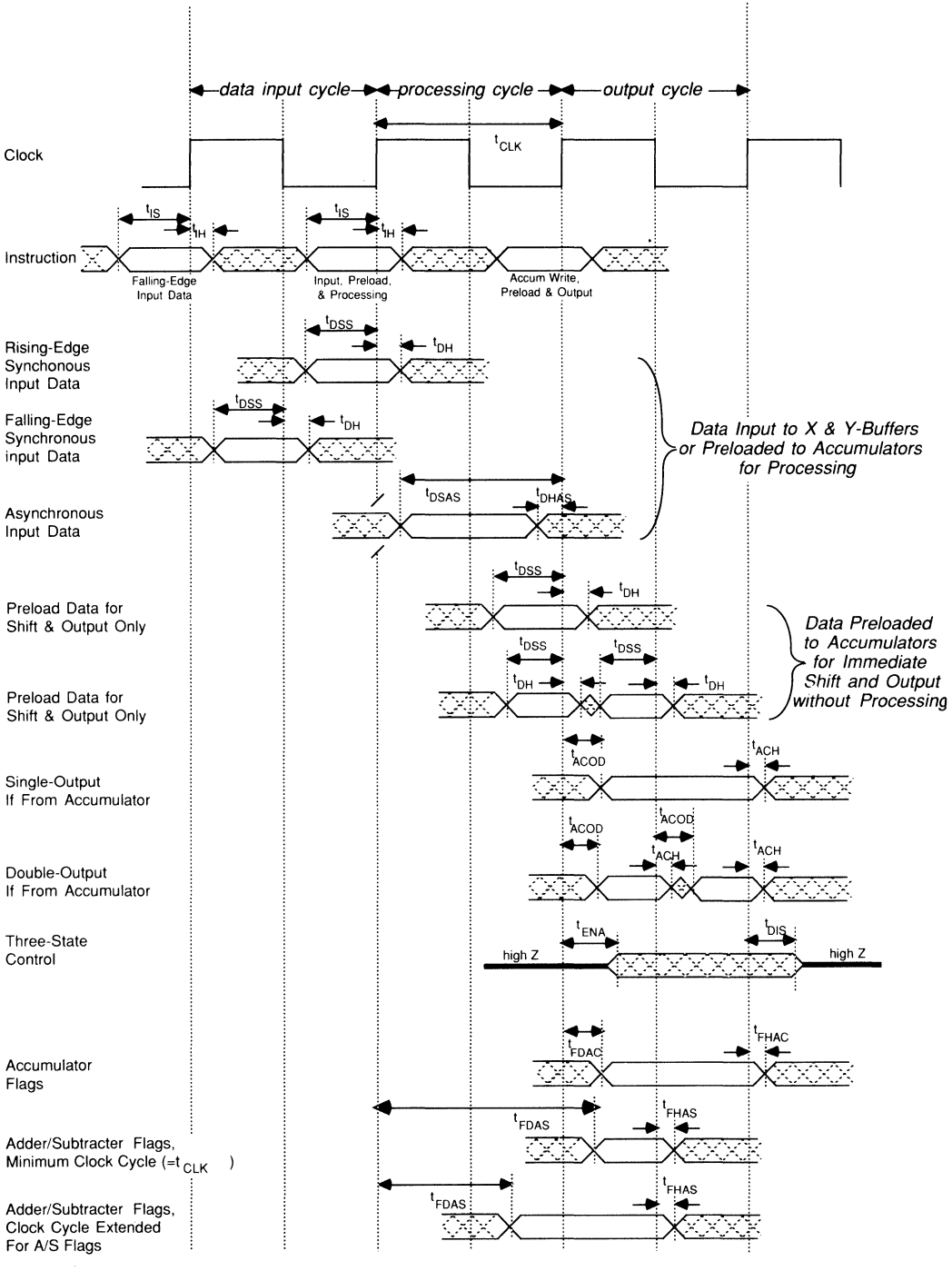
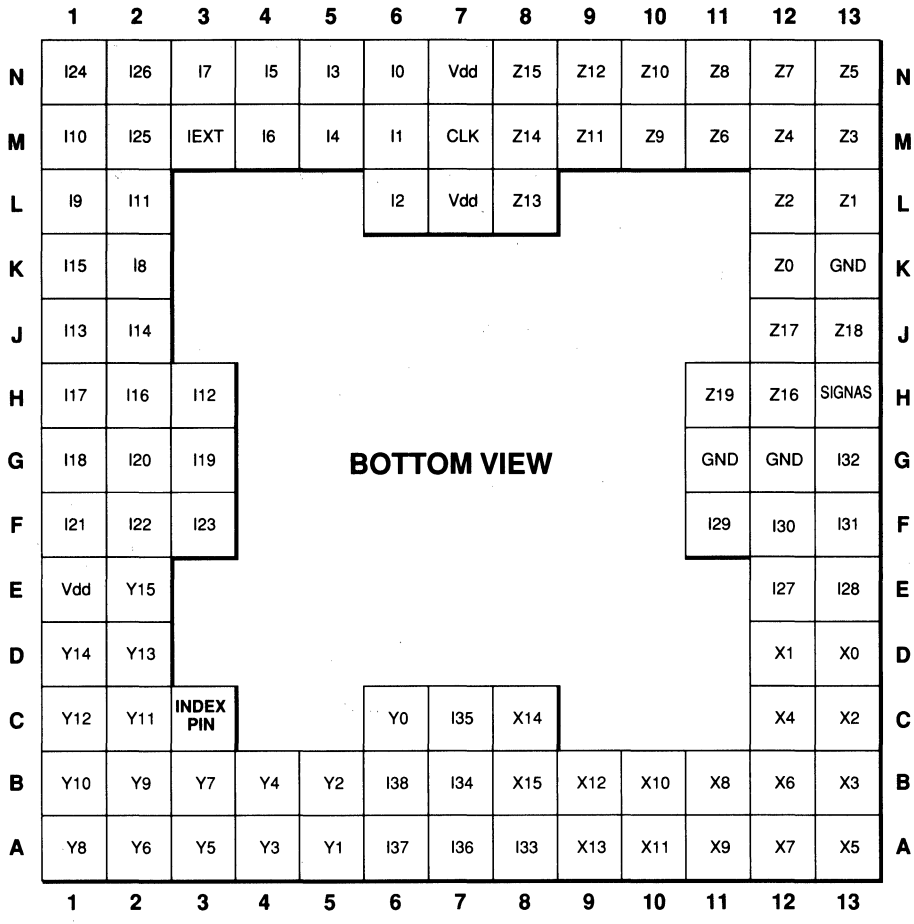


Figure 20. ADSP-1101 Timing For Synchronous Accumulator Outputs



ADSP-1101 Pin Grid Array Pinout

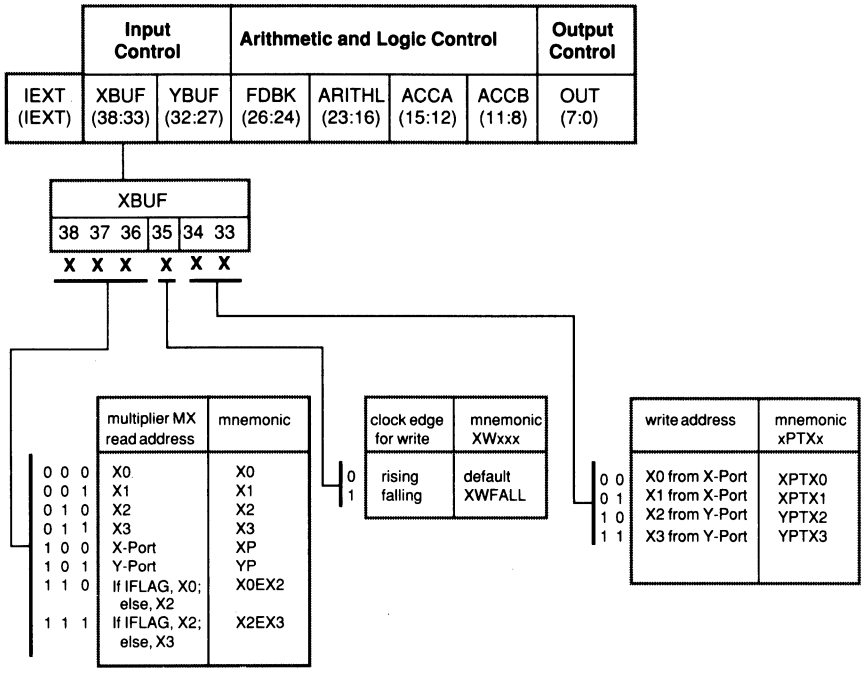


Table I. ADSP-1101 X-Buffer Instruction Set

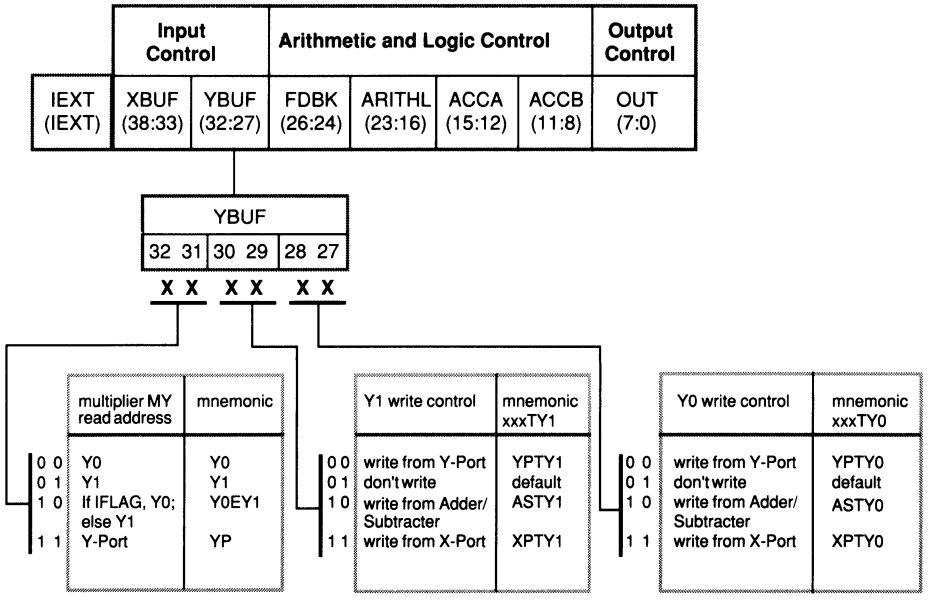
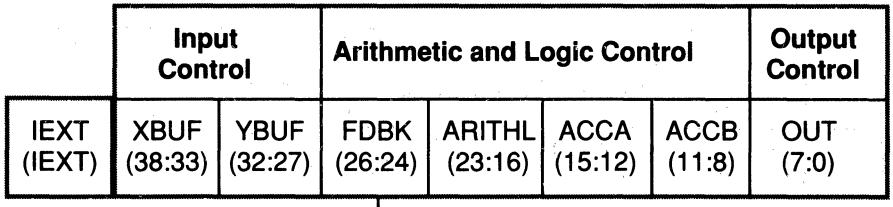
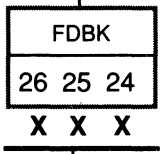


Table II. ADSP-1101 Y-Buffer Instruction Set





	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

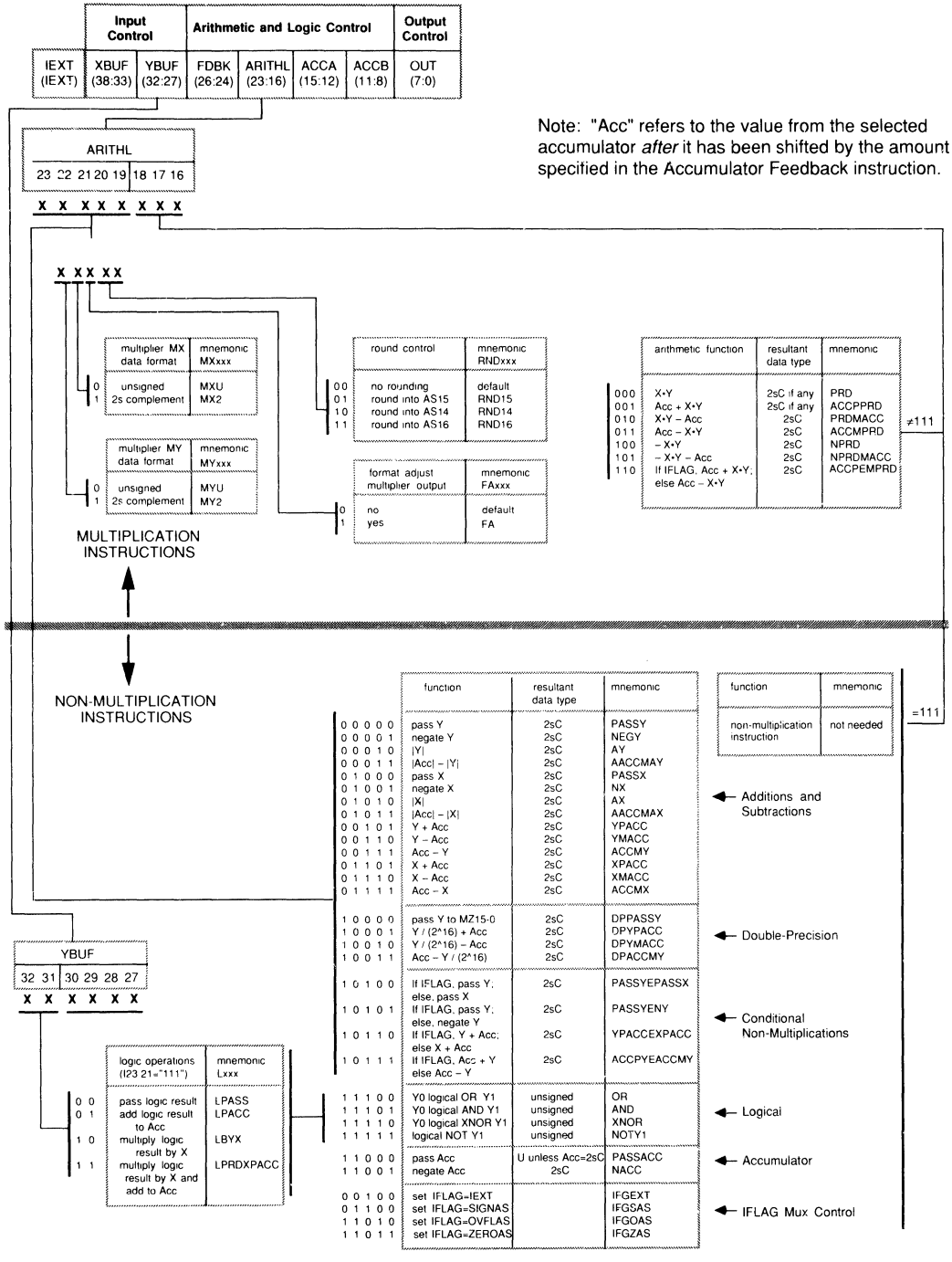
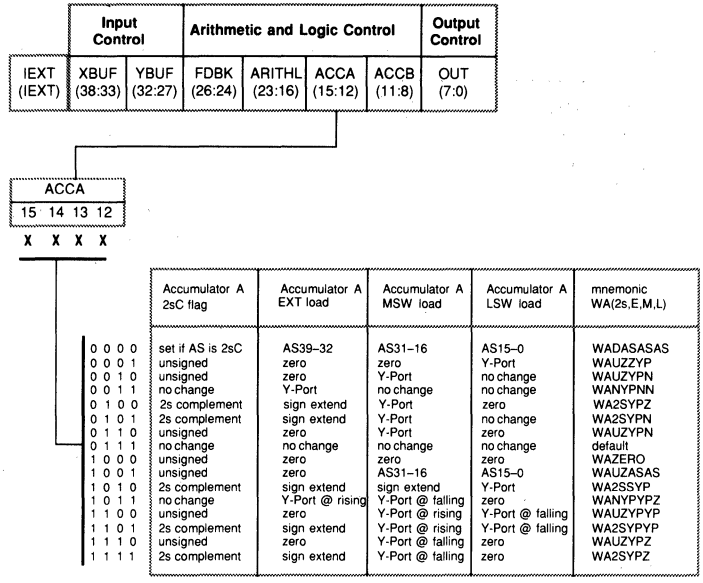
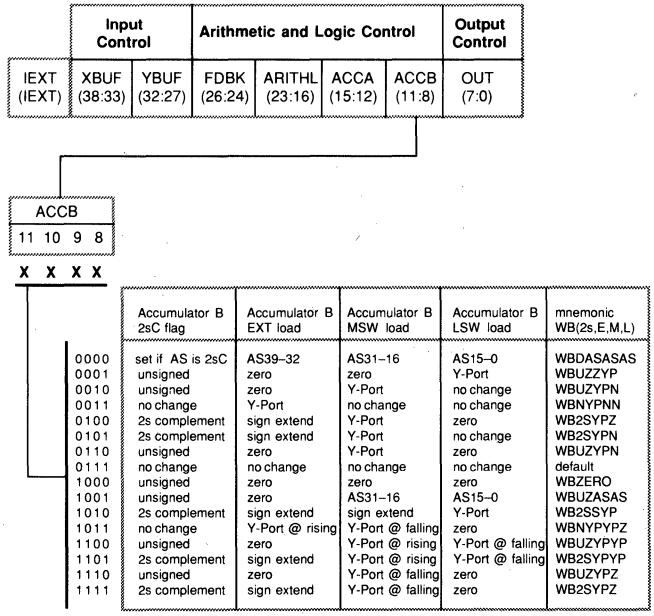


Table IV. ADSP-1101 Arithmetic and Logic Control



	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

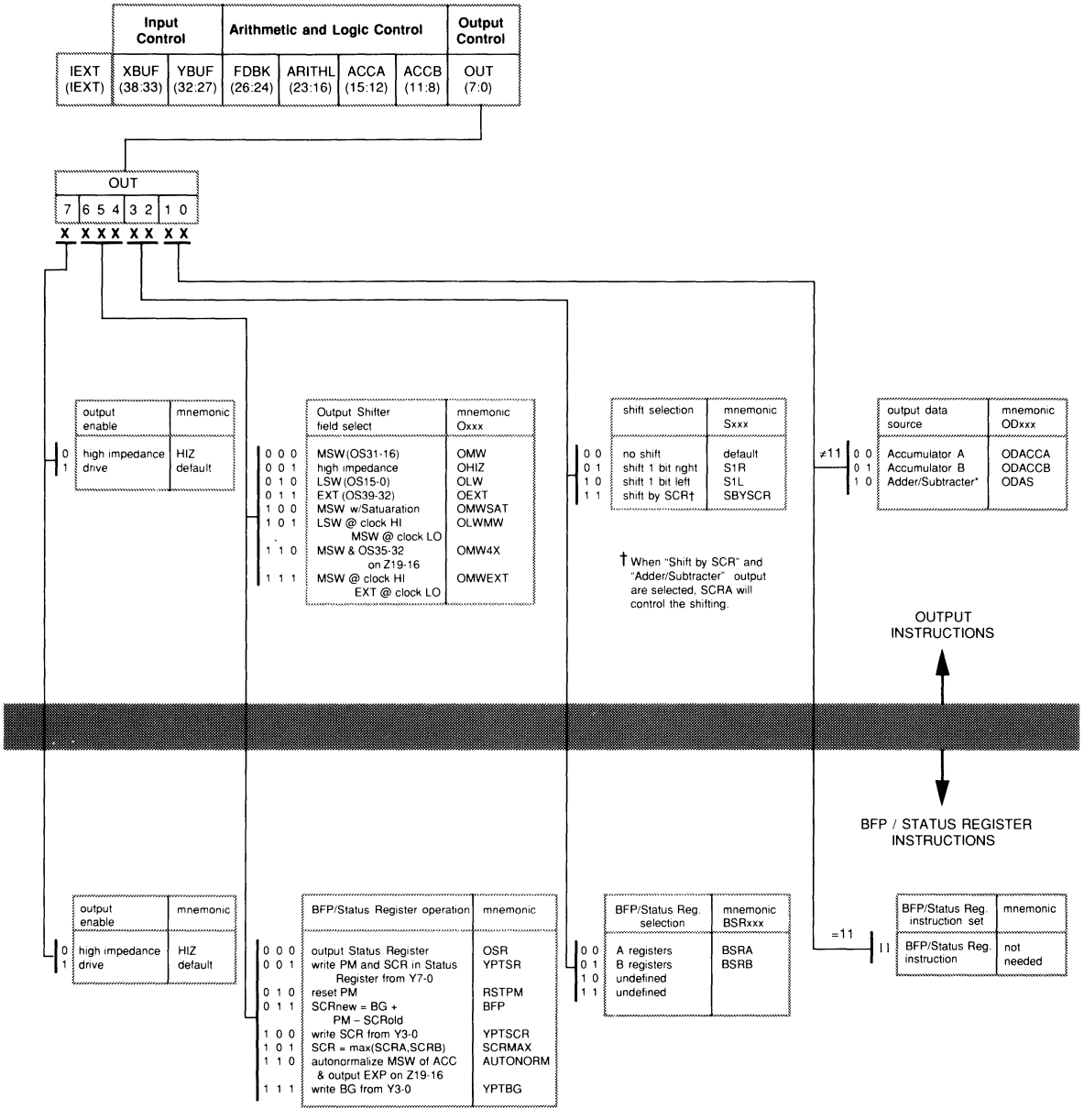
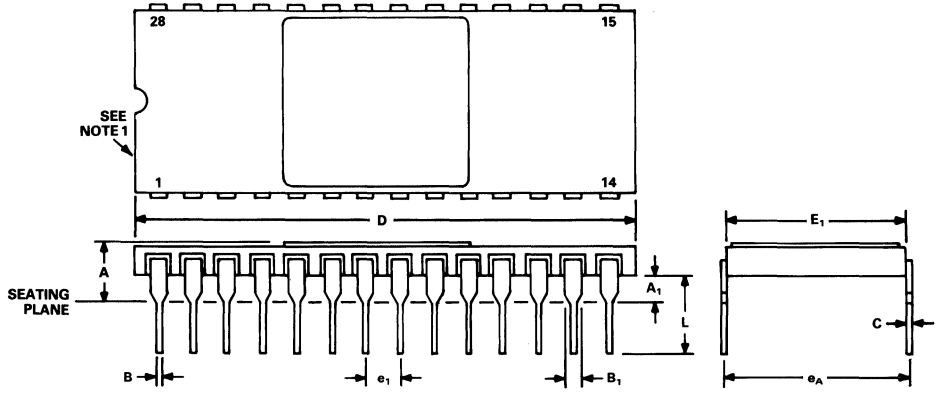


Table VIII. ADSP-1101 Output, Shift, Block Floating-Point, and Status Register Instructions

Package Information
Contents

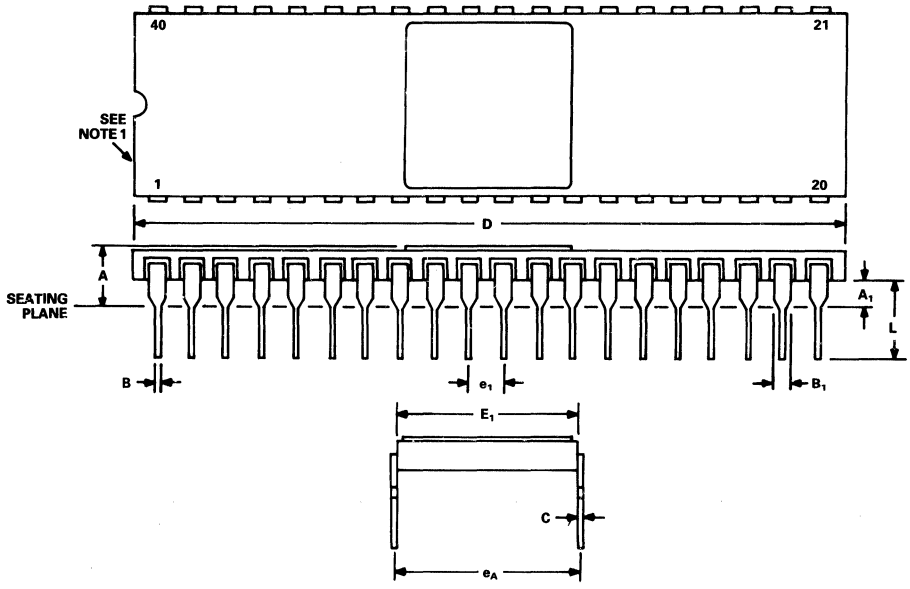
D-28
28-Pin Side Brazed



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.

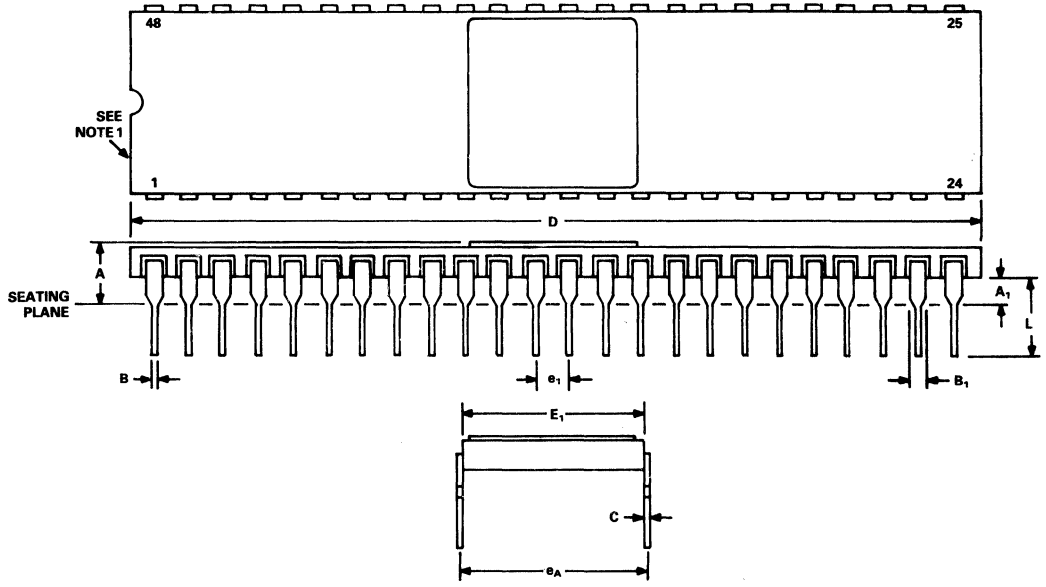
D-40A
40-Pin Side Brazed



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.

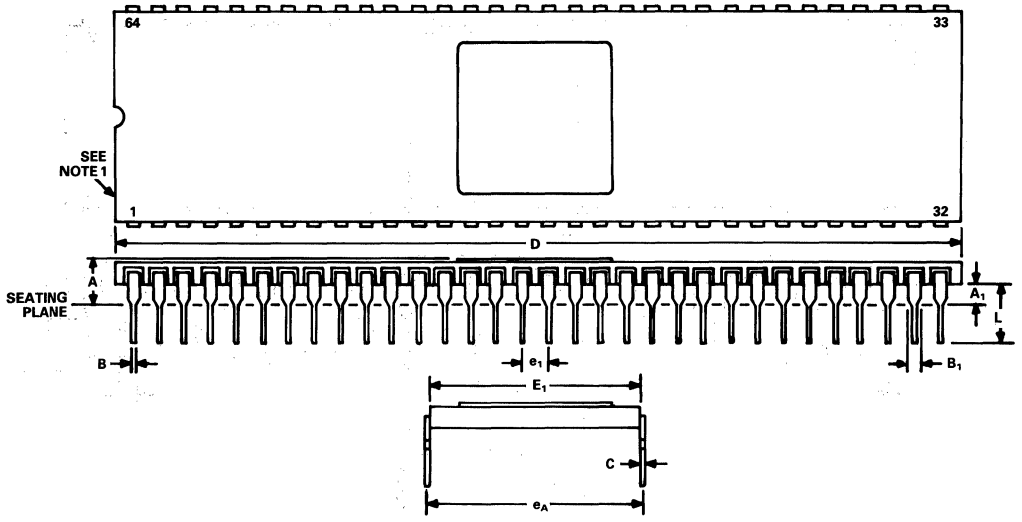
D-48A
48-Pin Side Brazed



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

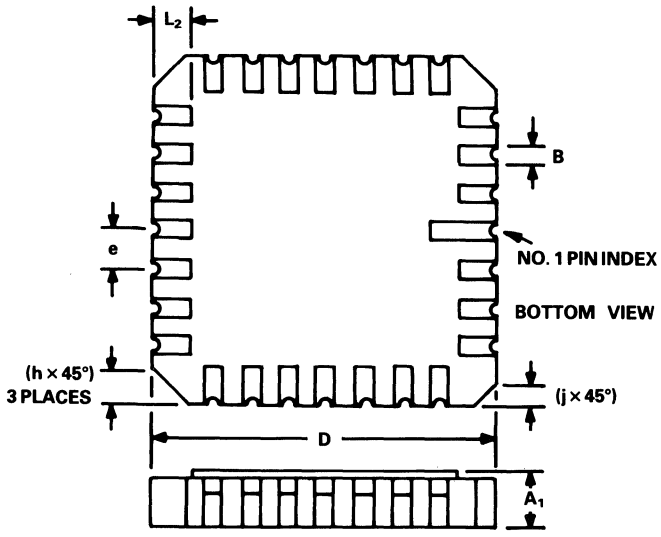


SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A		0.169		4.29	
A_1	0.040	0.060	1.02	1.52	3
B	0.016	0.020	0.41	0.51	5
B_1	0.045	0.055	1.14	1.40	2,5
C	0.009	0.012	0.23	0.30	5
D	3.170	3.230	80.52	82.04	4
E_1	0.880	0.905	22.35	22.99	4
e_A	0.900 TYP		22.86 TYP		
e_1	0.095	0.105	2.41	2.67	6
L	0.210	0.240	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-28A
28-Terminal Leadless Chip Carrier

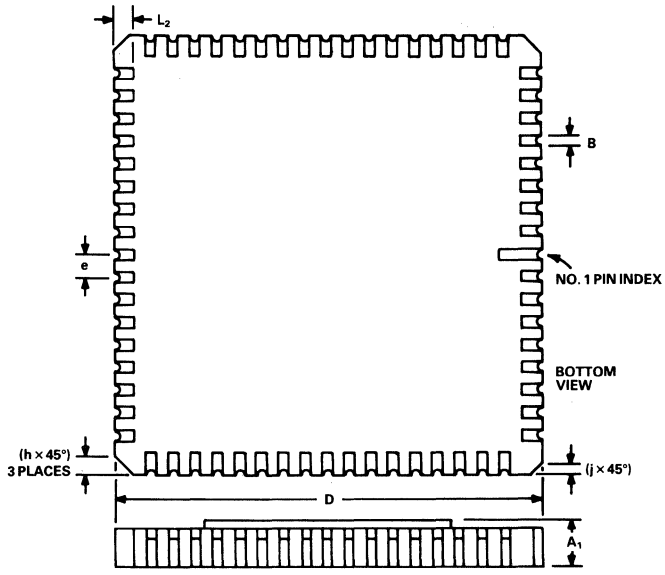


SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A_1	0.074	0.094	1.88	2.39	1
B	0.020	0.030	0.51	0.76	
D	0.442	0.458	11.23	11.63	2
e	0.045	0.055	1.14	1.40	
h	0.040 TYP		1.02 TYP		
j	0.020 TYP		0.51 TYP		
L_2	0.045	0.055	1.14	1.40	

NOTES
1. Dimension controls the overall package thickness.
2. Applies to all 4 sides.
3. All terminals are gold plated.

6

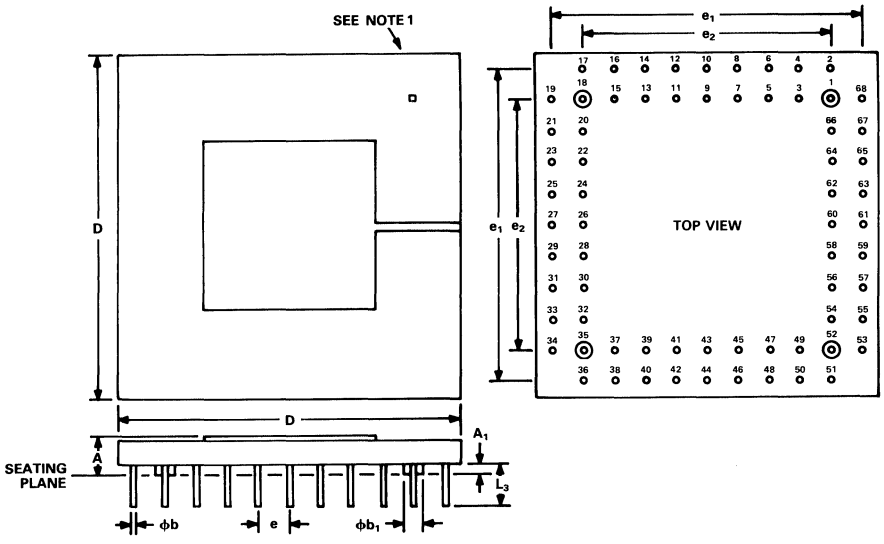
E-68A
68-Terminal Leadless Chip Carrier



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

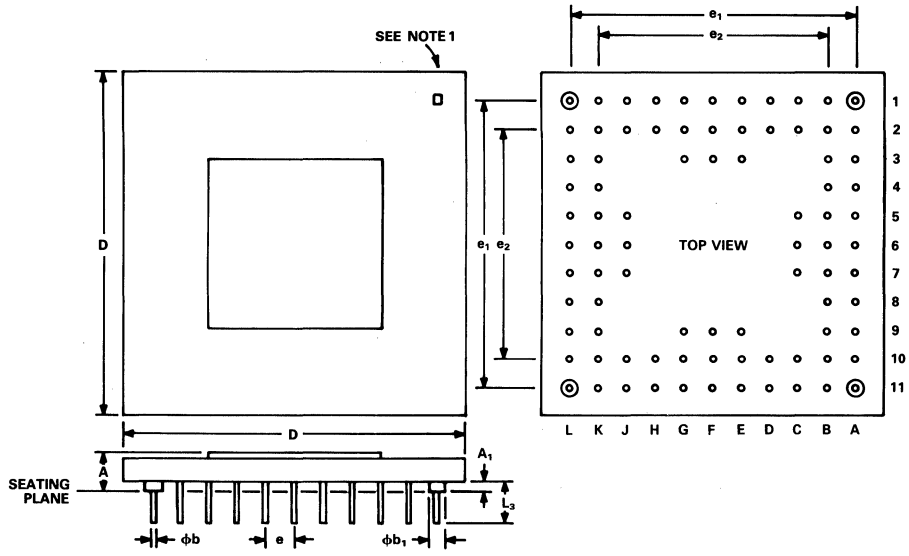


SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A	0.123	0.164	3.12	4.17	3
A_1	0.035	0.055	0.89	1.40	3
ϕb	0.016	0.021	0.41	0.53	8
ϕb_1	0.045	0.060	1.14	1.52	2, 8
D	1.080	1.110	27.43	28.19	4, 9
e_1	0.988	1.012	25.10	25.70	7
e_2	0.788	0.812	20.02	20.62	7
e	0.095	0.105	2.41	2.67	5
L_3	0.145	0.190	3.68	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.

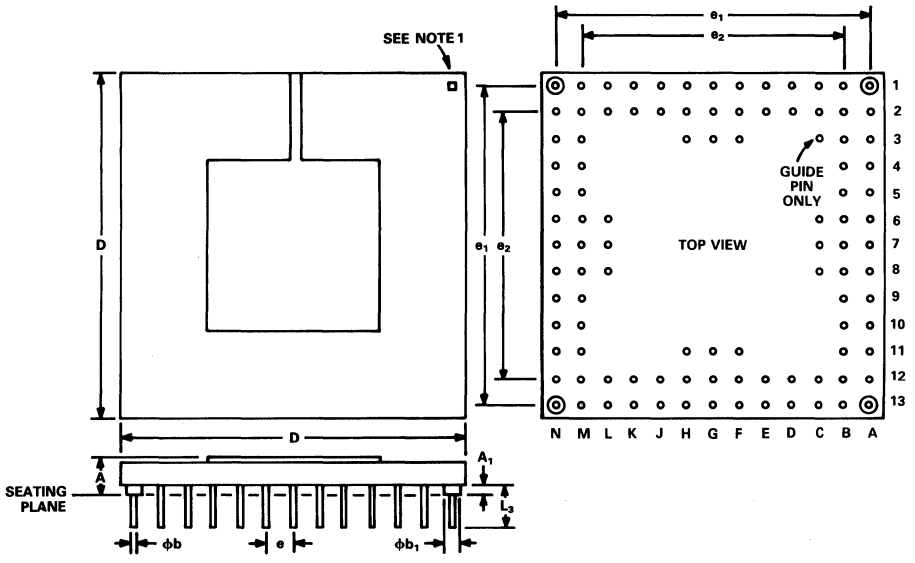
G-84A
84-Pin Grid Array



SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
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 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-100A
100-Pin Grid Array

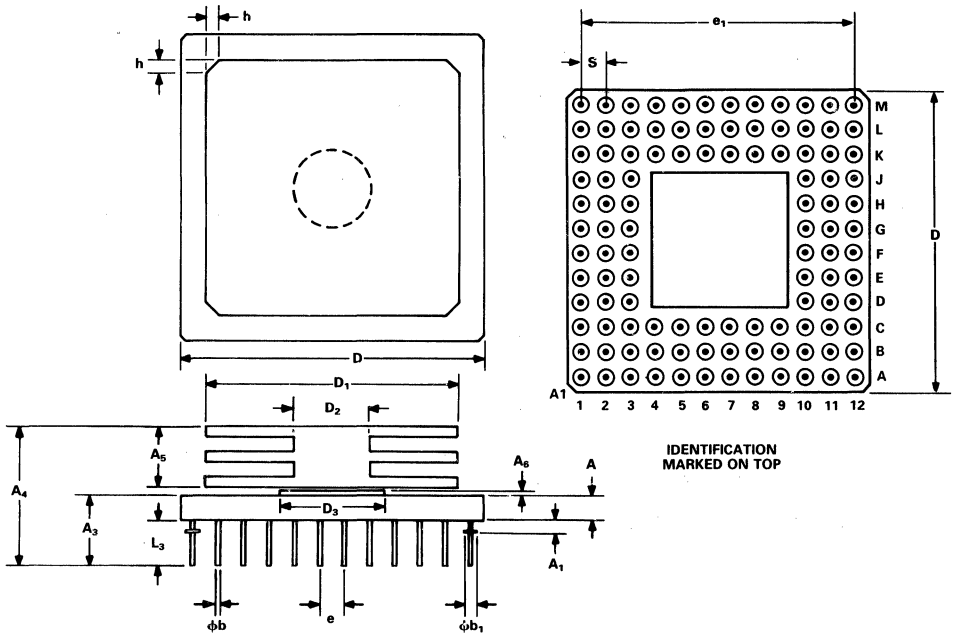


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A		0.169		4.29	3
A_1	0.025	0.055	0.64	1.40	3
ϕb	0.016	0.020	0.41	0.51	8
ϕb_1	0.040	0.055	1.02	1.40	2, 8
D	1.308	1.332	33.22	33.83	4, 9
e_1	1.188	1.212	30.18	30.78	7
e_2	0.988	1.024	25.10	26.01	7
e	0.095	0.105	2.41	2.67	5
L_3	0.165	0.190	4.19	4.83	

NOTES

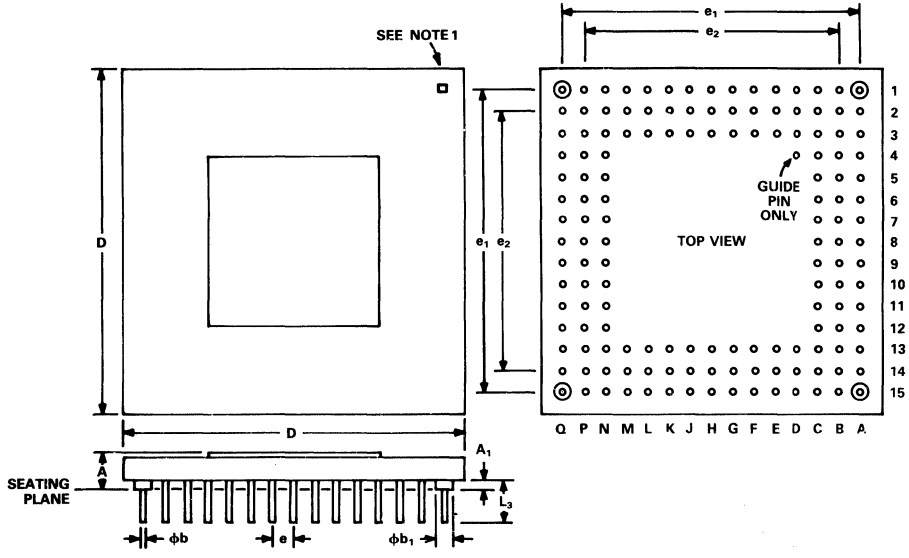
1. Index area; a notch or a lead one identification mark is located adjacent to lead one.
2. The minimum limit for dimension ϕb_1 may be 0.023" (0.58mm) for all four corner leads only.
3. Dimension shall be measured from the seating plane to the base plane.
4. This dimension allows for off-center lid, 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-108A
108-Pin Grid Array



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.090	0.110	2.29	2.79
A_1	0.045	0.055	1.14	1.40
A_3	0.275	0.285	6.99	7.24
A_4	0.520	0.560	13.21	14.22
A_5	0.240 TYP		6.10 TYP	
A_6	0.018 TYP		0.46 TYP	
ϕb	0.016	0.020	0.41	0.51
ϕb_1	0.045	0.055	1.14	1.40
D	1.200 TYP		30.48 TYP	
D_1	1.000 TYP		25.40 TYP	
D_2	0.300 TYP		7.62 TYP	
D_3	0.360 TYP		9.14 TYP	
e_1	1.090	1.110	27.69	28.19
e	0.095	0.105	2.41	2.67
h	0.045	0.055	1.14	1.40
L_3	0.180 TYP		4.57 TYP	
S	0.095	0.105	2.41	2.67

G-144
144-Pin Grid Array

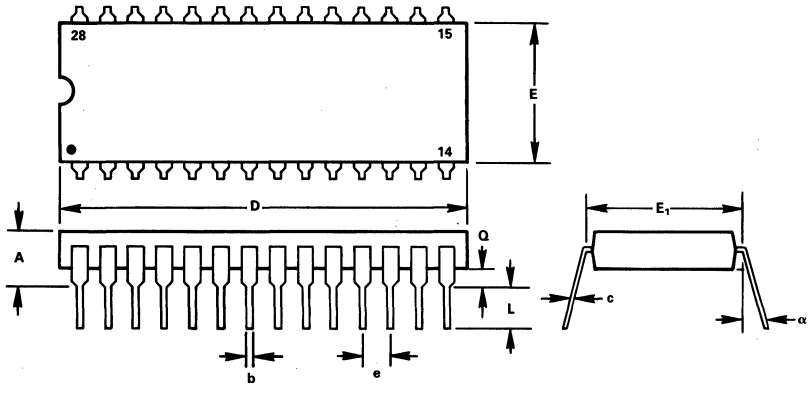


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX	NOTES
A	0.132	0.165	3.35	4.19	3
A_1	0.045	0.055	1.14	1.40	3
ϕb	0.016	0.020	0.41	0.51	8
ϕb_1	0.045	0.055	1.14	1.40	2,8
D	1.559	1.591	39.60	40.41	4,9
e_1	1.388	1.412	35.26	35.86	7
e_2	1.188	1.212	30.18	30.78	7
e	0.095	0.105	2.41	2.67	5
L_3	0.175	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.

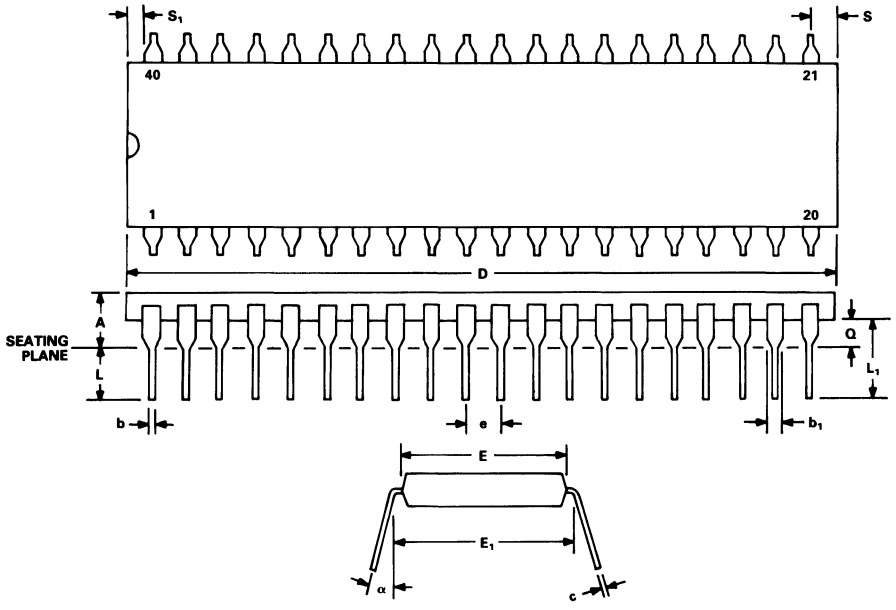
N-28A
28-Pin Plastic DIP



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.

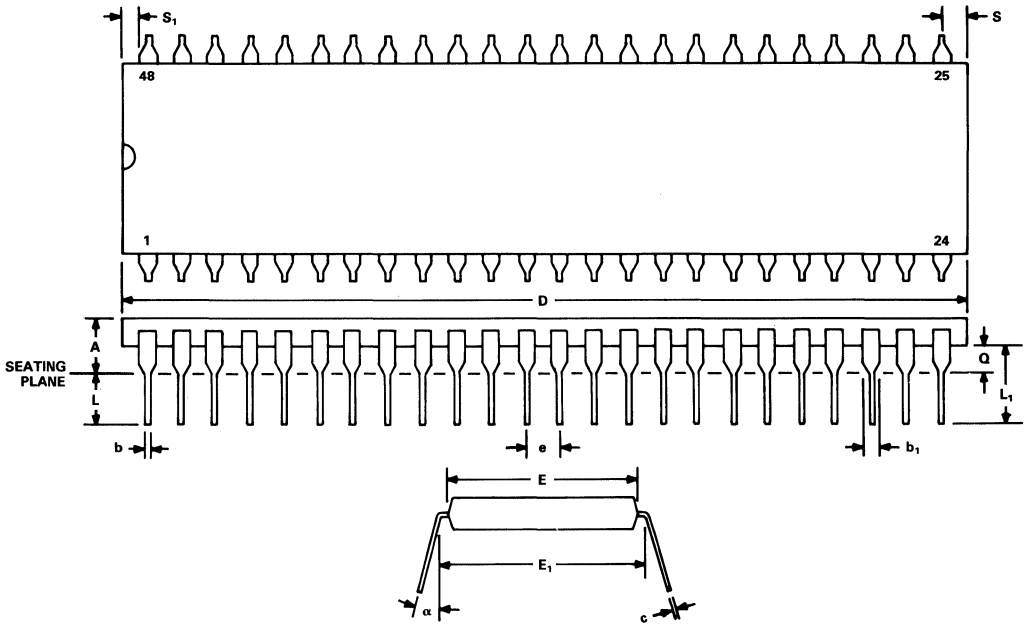
N-40A
40-Pin Plastic DIP



NOTE:
LEADS ARE SOLDER-PLATED KOVAR OR ALLOY 42

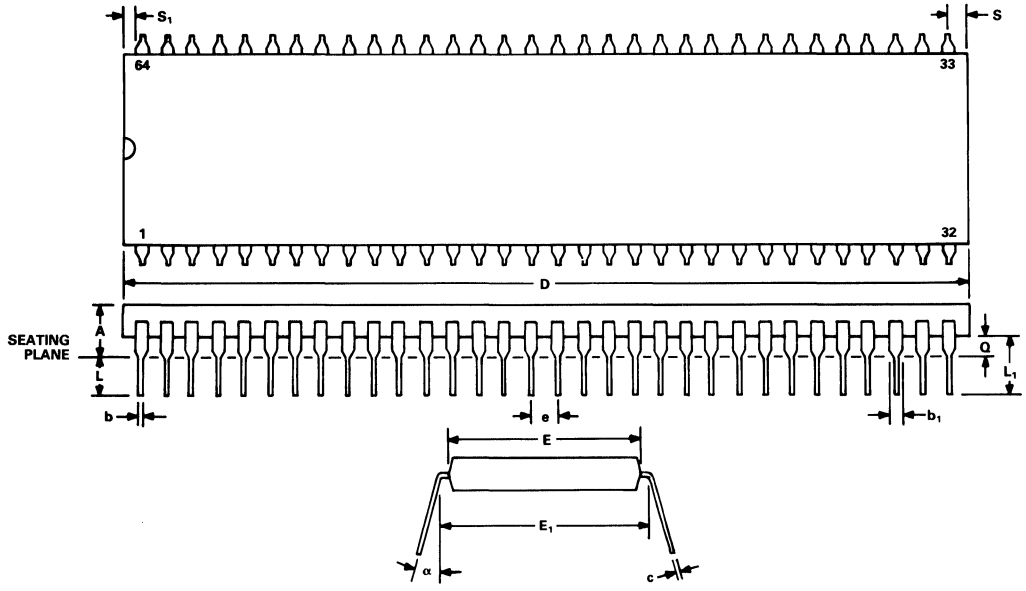
SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	–	0.200	–	5.08
b	0.015	0.025	0.38	0.64
b_1	0.040	0.060	1.02	1.52
c	0.008	0.015	0.20	0.38
D	–	2.08	–	52.83
E	0.550	0.550	13.46	13.97
E_1	0.580	0.620	14.73	15.75
e	0.100 BSC		2.54 BSC	
L	0.120	0.175	3.05	4.45
L_1	0.140	–	3.56	–
Q	0.015	0.060	0.38	1.52
S	–	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°

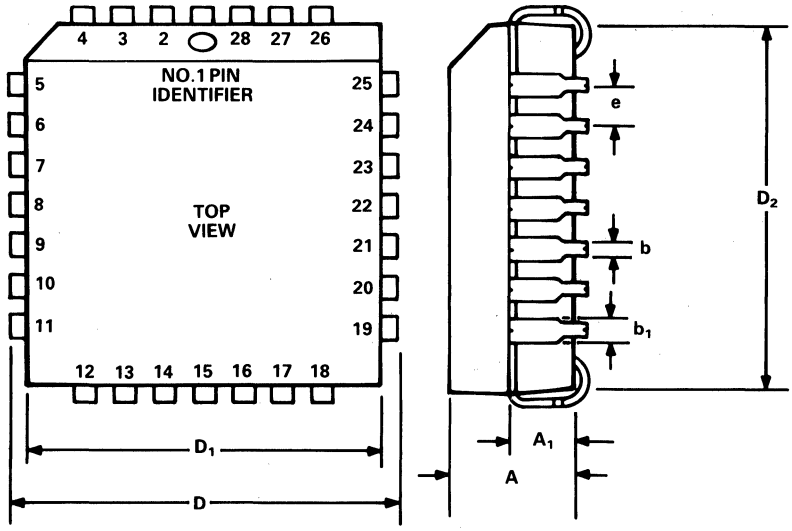
N-64A
64-Pin Plastic DIP



NOTE:
LEADS ARE SOLDER-PLATED KOVAR OR ALOY 42

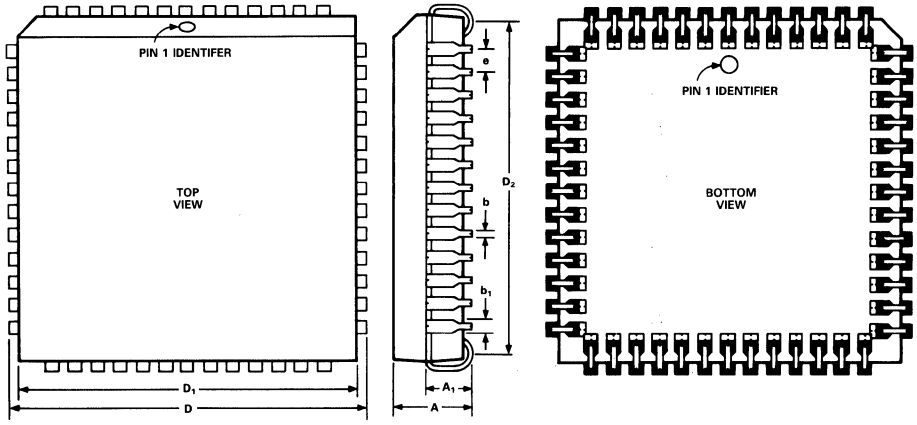
SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A		0.250		6.35
b	0.015	0.025	0.38	0.64
b_1	0.030	0.070	0.76	1.78
c	0.008	0.015	0.20	0.38
D		3.25		82.55
E	0.790	0.810	20.07	20.57
E_1	0.880	0.920	22.35	23.37
e	0.100 BSC		2.54 BSC	
L	0.125	0.200	3.18	5.08
L_1	0.150		3.81	
Q	0.015	0.060	0.38	1.52
S		0.098		2.49
S_1	0.005		0.13	–
α	0°	15°	0°	15°

P-28
28-Lead Plastic Leaded Chip Carrier



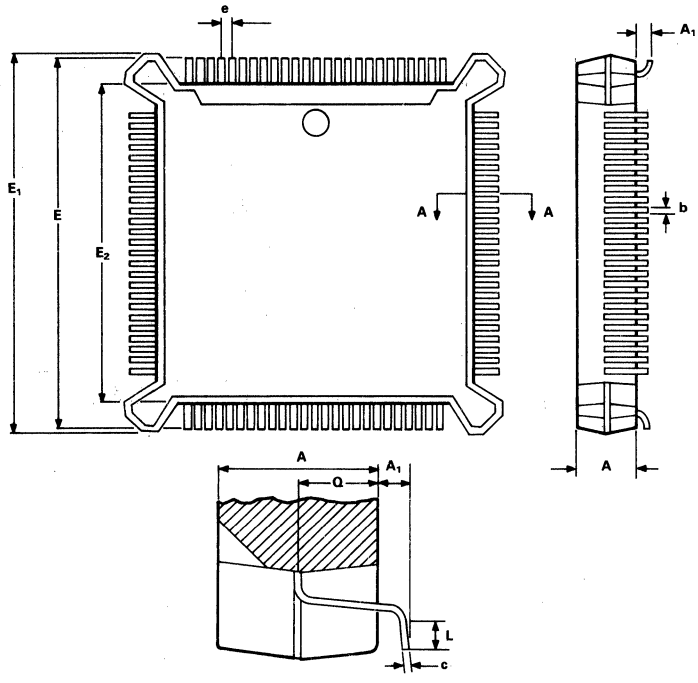
SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.165	0.180	4.19	4.57
A_1	0.090	0.120	2.29	3.05
b	0.013	0.021	0.33	0.53
b_1	0.026	0.032	0.66	0.81
D	0.485	0.495	12.32	12.57
D_1	0.450	0.456	11.43	11.58
D_2	0.390	0.430	9.91	10.92
e	0.045	0.055	1.14	1.40

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-100
100-Lead Plastic Leaded Chip Carrier



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

Application Notes
Contents

Introduction

This chapter contains eight application notes representative of the applications support provided by the Digital Signal Processing Division. Additional application notes are always being generated by Applications Engineering. For example, an application note on prototyping an ADSP-3212/22 system with the ADSP-3211/21 is in draft at press time. The complete list of current applications literature is printed in the divisional newsletter, *DSPatch*. Consult your Analog Devices Sales Office for more information.

These application notes focus on the microcode and numeric parts. Applications information for the ADSP-2100 DSP microprocessor has been published in a separate volume; the *ADSP-2100 Applications Handbook, Volume 1*, is 180 pages of discussion and subroutines for the ADSP-2100. Here is a brief summary of the material presented in the chapters of that book.

1. *Fixed-Point Arithmetic*
   Describes how basic fixed-point arithmetic operations are implemented in the hardware of the ADSP-2100.
2. *Floating-Point Arithmetic*
   Describes how to convert from fixed-point to floating-point and back and how to perform basic floating-point operations on the processor. Block floating-point is described under FFTs.
3. *Function Approximation*
   How to approximate several useful functions, such as sine and arctangent and a random number generator.
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-reverse 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.
8. *Linear Predictive Speech Coding*
   Techniques used to analyse, encode and synthesize speech signals.
9. *High-Speed Modem Algorithms*
   Describes several algorithms (stochastic gradient, etc.) used in implementing a high-speed modem with the ADSP-2100.

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.

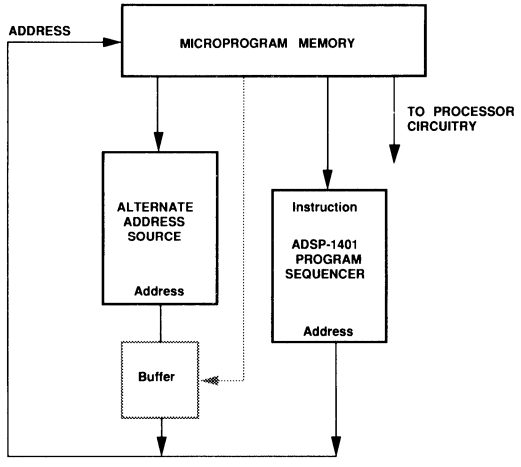


Figure 1. Basic Bus Sharing Architecture

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.)

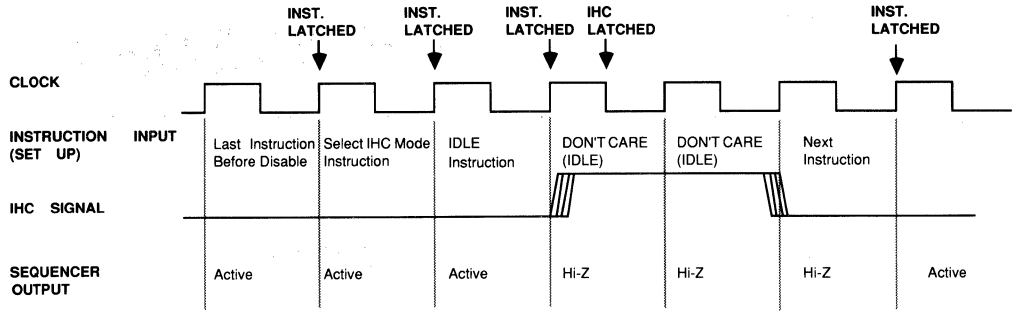


Figure 2. IHC Mode - IDLE Instruction Timing

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.

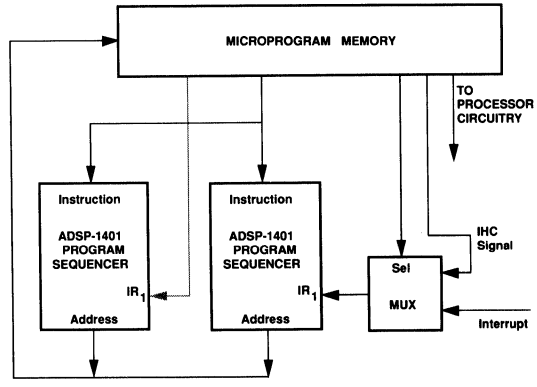


Figure 3. Block Diagram with IHC Connections

Stack Paging Expands Internal Ram of the ADSP-1401 Program Sequencer

by Matthew J. Johnson

INTRODUCTION

This application shows how the subroutine stack of the ADSP-1401 Program Sequencer may be expanded by employing a data memory stack paging technique. Although the bounds of the sequencer's subroutine stack are user configurable for larger or smaller stack space, the internal RAM must also support the global stack, the register stack, and indirect jump address storage. In context of complex or highly nested subroutines, allocation of sufficient subroutine stack space may become difficult, hence the need for stack expansion.

The implementation of subroutine stack paging is actually rather trivial (in principal) by virtue of a reserved stack recovery interrupt, IR_9, built into the sequencer especially for this purpose. This interrupt is a reserved system interrupt issued as a warning of stack overflow or underflow. Upon generation of IR_9, the associated interrupt service routine will page subroutine stack values between data memory and internal sequencer RAM, the direction depending upon whether the subroutine stack has overflowed or underflowed. This application note explains the programming and hardware considerations associated with implementing a stack recovery interrupt routine.

We start with a general overview of the program sequencer and a discussion of the constraints and implications of stack paging. A complete discussion of a sample subroutine stack paging interrupt routine is presented in the main text with supporting flowchart, listings and benchmarks in the appendix. Operation of the routine is illustrated using "snapshots" of the internal RAM and data memory paging areas taken at various points in time. As a convention, all numbers are given in decimal throughout this application note, unless otherwise noted.

GENERAL DESCRIPTION OF THE ADSP-1401

The ADSP-1401 is a high-speed microprogram controller optimized for the demanding tasks found in digital signal processors and general purpose computers (see Figure 1). It provides both high speed (25ns clock-to-address delay) and a large microcode address space (64K of microcode). In addition, it has various unique features including:

- On-chip control of ten prioritized and maskable interrupts
- Four decrementing event/loop counters
- Absolute, relative and indirect addressing modes
- Microcode download capability (writeable control store)
- Dynamically configurable internal 64x16 RAM.

During each microinstruction, the ADSP-1401 uses internal and external conditions and instructions to determine the next microprogram address. This address can come from either the stack, the jump address space in the RAM, the data port, the internal interrupt vector file or the microprogram counter. An extensive set of conditional instructions is available including jumps, branches, subroutines, interrupts, and writeable control store.

The ADSP-1401's internal 64-word by 16-bit RAM is used (among other things) for interrupt and subroutine linkage via the stack. It also provides for storage of internal and external system parameters in addition to providing the ability to associate counters, jump addresses, and the status register with various levels interrupts and subroutines.

Interrupts are handled real-time entirely on chip. The ADSP-1401's internal interrupt control logic incluides dedicated registers for storage of eight external (user) interrupt vectors, a 10-bit interrupt mask, and an interrupt priority encoder. Two additional vectors are reserved for processing

internally generated interrupts resulting from counter underflow and stack limit violation. The reserved interrupts are also maskable.

The ADSP-1401's four decrementing counters are used to track loops and events. These counters are referenced by several conditional instructions and can also trigger an internal interrupt.

The ADSP-1401's Look-Ahead pipeline eliminates the need for an external microcode pipeline register by internally latching instructions and addresses. A complementary latching arrangement allows the next instruction to be predecoded while the current address is held constant. In addition to eliminating the external microcode latch, this technique also allows for use of slower microcode memory for other system components (see the section *The Target System Hardware*).

See the ADSP-1401 Program Sequencer data sheet for complete specifications.

HOW THE STACKS ARE USED

In normal operation (when not stack paging), the internal 64-word RAM of the ADSP-1401 is configured to support two stacks (the subroutine stack and the register stack) and indirect jump addresses storage. The Subroutine Stack (SS) has a dedicated Subroutine Stack Pointer (SSP), while the Register Stack (RS) shares two pointers: the Local Stack Pointer (LSP) and the Global Stack Pointer (GSP), only one of which is active at a time. All stack pointers are user loadable.

The subroutine stack is used to temporarily save internal sequencer states such as counter values, the status register, return addresses and also external device data (as in context saves) in preparation for execution of subroutines or interrupts or simply as a convenient way station for transient data. Immediately after calls to interrupts or subroutines, the internal and external states which will be altered during execution of the routine are saved by the user (pushed on the subroutine stack) prior to modification. In this way, contextual parameters (which are later restored by

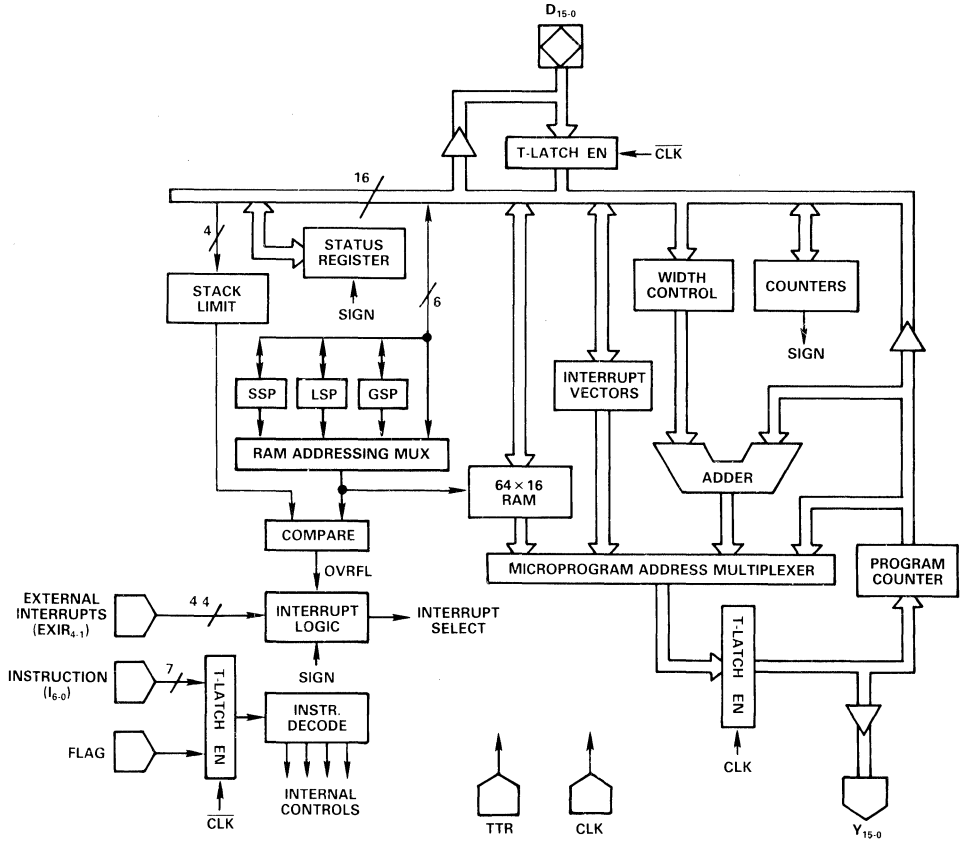


Figure 1. ADSP-1401 Program Sequencer Block Diagram

popping the stack upon return) are preserved during execution of the routine.

The register stack is used similarly to associate unique (local stack) or general (global stack) jump address sets with specific code segments. Selection of the active RS pointer (LSP or GSP) is made in the status register. The two register stack pointers are actually 6-bit *base* pointers for register-based instructions. The instruction provides a 2-bit *offset* to the RS base pointer as part of the instruction opcode. This allows selection among four registers for each register-based instruction.

Use of phrases such as *above vs. below, up vs. down* or *advance vs. retard* are all dependant upon a directional reference for their meaning. Throughout this application note, use of phrases such as these is made with the understanding that low memory is "above" high memory; memory addresses increase from top to bottom and advancing a memory pointer moves it downward. In contrast, nomenclature to be introduced in the *Principals and Precautions of Subroutine Stack Paging* defines concepts such as "bottom of stack" and "top of stack" as being located at the logical top and bottom of the sequencer's subroutine stack area, respectively. Be careful in thinking about the difference between *top of memory vs. top of stack* and *bottom of memory vs. bottom of stack*; literal memory addresses and logical subroutine stack boundaries have opposite organizations.

The two stacks (subroutine *vs.* register) migrate in opposite directions (see Figure 2); the SSP pushes in a "downward" direction of increasing addresses (a push increments the pointer), while the Register Stack Pointer (RSP) pushes in an "upward" direction of decreasing addresses (a push decrements the pointer).

The SSP is initialized to location zero upon reset, allowing the SS to grow downward. The RS pointers are initialized manually: typically, the LSP is placed above the GSP, allowing the local stack to grow upwards as the level of subroutine nesting increases. Indirect jump address space is reserved below the global stack (GS) as needed.

HOW THE STACK LIMIT REGISTER IS USED

Given the opposition in stack migration (SS *vs.* RS), it becomes obvious that these two stacks will eventually collide if enough pushes to either are made. To alert the user of impending stack *overflow* prior to stack collision, a Stack Limit Register (SLR) is provided which constantly compares the relevant stack pointer (be it a SS or RS operation) with a user initialized SLR value. At such time as either stack pointer violates the SLR value, the highest priority interrupt (IR_9) is generated which gives the user the opportunity to resolve the situation.

The SLR is 4-bits wide and left justified in the 6-bit RAM address field: the SLR may only be set to integer multiples of 2^2, i.e., RAM locations 0, 4, 8, 12, ..., 60. Comparisons of the 6-bit internal RAM address with the 4-bit SLR require

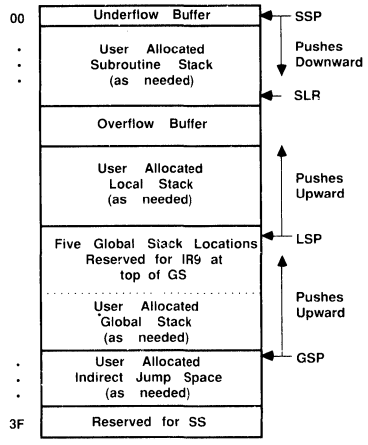


Figure 2. Typical Sequencer RAM Initialization for Subroutine Stack Paging

that the missing two lower bits of the SLR be filled appropriately so as to provide advance warning of *impending* stack collision. These two bits are filled with '00' if a SS push is made, or with '11' if a RS push is made. The net effect is that the stack overflow interrupt is issued whenever the SSP increments to the SLR or the RSP decrements to the SLR+3, allowing stack recovery *before* a collision occurs.

The three-location buffer between the SLR and the RSP allows for up to three possible stack pushes (either SS or RS) which may occur prior to entering the stack recovery interrupt routine:

- The push causing the initial overflow
- A possible push while IV_9 is output (depending on the interrupted instruction)
- The IR_9 return address push to the SS.

Subroutine stack *underflow*, on the other hand, will also generate IR_9 whenever SS location zero is popped.

In review, the three separate occurrences which will cause an IR_9 interrupt are:

- A push causes the SSP to increment to the value held in the SLR (SS overflow)
- A pop from SS location zero (SS underflow)
- A push causes the RSP to decrement to the value held in the SLR+3 (RS overflow).

Although the sequencer detects RS overflows, there is no mechanism for detecting RS underflows. Therefore, independent paging of the RS is almost impossible from a practical standpoint. Imposing a linkage between the RS and the SS would allow paging both stacks whenever the SS overflows or underflows, but this approach compromises performance and is very difficult in implementation. Such a

solution to RS paging is beyond the scope of this application note however, note that the SS can be made small compared to the RS to eliminate RS overflow.

PRINCIPALS AND PRECAUTIONS OF SUBROUTINE STACK PAGING

The principal of subroutine stack paging is to limit the size of the sequencer's internal stack by "shuffling" data back and forth between the stack and data memory. The shuffling is managed in such a way as to keep only the most current data "page" handy while also insuring sufficient room is always available for additional pushes; old SS data is stored away in data memory until needed.

General Discussion

When the internal SS grows too large, we have an *overflow*. The entire stack is copied to data memory and only the most recent data is written back to the upper area of the

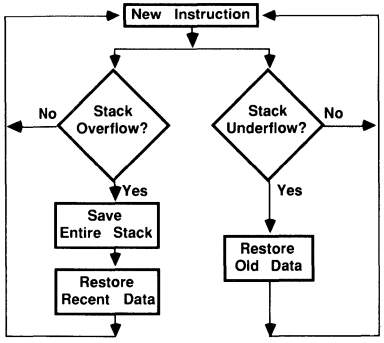


Figure 3. Basic Stack Paging Flow Chart

sequencer's RAM, leaving a block of space for additional subroutine stack pushes. The size of this block is equivalent to the net amount of data removed from the stack and left in data memory by the interrupt routine. Typically, the block size is set to about half of that allocated to the entire subroutine stack to equally distribute the likelihood of subsequent operations being pops as opposed to additional pushes.

As data already residing in the sequencer's subroutine stack is used up (popped from the SS), eventually the stack is emptied and some of the old data which was paged out to data memory earlier must be brought back. This is the *underflow* scenario and again, only half as much data as could be fit in the SS is actually returned.

The flowchart of Figure 3 illustrates these rudimentary operations. As each new stack push or pop instruction is executed, the question is posed: Has a stack overflow or underflow occurred? If so, data is shuffled between the SS and data memory so as to assure a sufficient record of recent data is always resident within the sequencer for further popping while also maintaining sufficient room for further pushing.

Note that in principal, a stack overflow or underflow interrupt represents a *Catch-22* situation in that any further processing may well "blow the stack" and yet, saving the stack requires rather extensive processing. In practice, the necessary registers needed to effect the interrupt processing are freed by saving them in preallocated storage registers of the global stack area reserved solely for this purpose.

For the purposes of discussing the details of this application, some terms are now established to differentiate between the various boundary data stored in the sequencer stack *vs.* data memory. Figure 4 illustrates the location of these boundaries:

TBS:	True Bottom of Stack
TTS:	True Top of Stack
CBS:	Current Bottom of Stack
CTS:	Current Top of Stack
BPD:	Bottom of Paged Data
TPD:	Top of Paged Data

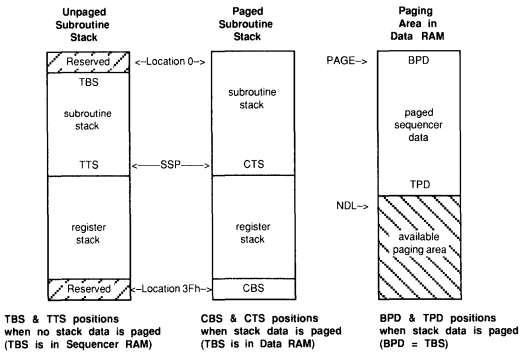


Figure 4. Stack Boundary Nomenclature

Underflow Buffer at the Top

In applications where stack paging is *not* utilized, internal RAM location zero is reserved as an underflow buffer. This avoids generating interrupts when the stack is emptied (every time TBS is popped) by never popping location zero. Since the SSP is initialized to location zero upon reset, the first push is automatically made to location one; location zero would never be popped except through programming error. If location zero must absolutely be used, these interrupts may be avoided by masking out IR_9 in the status register. Note that by masking out IR_9, stack overflows would also be masked.

Extra SS Location at the Bottom

In applications where SS paging *is* used, the first push (of TBS) should also be to location one, as in the unpaged scenario. Again, this is to avoid the underflow interrupt every time TBS is popped. Following the first and subsequent overflows, TBS will reside in data memory (at location BPD) and the new *bottom* of stack position (CBS) is moved to sequencer RAM location 3Fh (which is helpful to think of as

SS location –1) by the interrupt routine prior to restoring the most recent block of SS data. Location 3Fh is reserved expressly for this purpose by the programmer during μcode assemble.

Stack paging applications require this "extra" stack location (CBS) to accommodate the corresponding extra instruction which is executed after a stack overflow and prior to commencing the interrupt routine (the instruction executed during the interrupt dispatch). Consider the following situation: paged data already resides in data memory (TBS is at BPD and CBS is active) and a series of sequential SS pop instructions are executed, in the midst of which an underflow interrupt is generated.

Eventually, CTS migrates to and is popped from SS location zero. An interrupt request is then generated which is dispatched on the *following* instruction (which would pop location CBS from the SS, in our scenario). Although CBS is in fact popped, its address is supplanted with the interrupt vector by the sequencer's address multiplexer. Whenever an interrupt is dispatched, the address which *would* have been generated (were the interrupt not to have occurred) is pushed on the SS as the interrupt return address during the first cycle of the interrupt routine; a CONTinue instruction is always assumed (although a push is actually made) during this cycle.

Not knowing whether a pop instruction will follow an interrupt request or not requires that provision for this possible pop be made. Specifically, location 3Fh (CBS) is reserved for this extra pop, whether it occurs or not. One of the main jobs of the interrupt routine is to install the underflow buffer as the TBS is restored, or reorganize the bottom of stack to CBS whenever TBS is left in data memory.

TBS Always Returns to Stack Location One

Stack paging routine must guarantee the integrity of the stack after any sequence of overflows and underflows so that TBS always returns to SS location 01h. To do this, every data block about to be restored from data memory (whether due to overflow *or* underflow) is tested against the total amount of paged data currently held in data memory. If the block size (BSIZ. see program listings) is greater than this total amount (meaning that TBS will be returned), then special steps to restore the underflow buffer and truncate the block size are taken. Statistically, it is preferred to have the BSIZ ≈ SLR/2, but the relation BSIZ < SLR must also hold, otherwise the SSP will end up beyond the SLR after the block is restored and subsequent overflows will go undetected.

Locating the Paging Area in Data Memory

The base location of the paging area in data memory (PAGE) is also critical in preserving the stack integrity. This is because the CMP flag of the address generator is used to inform the sequencer whether TBS will be returned during any particular underflow. As underflows are processed, NDL (a dynamic memory pointer marking the Next Data Location available for paging in data memory) approaches the value of PAGE (in fact, NDL is initialized to PAGE). The interrupt routine checks that the relationship NDL–BSIZ ≤ PAGE (as evaluated by the address generator) is false by comparing the subtract result with the value of PAGE; the CMP flag conveys this information. If the test is true, then special steps are taken (see discussion of the MOPUP code segment in the *Line-by-Line Description* section).

If PAGE is located too close to data memory location zero (i.e., if PAGE ≤ BSIZ), the subtract will occasionally result in the address pointer being retarded "above" PAGE, wrapping it around zero to a large negative number (e.g., NDL–BSIZ = 0009h–0010h = FFF9h). Because the address generator comparator does not recognize twos complement numbers as such, negative results are interpreted as very large positive numbers by the comparator. In our example, the comparator will sense that indeed FFF9h ≥ PAGE, failing to indicate that TBS is about to be returned. The start of the SS paging area (PAGE) should be offset by at least the block size (PAGE > BSIZ) from location zero in data memory.

GENERAL ARCHITECTURAL DISCUSSION

A crucial decision is typically be made early in the system architecture design: whether or not to support interrupts in hardware. A decision to not explicitly accommodate interrupts in hardware will severely impact system performance because of pipelined data and instruction flow. The problem is that numerical results in the pipeline are permanently lost if an interrupt occurs because the instruction(s) intending to deal with this data are displaced by the interrupt code. Special steps must be taken to save this information.

A few possibilities to prevent the loss of interrupted data flow as it falls out of the pipeline are:

- Use dedicated hardware latches which grab the data whenever an interrupt occurs and which restore the data upon interrupt return
- Relegate the data saving task to the interrupt routine (a software approach)
- Preclude data transfers and arithmetic operations when interrupts may occur.

The use of dedicated latches solves the problem of pipelined data loss regardless of the number of data sources (be they memory, arithmetic processors, acquisition or display subsystems) at the cost of increased hardware.

The software solution incurs no hardware penalty, but can still result in lost data if more than one value becomes valid during any one cycle. The software solution is further complicated if stack recovery interrupts are used. In this context, we must assume all stacks are full and therefore, their use is precluded for data saves — an otherwise convenient way of capturing pipelined data. The only option then open is to reserve space solely for software data capture on the global stack. Global stack saves of *any* data

cannot be made during the first line of any interrupt routine, because this cycle is reserved to push the interrupt return address.

THE TARGET SYSTEM HARDWARE

In a real-world situation, latching pipelined data during interrupts is necessary to realize a fully interruptable system. However, these additional latches and associated control hardware are extraneous to the stack paging application *per se* and are not actually supported in the target system.

A simplified system block diagram of the hardware is shown in Figure 5. The key components are the program sequencer (ADSP-1401), two address generators (ADSP-1410, one each for data and coefficient memory), microcode and data memory, the microcode pipeline stage and the arithmetic processing block.

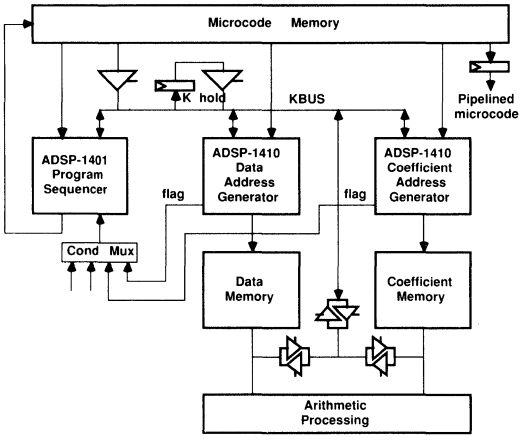


Figure 5. Simplified System Block Diagram

The system runs at 10MIPS. Because of the Look-Ahead internal microcode pipeline stage of both Word-Slice™ components, all other components of the system (without the Look-Ahead pipeline) only require 75ns microcode memory (subtracting the 25ns microcode address output delay of the sequencer from the total cycle). However, since the sequencer is addressing its own next instruction, 45ns microcode memory is used for its memory (100ns – 25ns address delay – 30ns setup time). The address generators use 45ns memory. The AGs could use even slower memory than this because the they do not address their own instructions as the sequencer does, yet not as slow as 75ns (used for the balance of the system) since the AGs still require sufficient instruction pre-decoding time. The access time of the microcode memory used for the address generator ultimately impacts the speed of the *data* memory needed (45ns data memory is used on the target system).

Although the Word-Slice components incorporate the microcode pipeline stage on-chip, unpipelined devices (e.g., memory control, the condition multiplexer, the arithmetic processing block, etc.) require an explicit microcode pipeline stage. This is why operations involving non-Word-Slice components (e.g., enabling the constant field data drivers) require the appropriate controls to be asserted in the instruction preceding the operation (i.e., driving a constant from microcode onto the K bus).

THE STACK RECOVERY INTERRUPT ROUTINE

This section provides a general discussion of a specific programming example used throughout the balance of this application note to illustrate the workings of the interrupt routine. For complete details on any aspect of the example, refer to the *Line-by-Line Description of IR9* section.

Figure 6 outlines a series of "snapshots" (shown in the appendix) showing a sequence of overflow and underflow interrupts which comprise our example. The purpose of showing these snapshots is to portray the actual execution of the interrupt routine. Reference to the actual interrupt code is encouraged while reading this section.

Each of the snapshots show three frames: the sequencer RAM at the beginning of the interrupt routine (immediately following the return address push); changes made to the data memory paging area during interrupt processing; and the sequencer RAM immediately following the interrupt return. Debugging code invokes the interrupts and performs "core" dumps of sequencer RAM at various times to generate the different snapshot frames.

Various arrows are used in the snapshots to show exactly how data migrates between data memory and the subroutine stack during interrupt processing. Note that stack locations 3Ah through 3Eh are global stack locations reserved expressly for context saves during interrupt processing.

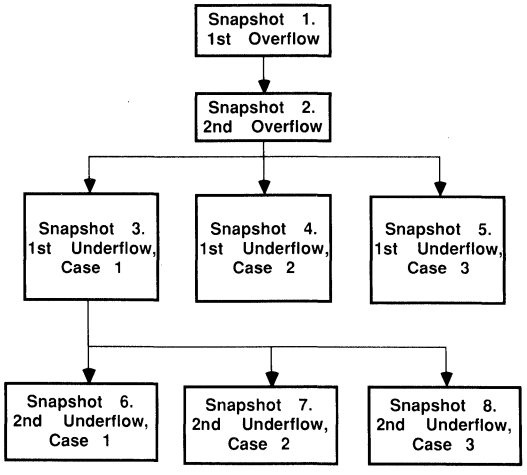


Figure 6. Snapshot Flowchart

The interrupt code makes extensive use of the address generator's ALU and various sequencer instructions (such as stack data and pointer operations) to effect the requisite data shuffling. The use of the global stack is crucial to the success of the interrupt routine because it provides a place to save the various address generator registers during the interrupt processing. Note that five global stack locations are reserved for this purpose.

Throughout the entire series of snapshots, the SLR remains at location 10h for the sake of simplifying the snapshots. All references to data memory locations are implicitly relative to PAGE, the base address of the paging area. The debug code has been pushing data values corresponding to the respective stack locations during the overflow sequence; one was pushed to location one, two to location two, and so on. The debug code has been popping these data values during the underflow sequences.

Overflows

Let's start with the first overflow, Snapshot 1. Frame 1 shows the presence of the underflow buffer at location zero of the sequencer RAM — note that Frame 3 of Snapshots 6, 7 and 8 (representing the return of TBS to the sequencer RAM after underflowing) show the underflow buffer restored. Pushing data to location 10h generated an interrupt because the SSP now equals the SLR. During the interrupt vectoring of the next cycle, the stack is pushed with the value 11h to location 11h; the interrupted instruction. The first instruction of the interrupt routine is now executed (microcode location IRV_9, which had better be a CONT) which is used to push the resultant address of the interrupted instruction onto the SS as the interrupt return address. This address is denoted RTI1 in the snapshot.

The first job of the interrupt routine (be it invoked by overflow *or* an underflow) is to free some registers on the address generator by pushing their values to the reserved area on the sequencer's global stack. This gains some computational "elbow room" for the various addressing and calculating tasks to be performed during interrupt processing (see Figure 7, the interrupt routine flowchart). The preallocated area of the global stack is used for this purpose because pushing on the register stack may destroy the SS.

Next, the SSP is examined to determine whether an overflow or an underflow has occurred. In this case, an overflow is detected and the interrupt routine proceeds to save the entire subroutine stack (locations 00h through 12h in this case) to a similarly sized block in data memory marked by NDL (a pointer to the next available data location in the paging area which is initialized to PAGE).

The transfer to data memory during overflow is set up by first loading NDL into an address comparator register and then adding NDL with the current value of the SSP. The resulting sum serves as the data memory starting address of the SS data to be transferred. This sum is also saved on the GS for later use. The transfer is made by repeatedly popping the SS to data memory while simultaneously decrementing the data memory pointer and comparing it to NDL (stored in the compare register). This process continues until the address generator comparator flags the sequencer that the data memory pointer has reached the original NDL value, meaning the transfer is complete (i.e., the SSP has reached the TBS location, sequencer RAM location one). This transfer is depicted by the arrows shown between Frames 1 and 2 of Snapshot 1. Note the length and direction of the arrows is consistent with the amount and sequence of data being transferred.

Having transferred the SS contents to the data memory paging area, the most recent SS values (BSIZ in number) are next transferred back to the sequencer starting at CBS. This transfer is set up by retrieving the sum NDL+SSP (previously saved on the global stack) and loading it into a compare register. The value of BSIZ is then subtracted from this to point to location 8h (NDL+SSP–BSIZ). This value (8h) is saved on the GS for updating the NDL pointer later.

Now the transfer begins (see arrows between Frames 2 and 3 of Snapshot 1). The most recent SS values are returned (from data memory location 8h) to the SS starting with the CBS location by writing the SSP to CBS. The transfer is then performed by incrementing the address register (until it equals NDL+SSP) while reading data memory and pushing it onto the SS. Again, note the arrow length and direction correspond to the number and sequence of transfers, including the wrapping of the SSP from the bottom (CBS) to the top of the sequencer RAM.

Various housekeeping tasks (such as updating NDL and restoring the address generator registers) are then performed and the RTNIR instruction is executed. This pops the subroutine stack (sequencer RAM location 8h, at this point), returning control to the calling code at the point where it was interrupted. This is the point in time (having just returned to the calling code) at which Frame 3 of the snapshot is taken. Note the presence of what is now residual garbage (the most recent stack values just returned to the sequencer) leftover from the interrupt processing. More significantly, note that the SS now has more room, having left its old data in data memory. The residual stack data held in data memory is indicated in Frame 2 as "Net Data Paged."

Snapshot 2 is a continuation of the series of mainline SS pushes, showing how additional overflows are appended to the existing paging area. Frame 1 of Snapshot 2 is a repetition of Frame 3 of Snapshot 1. Here we see that additional pushes have been made to locations 8h through 12h by the mainline code. Again the interrupt routine transfers the entire subroutine stack to the data memory paging area starting at the new NDL marker (now pointing to data memory location 8h). Note that the interrupt routine recognizes that TBS already resides in data memory (NDL ≠ PAGE) and therefore, pops the SS around the horn to location 3Fh (CBS) during the transfer.

The same approach of restoring the SS with the most recent

segment of the SS values is utilized here as was used in the first overflow (Snapshot 1). Again, paging back begins with CBS and again, more room is made.

Three Cases of the First Underflow

Before delving into underflows, it is helpful to understand the three possible underflow scenarios. Underflows arise because SS location zero has been popped. Depending upon the type of instruction executed during the interrupt vectoring, the SSP will end up pointing to one of three possible stack locations. The three instruction types are:

- Case 1, A stack pop from location 3Fh
- Case 2, A "benign" operation (neither pop nor push)
- Case 3, A stack push to location 00h

Depending upon which one of these operations take place, the SSP will be left pointing one of three positions after the interrupt return address is pushed: 3Fh for pop, 00h for benign, or 01h for push. Therefore, the interrupt routine must determine exactly where the SSP is and then jump to the correct location within the interrupt code to accommodate the corresponding special handling. The three underflow cases require that an additional one, two, or three extra locations be appended to the stack paging area prior to restoring the missing block from data memory. This is to guarantee that TBS is ultimately returned to SS location one. If the SSP is not at any of these three locations, then it is assumed by default that an overflow has occurred.

In Case 1, the interrupt return address (the result of the interrupted pop instruction) has just been pushed to SS location 3Fh (CBS) and must be appended to the existing stack image in data memory before paging back the missing data which caused the underflow. This is to insure that the return address is restored to the CTS position.

In Case 2, location 3Fh (CBS) still holds valid stack data, and the interrupt return address has been pushed to location 00h. Here, CBS is first appended to the stack image, followed by the interrupt return address of location 00h.

In Case 3, location 00h was just popped and then pushed with a new value. Here again, stack location 3Fh (CBS) is still valid, as are both locations 00h and 01h (containing the new push and the return address, respectively). Therefore, locations 3Fh, 00h, and 01h must be appended to the stack image, in that order. This is the benign case since no SS operations were made besides the usual return address push.

In conjunction with the addition of this miscellaneous SS data to the paging area of data memory prior to restoring the missing SS data, the block size (BSIZ) of data to be returned to the sequencer is also increased by an amount commensurate with the amount of data appended to the paging area. This is to guarantee that regardless of what particular series of interrupts occurred, the TBS is ultimately restored to sequencer RAM location one. If the block size were not modified on a case-by-case basis, miscellaneous data will be left lingering in the paging area by each underflow. This lingering data will accumulate and incur additional underflow interrupts to restore: an unwarranted overhead.

The addition of the miscellaneous data to the paging area is indicated by one, two or three additional arrows (depending on the case) between Frames 1 and 2 of each underflow snapshot. The increase in the block size is shown by the arrow of Frame 2 having increased length beyond the horizontal bar to include the miscellaneous data.

Snapshot 3 is the first of the underflow interrupts. This is a Case 1 underflow, in which an extra *pop* was performed during the vectoring. The extra pop removed CBS (location 3Fh) from the SS and replaced it with the interrupt return address. In order that the interrupt routine know where to go when done, this return address is appended to the next available paging location before restoring missing data to the sequencer. The block size is increased by one to preserve stack integrity. Therefore, BSIZ+1 locations are paged back to the sequencer, placing the first at the CBS location because the underflow buffer is not needed since TBS still resides in data memory.

Snapshot 4 shows an alternate scenario to the first underflow (of Snapshot 3) in which a benign operation (Case 2) was performed during vectoring. Here the return address is pushed to location 00h of the SS and therefore, both locations 3Fh and 00h are added to the data memory image and two extra locations are paged back.

Snapshot 5 is similar to Snapshots 3 and 4 (all representing the first underflow), except that this time, the interrupted instruction was a stack *push*, Case 3. We show the push (of data value 1234h) to location 00h and then the interrupt return address push to location 01h. Three extra locations are therefore appended to the stack image and paged back to the sequencer during the interrupt.

Three Cases of the Second Underflow

The snapshot flowchart (Figure 6) shows that three cases of the second underflow are shown, all of which derive from a Case 1 underflow. This sequence was selected to closely emulate what would typically occur in a real environment, namely, a sequential series of pops without regard to page faulting. All three of the second-underflow cases end up by returning the TBS to the sequencer. Again, miscellaneous data is appended to the data memory image of the SS before paging back anything.

As the TBS is finally returned to sequencer location one, the underflow buffer must be recreated. This means two extra data memory transfers which would usually be included (representing transfers to sequencer locations 3Fh and 00h) are foregone. This phenomenon is portrayed in Snapshots 6, 7 and 8 as a tendency of the arrow representing the data about to be paged back to the sequencer to go above the allocated paging area. The slash marks at the tail of the

arrow represent these two locations being disallowed just as the same two locations were not paged out on the first overflow.

LINE-BY-LINE DESCRIPTION OF IR9

This section gives detailed descriptions of the actual code of the sample interrupt code. The complete listing appears in the appendix "IR9 Interrupt Routine Source Code", and consists of four primary sections:

- Initialization Code (µcode locations 0040h through 0059h)
- Interrupt Calling Code (µcode locations 005Ah through 0088h)
- Debug Code (µcode locations 0089h through 0097h)
- Interrupt Code (µcode locations 0100h through 0173h)

Microcode locations 0000h through 003Fh contain various macro routines for transferring data between the host and the application board. These console macros are hardware and firmware specific and change significantly with different implementations. For this reason, they have been omitted from detailed description.

Initialization Code (µcode locations 0040h through 0059h)

The initialization code is designed to prepare the Word-Slice board for execution of the interrupt routine. After equating a series of constants used during the interrupt, both address generators are reset and initialized. Only address generator one (addressing data memory) is used by the interrupt routine. Address generator two (addressing coefficient memory) is used during the interrupt routine as a benchmark generator and is otherwise, not necessary. All subsequent references to the "address generator" (AG) will actually be to address generator one.

The next task performed is to initialize the subroutine stack paging area in data memory to a known state so as to help identify any abnormalities which may arise during program execution — this is not a required step. 256 (in this case) locations of the paging area are initialized by setting address generator compare register C_0 to point to the end of the allotted subroutine stack paging area in data memory (PAGE+FFh in this case), followed by loading address register R_0 with the value PAGE. Thereupon, the INIRAM loop is entered (µcode locations 0045h to 0047h) which initializes this block of data memory (locations PAGE through PAGE+FFh) to FFFFh.

Next, the stack paging area memory pointer is initialized. AG register R_0 is loaded with the pointer to NDL, PNDL (a constant with value 0100h). This allows access to the NDL pointer for *its* initialization. PNDL is placed on the data memory address bus while the initialization value of PAGE is placed on the data bus. A data memory write effects the initialization of NDL with the starting position of the stack paging area. PAGE (data memory location 0200h). PNDL is used throughout the program to access NDL.

Microcode locations 004Ah through 004Dh are in preparation for sequencer initialization. All interrupts are disabled because this initialization process will generate an unwanted IR_9 interrupt request; the SSP traverses the SLR. The AG compare flag is used to control the sequencer initialization (although a sequencer counter would do just as well) by loading a compare register with the size of the sequencer RAM (3Fh) and a companion address register with zero. At µcode location 004Dh, the SSP is initialized to point to sequencer RAM location zero as well.

The INISEQ loop of µcode locations 004Eh and 004Fh initializes the entire sequencer RAM to the arbitrary value FFFFh for the same reason: to help spot problems during initial debugging. The SSP is left at location zero after the internal RAM is initialized.

Microcode location 0050h executes the sequencer instruction SLRIVP which sets the stack limit register (SLR) to sequencer RAM location 10h and simultaneously initializes the interrupt vector pointer (IVP) to point to IV_9. Microcode location 0051h then writes the interrupt vector (label IR9, which equals µcode location 0100h) to interrupt vector file location 9. Note that executing the SLRIVP instruction also clears the pending interrupt request generated by initializing the sequencer RAM.

Microcode locations 0052h through 0059h then proceed to configure the sequencer RAM in accord with Figure 2. No indirect jump address space is reserved. The global stack is configured for only the five locations required by the interrupt routine, and the local stack pointer is positioned above the GS, although it is not used here and could be neglected.

A minor restriction is imposed on the use of the RS by the interrupt routine. The assumption is made that the local stack pointer is always active in the user code. The interrupt routine therefore blindly activates the local stack upon exit. This could be a problem if a stack overflow/underflow interrupt is encountered while the GS is active; mysteriously, the local stack pointer will become the active RS pointer after an interrupt. A solution to this problem is to insure that the GS is active whenever SS pushes or pops are performed by only using the GS momentarily, always reverting the active RS to the local stack.

Finally, µcode locations 0056h through 0059h mask out all but IR_9 interrupts and in turn, enable interrupting.

Interrupt Calling Code (µcode locations 005Ah through 0088h)

The sequencer is now ready to support stack overflow/-underflow, and the interrupt calling code segment is simply a series of pushes and pops designed to invoke the interrupt routine for test purposes. The snapshots were generated by selectively commenting out any pushes and/or pops subsequent to the point in time which the snapshot is intended to portray. For example, Snapshot 1 (the first overflow) was created by executing the pushes of µcode lo-

flow) was created by executing the pushes of µcode locations 005Bh through 006Bh, resulting in one more push of data beyond the SLR marker which occurred during the interrupt vectoring initiated by the push to SS location 10h (µcode location 6Ah).

As more interrupt calls were forced in developing the snapshot series, gaps between µcode instructions (at line numbers 181, 192, 202, 204, and 215) were placed in the source code to indicate that an interrupt was executed there; Snapshot 1 was made at line 181, Snapshot 2 at 192, etc.

Two of the three underflow cases associated with each underflow (see the snapshot flowchart, Figure 6) are also shown as gaps in the µcode: the first underflow shows Case 1 at line 202 and Case 2 at line 204. Case 3 (a SS push during vectoring) is a special one which had to be modified for each snapshot. When generating the Case 3 first underflow interrupt (actually invoked on line 201), the µcode instruction appearing on line 225 is repositioned to the gap shown at line 202 and the KEN (constant enable) bit of the previous line set in order to effect the push.

The interrupt calling code, as it appears in the listings, corresponds to the generation of Snapshot 8, the last snapshot. The last series of pops (which up to µcode location 0088h had been causing stack underflows) are truncated (commented out) for pops of data values 0008h through 0001h because the TBS was just returned by the previous interrupt (called by having popped data value 9h); popping all the way to the TBS would produce no further interrupts.

Debug Code (µcode locations 0089h through 0097h)

The purpose of the debug code is to provide a vehicle whereby the results of the interrupt routine may be analyzed. It copies the entire sequencer RAM (locations 0016h through 3F16h) to data memory locations 0300h through 033Fh. The last statement of the debug code (µcode location 0096h) is a jump back to µcode location 0000h; where the program awaits a macro fetch from the command console. Data memory locations 0200h through 0220h (the stack paging area) and 0300h through 033Fh (the image of the sequencer's internal RAM) are then uploaded to the host computer for analysis.

Interrupt Code (µcode locations 0100h through 0173h)

The actual interrupt discussion should be followed using the IR9 Interrupt Routine Flow Chart, Figure 7. The first two instructions (µcode locations 0100h and 0101h) initialize the benchmark counter so as to provide a performance measure. This is followed by activation of the GS and a series of saves onto the GS, making various AG registers available for processing during the interrupt routine (µcode locations 0102h through 0107h).

The next task is to read the SSP value to control branching to the appropriate interrupt code segment. This requires, in addition to simply reading the SSP, masking its upper order bits since they are indeterminate on the read. The SSP is read into the address generator, which then performs the required masking in lines 0108h through 010Dh.

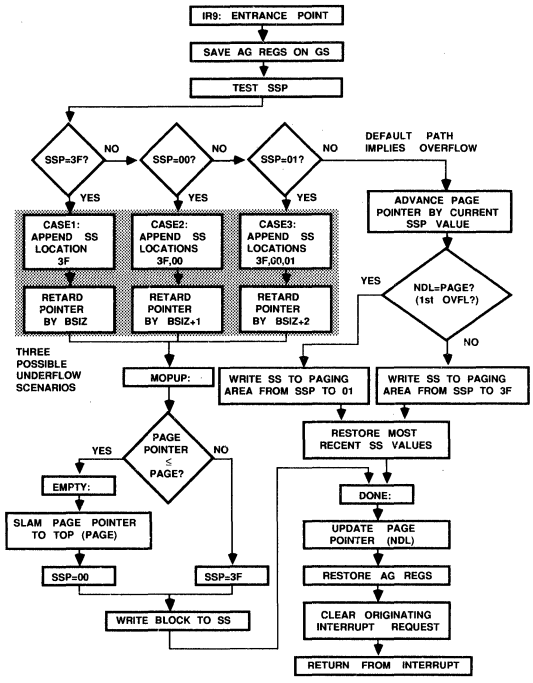


Figure 7. IR9 Interrupt Routine Flow Chart

The SSP is then tested (µcode locations 010Eh through 0113h) against the possible values comprising the three underflow scenarios. If none of the tests are true, the default action is a fall-through to the overflow code segment. Therefore, four possible pathways through the interrupt code are possible, all of which eventually merge again at the code segment DONE:

- Overflow
- Case 1 Underflow
- Case 2 Underflow
- Case 3 Underflow

Let's discuss the overflow path first. The first thing is to get the current NDL pointer and test if it equals PAGE. If NDL = PAGE, then we know that the sequencer's underflow buffer is in place (the TBS is resident) and we do not want to page out sequencer locations 3Fh and 00h. The fetching and testing of NDL is made in µcode locations 0114h through 0119h. If the test on NDL is true (NDL = PAGE), then µcode locations 011Ah through 0120h are executed, whereas if the test is false (NDL ≠ PAGE), then µcode locations 0121h through 0126h are executed.

The purpose of these two alternate code segments is to prepare to page out the entire SS to data memory (from either sequencer RAM location 3Fh or 01h, depending upon the test). NDL is transferred from an address register into a compare register. The value of the SSP (which had been saved on the GS) is then added to the address register, giving NDL+SSP. We are then ready to start popping the SS to data memory while the address generator decrements the address register until it hits the compare register, see Snapshot 1. This is done by LOOP1 (μcode locations 0127h through 0129h). Note that the sequencer is monitoring the AG CMP flag as the loop control.

The next phase, having gotten the SS into data memory, is to restore the most recent block. The amount of data to be paged back is set by the parameter BSIZ, the block size. First, we retrieve the advanced NDL pointer (NDL+SSP) which had been saved on the GS and load it into both an address and a compare register. The SSP is then set to sequencer RAM location 3Eh so that the first push will be to 3Fh. Next, we subtract the block size from the address register (retarding the pointer) and save this value to update NDL later. The system is now ready to page back the most recent stack entries to the sequencer as shown between Frames 2 & 3 of Snapshot 1. Microcode locations 012Ah through 012Fh implement these setups for paging back to the SS.

The actual paging back is performed in LOOP2 (μcode locations 0130h through 0132h). Note that again, the AG flag controls the loop.

Microcode locations 0133h and 0134h conclude the overflow-specific code segment by executing a jump to DONE, described later.

Underflows may be of either Case 1, 2 or 3. All underflows branch from the SSP test to the appropriate code segment for that case. Each these code segments comprises a unique initialization segment denoted CASE1, CASE2 or CASE3, respectively. The purpose of these initialization segments is to append the additional one, two or three SS locations to the data memory paging area corresponding to the case number. Microcode locations 0135h through 013Dh append sequencer location 3Fh to data memory, comprising the CASE1 segment. Microcode locations 013Eh through 014Ah append sequencer locations 3Fh and 00h to data memory, comprising the CASE2 segment. Microcode locations 014Bh through 0158h append sequencer locations 3Fh, 00h and 01h to data memory, comprising the CASE3 segment.

All three underflow scenarios then jump to the MOPUP code segment (μcode locations 0159h through 0165h) which does some miscellaneous housekeeping prior to writing back the most recent block to the sequencer. MOPUP has the responsibility of determining whether the TBS is about to be returned and if so, restoring the underflow buffer.

MOPUP tests for the imminent return of the TBS. The test is made by comparing the address register (which just finished addressing data memory during the appending of the extra SS locations) to the paging area. If this value is less than or equal to PAGE (the marker designating the starting point of the paging area), then the TBS return is imminent and the underflow buffer should be installed.

If the test SSP ≤ PAGE is false, then a "full" block of SS data is restored to the SS starting with location 3Fh. Here, the SSP is simply positioned at sequencer RAM location 3Fh and program flow resumes with DONE (which does the actual transfer). However, if SSP ≤ PAGE is true, then the TBS is about to be returned and a "partial" block is restored. In this case, two things are done: the address pointer (R_0) is forced to the value PAGE, and the SSP is positioned to sequencer RAM location 00h.

In either case, the two alternate MOPUP segments merge together again at LOOP3 (microcode locations 0166h through 0168h). This loop is what actually effects the transfer of the data block back to the SS. Comparing Snapshots 6, 7 & 8 with Snapshot 3 readily shows the distinction between the two alternate pathways through MOPUP.

Now that the right sized data block has been restored to the SS, the final segment is executed: DONE (microcode locations 0169h through 0173h). DONE simply updates the NDL pointer with the value saved on the GS, restores the various registers to the address generator, clears the calling IR_9 interrupt request, and finally, returns to the calling code.

CONCLUSION

We have shown how subroutine stack paging for the ADSP-1401 Program Sequencer is performed: now let's consider its justification. The decision as to whether to implement stack paging should encompass at least these three considerations: necessity, complexity, and execution speed.

First of all, don't immediately jump to the conclusion that stack paging is needed. There is a lot of overhead associated with stack paging and a careful analysis of the troublesome user code should be made before resorting to stack paging. Poor programming practices, such as unnecessary or indiscriminate use of the stack, excessive use of subroutines for trivial tasks (over-structured programming), and over-allocation of sequencer RAM to the register stacks or indirect jump address space when direct data via the data port might suffice, can mislead the user into thinking there are no alternatives to stack paging.

The second issue to consider is whether the complexity of the interrupt code and the associated hardware provisions necessary to support an interruptable system is too great in view of the design objectives. We have ignored a detailed discussion of the hardware aspects of building a fully interruptable system because it does not pertain to the stack paging application *per se*. However, a truly interruptable system can quickly become a design nightmare, especially if multiple data paths and/or pipeline stages are involved.

Make sure the task truly warrants an interruptable system and furthermore, that the implementation is kept as simple as possible.

Finally, assuming one has determined that stack paging seems necessary and that the requisite support hardware is appropriate, one must evaluate whether the overhead is acceptable. This application note offers a sample solution to the stack paging problem. Although careful attention was paid to minimizing the code and execution time, this solution should not be construed as definitive. The benchmarks shown in the appendix should be considered maximum measures, realizing that the interrupt routine ultimately employed may improve upon these measures, depending upon the system requirements and the ingenuity of the programmer.

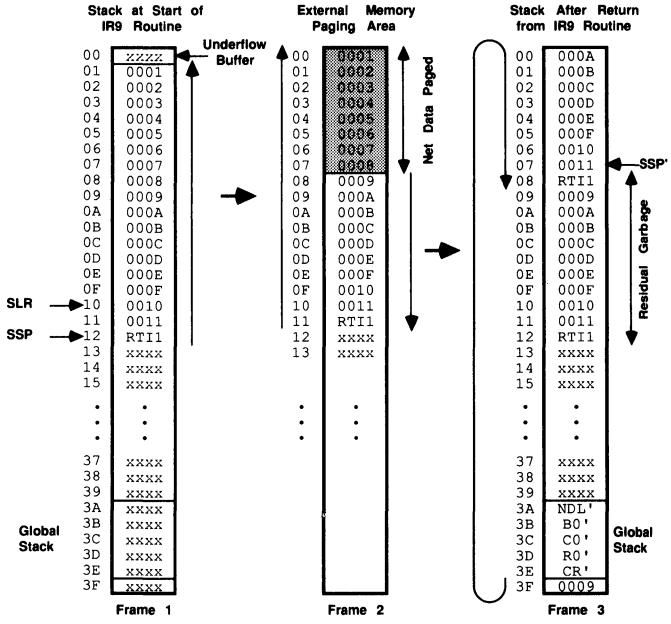
The advantage of implementing a paged subroutine stack is twofold. Obviously, the stack may be considered as infinitely large, provided sufficient data memory is available. In conjunction with paging the subroutine stack, the allocation of the sequencer's internal subroutine stack area may be reduced. This reduction brings with it the possibility of utilizing this space for other sequencing tasks.

One final note: reread and throughly understand the *Principals and Precautions of Subroutine Stack Paging* section. This is where the more difficult problems encountered in developing this application note are recounted.

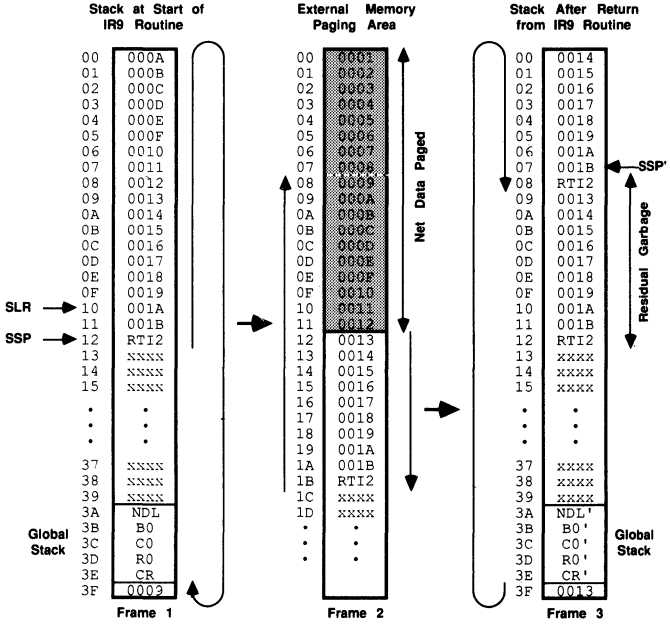
BENCHMARKS

The following benchmarks were made by initializing a second unused address generator (the one normally used for coefficient addressing; see Figure 5) register to zero at the start of the interrupt routine and then always incrementing the register, regardless of what the actual routine was doing. Obviously, these benchmarks are strongly dependent upon the stack and block sizes used because of the various loops employed in transferring data, but they do give a good ballpark estimation of the overhead associated with stack paging applications.

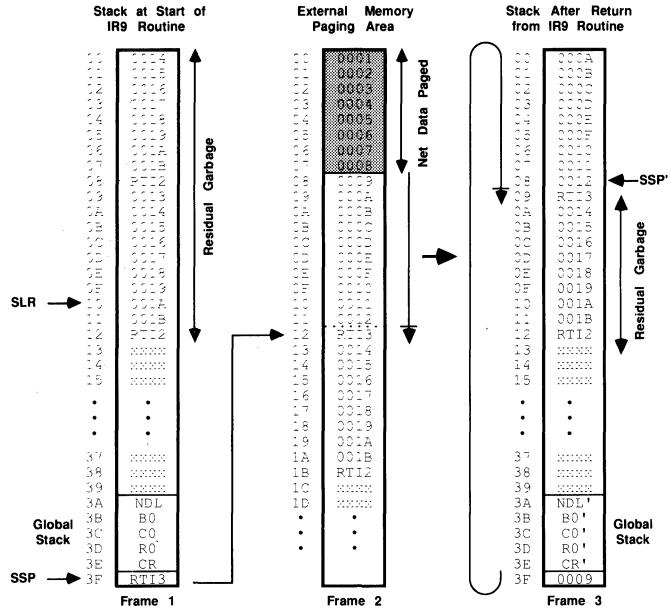
Snapshot	*Benchmark (cycles)*
1. First Overflow	134
2. Second Overflow	139
3. First Underflow (Case 1)	76
4. First Underflow (Case 2)	85
5. First Underflow (Case 3)	91
6. Second Underflow (Case 1)	68
7. Second Underflow (Case 2)	77
8. Second Underflow (Case 3)	83



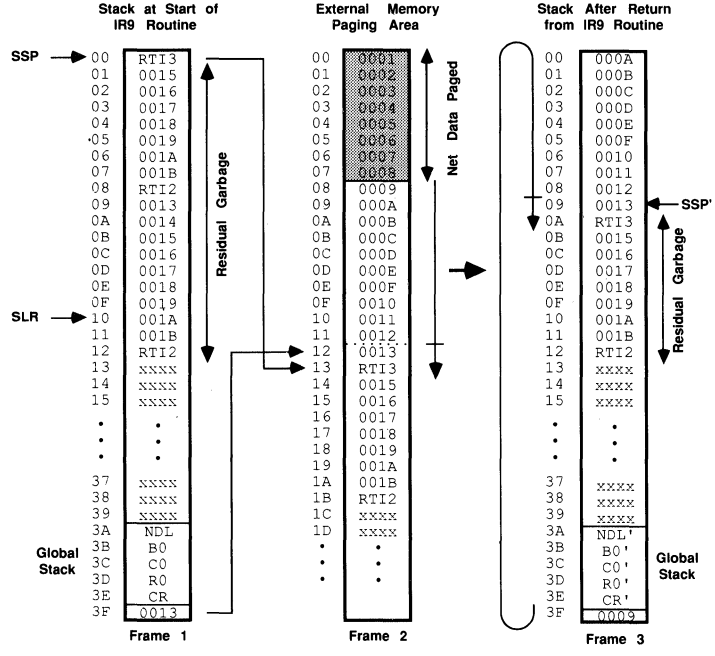
Snapshot 1. FIRST Overflow



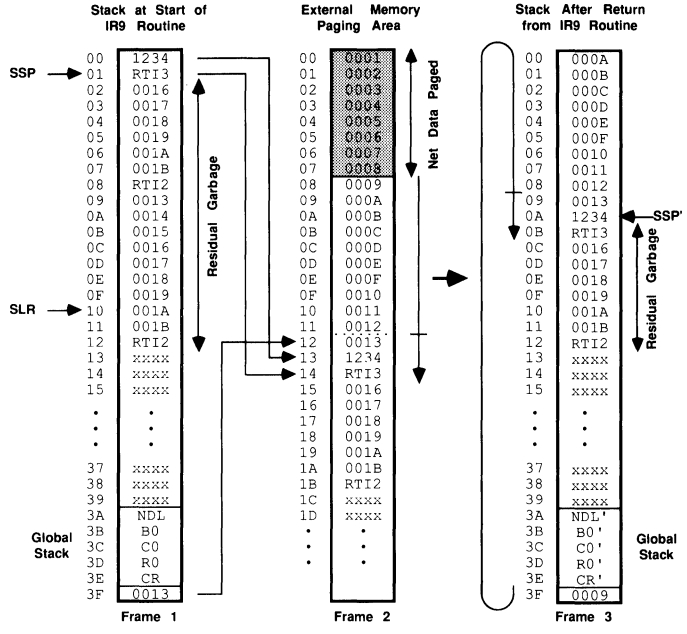
Snapshot 2. SECOND Overflow



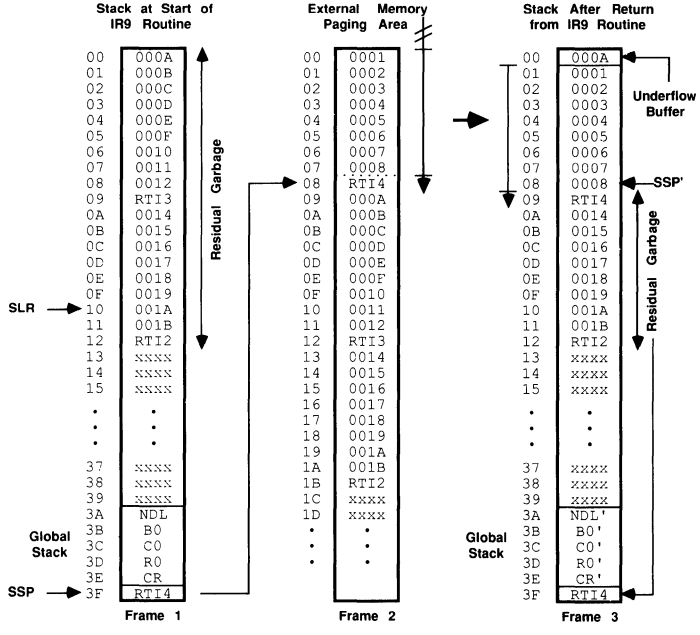
Snapshot 3. FIRST Underflow (Case 1)



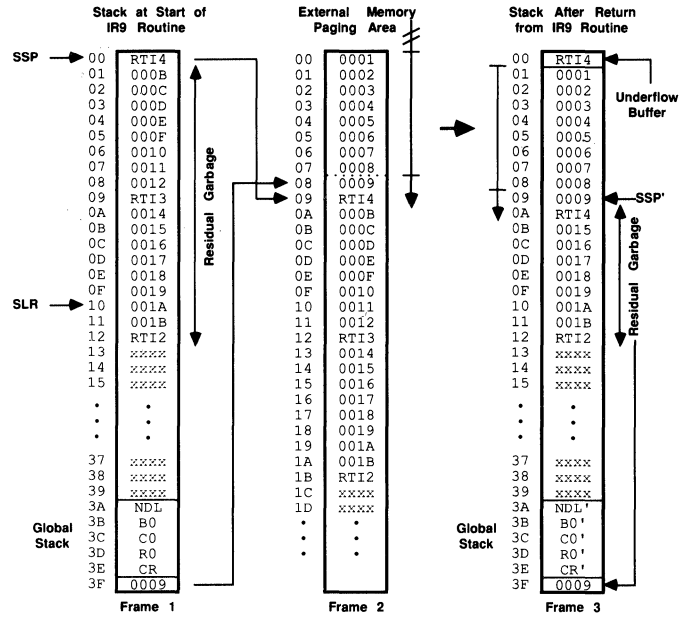
Snapshot 4. FIRST Underflow (Case 2)



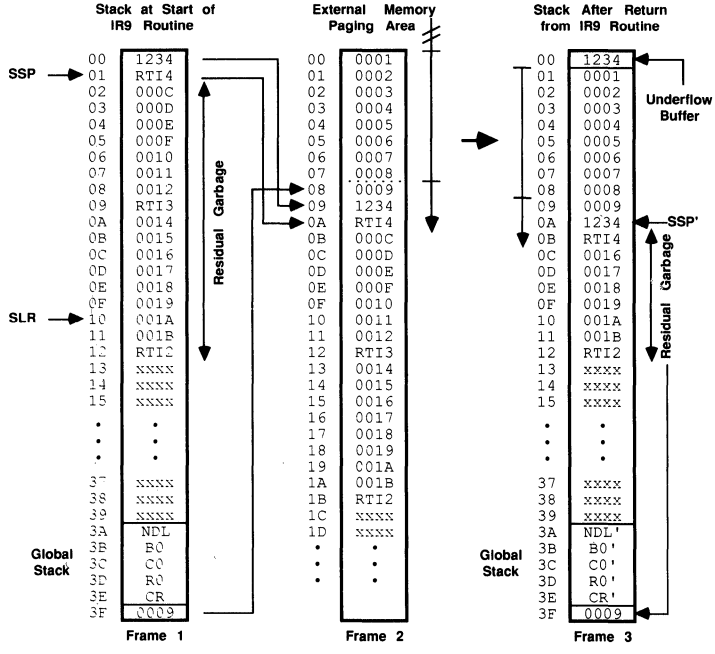
Snapshot 5. FIRST Underflow (Case 3)



Snapshot 6. SECOND Underflow (Case 1), Following FIRST Underflow (Case 1)



Snapshot 7. SECOND Underflow (Case 2), Following FIRST Underflow (Case 1)



Snapshot 8. SECOND Underflow (Case 3), Following FIRST Underflow (Case 1)

```
 95          ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 96          ;
 97          ;
 98          ;ADSP-1401 SUBROUTINE STACK OVERFLOW/UNDERFLOW DEBUG CODE
 99
100 00040    ORG     H#0040
101
102          ZERO:   EQU     H#0000  ;zip
103          PNDL:   EQU     H#0100  ;pointer to Next Data Location (NDL) holder
104          PAGE:   EQU     H#0200  ;start of data RAM paging area
105          DBUF:   EQU     H#0300  ;dump buffer for debugging (dump stack here)
106          BSIZ:   EQU     H#000A  ;block size to pageback after pageout is 2+SLR/2=A
107          INIT:   EQU     H#9010  ;IVP=9, and SLR=16
108
109          START:
110 00040    JPCNF & RST1 & RST2                                    ;required handshake
111
112          ;init address generators
113 00041    CONT & RST1 & RST2
114
115          ;init data memory paging area to FFFF
116 00042    CONT & NOP1 & NOP2 & KEN                                        ;enable K bus
117 00043    CONT & KA PAGE+H#FF & YRTC1 C0,R0 & DSEL1 & NOP2 & KEN          ;FF page length
118
119 00044    CONT & KA PAGE & YRTR1 R0 & DSEL1 & NOP2                        ;start of page
120          INIRAM:
121 00045    CONT & YRTR1 R0 & NOP2 & KEN & KTOL & WR1 FROML                 ;write RAM
122 00046    CONT & KA H#FFFF & YINC1 C0,R0 & NOP2 & KEN & MUX CZ1F          ;with FFFF
123 00047    JDA NF & KA INIRAM & NOP1 & NOP2 & KEN                          ;till flag
124
125          ;init the next available data location pointer (NDL) to PAGE
126 00048    CONT & KA PNDL & YRTR1 R0 & DSEL1 & NOP2 & KTOL & WR1 FROML & KEN
127 00049    CONT & KA PAGE & NOP1 & NOP2
128
129          ;disable interrupting and init entire sequencer RAM to FFFF using
130          ;AG1 flag as loop control (note: SSP ends up at 00)
131 0004A    DISIR & NOP1 & NOP2 & KEN
132 0004B    CONT & KA H#003F & YRTC1 C0,R0 & DSEL1 & NOP2 & KEN             ;C0=3F
133 0004C    CONT & KA ZERO & YRTR1 R0 & DSEL1 & NOP2 & KEN                  ;R0=00
134 0004D    WRSSP & KA ZERO & NOP1 & NOP2 & KEN                             ;SSP=00
135          INISEQ:
136 0004E    PSDSS & KA H#FFFF & YINC1 C0,R0 & NOP2 & KEN & MUX CZ1F         ;push FFFF
137 0004F    JDA NF & KA INISEQ & NOP1 & NOP2 & KEN                          ;til flag
138
139          ;init SLR and load IR9 vector
140 00050    SLRIVP & KA INIT & NOP1 & NOP2 & KEN            ;write SLR and IVP
141 00051    WRIV & KA IR9 & NOP1 & NOP2 & KEN               ;write IV9
142
143          ;init seq pointers, creating 5 location GS
144 00052    SGSP & NOP1 & NOP2 & KEN                        ;select GSP
145 00053    WRRSP & KA H#003F & NOP1 & NOP2                 ;GSP=3F
146 00054    SLSP & NOP1 & NOP2 & KEN                        ;select LSP
147 00055    WRRSP & KA H#003A & NOP1 & NOP2 & KEN           ;LSP=3A
148          ;note: the LSP is left active at this point (although the GS is used
149          ;exclusively in the actual interrupt routine).
150
151          ;disable all but IR9 and enable interrupting
152 00056    IRMBS & KA H#7FC0 & NOP1 & NOP2 & KEN           ;disable all but IR9
153 00057    IRMBC & KA H#8000 & NOP1 & NOP2 & KEN           ;enable only IR9
154 00058    SLRIVP & KA INIT & NOP1 & NOP2                  ;clear pending IR9
155 00059    ENAIR & NOP1 & NOP2                             ;enable interrupts
156
```

```
157                ;The debug/dump code following this series of pushes and pops allows
                   incremental
158
159                ;examination of the SS at various states.  Simply comment out the push/pop
160                ;instructions following the SS state of interest.
161
162                ;start pushing, man...
163 0005A          CONT & NOP1 & NOP2 & KEN
164 0005B          PSDSS & KA H#0001 & NOP1 & NOP2 & KEN   ;1
165 0005C          PSDSS & KA H#0002 & NOP1 & NOP2 & KEN   ;2
166 0005D          PSDSS & KA H#0003 & NOP1 & NOP2 & KEN   ;3
167 0005E          PSDSS & KA H#0004 & NOP1 & NOP2 & KEN   ;4
168 0005F          PSDSS & KA H#0005 & NOP1 & NOP2 & KEN   ;5
169 00060          PSDSS & KA H#0006 & NOP1 & NOP2 & KEN   ;6
170 00061          PSDSS & KA H#0007 & NOP1 & NOP2 & KEN   ;7
171 00062          PSDSS & KA H#0008 & NOP1 & NOP2 & KEN   ;8
172 00063          PSDSS & KA H#0009 & NOP1 & NOP2 & KEN   ;9
173 00064          PSDSS & KA H#000A & NOP1 & NOP2 & KEN   ;A
174 00065          PSDSS & KA H#000B & NOP1 & NOP2 & KEN   ;B
175 00066          PSDSS & KA H#000C & NOP1 & NOP2 & KEN   ;C
176 00067          PSDSS & KA H#000D & NOP1 & NOP2 & KEN   ;D
177 00068          PSDSS & KA H#000E & NOP1 & NOP2 & KEN   ;E
178 00069          PSDSS & KA H#000F & NOP1 & NOP2 & KEN   ;F
179 0006A          PSDSS & KA H#0010 & NOP1 & NOP2 & KEN   ;10
180 0006B          PSDSS & KA H#0011 & NOP1 & NOP2 & KEN   ;11
181
182 0006C          PSDSS & KA H#0012 & NOP1 & NOP2 & KEN   ;12
183 0006D          PSDSS & KA H#0013 & NOP1 & NOP2 & KEN   ;13
184 0006E          PSDSS & KA H#0014 & NOP1 & NOP2 & KEN   ;14
185 0006F          PSDSS & KA H#0015 & NOP1 & NOP2 & KEN   ;15
186 00070          PSDSS & KA H#0016 & NOP1 & NOP2 & KEN   ;16
187 00071          PSDSS & KA H#0017 & NOP1 & NOP2 & KEN   ;17
188 00072          PSDSS & KA H#0018 & NOP1 & NOP2 & KEN   ;18
189 00073          PSDSS & KA H#0019 & NOP1 & NOP2 & KEN   ;19
190 00074          PSDSS & KA H#001A & NOP1 & NOP2 & KEN   ;1A
191 00075          PSDSS & KA H#001B & NOP1 & NOP2         ;1B
192
193                ;start popping, where I don't care.
194 00076          PPSSD & NOP1 & NOP2                     ;1B
195 00077          PPSSD & NOP1 & NOP2                     ;1A
196 00078          PPSSD & NOP1 & NOP2                     ;19
197 00079          PPSSD & NOP1 & NOP2                     ;18
198 0007A          PPSSD & NOP1 & NOP2                     ;17
199 0007B          PPSSD & NOP1 & NOP2                     ;16
200 0007C          PPSSD & NOP1 & NOP2                     ;15
201 0007D          PPSSD & NOP1 & NOP2                     ;14
202
203 0007E          PPSSD & NOP1 & NOP2                     ;13
204
205 0007F          PPSSD & NOP1 & NOP2                     ;12
206 00080          PPSSD & NOP1 & NOP2                     ;11
207 00081          PPSSD & NOP1 & NOP2                     ;10
208 00082          PPSSD & NOP1 & NOP2                     ;F
209 00083          PPSSD & NOP1 & NOP2                     ;E
210 00084          PPSSD & NOP1 & NOP2                     ;D
211 00085          PPSSD & NOP1 & NOP2                     ;C
212 00086          PPSSD & NOP1 & NOP2                     ;B
213 00087          PPSSD & NOP1 & NOP2                     ;A
214 00088          PPSSD & NOP1 & NOP2                     ;9
215
216                ;       PPSSD & NOP1 & NOP2             ;8
217                ;       PPSSD & NOP1 & NOP2             ;7
```

```
                ;       PPSSD & NOP1 & NOP2             ;6
                ;       PPSSD & NOP1 & NOP2             ;5
                ;       PPSSD & NOP1 & NOP2             ;4
                ;       PPSSD & NOP1 & NOP2             ;3
                ;       PPSSD & NOP1 & NOP2             ;2
                ;       PPSSD & NOP1 & NOP2             ;1

                ;       PSDSS & KA H#1234 & NOP1 & NOP2         ;CASE 3 during vectoring (KEN)

                ;done with interrupt calling code, now dump sequencer RAM
                ;to data memory and upload to host for verification
                DEBUG:
00089           CONT & NOP1 & NOP2

                ;disable interrupts
0008A           DISIR & NOP1 & NOP2

                ;do a push of benchmark counter to indicate where SSP ended up
                ;and also to note the count.
0008B           CONT & NOP1 & RTD2 R0 & KHOLD                   ;read r0
0008C           PSDSS & NOP1 & NOP2 & KEN                       ;push it on SS

                ;dump seq RAM 0-3F into data memory 0300-033F using sequencer counter
                ;as loop control
0008D           WRCNTR C0 & KA H#803E & NOP1 & NOP2 & KEN               ;preload cntr
0008E           CONT & KA DBUF+H#3F & YRTR1 R0 & DSEL1 & NOP2 & KEN     ;3F buffer
0008F           WRSSP & KA H#003F & NOP1 & NOP2                         ;SSP=3F
                DUMP:
00090           PPSSD & YDEC1 C0,R0 & NOP2 & KHOLD & KTOL & WR1 FROML   ;stack to mem
00091           DCCNTR C0 & NOP1 & NOP2 & KEN                           ;dec cntr
00092           JDA SN & KA DUMP & NOP1 & NOP2                          ;till cntr undfl

                ;clear dummy IR9 request caused by dumping stack
00093           CONT & NOP1 & NOP2 & KEN                        ;enable K bus
00094           SLRIVP & KA INIT & NOP1 & NOP2                  ;clear IR9
00095           ENAIR & NOP1 & NOP2                             ;enable interrupts

                ;jump back for a macro fetch
00096           CONT & NOP1 & NOP2 & KEN
00097           JDA & KA ZERO & NOP1 & NOP2


                ;===============================================================================

                ;       ADSP-1401 SUBROUTINE STACK OVERFLOW/UNDERFLOW ROUTINE (IR9)
                ;===============================================================================

00100           ORG     H#0100
                ;Component assumptions:
                ;       1401 SS space is reserved from location 3F around to SLR
                ;       1401 has 5 GS locations reserved for this routine (3E-3A, inclusive)
                ;       1401 SS starts at location 3F and ends at SLR (SLR+2 in length)
                ;       1401 has the LSP activated
                ;       1410 status will be preserved (CR saved on seq GS)
                ;
                ;
                ;Data memory:
                ;NDL    Next Data Location available for writing SS values
                ;               (allocated in calling code)
                ;PNDL   a Pointer to NDL (defined in calling code)
                ;
                ;Immediate data from microcode (defined in calling code)
```

```
280           ;BSIZ   Block SIZe: SS is SLR+2 long, BSIZ should be 2+SLR/2
281           ;               to minimize page faulting
282           ;INIT   INITialization word for the sequencer SLRIVP instruction
283           ;
284           ;Labels used in IR9 interrupt routine:
285           ;IR9    entrance point (saves various machine states and
286           ;       determines type of interrupt -- overflow or underflow)
287           ;OVFL   SS OVerFLow segment
288           ;LOOP1  dump SS to data RAM, always followed by...
289           ;LOOP2  restore most recent SS values from data RAM (after overflow), or...
290           ;LOOP3  restore most recent SS values from data RAM (after underflow)
291           ;three underflow scenarios...
292           ;       CASE1   SS UNderFLow routine starting address: for pop,pop,psh(rtn
                      addr)
293
294           ;       CASE2   SS UNderFLow routine starting address: for pop,xxx,psh(rtn
                      addr)
295
296           ;       CASE3   SS UNderFLow routine starting address: for pop,psh,psh(rtn
                      addr)
297
298           ;MOPUP  prepare to restore SS in LOOP3 (called after UNFLxx)
299           ;DONE   restore saved states and return
300           ;
301           ;NOTES:
302           ;This routine does not discriminate TRAP type IR9 requests!
303           ;
304           ;This routine will attempt to support infinite subroutine stack paging.
305           ;No attempt is made to limit paging to data memory.  It is the user's
306           ;responsibility to allot sufficient memory to accommodate their needs so
307           ;as to not corrupt data memory used for other purposes.
308           ;
309           ;Paging of the Register Stack (LS or GS) is not supported.
310
311
312           ;Subroutine stack overflow/underflow interrupt routine
313           IR9:
314           ;init r0 of AG2 with zero to serve as benchmark counter.
315 00100     CONT & NOP1 & NOP2 & KEN
316 00101     CONT & KA ZERO & NOP1 & YRTR2 R0 & DSEL2
317
318           ;save AG1 registers onto global stack and init
319 00102     SGSP & NOP1 & yinc2 c0,r0               ;select GSP
320 00103     CONT & CRTD1 & yinc2 c0,r0 & KHOLD      ;save CR on GS
321 00104     PSDRS & RTD1 R0 & yinc2 c0,r0 & KHOLD   ;save R0 on GS
322 00105     PSDRS & CTD1 C0 & yinc2 c0,r0 & KHOLD   ;save C0 on GS
323 00106     PSDRS & BTD1 B0 & yinc2 c0,r0 & KHOLD   ;save B0 on GS
324 00107     PSDRS & RST1 & yinc2 c0,r0 & KEN        ;clear CR (mask init regs!)
325
326           ;get the SSP right-filled with zeros
327 00108     CONT & KA H#003F & YRTR1 R0 & DSEL1 & yinc2 c0,r0       ;leading zeros to R0
328 00109     RDSSP & NOP1 & yinc2 c0,r0 & KHOLD                      ;get SS pointer
329 0010A     CONT & YAND1 B3,R0 & yinc2 c0,r0                        ;AND in with mask
330 0010B     CONT & YRTB1 B0,R0 & yinc2 c0,r0                        ;zero filled SSP to B0
331 0010C     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                    ;save on GS for OVFL
332 0010D     PSDRS & NOP1 & yinc2 c0,r0 & KEN
333
334           ;find out where to go...
335           ;       if SSP=63, then we had a pop,pop,psh(rtn addr): go to CASE1
336           ;       if SSP=00, then we had a pop,xxx,psh(rtn addr): go to CASE2
337           ;       if SSP=01, then we had a pop,psh,psh(rtn addr): go to CASE3
338           ;       otherwise, we implicitly have an overflow: fallthrough to OVFL
```

```
339
340 0010E     CONT & KA H#003F & YXOR1 B0,R0 & DSEL1 & yinc2 c0,r0 & KEN & MUX CZ1F
341 0010F     JDA FL & KA CASE1 & NOP1 & yinc2 c0,r0 & KEN    ;test for 3F
342
343 00110     CONT & KA ZERO & YXOR1 B0,R0 & DSEL1 & yinc2 c0,r0 & KEN & MUX CZ1F
344 00111     JDA FL & KA CASE2 & NOP1 & yinc2 c0,r0 & KEN    ;test for 00
345
346 00112     CONT & KA H#0001 & YXOR1 B0,R0 & DSEL1 & yinc2 c0,r0 & KEN & MUX CZ1F
347 00113     JDA FL & KA CASE3 & NOP1 & yinc2 c0,r0 & KEN    ;test for 01
348
349           ;fallthrough implies overflow
350
351           OVFL:
352           ;SS overflow routine
353
354           ;move mem pointer ahead by stack size...
355 00114     CONT & KA PNDL & YRTR1 R0 & DSEL1 & yinc2 c0,r0         ;get NDL pointer
356 00115     CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL          ;read NDL
357 00116     CONT & YRTB1 B0,R0 & DSEL1 & yinc2 c0,r0                ;NDL into B0
358
359           ;test for first underflow (NDL=PAGE)
360 00117     CONT & NOP1 & yinc2 c0,r0 & KEN
361 00118     CONT & KA PAGE & YXOR1 B0,R0 & DSEL1 & yinc2 c0,r0 & KEN & MUX CZ1F
362 00119     JDA NF & KA REST & NOP1 & yinc2 c0,r0 & KEN             ;jump if not first
363
364           ;fallthrough implies first underflow...
365           ;so avoid paging of locations 3F and 00.
366 0011A     CONT & BTR1 B0,R0 & yinc2 c0,r0                 ;NDL into R0
367 0011B     CONT & YRTC1 C0,R0 & yinc2 c0,r0                ;NDL into C0
368 0011C     PPRSD & YDEC1 C0,R0 & yinc2 c0,r0 & KHOLD       ;R0<=NDL-1
369 0011D     CONT & YADD1 C0,B3,R0 & yinc2 c0,r0             ;add in stack size
370 0011E     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD            ;R0<=NDL+SSP-1, and
371 0011F     PSDRS & NOP1 & yinc2 c0,r0 & KEN                ;save as readback comp
372 00120     JDA & KA LOOP1 & NOP1 & yinc2 c0,r0             ;jump to LOOP1
373
374           ;not first underflow...
375           ;so perform paging of locations 3F and 00.
376           REST:
377 00121     CONT & BTR1 B0,R0 & yinc2 c0,r0                 ;NDL into R0
378 00122     CONT & YRTC1 C0,R0 & yinc2 c0,r0                ;NDL into C0
379 00123     PPRSD & YINC1 C0,R0 & yinc2 c0,r0 & KHOLD       ;R0<=NDL+1
380 00124     CONT & YADD1 C0,B3,R0 & yinc2 c0,r0             ;add in stack size
381 00125     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD            ;R0<=NDL+SSP+1, and
382 00126     PSDRS & NOP1 & yinc2 c0,r0                      ;save as readback comp
383           ;fall through to LOOP1
384
385           ;pop SS to mem from current SSP to 01 or 3F (depending on whether
386           ;this is the first underflow or not, respectively).
387           LOOP1:
388 00127     PPSSD & YRTR1 R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
389                 ;dec & read SSP
390 00128     CONT & YDEC1 C0,R0 & yinc2 c0,r0 & KEN & MUX CZ1F
391                 ;extra line for KEN
392 00129     JDA NF & KA LOOP1 & NOP1 & yinc2 c0,r0          ;until flag
393
394           ;setup to read back latest page
395 0012A     PPRSD & NOP1 & yinc2 c0,r0 & KHOLD                      ;get NDL+SSP+/-1
396 0012B     CONT & YRTR1 R0 & DSEL1 & yinc2 c0,r0 & KEN             ;into R0
397 0012C     WRSSP & KA H#003E & YRTC1 C0,R0 & yinc2 c0,r0 & KEN     ;and C0, SSP=3E
398
399           ;BSIL is decreased here to match up the first stack entry to location
400           ;one after a series of overflows and underflows:
```

```
401           ;overflows write back one less than underflows to SS.
402 0012D     CONT & KA BSIZ-1 & YSUB1 C0,B3,R0 & yinc2 c0,r0          ;R0<=NDL+SSP+/-1-BSIZ-1
403 0012E     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                    ;save it on GS as NDL'
404 0012F     PSDRS & NOP1 & yinc2 c0,r0                              ;to update [PNDL] at end
405
406           ;write block of paged data into SS (starting with location 3F)
407           LOOP2:
408 00130     CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL          ;read data mem loc
409 00131     PSDSS & YINC1 C0,R0 & yinc2 c0,r0 & KEN & MUX CZ1F
410                   ;inc SSP and write seq
411 00132     JDA NF & KA LOOP2 & NOP1 & yinc2 c0,r0                  ;until flag
412
413 00133     CONT & NOP1 & yinc2 c0,r0 & KEN                         ;extra line for ken
414 00134     JDA & KA DONE & NOP1 & yinc2 c0,r0                      ;end of overflow routine
415
416
417
418           CASE1:
419           ;SS underflow by pop,pop,psh(rtn addr), i.e., rtn addr @ loc 3F
420
421           ;get NDL
422 00135     PPRSD & KA PNDL & YRTR1 R0 & DSEL1 & yinc2 c0,r0        ;clear SSP from GS
423 00136     CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL          ;read [PNDL]
424 00137     CONT & YRTR1 R0 & DSEL1 & yinc2 c0,r0                   ;into R0
425
426           ;move RTI address [3F] to data mem without inc on R0 (NDL),
427           ;thereby wiping RTI on next pagefault.
428 00138     PPSSD & YRTR1 R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
429 00139     CONT & NOP1 & yinc2 c0,r0 & KEN
430
431           ;save current NDL on GS for readback compare
432 0013A     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                    ;~NDL to GS
433 0013B     PSDRS & NOP1 & yinc2 c0,r0 & KEN
434
435           ;retard page pointer by BSIZ
436 0013C     CONT & KA BSIZ & YSUB1 C0,B3,R0 & yinc2 c0,r0 & KEN
437
438 0013D     JDA & KA MOPUP & NOP1 & yinc2 c0,r0 & KEN               ;write page to seq
439
440           CASE2:
441           ;SS underflow by pop,xxx,psh(rtn addr), i.e., rtn addr @ loc 00
442
443           ;get NDL
444 0013E     PPRSD & KA PNDL & YRTR1 R0 & DSEL1 & yinc2 c0,r0        ;clear SSP from GS
445 0013F     CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL          ;read [PNDL]
446 00140     CONT & YRTR1 R0 & DSEL1 & yinc2 c0,r0 & KEN             ;into R0
447
448           ;move locations 3F and 0 into data mem (in that order)
449 00141     WRSSP & KA H#003F & NOP1 & yinc2 c0,r0                  ;point SSP to 3F
450 00142     PPSSD & YINC1 C0,R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
451                   ;wr [3F] to mem
452 00143     CONT & NOP1 & yinc2 c0,r0 & KEN                         ;point SSP to 00
453
454           ;move RTI address [00] to data mem without inc on R0 (NDL),
455           ;thereby wiping RTI on next pagefault.
456 00144     WRSSP & KA ZERO & NOP1 & yinc2 c0,r0
457 00145     PPSSD & YRTR1 R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
458 00146     CONT & NOP1 & yinc2 c0,r0 & KEN
459
460           ;save current NDL on GS for readback compare
461 00147     CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                    ;~NDL to GS
462 00148     PSDRS & NOP1 & yinc2 c0,r0 & KEN
```

```
463
464           ;retard page pointer by BSIZ+1 (for one extra datum paged to data mem)
465 00149      CONT & KA BSIZ+1 & YSUB1 C0,B3,R0 & yinc2 c0,r0 & KEN
466
467 0014A      JDA & KA MOPUP & NOP1 & yinc2 c0,r0 & KEN                ;write page to seq
468
469           CASE3:
470           ;SS underflow by pop,psh,psh(rtn addr). i.e., rtn addr @ loc 01
471
472           ;get NDL
473 0014B      PPRSD & KA PNDL & YRTR1 R0 & DSEL1 & yinc2 c0,r0         ;clear SSP from GS
474 0014C      CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL           ;read [PNDL]
475 0014D      CONT & YRTR1 R0 & DSEL1 & yinc2 c0,r0 & KEN              ;into R0
476
477           ;move locations 3F, 0, and 1 into data mem (in that order)
478 0014E      WRSSP & KA H#003F & NOP1 & yinc2 c0,r0                   ;point SSP to 3F
479 0014F      PPSSD & YINC1 C0,R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
480                  ;wr [3F] to mem
481
482 00150      CONT & NOP1 & yinc2 c0,r0 & KEN                          ;point SSP to 00
483 00151      WRSSP & KA ZERO & NOP1 & yinc2 c0,r0
484 00152      PPSSD & YINC1 C0,R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
485                  ;wr [00] to mem
486
487           ;move RTI address [01] to data mem without inc on R0 (NDL),
488           ;thereby wiping RTI on next pagefault.
489 00153      CONT & NOP1 & yinc2 c0,r0 & KEN                          ;point SSP to 01
490 00154      WRSSP & KA H#0001 & NOP1 & yinc2 c0,r0
491 00155      PPSSD & YRTR1 R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML
492
493           ;save current NDL (~NDL) on GS for readback compare
494 00156      CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                     ;~NDL to GS
495 00157      PSDRS & NOP1 & yinc2 c0,r0 & KEN
496
497           ;retard page pointer by BSIZ+2 (for two extra datums paged to data mem)
498 00158      CONT & KA BSIZ+2 & YSUB1 C0,B3,R0 & yinc2 c0,r0 & KEN
499
500           ;fallthrough to MOPUP
501
502           MOPUP:
503           ;actual pageback routine
504
505           ;test if time to slam things, i.e., if stack is empty enough
506           ;to go ahead and clear out the paging area.
507           ;note that in so doing, R0 gets blown away (hence ~NDL is on GS)
508
509           ;is R0(ndl')<=PAGE?, if so then time to initialize...
510 00159      CONT & KA PAGE & YRTC1 C0,R0 & DSEL1 & yinc2 c0,r0 & KEN
511 0015A      CONT & KA ZERO & YSUB1 C0,B3,R0 & yinc2 c0,r0 & KEN & MUX CZ1F
512 0015B      JDA FL & KA EMPTY & NOP1 & yinc2 c0,r0
513
514           ;fallthrough means not close enough to slam, so do the usual...
515
516           ;pop ~NDL from GS and write to C0 as pageback limit
517 0015C      PPRSD & NOP1 & yinc2 c0,r0 & KHOLD                       ;~NDL to data bus
518 0015D      CONT & YRTC1 C0,R0, & DSEL1 & yinc2 c0,r0                ;and to C0
519
520           ;push R0 (NDL') on GS for update during DONE.
521 0015E      CONT & RTD1 R0 & yinc2 c0,r0 & KHOLD                     ;NDL' to data bus
522 0015F      PSDRS & NOP1 & yinc2 c0,r0 & KEN                         ;NDL' to GS
523
524           ;initialize SSP to location 3E
```

```
525 00160     WRSSP & KA H#003E & NOP1 & yinc2 c0,r0 & KEN
526
527           ;skip around EMPTY to LOOP3
528 00161     JDA & KA LOOP3 & NOP1 & yinc2 c0,r0                    ;write page to seq
529
530           EMPTY:
531           ;time to slam things...
532
533           ;pop ~NDL from GS and write to C0 as pageback limit
534 00162     PPRSD & NOP1 & yinc2 c0,r0 & KHOLD
535 00163     CONT & YRTC1 C0,R0 & DSEL1 & yinc2 c0,r0 & KEN
536
537           ;force R0 to PAGE and also push it on GS as NDL'
538 00164     PSDRS & KA PAGE & YRTR1 R0 & DSEL1 & yinc2 c0,r0 & KEN
539
540           ;initialize SSP to location zero
541 00165     WRSSP & KA ZERO & NOP1 & yinc2 c0,r0
542
543           ;fallthrough to LOOP3
544
545           LOOP3:
546           ;write block of paged data into SS (starting with location 3F)
547 00166     CONT & YRTR1 R0 & yinc2 c0,r0 & LTOK & RD1 TOL         ;read a paged datum
548 00167     PSDSS & YINC1 C0,R0 & yinc2 c0,r0 & KEN & MUX CZ1F     ;and push onto SS
549 00168     JDA NF & KA LOOP3 & NOP1 & yinc2 c0,r0                 ;until flag
550
551           ;fallthrough to DONE
552
553           DONE:
554           ;update [PNDL] with NDL' (saved on GS)
555 00169     CONT & NOP1 & yinc2 c0,r0 & KEN                        ;enable KA field

556 0016A     CONT & KA PNDL & YRTR1 R0 & DSEL1 & yinc2 c0,r0              ;get NDL pointer
557 0016B     PPRSD & YRTR1 R0 & yinc2 c0,r0 & KHOLD & KTOL & WR1 FROML    ;write NDL'
558
559           ;restore stuff
560 0016C     PPRSD & NOP1 & yinc2 c0,r0 & KHOLD                     ;pop old B0
561 0016D     PPRSD & YRTB1 B0,R0 & DSEL1 & yinc2 c0,r0 & KHOLD      ;restore it, pop C0
562 0016E     CONT & YRTR1 R0 & DSEL1 & yinc2 c0,r0                  ;to R0
563 0016F     PPRSD & YRTC1 C0,R0 & yinc2 c0,r0 & KHOLD              ;then to C0, pop R0
564 00170     PPRSD & YRTR1 R0 & DSEL1 & yinc2 c0,r0 & KHOLD         ;restore it, pop CR
565 00171     SLSP & DTCR1 & yinc2 c0,r0 & KEN                       ;restore it, select LSP
566 00172     SLRIVP & KA INIT & NOP1 & yinc2 c0,r0                  ;clear calling IR9
567
568           ;turn on KEN for resumed calling code in case it's a push
569           ;WARNING! this could be catastrophic if first resumed line is not a PUSH
570
571 00173     RTNIR & NOP1 & yinc2 c0,r0 & KEN                       ;jump back, jack
572
573           END


 TOTAL ASSEMBLY ERRORS =      0
```

Using the Counters of the ADSP-1401 Program Sequencer for Loop and Event Counting

by Bob Fine

INTRODUCTION

The ADSP-1401 has four internal, independent, 16-bit counters that are provided for maintaining loops and event tracking. These counters (denoted as C0, C1, C2, and C3) hold twos complement values that may be decremented or preloaded through dedicated instructions. This application note investigates these instructions and shows how to initialize the counters and perform conditional jump instructions based on the value of the sign of the counters. Event counting, in conjunction with interrupt upon counter underflow, is also investigated. Examples show the initialization of the register stack with jump addresses for conditional loops and the initialization of the interrupt logic.

COUNTER OPERATION

The sign bit of the most recently used counter is always saved in the status register (SR1). Simultaneously, the sign bit is also available to control various conditional instructions or for asserting the lowest priority interrupt, IR0, upon counter underflow. In general, actions are taken by the ADSP-1401 when the sign bit (most significant bit of the counter) is a "1" (count value is negative). In most cases, the sign bit is first checked and then the counter is decremented.

Timing

Depending on the specific instruction being executed, the sign of the count value is checked at either the status register bit SR1 or at the counter itself. Whenever an instruction which effects the value of the counter is executed, the sign bit of the counter is transferred to the status register bit SR1 before the count change takes place. The timing diagram of Figure 1 details the series of events for this type of instruction.

Due to the Look-Ahead pipeline of the ADSP-1401, the instruction is pre-decoded before the rising edge of the clock. During this time the ADSP-1401 recognizes the instruction as one that will effect the counter. The rising edge of the clock occurs and the sign bit of the counter is first transferred into the status register bit SR1 and then the counter is decremented.

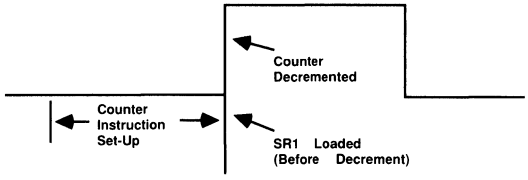


Figure 1. Timing for Counter Decrement

Conditional Instructions

There are seven conditional jump instructions where the sign of the specified counter can be selected as the condition. These instructions look for the sign bit in the status register (SR1) rather than at the specified counter itself. The value of the counter is not affected by these instructions. In most cases, these instructions will be used in conjunction with a decrement counter (DCCNTR) instruction. These seven conditional instructions are listed in Table 1.

JTWO	If Condition: Jump PC+2
JDA	If Condition: Jump Data, Absolute
JDR	If Condition: Jump Data, Relative
JDI	If Condition: Jump Data, Indirect
JSA	If Condition: Jump Subroutine, Absolute
JSR	If Condition: Jump Subroutine, Relative
RTN	If Condition: Return From Subroutine

Table 1. Conditional Jump Instructions

There are 3 conditional jump instructions that explicitly jump only on the sign being true ("1"). These instructions are listed in Table 2 which is located on the following page.

JDRST	If Sign: Jump Data and Reset Count
JRS	If Sign: Jump Register
BRANCH	If Sign: Jump Register; If Condition: Jump Data

Table 2. Jump on Sign and Decrement Instructions

In addition to checking the sign bit of the counter, these instructions also decrement the count value. Whenever the value of the specified counter is changed, the state of the old sign bit is saved in the status register before the counter is decremented. Any operation, be it decrementing or transferring the sign bit to the status register, occurs on the rising edge of the clock.

PROGRAM EXAMPLES

The following program examples illustrate how the instructions listed above are used. (For a detailed description of these instructions, see the ADSP-1401 Data Sheet.) For repeated operations such as a "DO UNTIL" type loop, where a task is to be repeated *until* the count expires, the counter is initialized with a positive value and decremented until the count becomes negative or the sign becomes true ("1"). The example of Table 3 shows the counter decrementing from positive values to a negative value.

Hex Counter Value	*Sign Bit*
0004	0 (Positive)
0003	0 (Positive)
0002	0 (Positive)
0001	0 (Positive)
0000	0 (Positive)
FFFF	1 (Negative)

Table 3. Decrementing Positive Count Values

With DO UNTIL type loops, the initial count value will depend on where the jump instruction is placed in the loop. In general, if the jump is placed at the top of the loop (before task execution), the counter is initialized with N-1. If the jump is placed at the bottom of the loop (after task execution), the counter is initialized with N-2 where N is the number of iterations desired. Specific examples will be shown.

For repeated operations such as a "DO WHILE" type loop, where a task is to be repeated *while* the sign is true ("1"), an initial negative count value can be used. The counter is decremented until the negative value wraps around to a positive value as illustrated in Table 4.

Hex Counter Value	*Sign Bit*
8005	1 (Negative)
8004	1 (Negative)
8003	1 (Negative)
8002	1 (Negative)
8001	1 (Negative)
8000	1 (Negative)
7FFF	0 (Positive)

Table 4. Decrementing Negative Count Values

As fully described in the ADSP-1401 Data Sheet, the counter should be initialized with 2^{15} + (N-2) for the type of loop shown above where the loop iterates until the sign becomes ("0").

Loop Example 1

The following example illustrates the use of a conditional instruction, where the sign is specified as the condition in conjunction with an explicit counter decrement. A subroutine ending in a conditional return instruction is to be repeated a specified number of times. The example program is shown in Table 5.

Microprogram Memory Address		*Instruction*
SUB:	200	CONT
	201	CONT
	202	CONT
	203	DCCNTR C1
	204	RTN
	205	JDR

Table 5. Conditional Return from Subroutine Example

The subroutine starting at label SUB (location 200) is to be repeated 6 times (N=6). The counter C1 is loaded with N-2 or 4. At the end of the execution of the subroutine, the counter C1 is decremented by the use of the DCCNTR instruction. The sign bit of counter C1 is transferred to the status register then the counter is decremented. The RTN instruction looks at the sign bit in the status register and, if true ("1"), returns from the subroutine. If the sign bit is false ("0"), the execution continues with the JDR instruction which causes the sequencer to jump back to the top of the subroutine (SUB). A relative address of -6, which is placed on the data bus, is used to get back to the top of the routine. (PC increments to 206, 206 - 6 = 200. See the ADSP-1401 Data Sheet for details on JDR instruction.)

The flow chart of this example is shown in Figure 2.

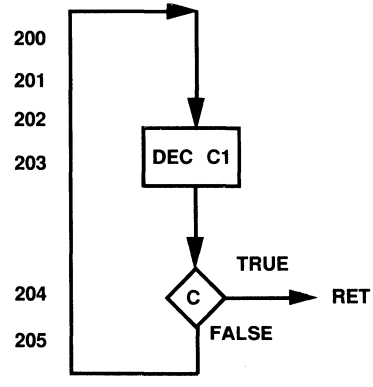


Figure 2. Flow Chart for Conditional Return Example

The subroutine starting at location 200 and ending at location 204 is repeated 6 times before the return is executed and program control is returned to the calling program.

The timing diagram for this example is shown in Figure 3.

It should be noted from the timing diagram that the sign bit of the counter is transferred to the status register (SR1)

Microprogram Memory Address		*Instruction*
	100	CONT
TOP:	101	CONT
	102	CONT
	103	JRS
	104	CONT
	105	CONT

Table 6. Example 2 Instruction Sequence

Instruction	DCCNTR	RET	DCCNTR	RET	DCCNTR	RET	DCCNTR	RET	DCCNTR	RET	DCCNTR	RET	
Counter Value 0004	0003		0002		0001		0000		FFFF		FFFE		
Status Bit SR1	0		0		0		0		0		1		
Loop Iteration		1		2		3		4		5		6	
Action		Fall Thru		Fall Thru		Fall Thru		Fall Thru		Fall Thru		RET	

Figure 3. Timing Diagram for Example 1

before the counter is decremented. The conditional RET instruction checks the sign bit at the status register (SR1).

It should also be noted that conditional instructions which check the sign at status register bit SR1 should be preceded by the DCCNTR instruction as opposed to having the DCCNTR follow. The reason for this is that the status register bit SR1 is loaded as a result of the counter operation. If a counter operation has not yet occurred and the conditional instruction is executed, it can not be guaranteed that the status bit will be valid. Having a counter instruction before the conditional instruction guarantees that the status register bit SR1 is valid.

All the conditional instructions, listed in Table 1, will operate in a similar manner as long as they are preceded by an instruction that updates one of the four counters.

Loop Example 2

The following example illustrates the use of the counter in conjunction with the JRS (Jump Register Sign) instruction. This instruction first tests the sign of the counter, C_i, and if true ("1"), outputs the register in RAM at the location (RSP+i), where i is given by the least significant instruction bits. The counter is always decremented after the test. If the sign is true, the jump is performed. If the sign is false ("0"), the next sequential microprogram address is output.

The example program is shown in Table 6.

The above example only shows the program sequencer instructions. Of course in a real system, other logic would be performing meaningful operations at the same time. The loop begins at location 101 and ends at location 103. This is illustrated in the flow graph of Figure 4.

It is desired to loop through the code 6 times and then to continue on with the remaining program. The JRS instruction is placed at the end of the loop to instruct the program sequencer to jump back to the top of the loop if the count has not yet expired or to fall out of the loop and continue executing the microprogram.

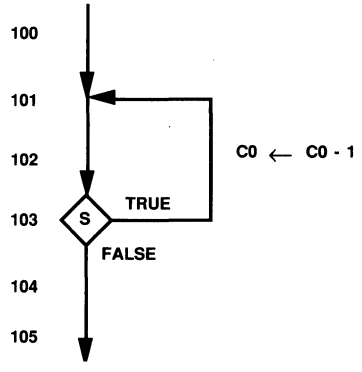


Figure 4. Example 2 Flow Graph

Before the loop can be executed, the top of loop address must be stored in the internal RAM of the program sequencer and a count value must be stored in one of the four counters. From the example program shown above, it can be seen that the top of loop address is 101 ; the initial counter value for N=6 (6 iterations) is 2^{15} + (6-2) which equals 8004 Hex. The initialization routine which sets up the register stack with the top of loop address and the counter with the count value is shown in Table 7.

Instruction	*Comment*
WRCNTR C0	Load Counter with 8004 Hex
WRRSP	Write Stack Pointer With Register Address
AIRSP 01	Add 1 to the Stack Pointer
PSDRS	Push Top of Loop Address on to Stack

Table 7. Initialization Routine for Example 2

The first instruction (WRCNTR) loads a value at the data port into counter C0. The data and instruction are both set up before the rising edge of the clock and are latched into the program sequencer by the rising edge of the clock.

The next three instructions set up the register stack pointer and load the register stack location with the top of loop address. The WRRSP instruction writes the register stack pointer with the value found at the data port. For this discussion the value 34 Hex is used. After the execution of the WRRSP instruction, the register stack pointer contains the value 34 Hex. If the top of loop address is to be written into location 34 Hex, the stack pointer must first be incremented. The reason for this is that the top of loop address is to be pushed on to the stack. A push first decrements the stack pointer then writes the stack. The stack pointer is incremented to point to location 35 Hex. The PSDRS instruction first decrements the stack pointer to 34 Hex and takes the value found at the data port and writes it to stack location 34 Hex. This sequence is illustrated in Table 8.

Instruction	*Register Stack Pointer*
WRRSP 34	34
AIRSP 01	35
PSDRS	34

Table 8. Stack Addressing Sequence

As shown, the top of loop address is loaded into stack location 34 Hex.

The example used counter C0; any counter can be used. If another counter is used, the program will differ a bit. The AIRSP instruction will add i+1, where i is the counter index (C_i). After the PSDRS instruction, the stack pointer must be rewritten to point to location 34 (example address). It should be noted that the JRS instruction used in this example (see Table 6) will access the register found at ram location RSP+i, where RSP is the value of the register stack pointer and i is the counter index (C_i).

The timing diagram for this example is shown in Figure 5.

Loop Example 3

Another method of terminating a loop or acting upon the occurrence of a specified number of events is to use the interrupt upon counter underflow feature of the ADSP-1401. Internal interrupt vector IR_0, the lowest priority of the 10 available interrupts, will be accessed upon the counter

Instruction		**JRS**	**JRS**	**JRS**	**JRS**	**JRS**	**JRS**
Count Value	**8004**	**8003**	**8002**	**8001**	**8000**	**7FFF**	**7FFE**
Counter Sign Bit (Before Decrement)		**1**	**1**	**1**	**1**	**1**	**0**
Loop Iteration		**1**	**2**	**3**	**4**	**5**	**6**
Action		**Jump Register**	**Jump Register**	**Jump Register**	**Jump Register**	**Jump Register**	**Fall Through**

Figure 5. DO WHILE Example 2, C=2^{15} + (N-2)

underflowing (going from positive to negative).

The following example illustrates the use of interrupt upon counter underflow to exit from a loop after a specified number of operations. The loop is exited and the interrupt service routine, referenced by IRV_0, entered. The program instructions for this example are shown in Table 9.

Microprogram Memory Address		*Instruction*
LOOP:	100	CONT
	101	CONT
	102	CONT
	103	DCCNTR C1
	104	JDR
	.	
	.	
INTRPT:	200	CONT
	201	WRCNTR
	202	CLRS
	203	RTNIR

Table 9. Interrupt upon Counter Underflow Example

The program listing of Table 9 shows a loop starting at location 100 and ending at location 104. Upon counter underflow, the loop will be exited and the interrupt service routine starting at location 200 will be entered. In this example, the interrupt routine reloads the counter, clears the status register sign bit and returns back into the loop. Note that clearing the sign bit also clears the interrupt request. A more useful application would involve more meaningful instructions in the interrupt routine. This simple program is used for example purposes only. The flow chart of Figure 6 further illustrates program execution.

The timing diagram of Figure 7 shows the value of the counter for each iteration of the loop. Note that the counter is initialized with N-2, where N is the number of iterations desired, and that the DCCNTR instruction is placed *after* the task or at the end of the loop.

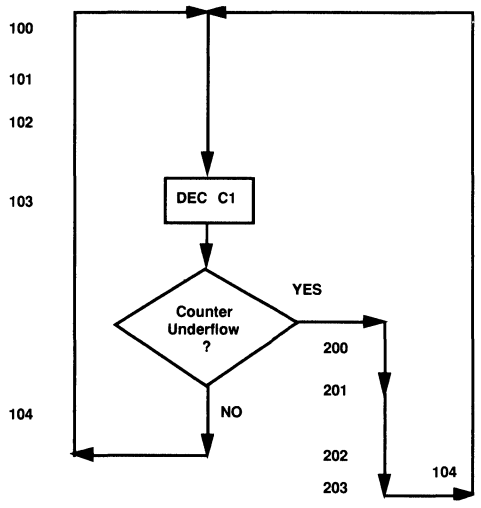


Figure 6. Flow Chart for Interrupt upon Counter Underflow Example

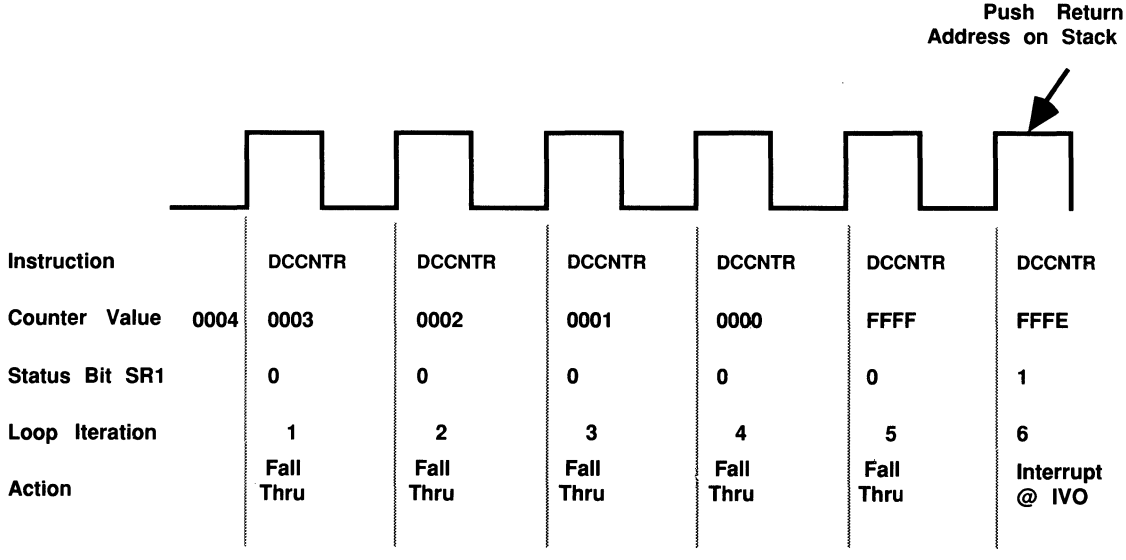


Figure 7. Timing Diagram for Example 3

The DCCNTR instruction results in the sign bit of the counter (MSB) to be checked rather than the status register bit SR1. The detailed timing for the execution of the DCCNTR instruction prior to the counter underflow interrupt is shown in Figure 8.

The ADSP-1401 performs the following sequence of operations when a DCCNTR instruction is presented to the instruction port.

1) Pre-decode the instruction (prior to rising clock edge)
2) Recognize instruction as DCCNTR
3) Look at sign bit of specified counter (pre-decrement)
4) If sign bit = 1 and interrupts enabled assert interrupt

----- Rising edge of clock occurs -----

5) Decrement Counter
6) Push PC on to stack
7) Output interrupt vector address

Due to the Look-Ahead pipeline architecture of the ADSP-1401, it can be decided before the rising edge of the clock if an underflow interrupt should be asserted. If the sign bit is set (count value is negative) and interrupts are enabled (IV0 not masked), an interrupt request is asserted (prior to the rising edge of the clock). The rising edge of the clock occurs and the counter is decremented. This is shown in the timing diagram of Figure 8.

Initialization

Before execution can take place, the interrupt vector pointer must be initialized, the interrupt vector IRV_0 must be loaded with the address 200, the interrupt mask must be properly set and the counter initialized. The instructions of Table 10 perform these necessary set-up tasks.

Microprogram Memory Address		*Instruction*
INIT:	10	SLRIVP IVP0
	11	WRIV 200
	12	WRSR FF84
	13	WRCNTR C1

Table 10. Interrupt/Counter Initialization Routine

The initialization program of Table 10 first sets the interrupt vector pointer to point to IVP0. The next instruction writes the address 200 into IVP0. The status register is then loaded with the Hex value FF84 which enables interrupts and also masks out all interrupts except IR0. Finally, the counter is initialized with the appropriate count value. If N is defined as the number of iterations before the interrupt

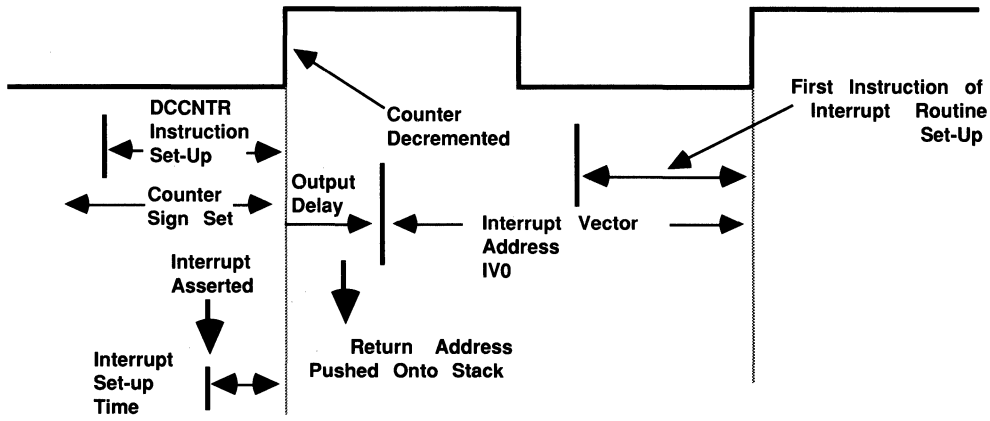


Figure 8. Interrupt upon Counter Underflow Timing for DCCNTR Instruction

occurs, the counter is initialized with the value N-2.

Event Counting Example

In this example, the IFCDEC instruction is used to decrement the counter upon the occurrence of some external condition presented to the ADSP-1401 via the flag pin. After a predetermined number of external events, the internal interrupt upon counter underflow is asserted. This technique of counting events is independent of utilizing loops. Anytime the IFCDEC is encountered and the external flag or condition is true, the counter will be decremented; the IFCDEC instruction need not be part of a loop. When the counter underflows, the interrupt IV0 is activated. As in loop Example 3, the interrupt logic must be initialized.

The timing associated with the IFCDEC and the interrupt upon counter underflow differs slightly from the interrupt in loop Example 3. The IFCDEC instruction must first look at the condition specified and then look at the sign of the counter to determine if an underflow status exists. The sign bit is checked not at the counter itself but at the status register bit SR1. The timing diagram, which details this operation, is shown in Figure 9.

Note that the instruction after the IFCDEC instruction is fetched even though the counter has underflowed. This one cycle of interrupt latency exists due to the way the sign bit is checked. Unlike the method used with the DCCNTR instruction (loop Example 3) where the sign of the counter causes the interrupt, the IFCDEC instruction uses the status

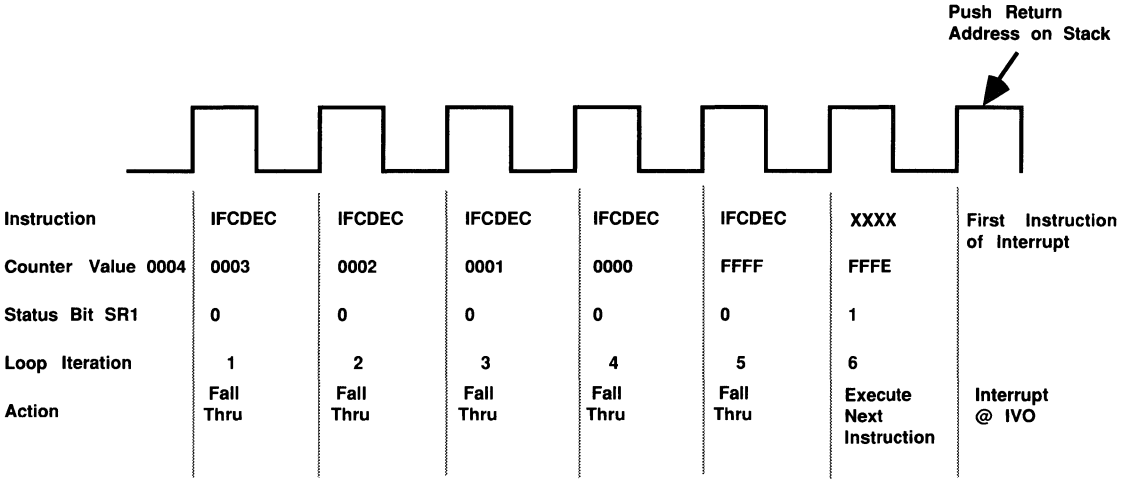


Figure 9. Timing Diagram for Event Counting Example

register bit SR1 to initiate the interrupt. This timing is detailed in Figure 10.

The ADSP-1401 performs the following sequence of operations when an IFCDEC instruction is presented to the instruction port.

1) Pre-decode the instruction (prior to rising clock edge)
2) Recognize instruction as IFCDEC
3) Look at state of specified condition

----- Rising edge of clock occurs -----

4) Transfer sign to status register SR1 (pre-decrement)
5) If condition true decrement counter
6) Output address for next instruction
7) Look at sign bit at SR1
4) If sign bit = 1 and interrupts enabled assert interrupt

----- Rising edge of clock occurs -----

6) Push PC on to stack
7) Output interrupt vector address

CONCLUSION

The four counters of the ADSP-1401 in conjunction with a powerful and flexible conditional instruction set, facilitate the use of compact looped code. The interrupting capabilities also enable effective event counting without the need for external count and compare logic.

Four examples of using the counters of the ADSP-1401 have been presented to give a representation of the techniques used for loop and event counting. These examples have been generalized so that they may be easily tailored for any application.

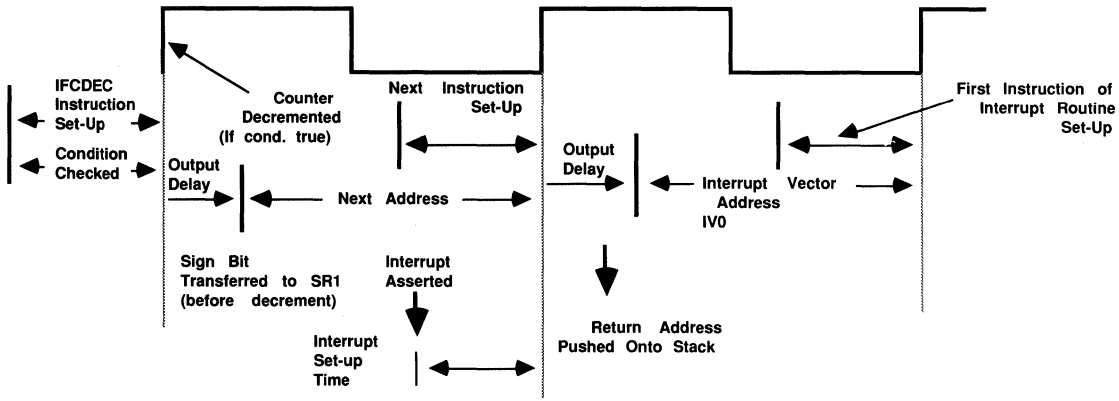


Figure 10. Interrupt upon Counter Underflow Timing for IFCDEC Instruction

Variable Width Bit-Reversing with the ADSP-1410 Address Generator

by Bob Fine

INTRODUCTION

In algorithms such as the FFT, bit-reversed addresses are required. The ADSP-1410 Address Generator contains a 16-bit hardware bit-reverser and can provide bit-reversed addresses with no external logic.

Addresses can be bit-reversed as they are output. The bit-reverse mapping is as follows, where K_i and Y_j denote the i^{th} bit of K (either an address register or the data bus) and the j^{th} bit of Y (the address port).

$K_0 \rightarrow Y_{15}$
$K_1 \rightarrow Y_{14}$
.
.
.
$K_{15} \rightarrow Y_0$

The hardware bit-reverser operates only on 16-bit addresses but the techniques described below allow proper bit-reversed access to data records less than 64K locations which reside anywhere within a larger data space.

ADSP-1410 IMPLEMENTATION

This application note describes the bit-reversing process for an N-bit wide address in general terms. Specific examples are given to clarify the process. The only restriction on the base address or starting address of the data record is that the least significant N bits be 0's.

YREV Instruction

The YREV instruction will output the selected address register (R) bit-reversed. The original (unreversed) R value is added to the selected offset (B) register, and written back into the specified R register. Condition testing is not performed. Bit-reversing affects only output data, not register contents. The bit-reversing process, regardless of the size of the data space being accessed, requires no extra overhead for address calculations.

Register Initialization

The task at hand is to determine the initial values for R and B registers to properly access the bit-reversed data.

In general, R will be initialized with the bit-reverse value of the first 16-bit data address (no matter where in memory it resides) and the B register will be initialized with 2^{16-N} (where 2^N is the size of the data space).

Variable Width Bit-Reversing

To illustrate the derivation of the generalized rules for bit-reversing, the example of data results from an eight point FFT is shown in Figure 1.

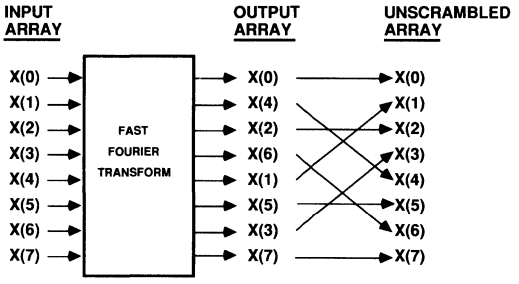


Figure 1. Results of 8-point FFT

As shown in Figure 1, the bit-reversing process results in data reads from location 0 first, followed by locations 4, 2, 6, 1, 5, 3, and 7. The first data point of the record is also the first bit-reversed data point since the bit-reverse of 000 is 000. In fact, no matter where the first data point resides in memory, it will always be read as the first address of bit-reversed addresses.

Three bits out of the 16-bit address provided by the ADSP-1410 are required to access the eight locations containing the transformed data points. These eight data locations may reside anywhere within a larger memory space. Only the three least significant bits need be bit-reversed with most significant address bits remaining unchanged. This is graphically illustrated in Figure 2 where an arbitrary base address has been selected.

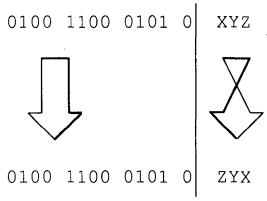


Figure 2. Desired Bit-Reversed Address

This can easily be done with the ADSP-1410 Address Generator. The first step is to calculate the bit-reverse value of the base address (4C50 Hex in this example). This is shown in Figure 3.

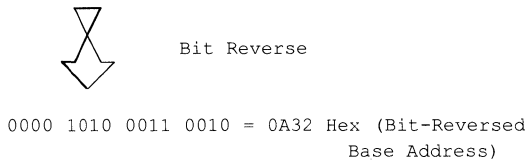


Figure 3. Bit-Reversed Base Address

When the bit-reversed base address is output by the ADSP-1410 with its bit-reverser enabled, the base address will be put back into its normal state. This bit-reversed base address is stored in one of the address (R) registers of the ADSP-1410.

To produce the bit-reversed least significant bits of the address, an offset corresponding to a data sample spacing of one is added to the bit-reversed base address stored in the R register. This offset value, for a data spacing of one, is 2^{16-N}, where 2^N is the size of the data area to be bit-reversed and is stored in one of the offset (B) registers of the ADSP-1410.

In this example N =3 and $2^{16-N} = 2^{13}$ = 2000 Hex.
This is shown in Figure 4.

```
   000 | 0 1010 0011 0010 = 0A32 Hex
+
   001 | 0 0000 0000 0000 = 2000 Hex
  -----------------------------------
   001 | 0 1010 0011 0010 = 2A32 Hex
```

Figure 4. Bit-Reversed Base Address Plus Offset

The offset value is essentially the value 001 (shifted left the appropriate number of bits to fit in the most significant bit positions). When the full 16-bit address is hardware bit-reversed, the most significant N bits are put into the least significant N bit positions (in a bit-reversed fashion) and the pre-reversed base address is output as the base address. This is shown in Figure 5.

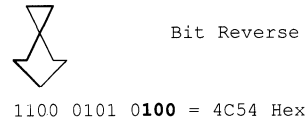


Figure 5. 16-Bit, Bit-Reversed Address

EXAMPLES

To further clarify the variable width bit-reversing process, several examples will be shown using the eight point FFT with the eight data points at the beginning of memory, at the end of a 64K memory, and at an arbitrary place in the middle of memory.

Beginning of Memory

Here a base address of 0000 Hex is used. ADSP-1410 instructions are shown with the value of the address being output, the results of the R+B operation, and, the contents of R. All address values are shown in Hexidecimal format.

ADSP-1410 Instruction	*R register Contents*	*Bit-Reversed Output*	*R+B*
YREV	0000	0000	2000
YREV	2000	0004	4000
YREV	4000	0002	6000
YREV	6000	0006	8000
YREV	8000	0001	A000
YREV	A000	0005	C000
YREV	C000	0003	E000
YREV	E000	0007	

Note that with the method described, the least significant three-bit field is properly bit-reversed and the upper order base address bits are zero (not bit-reversed).

Middle of Memory

This example shows the same eight point FFT results with a base address of 1000 Hex.

Register R is initialized with the bit-reverse value of 1000 Hex (0008 Hex) and B is initialized with 2000 Hex as before.

ADSP-1410 Instruction	*R register Contents*	*Bit-Reversed Output*	*R+B*
YREV	0008	1000	2008
YREV	2008	1004	4008
YREV	4008	1002	6008
YREV	6008	1006	8008
YREV	8008	1001	A008
YREV	A008	1005	C008
YREV	C008	1003	E008
YREV	E008	1007	

End of Memory

Finally, the eight data points are placed at the end of memory. This example shows the first data point to be at location FFF8 Hex. R is initialized with the bit-reverse of FFF8 Hex (1FFF Hex) and B is initialized with 2000 Hex.

ADSP-1410 Instruction	*R register Contents*	*Bit-Reversed Output*	*R+B*
YREV	1FFF	FFF8	3FFF
YREV	3FFF	FFFC	5FFF
YREV	5FFF	FFFA	7FFF
YREV	7FFF	FFFE	9FFF
YREV	9FFF	FFF9	BFFF
YREV	BFFF	FFFD	DFFF
YREV	DFFF	FFFB	FFFF
YREV	FFFF	FFFF	

Working with Complex Data

In some cases, it may be desirable to interleave real and imaginary data as shown below.

Loc	*Data*
N	Real
N+1	Imaginary
N+2	Real
N+3	Imaginary
N+4	Real
.	.
.	.
.	.
.	.

To properly bit-reverse interleaved data and keep the real imaginary data pairs intact, the offset stored in the B register should be:

$$\frac{2^{16-N}}{2}$$

where 2^N is the size of the data buffer.

CONCLUSION

The ADSP-1410 is a general purpose address generator which provides great flexibility in many data addressing tasks. An internal hardware bit-reverser simplifies the task of providing bit-reversed addresses for algorithms such as the FFT. By using the ADSP-1410's internal hardware bit-reverser (YREV instruction), zero-overhead, single-cycle, bit-reversing of address spaces less than 64K (less than 16 bits) can be achieved. The initialization of the R and B registers are as follows:

R = bit-reverse of base address

$B = 2^{16-N}$,

where 2^N is the size of data space.

Implement a Cache Memory in Your Word-Slice™ System

by Bob Fine

INTRODUCTION

The task of designing a high-speed processor is simplified with Analog Devices' Word-Slice family of components. In these high speed processors, memory management for both program and data memories becomes more complex as speed and memory size increases. Part of this memory management is cache control. Several features of the ADSP-1401 Program Sequencer simplify the cache update task and minimize the need for extra circuitry.

It is often desirable to have a processing system with two basic memory sections; one small, limited amount of high-speed (cache) memory that can be accessed directly by the processor and a much larger and slower memory that contains the complete program or data image.

The user program or data base can be written as though a large area of memory is available to the processor but in reality resides in some secondary storage. Small pages of the larger image are stored in the smaller high-speed memory cache available to the processor. When the processor attempts to access a location not residing in the cache memory, a "cache miss" is detected and program execution is temporarily suspended while a new section of memory is loaded into the cache.

CACHE CONTROL

A trap feature of the ADSP-1401 Program Sequencer supports asynchronous interrupts which are generated by the cache memory controller when a cache miss is detected. The interrupt vector address points to a download routine which will update the appropriate data or program cache depending on where the cache miss was initiated.

The application described details the use of the program sequencer and address generator with a cache memory. If the cache miss is caused from a program memory access, the sequencer is utilized to supply program memory addresses for download. If the cache miss is caused from a data memory access, the address generator is used to supply data memory addresses for download.

After the download is complete, operation must resume from the microprogram or data memory address that caused the cache miss. The sequencer is able to save this address on its internal stack before the interrupt service routine is entered. When the address generator is used to supply addresses for data cache update, an external register is needed to store the return address (next location to be fetched). This process is described in detail in the following sections.

Hardware Description

The block diagram shown in Figure 1 details the interconnection of the program sequencer, program cache, program cache controller, and memory (or host) interface for handling a program memory cache miss and the interconnection of the address generator, data cache, data cache controller, and memory (or host) interface for handling a data memory cache miss.

Trap Signals

The program and data cache controllers monitor the address busses to produce trap signals on a cache miss for either the program or data memory. The TTR input pin of the program sequencer serves as a multipurpose input. Reset, trap, and output enable functions are accessed by this single pin. A time division multiplexing scheme is used to differentiate between the three functions. The program sequencer looks at the TTR signal during the first half of each clock period and interprets it as an output enable control signal (HI = Output Disabled, LO = Output Enabled). During the second half of the clock period, it is interpreted as a trap signal (Active High). A reset is recognized when both trap and three-state output controls are active for a complete cycle (a trap is not allowed to occur when the output of the sequencer is disabled). A reset must be active (high) for at least three cycles to insure that all internal states are correctly reset. (See the ADSP-1401 Data Sheet.)

The TTR multiplexer selects the trap or output control signal with the clock line so that the output control is passed

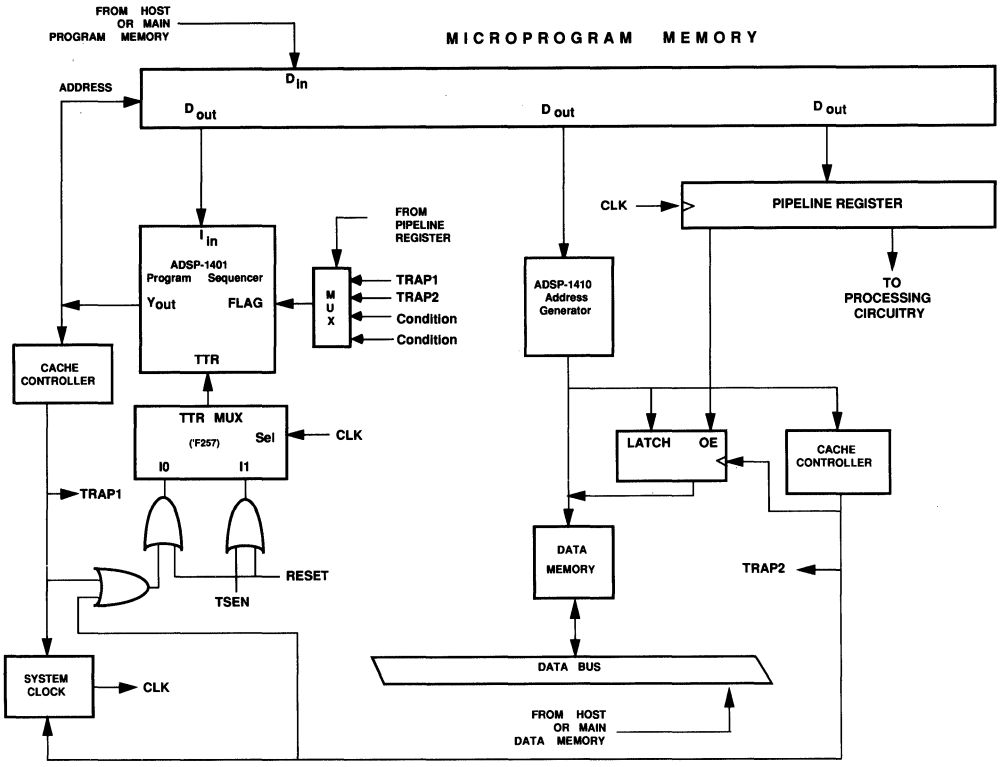


Figure 1. Block Diagram of System with Cache

during the first half of the clock and the trap is passed during the second half.

In the case shown, where both data and program memory may produce a cache miss, the two trap signals are also fed to the condition code multiplexer which feeds the flag input of the sequencer. The cache update service routine then instructs the sequencer to perform a jump on condition and conditionally service either cache miss.

The two cases of cache miss (program memory and data memory) are described separately, detailing both hardware and software. Even though both types of cache error handling are similar, the topics are treated individually to focus on the differences in programming techniques.

PROGRAM CACHE MEMORY

Small sections of the total program image are made available to the processor by using a cache memory. This cache memory has a much faster access time than the secondary storage holding the total program image (main memory). By utilizing this small, high-speed memory section as a cache instead of a large high-speed memory, a tremendous cost savings can be realized. The task of transferring program code from the slower main memory to the smaller high-speed cache at the appropriate time is handled by a cache memory controller. It should be noted that downloading the cache memory presents a time overhead. If cache memory is to be used and if a routine can not be interrupted, a block size should be chosen so that the complete routine will fit into program memory, otherwise a cache memory should not be used in time critical routines.

Cache Controller

To implement a cache memory system, a cache controller is needed. The cache controller monitors the addresses output by the program sequencer or address generator and flags cache misses. A comparator, internal to the cache controller, detects when a block not residing in the cache memory is addressed.

Before the sequencer can be used with the cache controller, interrupt vector, IV9, must be initialized with the address of the cache management routine. Care must be taken when using interrupt IV9 since this interrupt is also

used for stack overflow. (See the ADSP-1401 Data Sheet.) A trap input causes an asynchrounous, non-maskable interrupt. This means that the interrupt vector address is output by the sequencer immediately upon recognition, independent of the clock edge. There exists a 30ns delay time from trap input to interrupt vector address output so the system clock must be suspended (cycle stretch) to allow time to access the new instruction (first instruction of the interrupt routine specified by IV9) before the next rising edge of the clock. As shown in the block diagram, the cache controller is connected to the system clock for this reason.

Figure 2 shows the timing for the trap interrupt in comparison to a normal fetch cycle. The sequencer takes a time period, T_{seq}, to output the microprogram address. A time period, T_{acc}, is then needed to access the microprogram location. The top portion of Figure 2 illustrates this normal instruction fetch sequence. In the case of a cache miss, the sequencer will output an address as in a normal instruction fetch but after a time period, T_{cc}, the cache controller decides that the microprogram address is invalid. The cache controller produces the trap signal and the sequencer requires a time period, T_{int}, to output the interrupt vector address pointing to the cache update service routine. Once this address is output, the microprogram memory has to be accessed and another time period of T_{acc} is required. As Figure 2 shows, the clock must be stretched to account for the extra delay.

When the clock is held low, the system is suspended since no clock edge occurs. Care must be taken that no glitches occur on the clock line during this clock stretch procedure.

The trap interrupt causes the last address (the address which caused the cache miss) to be pushed on the sequencer stack. The interrupt vector stored in IV9 is then output pointing to the cache download routine in microprogram memory. All system circuitry not involved in the update procedure is suspended while the cache download occurs.

The interrupt vector, IV9, is reserved for both trap and stack overflow. If a system utilizes both these functions, the interrupt service routine must determine the source of the interrupt. A typical application implements a branching technique in the interrupt service routine. The flag input of

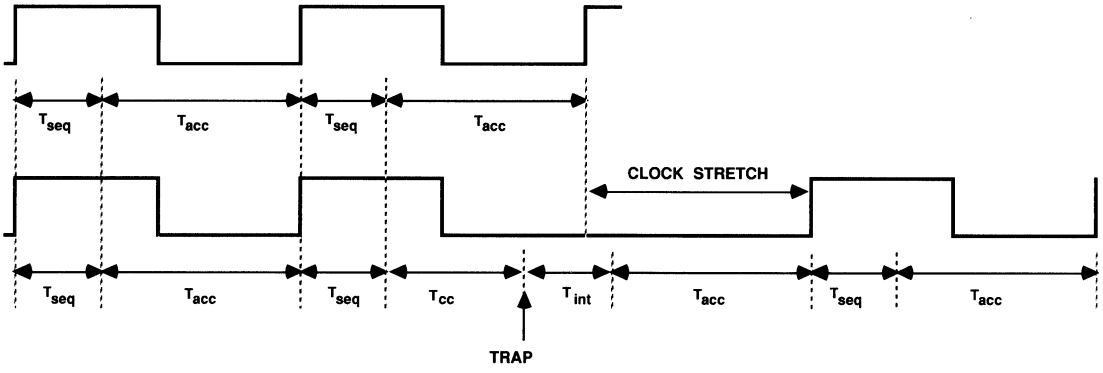


Figure 2. Cycle Stretch Timing

the sequencer accepts signals routed through a condition code multiplexer. Condition codes are systematically checked to determine the source of the interrupt and the appropriate jump is taken to properly service the interrupt. The download is accomplished by the use of the WCS sequencer instruction. (See "Implement A Writeable Control Store In Your Word-Slice™ System" Application Note.)
The last instruction of the download routine must be a return from interrupt (RTI) which pops the sequencer stack, thus resuming program flow transparent to the cache download. The flow chart in Figure 3 illustrates this sequence.

DATA CACHE MEMORY

The data cache memory is implemented in much the same way as the program cache memory. Since data memory is independent of the program memory, the consequences of changing its contents are different. No instructions in program memory have to be changed, only data in data memory. The sequencer merely jumps to an interrupt routine that services the data memory cache at the time of a cache miss to reload the data memory. Since the trap function is used to detect a data memory cache miss, the system clock cycle has to be stretched in the same manner as for the program memory cache update.

Normally the address generator updates address registers in the same cycle in which an address is output. (See the ADSP-1410 Address Generator Data Sheet.) The address which causes a cache miss must be latched externally since the contents of the internal register have already been updated (the current address will be needed when normal operation resumes). In addition to any pointers being used for normal data memory access, an extra pointer or address generator register is required to be used during cache memory update.

After the download is complete, data accesses from memory must resume as if no cache miss had occurred; the first memory address must come from the external latch. The first address generator instruction after the service routine must be a NOP. The NOP instruction disables the output port of the address generator avoiding contention while the external latch provides the proper data memory address. The second memory access after cache memory update utilizes the internal register of the address generator resuming normal address generator operation. The output of the external latch is also disabled.

CONCLUSION

With minimal external circuitry and the use of the TRAP function of the ADSP-1401 Program Sequencer, cache memory control can be simplified allowing for higher-speed, low-cost systems compared to systems using conventional memory.

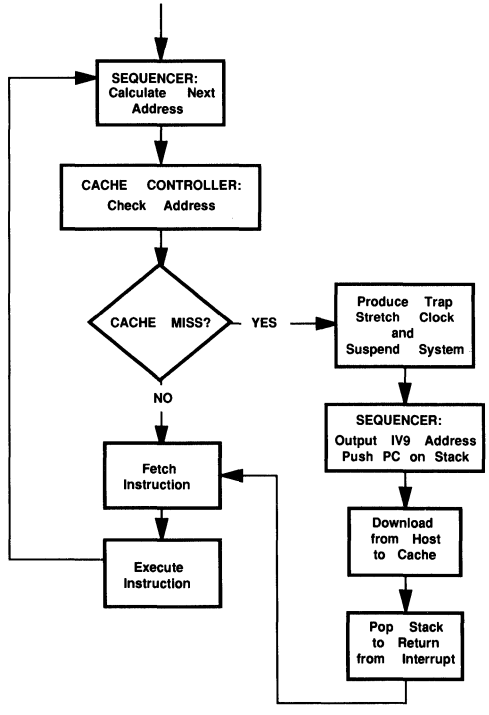


Figure 3. Flow Chart for Cache Update Sequence

Implement A Writeable Control Store In Your Word-Slice™ System

By Bob Fine

INTRODUCTION

The ADSP-1401 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.

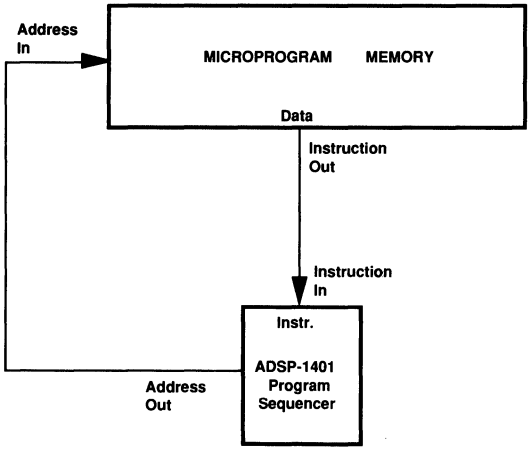


Figure 1. Program Sequencer Connection to Microprogram Memory

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 (25 - 50 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-1401 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, 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.

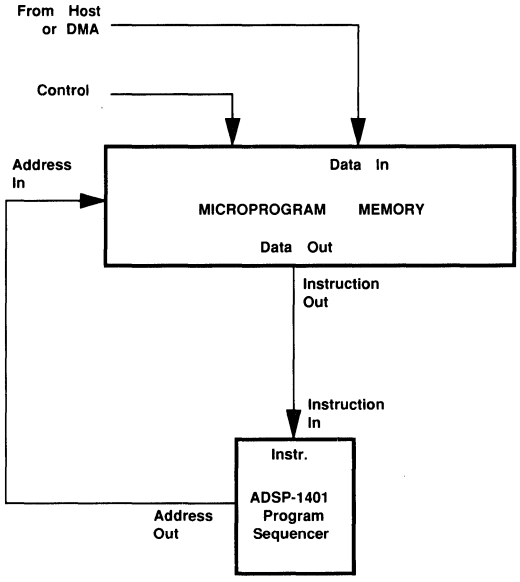


Figure 2. Host Data and Control Connections for WCS

The following description contains the details of implementing a writeable control store with the ADSP-1401 and some RAM, and details the initialization procedures for loading and starting program execution.

Two examples are described showing:

1) 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.

2) 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.

The ADSP-1401 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.

WCS INSTRUCTION

A sequencer instruction, dedicated for writeable control store download (WCS), is provided to facilitate the implementation of the above mentioned architectures. The WCS instruction will put the sequencer into a mode in which sequencial addresses are output every clock cycle and instruction inputs are ignored. The flag input may be used for handshaking.

The WCS instruction works in the following manner; the WCS instruction code (20 Hex) is presented to the instruction input of the program sequencer and a starting address value is presented to the data input (the flag input should be at a normally low logic state). Once the instruction is clocked into the sequencer by the rising edge of the system clock, the instruction and data inputs are ignored. Thus, the WCS instruction and starting address need only be presented to the sequencer for one cycle (but are allowed to remain for several cycles since these ports are ignored during WCS execution).

The program sequencer will output the starting address, which was fed to it through the data port, and will wait for the flag input to go to a logic high level. The flag input can be held high for full speed operation or the flag can be used for handshaking with a slow host. If handshaking is used, the flag is asserted (brought to a high logic level) when data is ready and the program sequencer is to increment its output address.

The instruction input will continually be ignored while the sequencer outputs sequential addresses. Depending on the termination technique used, the sequencer will continue to increment address values every cycle that the flag is high until either an internal or external interrupt occurs.

In most cases, the flag input is also used as a condition code input for system circuitry and a condition code multiplexer selects which signal is to be fed to the flag input.

There are two methods of terminating download utilizing the program sequencer's interrupt circuits. One is to use the internal counter register C0 to count a specific number of memory locations to be filled and have the sequencer stop automatically via an internal interrupt. The second is to interrupt the sequencer via the external interrupt pins. With both these methods, an interrupt vector address must be programmed into the sequencer; and with the first method, a count value must be put into C0. The count value must be *two less* than the number of instructions being downloaded. When the internal counter underflows, the interrupt vector IV0 is placed at the output of the sequencer during the next cycle. Since an interrupt pushes a return address on to the stack yet no real interrupt return exists, the stack must be popped after the interrupt jump occurs to clean-up the stack.

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

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.

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.

WRITEABLE CONTROL STORE WITH NO ROM

It is possible to build a Word-Slice system with no ROM by adding a minimal amount of circuitry to accommodate a writeable control store download. This hardware must present the WCS instruction and starting address to the sequencer to start the download and then assert the reset input to start program execution after download is complete. It should be noted that the first instruction after a reset must be a CONT. Therefore, the CONT should be the first instruction downloaded to location 0. This hardware is shown in Figure 3.

By decoding address bits of the host address together with the host write pulse a signal is produced (WCS START) which is used to initiate the download sequence. This signal enables the outputs of buffers which drive the instruction and data bus of the ADSP-1401. The WCS opcode (20 Hex) and the starting address (0000) are placed on the appropriate busses. The ADSP-1401 is put into the WCS

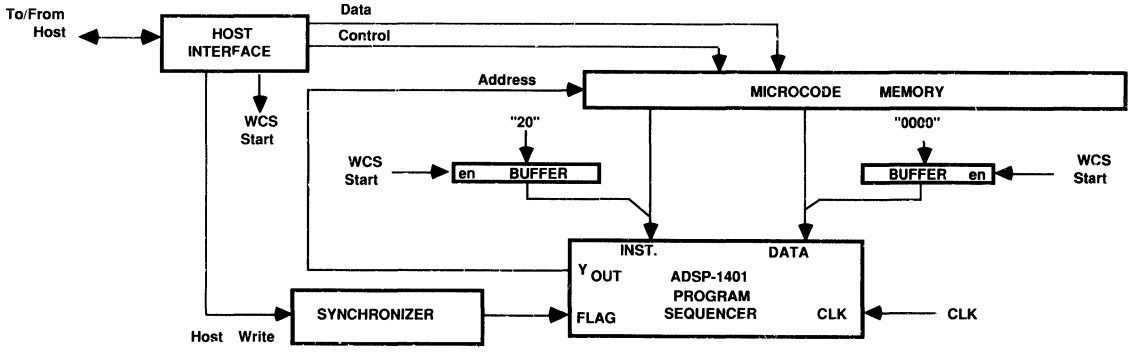


Figure 3. Block Diagram of WCS Circuit with No ROM

mode at the rising clock edge and after the first address is output awaits the assertion of the flag input.

The handshake logic shown in Figure 3 is used in both ROM and no-ROM WCS techniques. The timing for the handshake is shown in Figure 4.

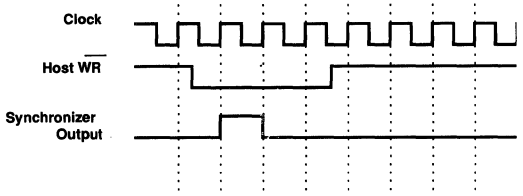


Figure 4. WCS Handshake Timing Diagram

This host write pulse must be synchronized to the Word-Slice circuit's clock. This is typically done with a D-type flip-flop. The synchronizer circuit must also limit the duration of the flag or synchronized host write to one clock cycle.

After the final memory location is filled, the host resets the sequencer, clearing the WCS and causing a jump to location 0 (the first instruction of the writeable control store). It should also be noted that since the microcode memory and the instruction buffer share the same bus, the output of the microcode memory must be disabled while the buffers drive the bus. The host must also control the write line of the microcode memory.

It should be noted that since neither the counter nor the interrupt vectors of the sequencer were initialized, the interrupting methods for terminating the download sequence can not be used. Once the writeable control store is filled, it is still possible to rewrite sections of the RAM by having a download routine similar to that of the ROM system included in the control software.

WCS HANDSHAKE SYNCHRONIZER FOR ADSP-1401

WCS selects a mode which greatly simplifies downloading programs to microcode memory from an external device, such as a microprocessor or microcontroller. In this mode, the ADSP-1401 outputs sequential addresses as the microcode RAM or Control Store is being written. A Flag input is needed for handshaking because, in almost all cases, this external device will download microcode at a rate much slower than the program sequencer's clock. Since this handshake signal will be asynchronous with the ADSP-1401's clock and usually will be longer in duration, a synchronization circuit must be used.

Figure 5 shows such a synchronizer circuit, actually a digital one-shot. The downloading device's handshake signal is synchronized to the Word-Slice system clock and its duration is limited to one clock cycle. The timing for this logic is shown in Figure 6. One timing constraint on the input handshake signal is that it be greater than the ADSP-1401

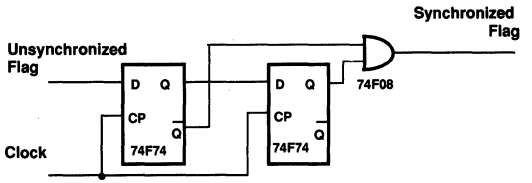


Figure 5. Flag Synchronizer Circuit

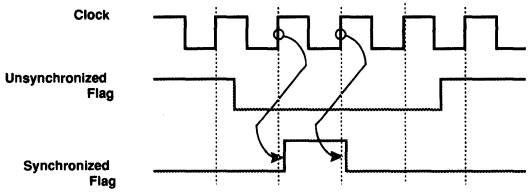


Figure 6. Flag Synchronizer Timing

clock cycle so that it is recognized before one of the clock edges. This signal must be a negative true logic level.

This circuit can also be used with the program sequencer's conditional jump and conditional branch instructions, for synchronizing any asynchronous external condition flag.

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. 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-1401 Program Sequencer simplifies the task of downloading to a writeable control store and minimizes the need for added external circuitry.

Optimize Data Transfers Between Word-Slice™ Components

by Steven Cox

INTRODUCTION

System designs using the ADSP-1401 Microprogram Sequencer and the ADSP-1410 Data Address Generator have the advantage of fast, flexible data transfers between system components. The 16-bit bidirectional data port on both of these Word-Slice components is key to this advantage. This application note discusses the use of these data ports as well as the design considerations required to operate the ports in an optimal configuration.

TYPICAL SYSTEM

Figure 1 shows a typical Word-Slice system. Here we have a general purpose configuration with an ADSP-1401 Program Sequencer for controlling program flow, an ADSP-1410 Address Generator for data memory address generation, a data memory (DM), and an arithmetic unit. Two separate 16-bit data buses join these functional units, the Constant Bus and the Data Bus. The Constant Bus connects the ADSP-1401's data port, the ADSP-1410's data port, and the microprogram constant field (CF) together with a bidirectional buffer. This buffer leads to the Data Bus which joins the arithmetic unit and data memory. Although we are discussing a specific system configuration, the ideas presented here apply also to various configurations. The abbreviation, (WS), refers to either of the Word-Slice components; the ADSP-1401 or the ADSP-1410.

WORD-SLICE DATA PORTS

The 16-bit bidirectional data ports on both the program sequencer and the address generator permit loading or

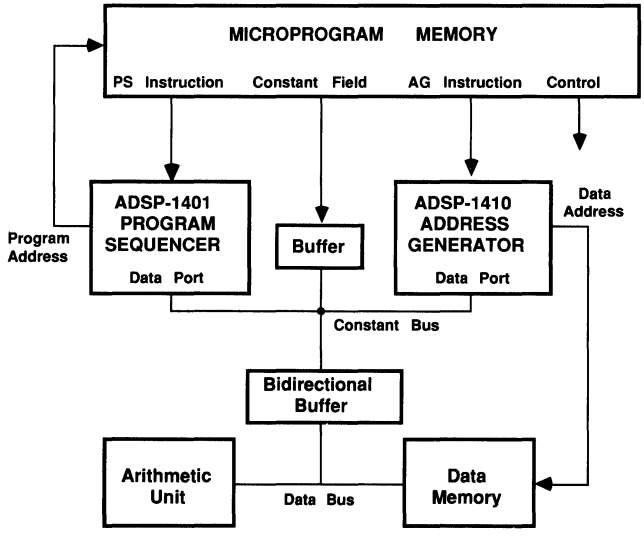


Figure 1. Word-Slice System Configuration

Word-Slice is a trademark of Analog Devices, Inc.

dumping of all internal registers. An internal input data latch is transparent during the clock low period allowing incoming data to be setup before the clock edge. Input data is latched for the entire first phase, (clock high). In turn, output data operations will drive the data port only during the clock high period. Whether a data input or output is done during phase one is instruction dependent, but the data port driver will always be in a high impedance state for the second phase, (clock low), allowing setup time for any subsequent inputs.

This I/O arrangement permits data to be output from the 1401 or 1410 during clock phase one, while allowing external input data setups, (for the next cycle), during phase two. A data read and write can occur in a single cycle. Therefore the number of overhead instructions for program

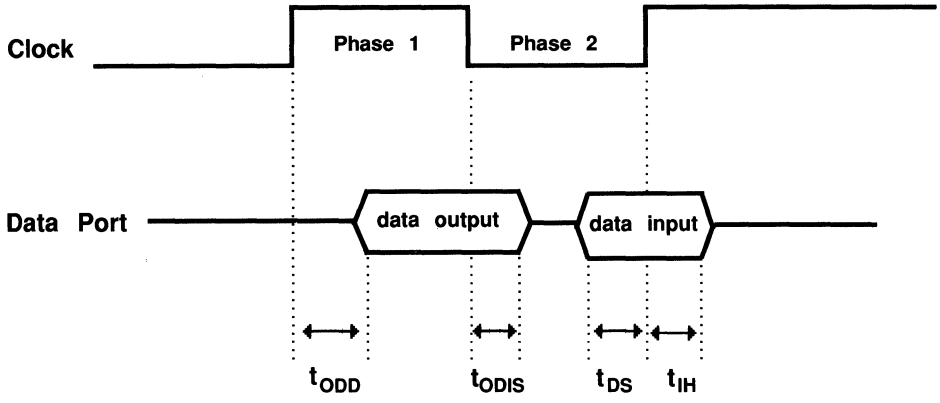


Figure 2. Data Port Timing

housekeeping is reduced and tighter loops can be written. Figure 2 is the timing diagram for the data port of the 1401 or 1410 while executing an input and output in one cycle. The timing parameters are as follows: t_{odd} is the clock to data delay; t_{odis} is the output data disable time; t_{ds} is the input data setup time; and t_{ih} is the input data hold time. See the ADSP-1401 and the ADSP-1410 Data Sheets for detailed timing information.

DATA TRANSFERS

Listed below are the five basic data transfers supported by our typical Word-Slice system. Transfers from the microprogram constant field are one way while all others are bidirectional.

1401 <-> DM : Primarily for saving the contents of the 1401's internal stack after a stack overflow. Also for dumping the counters, RAM, status etc. for a context switch.

1410 <-> DM : Saving 1410 registers for a context switch, enables the use of the 1410 as a data ALU, and allows an address lookup table to reside in data RAM.

1401 <-> 1410 : Used for saving the 1410 registers on the 1401's subroutine stack during a context switch or subroutine call. In addition, it allows use of the 1410 as an ALU for program addresses. For instance, if this system were performing a Fast Fourier Transform, the 1401 would need the shifting function of the 1410's ALU for calculating the number of butterflies per group and the number of groups per stage. Both are used by the 1401 for counting loops in the FFT algorithm.

CF -> WS : For initialization of all 1401 and 1410 registers. Supplies the 1401 with direct, indirect, and relative program addresses, and supplies the 1410 with immediate data addresses or offsets.

CF -> DM : This is used for initializing data RAM with constants stored in the microprogram code.

DATA TRANSFER HOLD LATCH

All components receiving data from a Word-Slice data port, i.e., Data RAM or another Word-Slice data port, require that the data be stable in phase two, a certain setup time prior to the clock going high. Since the data port drives only while the clock is high we must have a way to hold this data for the remaining clock low period. One way this can be accomplished is by modifying the bidirectional buffer between the Constant Bus and the Data Bus.

Figure 3, on the following page, shows the Word-Slice system with this modification. The 16-bit buffer is replaced with 74F543 octal registered transceivers which form the Data Transfer Hold Latch. The Data Transfer Hold Latch has two 16-bit I/O ports, 32 three-state buffers, and 32

transparent latches which allow the transfer of data between two buses with the ability to latch data from either. Four signals, driven by a microcode pipeline latch, control the transfer operations. Several of these signals are gated with the clock to ensure proper timing for the half-cycle latching and enabling operations which must occur during a transfer. Two of the signals, LECD and LEDC, control the transparent latches. The latches hold data when these controls are high and pass data to the corresponding three-state buffer when low. OECD and OEDC, the other two signals, control data flow direction between the buses. When OECD is low, the latch sourced by the Constant Bus drives the Data Bus. Similarly when OEDC is low the latch sourced by the Data Bus drives the Constant Bus.

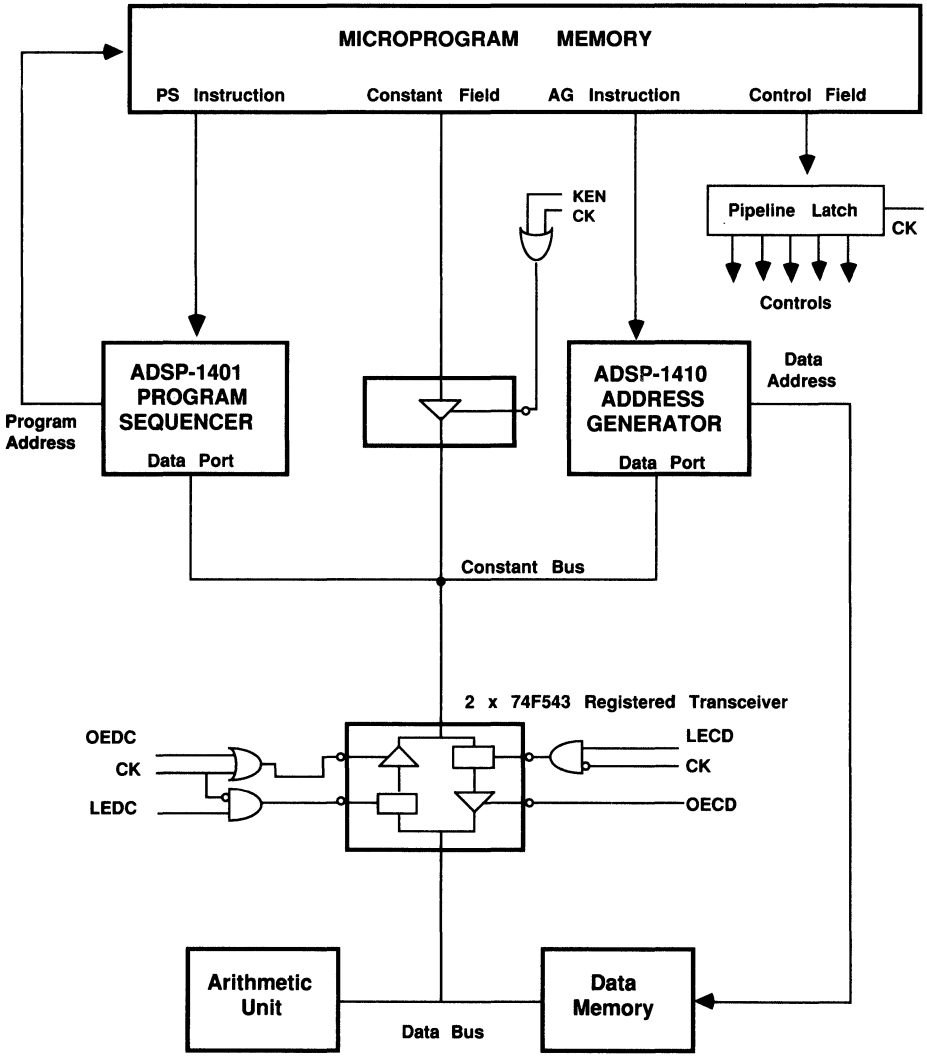


Figure 3. System with Data Transfer Hold Latch

Two data transfer scenarios, WS -> WS and WS -> DM, require the latches to hold the data during clock low, while the remaining three: DM -> WS, CF -> DM, and CF -> WS; do not. The relative timing for these five scenarios is given in Figure 4 and a control signal truth table is given in Figure 5 (see following page). The most complicated of the five scenarios is the WS -> WS transfer. For the example of transferring the contents of a 1410 address register to a counter register in the 1401 as used in an FFT, the 1410 outputs the data onto the Constant Bus during clock high. Then the latch sourced by the Constant Bus latches the data as the clock goes low, and holds the data on the Data Bus while the clock is low. Simultaneously the latch sourced by the Data Bus is in a transparent mode and during phase two drives the data back onto the Constant Bus. Consequently, the data from the clock-high period is held throughout the clock-low period so the 1401 can load it into one of its counter registers.

SEQUENCER OUTPUT AND DIRECT JUMP

The pipelined control signal, KEN, when low, enables the microcode constant field onto the Constant Bus. KEN is ORed with the clock so that a data port input and output, as discussed above, can occur in the same cycle without any bus contention. This allows data transfers such as WS -> DM and CF -> WS to be executed simultaneously.

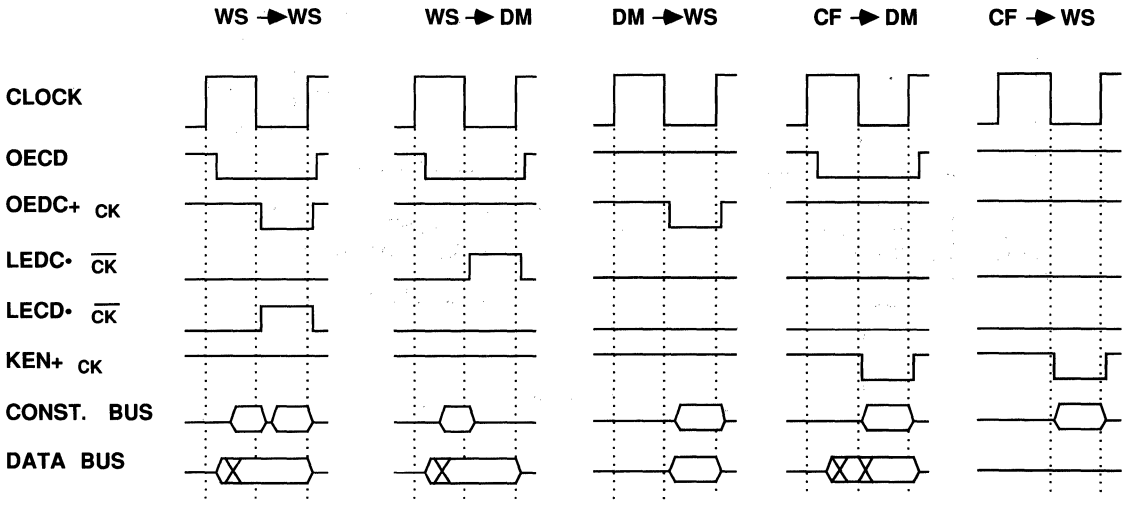


Figure 4. Data Transfer Timing

TRANSFER

CONTROL	DEFAULT	WS --> WS	WS --> DM	DM --> WS	CF --> DM	CF --> WS	WS --> DM & CF --> WS
OECD	1	0	0	1	0	1	0
OEDC	1	0	1	0	1	1	1
LECD	0	0	1	0	0	0	1
LEDC	0	1	0	0	0	0	0
KEN	1	1	1	1	0	0	0

Figure 5. Data Transfer Truth Table

Therefore one can pop data from the 1401's stack, save it in Data RAM, and perform a direct jump indexed by an immediate value from the microcode constant field. This transfer is key, for example, in the fast execution of a loop that pages out the contents of the 1401's stack to data memory.

STACK PAGING EXAMPLE

Figure 6 is an example of the code required for this stack paging loop. For initialization, this code assumes that the 1401's subroutine stack pointer is pointing to the bottom of data to be paged, the 1410's R0 address register contains the first data memory address to be loaded, and the 1410's C0 compare register contains the final data memory address to be loaded. Also the compare/zero flag of the 1410 is connected to the 1401's condition flag input so the program sequencer can decide when the data is completely paged out.

This loop consists of two microcode instructions, each made up of five operation fields (labeled 1-5 in Figure 6) preceded by an optional label. Each field is separated by a comma. The first field is the program sequencer operation, the second field is the constant field, the third is the address generator operation, and the fourth operation controls the signals for the Data Transfer Hold Latch and constant field output enable. The last field controls the write enable signals for the data RAM.

In the first instruction, PPSSD pops data from the 1401's internal stack onto the Constant Bus, while YINC supplies the data memory with a write address and increments the address register R0 to the next write address. The popped data is held on the Data Bus during the second half clock cycle for the data memory write and simultaneously the microcode constant field supplies the 1401 with an absolute jump address, (pageout), over the Constant Bus. Here the

	1	2	3	4	5
pageout	**PPSSD,**	**NOP,**	**YINC C0 R0,**	**XFER,**	**WRITE DM,**
	JDA NF,	**CONST pageout,**	**NOP,**	**NOP,**	**NOP,**

Figure 6. Stack Paging Loop Code

Data Transfer Hold Latch separates the Constant Bus and the Data Bus so that the data write and the address setup for the absolute jump can occur in the same cycle. The instruction XFER refers to assertion of the control signals as shown in the truth table for the WS -> DM & CF -> WS transfer. In the second instruction the JDA NF operation loops back to the start address (pageout) if the 1410's compare flag is not set or ends the loop if it is set.

CONSTRAINTS

The assembler used for generating microcode for this or any system should place error flags on several illegal operations which could result in bus contention. For instance the Data Bus cannot be in use during a WS -> WS transfer. The 1401 and 1410 cannot drive the Constant Bus at the same time, and the microcode constant field cannot drive the Constant Bus during a DM -> WS transfer.

CONCLUSION

The bidirectional data port of the ADSP-1401 Program Sequencer and ADSP-1410 Address Generator provides for fast and flexible data transfers which can contribute to your Word-Slice system's performance. The inclusion of a Bus Transfer Hold Latch simplifies the timing of these transfers and allows additional efficiency for time critical code.

A Guide to Designing Microcoded Circuits

by Bob Fine

INTRODUCTION

The majority of "smart" circuits designed-to-date utilize microprocessors to control data flow and perform computations. Microprocessors can be thought of as sequential processors where one operation is performed at a time and multiple operations are performed in a sequence. Many applications require very high computational throughput, not available with microprocessors. A microcoded approach to designing a system offers a high degree of functional parallelism (where many operations can be performed simultaneously) which gives rise to high computational throughput.

This application note describes some microcode circuit basics and the related design issues. It details the steps to be taken when embarking on a microcoded design.

MICROCODE PRINCIPLES

The basic difference in design philosophy between microprocessor circuits and microcoded circuits is that the functions which are integrated onto the single microprocessor device are broken out as separate devices or building blocks in the microcoded circuit. With this configuration, each functional block is allowed to operate independently and simultaneously. When more operations occur within any given time frame, the complete task requires less time. This is especially important when the operations are arithmetic.

Since each device in the microcoded circuit can operate independently, a method is required for coordinating their operation to produce a synchronous system. If all devices were to receive their operating instructions from a single source which is synchronous to a system clock, these devices would be able to operate in tandem to comprise a complete system. The larger or more complex the system, the more devices required. The instruction source for all the microcode devices in the system is the microcode memory which contains instructions at each of its locations. During each system clock cycle, a microcode memory location is accessed and the instruction residing at that location is presented to the microcode devices. The width of the microcode memory will depend on the number of devices in the system. A larger number of devices requires more instruction bits to control the devices.

Figure 1, on the following page, shows an example of a microcode memory. As we sequence from location to location, different bit patterns are output from the memory. Figure 1 focuses on the most significant instruction bit showing that as sequential locations are accessed, a digital waveform is produced at the memory output. The state of the digital waveform depends on whether the contents of that location of microcode memory in that bit position is a "1" or a "0" corresponding to a logic high or low respectively. The outputs of the microcode memory can be grouped to form smaller instruction words feeding VLSI devices or can be used as individual control lines to drive SSI and MSI circuits such as buffers, latches, memories, etc.. This is illustrated in Figure 2 (see next page).

Pipeline Latches

Microcode memories are constructed with either RAM (random access memory) or ROM (read only memory) components. Depending on the depth and width of the microcode memory needed, many memory ICs may be cascaded. The accessing of a memory involves an address becoming stable, the location contents being accessed and the data output becoming stable. Between memory accesses, as the address and data become stable, the output of the memory is in an undetermined state. The consequence of having these undetermined bits driving the devices of the microcode circuit is that unpredictable circuit operation is imminent. To prevent such operation, pipeline latches are required. A D-type latch is placed at the output of the microcode memory and is clocked at every rising edge of the system clock as shown in Figure 3 (see page 3).

The timing of the microcode memory is such that valid outputs exist prior to the rising edge of the clock and stable information is loaded into the latch. The latch will hold this valid microcode instruction until the next clock edge. During

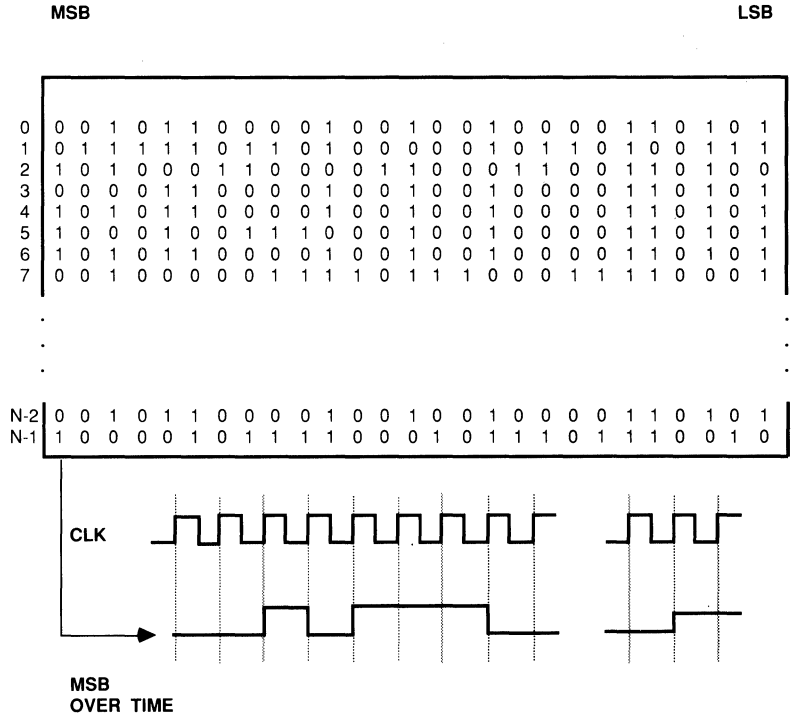


Figure 1. Example of Microcode Memory Contents

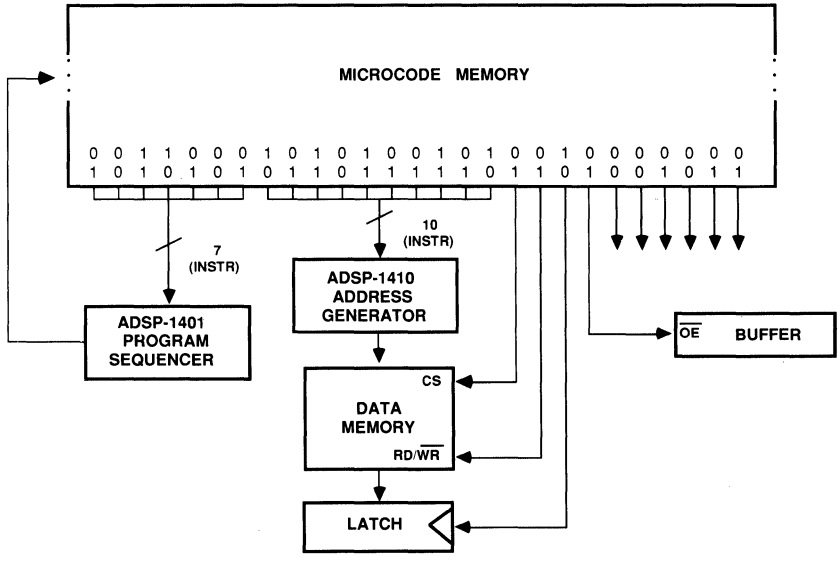


Figure 2. Microcode Memory to Device Connection

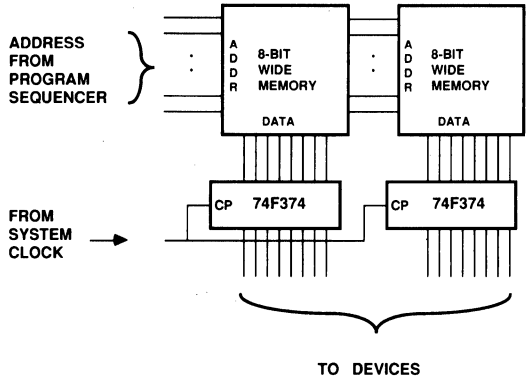


Figure 3. Microcode Pipeline Latch Connection

this time, a new microcode memory location can be accessed. The process of holding a valid instruction while the next is being fetched is called pipelining, hence the name pipeline latch. The timing for the microcode memory access and pipeline latch is shown in Figure 4.

Note that, within the time of one clock cycle, the memory output is in both undetermined states during memory access and valid data states after access is complete. The latch output, however, is valid for a complete clock cycle.

With a working knowledge of microcode basics and an understanding of microcode memories and pipeline latches, we can now handle the task of designing a microcode system.

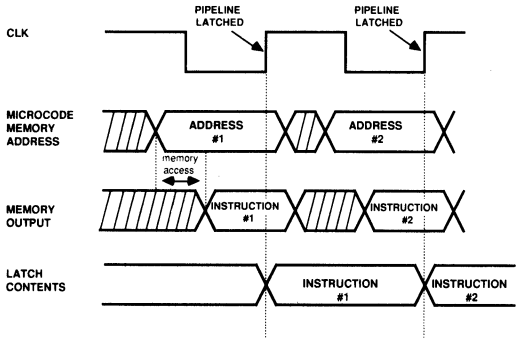


Figure 4. Microcode Pipeline Latch Timing

SYSTEM DESIGN EXAMPLE

The first step in designing a microcode circuit is to determine what functions the circuit must perform. By doing so, it can then be determined what components or devices are required. The data flow between devices must also be defined.

Once this step is complete, a microcode memory organization must be done. Each device in the system must receive its instructions from the microcode memory. Each instruction subfield will be placed side by side along with any miscellaneous control lines.

Design Goal

The goal of this design example is to provide a general purpose array processor circuit that can execute a variety of DSP and numerical processing functions. A simple example is desired which will make a trade-off in performance for a lower price and fewer parts count.

A single data bus structure is chosen for design compactness and simplicity as opposed to a multiple bus structure with multiple memories. The functions which need to be executed on this example design are an FIR filter, correlation, an FFT (up to 4K points), and a matrix multiply.

From the design goal, it becomes apparent that a data memory large enough to contain 4K words of real data, 4k words of imaginary data and 4k coefficients is needed. Also, an address generator is required to supply addresses to data memory. A multiplier/accumulator is required that will easily interface to the single data memory, single bus structure of this system. To control the activity of the system circuitry, a program sequencer and microprogram memory is needed. Finally, for a "real world" interface an A/D and D/A are to be used.

Circuit Description

The design example described consists of a program sequencer to control microcode instruction flow, a 16-bit data bus, an address generator to provide data memory addresses, a data memory, several data buffers, a data latch, a multiplier/accumulator and an I/O port. Figure 5 (see next page) illustrates this architecture. The assignment of instruction subfield to a specific location within the microcode instruction is purely arbitrary and will usually correspond to hardware relationships. Related hardware will have adjacent instruction fields.

Microprogram Sequencing

In a complex, high performance system the instructions of the microcode memory will not be fetched sequentially as previously described. In most cases, sophisticated program flow will be needed. A microcode program is not any different than any other program, be it for a microprocessor or mainframe. It is desired to have program flow that can accommodate nested loops, subroutines, interrupts and complex jumps and branching.

The ADSP-1401 is a high-speed microprogram sequencer optimized for the demanding sequencing tasks found in digital signal processors and general purpose computers. Unique features such as 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,

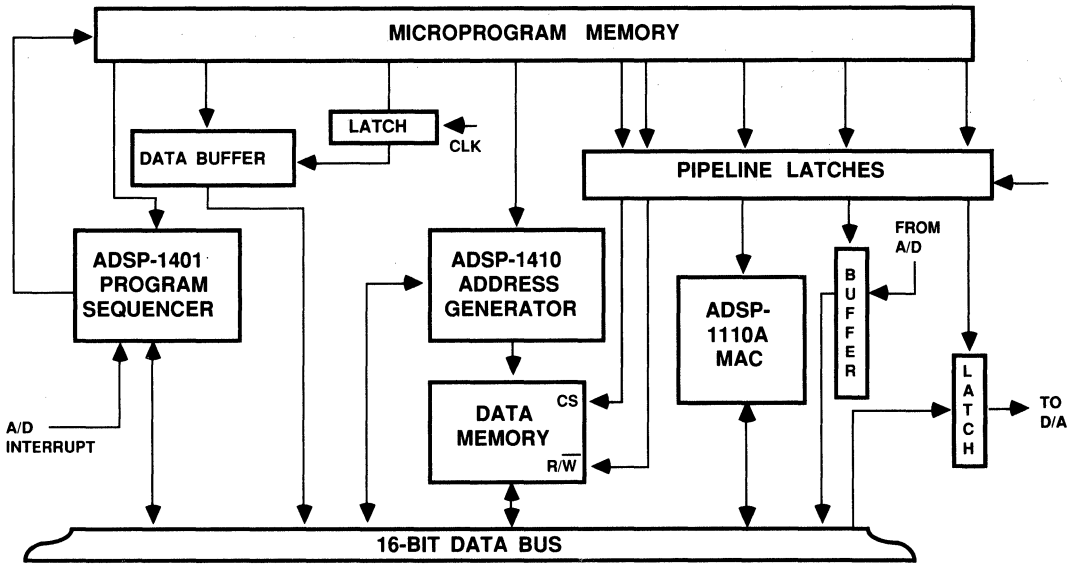


Figure 5. Block Diagram of Example System

a dynamically configurable 64-word RAM ideally meet the sequencing demands of any microcode circuit.

As with other devices in the system, the program sequencer will receive its instructions from the microcode memory. In essence, the microcode memory is instructing the program sequencer how to get the next instruction. Figure 6 shows an example of a non-sequential program flow. Each instruction gives sequencer information on how to get to the next instruction to be executed.

CURRENT MICROPROGRAM ADDRESS	PROGRAM SEQUENCER INSTRUCTION	NEXT MICROPROGRAM ADDRESS
1	JUMP TO LOCATION 10	10
10	JUMP TO SUBROUTINE	100
100	RETURN FROM SUBROUTINE	11
11		

Figure 6. Non-Sequential Program Flow Example

In Figure 6, the instruction at location 1 in microcode memory instructs the program sequencer to perform a program jump to location 10. The sequencer responds by outputting an address 10 resulting in a jump subroutine instruction to be fetched. The program sequencer responds by outputting an address 100 (the starting location of the subroutine). The instruction at location 100 is a return from subroutine resulting in the program sequencer outputting an address 11. These jumps and subroutine calls can also be made conditional upon some external event.

The ADSP-1401 contains an internal pipeline register and can be connected directly to the output of the microcode memory. No external pipeline latch is required as with other devices.

Address Generation

Now that the first device (the program sequencer) has been selected, it is time to further define the architecture of the microcode system to be built. In the general purpose design example used throughout this paper, a 16-bit data bus structure is used. Data operands must reside in a data memory and all data busses and memories are 16 bits wide. The length of the data memory is determined by the maximum size of any data record that must be stored. In this example, a 16K word memory is chosen.

This 16K x 16 data memory may contain many smaller data records and a method for properly accessing these data points is needed. The ADSP-1410 is a fast, flexible address generator which rapidly generates the data memory addresses required for a wide range of digital signal/array processors and other high-performance computers. The ADSP-1410's architecture features a 16-bit ALU, a comparator, and 30 16-bit registers. In a single 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 contains an internal microcode pipeline latch and can be connected directly to the output of the microcode memory. No external pipeline latch is required as with other devices.

Since 16K words of data memory are required, only fourteen of the sixteen address lines of the address generator are to be used. The two most significant address output lines are left unconnected.

Arithmetic Devices

Since the system to be built is to perform high speed arithmetic computations, an appropriate arithmetic device or devices must be selected. This selection will depend on desired port structure, computation speed and overall features. For this design, an ADSP-1110A single-port multiplier/accumulator has be selected.

The ADSP-1110A is a high-speed, low-power, single-port 16 x 16-bit multiplier/accumulator (MAC) offering unique advantages such as compact 28-pin DIP package, simpler system interface to single-bus peripherals, reduced cost and features such as overflow and saturation. This device will perform multiplications, additions and subtractions which are the core operations in most algorithms. One of the advantages of a microcode system is that many arithmetic devices can be organized in parallel to improve the computational throughput of the system. In this design, only one arithmetic device is used.

I/O Ports

The example system also has an I/O requirement for a 12-bit analog to digital (A/D) converter and a 16-bit digital to analog (D/A) converter. The microcode system will be constructed so that the A/D converter can interrupt the system when a sample is ready.

Miscellaneous Devices

To connect the A/D and D/A converters to the microcode system's data bus, an input buffer and an output latch are needed. Microcode control lines are needed to instruct the buffer and latch what to do.

SYSTEM ARCHITECTURE

Now that the necessary components have been selected, the system architecture can be defined. A block diagram of the example system described is shown in Figure 5 (on previous page).

The only component specification that has not yet been completed is that of the microcode memory width. Table 1 shows the components selected and the number of microcode bits required for each component.

COMPONENT	*MICROCODE BITS*
1) Program Sequencer	7
2) Address Generator	11
3) Data Memory	2
4) Multiplier/Accumulator	8
5) Input I/O Buffer	1
6) Output I/O Latch	1
TOTAL	30

Table 1. Required Microcode Bits

Thirty microcode bits are required to drive the system components. A 16-bit data constant field is also required so that data values can be specified immediately via the microprogram. An additional enable control bit is required to allow the data constant to be driven on to the bus. This brings the total number of microcode instruction bits to 47.

MICROCODE INSTRUCTION DEFINITION

The flexibility of the microcode system allows the designer the freedom to place the instruction subfields in any order. To make microprogramming easier and microprograms easier to read, related functions should be placed in adjacent fields. Figure 7 shows the instruction subfield placement within the 47-bit microinstruction described above.

The program sequencer instruction is placed in the most significant bits or left-most portion of the microcode instruction word. This position was chosen because the program sequencer determines program flow and will be important when the microprogram has to be read by a

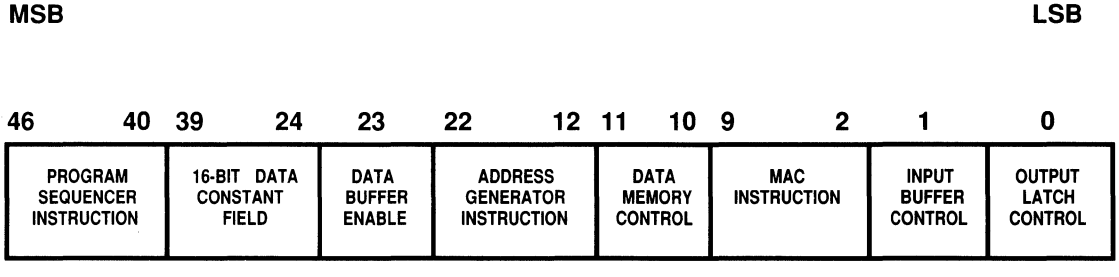


Figure 7. Microcode Instruction Subfield Placement

person. The program sequencer instructions will be easily identified as the left-most part of the microcode instruction. The data constant field was arbitrarily positioned next with the related data buffer enable-bit adjacent.

The next portion of the microcode instruction relates to data memory with the address generator instruction and memory control lines. The remaining portions of microcode instruction deal with the MAC and I/O port.

Figure 8 details the hardware components and their connection to the specific microcode instruction fields.

TIMING ANALYSIS

With the microcode instruction word and functional blocks of the system defined, the detail design of the system can be done. This involves a careful analysis of instruction timing, data flow, and pipeline skew.

Microcode Memory Timing

A first step in analyzing system timing is to determine the microcode memory access time required for a given system clock cycle. A 10 MHz system clock (100ns cycle time) is used for this design example. To determine the required microcode memory access time, the program sequencer

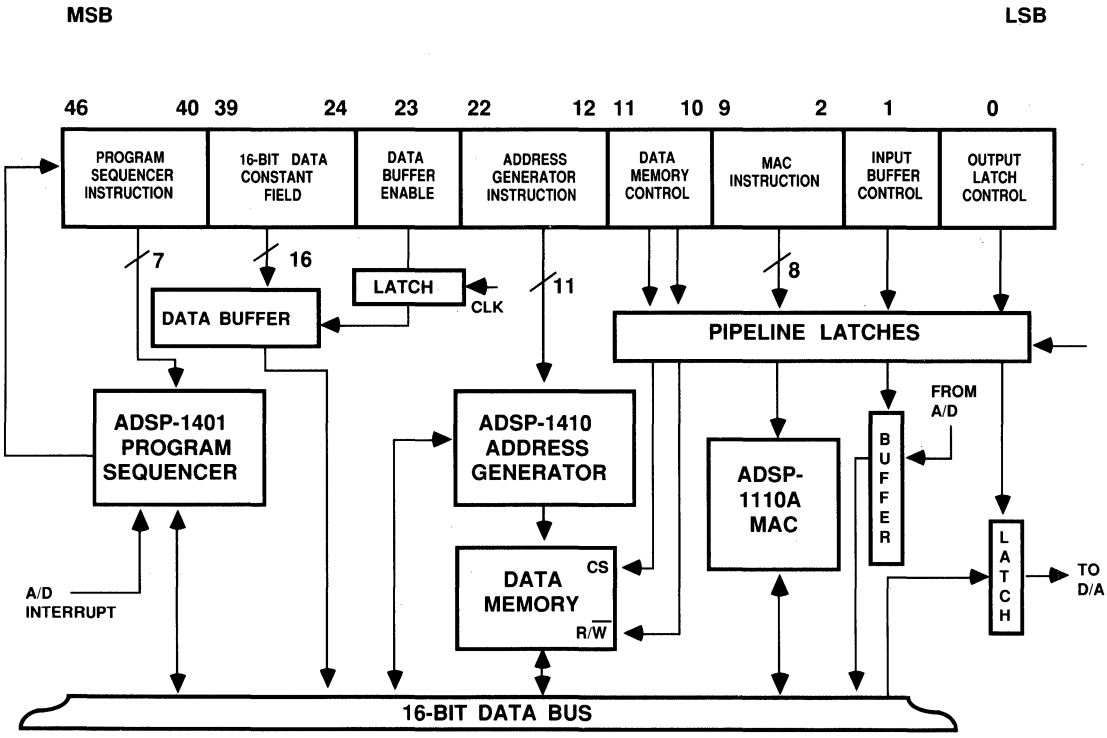


Figure 8. Example System Block Diagram

timing must first be analyzed. The ADSP-1401 Program Sequencer used in this example has an instruction set-up time (T_{SU}) of 25ns and an output address delay time (T_{OD}) of 20ns. This is graphically illustrated in Figure 9.

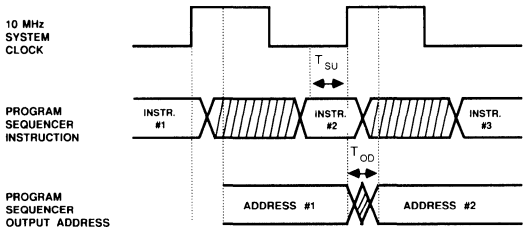


Figure 9. ADSP-1401 Instruction/Address Timing

The sequence of events which occurs during a microcode memory instruction fetch is as follows -- set-up program sequencer instruction, wait for valid address from program sequencer, access microcode memory location, wait for new instruction to be output from microcode memory, set-up new program sequencer instruction, etc.. This sequence is shown in Figure 10.

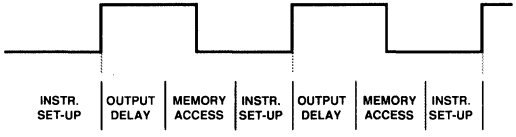


Figure 10. Microcode Memory Instruction Fetch Sequence

As shown in Figure 10, the instruction set-up, the output delay, and the memory access must fit into the cycle time. The instruction set-up and output delay times are known to be 25ns and 20ns respectively. This leaves 55ns remaining in the 100ns cycle for microcode memory access. If 5ns are left for overhead or extra margin, then a 50ns access time memory should be selected. *It should be noted that this access time is only required for the microcode memory portion that feeds the program sequencer.* The access of the remaining microcode memory will not involve the 25ns set-up time. Figure 11 shows the timing for remaining microcode memory feeding the instruction pipeline latches.

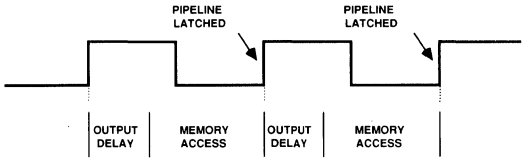


Figure 11. Microcode Memory/Pipeline Latch Timing

As shown in Figure 11, the microcode instruction is valid to the end of the cycle. The rising edge of the clock latches valid microcode instruction bits into the pipeline latch. The pipeline latch holds these instruction bits until the next rising edge of the clock.

All that is involved in this case is the 20ns output delay, the 2ns set-up time for the 74F374 pipeline latch and the remaining 78ns for the microcode memory access. Leaving 3ns for margin, a 75ns access time memory is selected for the remaining microcode memory.

Data Flow

Data flow consists of the transfer of data words between the I/O port, the data memory, the arithmetic device, the program sequencer and the address generator. Each device will have a specific data set-up time for inputs and a specific output delay and hold time for outputs. A data memory access time can be determined by combining these data set-up, delay, and hold times. Since many data paths may exist, the access time for data memory must be verified for each data path.

In this example design, the data paths are as follows.

1) Data Memory to Multiplier/Accumulator,
2) Multiplier/Accumulator to Data memory,
3) A/D to Data memory,
4) Data memory to D/A.

These paths are shown in Figure 12 (see next page).

Adequate time must be allowed for data input set-up, output delay and hold times with additional margins to compensate for bus capacitances.

Instructions and control lines will also have a specific set-up and hold time requirement. It is possible, due to pipeline delays in the system, that an instruction may be specified for a data transfer several cycles before that data is actually used by any device. A careful analysis of data flow will determine the pipeline skews required in programming.

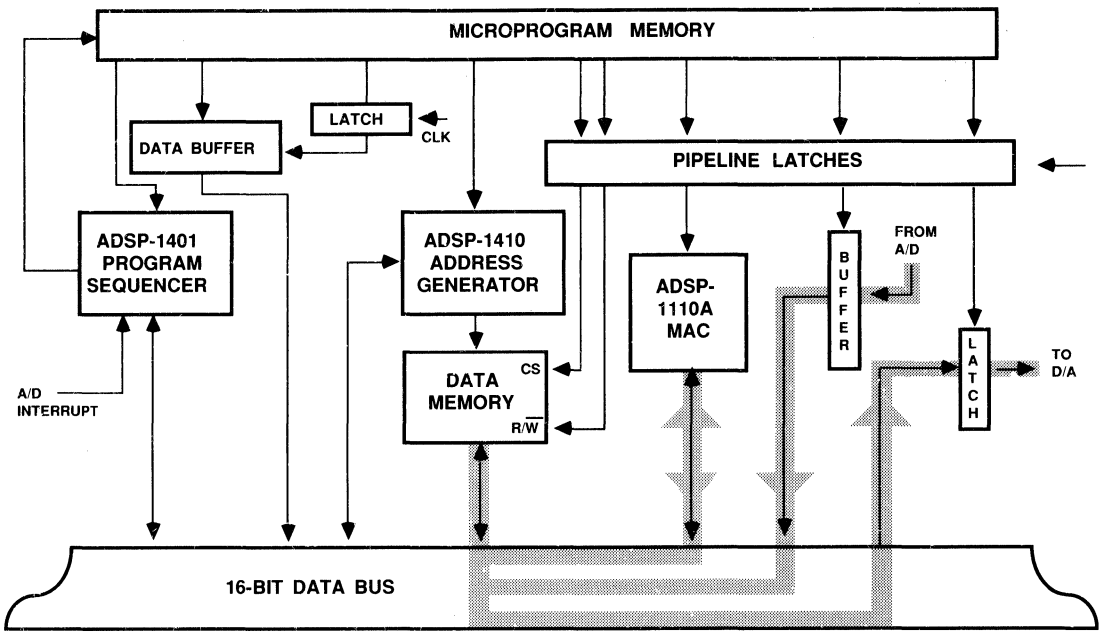


Figure 12. Data Paths for Design Example

SOFTWARE DEVELOPMENT

With the detailed design of the system complete, software development can begin. The first development tool required is a meta-assembler such as the METASTEP assembler available from STEP ENGINEERING in Sunnyvale, CA. A meta-assembler will allow the programmer to specify the microcode instruction word which has been defined and to represent machine code by assembly mnemonics. The meta-assembler can then be used, like any other assembler, to allow the programmer to write microcode on a higher level using address labels, variables and constants. A linker can join different program modules together and a PROM formatter prepares data for the user specific PROM configuration.

CIRCUIT DEBUGGING

As with many circuits, unpredictable problems in software or hardware occur. it is now the job of the designer or technician to debug the microcode circuit. There are several approaches available, each differing in cost of development tools and manpower required. In general, the better the debug tools, the easier it is to find the problem.

The most basic method of debugging involves an oscilloscope, a logic analyzer and a lot of sweat. A single-step clock feature, where the clock can be manually controlled, is a useful debug tool. Program instructions can be checked and control line states and data can be verified.

CONCLUSION

A systematic approach to microcoded design can minimize the design effort in addition to providing a flexible and well thought out design. A microcoded circuit's computation speed, flexibility, and overall performance are far better than conventional microprocessor circuits.

For more information on specific design issues concerning Analog Devices' Word-Slice™ microcode components and arithmetic components, contact the DSP Applications Engineering group.

Appendix
Contents

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 REPRINTS

A collection of articles about the ADSP-2100 and its applications from *Electronic Design*: February, March, April, June, 1986.

MANUALS (Obtain from your nearby sales office)
ADSP-2100 USER'S MANUAL—Introduction, Architecture, System Interface, Instruction Set, Appendix—180 pages.

ADSP-2100 CROSS-SOFTWARE MANUAL—Development System, System Builder, Assembler, Linker, Simulator, PROM Splitter, Appendix—102 pages.

ADSP-2100 APPLICATIONS HANDBOOK—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.

WORD-SLICE USER'S MANUAL (ADSP-1401 Program Sequencer and ADSP-1410 Address Generator)—Introduction; ADSP-1401: 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 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

DATA 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. (Available FREE with the Linear Products Databook as a 2-volume set.)

LINEAR PRODUCTS DATABOOK—1988. Data Sheets, Selection Guides, and Application Notes on Op Amps, Instrumentation Amplifiers, Isolators, RMS-to-DC Converters, 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. (Available FREE with the Conversion Products Databook as a 2-volume set.)

SHORT-FORM SELECTION GUIDE—1987. Selection Guides and Characteristics Tables on Converters and Converter-Support Components, Amplifiers, Analog and Digital Signal-Processing Components, and other products—150 pages.

APPLICATION NOTES AND GUIDES

Application Notes. All are available individually upon request.

A/D Converters:
- "AD670 8-Bit A/D Converter Applications."
- "Exploring the AD667 12-Bit Analog Output Port."
- "The AD7574 Analog-to-Microprocessor Interface."
- "Interfacing the AD7572 to High-Speed DSP Processors."
- "Bipolar Operation with the AD7572."
- "How to Obtain the Best Performance from the AD7572."
- "ADG201A/202A and ADG221/222 Performance with Reduced Power Supplies."

Amplifiers:
- "An IC Amplifier User's Guide to Decoupling, Grounding, and Making Things Go Right for a Change."
- "Applications of High-Performance BiFET Op Amps."
- "How to Select Operational Amplifiers."
- "Low-Cost, Two-Chip Voltage-Controlled Amplifier and Video Switch."

D/A Converters:
- "AD7528 Dual 8-Bit CMOS DAC."
- "Gain Error and Tempco of CMOS Multiplying DACs."
- "Interfacing the AD7549 Dual 12-Bit DAC to the MCS-48 and MCS-51 Microcomputer Families."
- "Methods for Generating Complex Waveforms and Vectors Using Multiplying D/A Converters."
- "Simple Interface Between D/A Converter and Microcomputer Leads to Programmable Sine-Wave Oscillator."
- "The AD7224 DAC Provides Programmable Voltages Over Varying Ranges."
- "Three-Phase Sine-Wave Generation Using the AD7226 Quad DAC."
- "14-Bit DACs (AD7534/AD7535) Maintain High Performance Over Extended Temperature Range."
- "Analog Panning Circuits Provide Almost Constant Output Power."

Resolver-to-Digital Conversion:
- "Dynamic Resolution-Switching on the 1S74 Resolver-to-Digital Converter."
- "Resolver-to-Digital Conversion—A Simple and Cost-Effective Alternative to Optical Shaft Encoders."
- "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."
- "Generate 4 Channels of Analog Output Using AD7542 12-Bit D/A Converters and Control It All with Only Two Wires."

V/F Converters:
- "Operation and Applications of the AD654 IC V-to-F Converter."
- "Analog-to-Digital Conversion Using Voltage-to-Frequency Converters (AD651)."

Application Guides available upon Request

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.

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.

Angular and Linear Data Conversion—A 12-page short-form guide to analog-digital conversion products for synchros, resolvers, and Inductosyns*, in forms ranging from hybrid ICs to instruments and systems.

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.

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.

Surface Mount IC—A 28-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.

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.

***Inductosyn is a registered trademark of Farrand Industries, Inc.**

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 Technolog Way, P.O. Box 9106, Norwood MA 02062-9106.

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

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. 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

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.

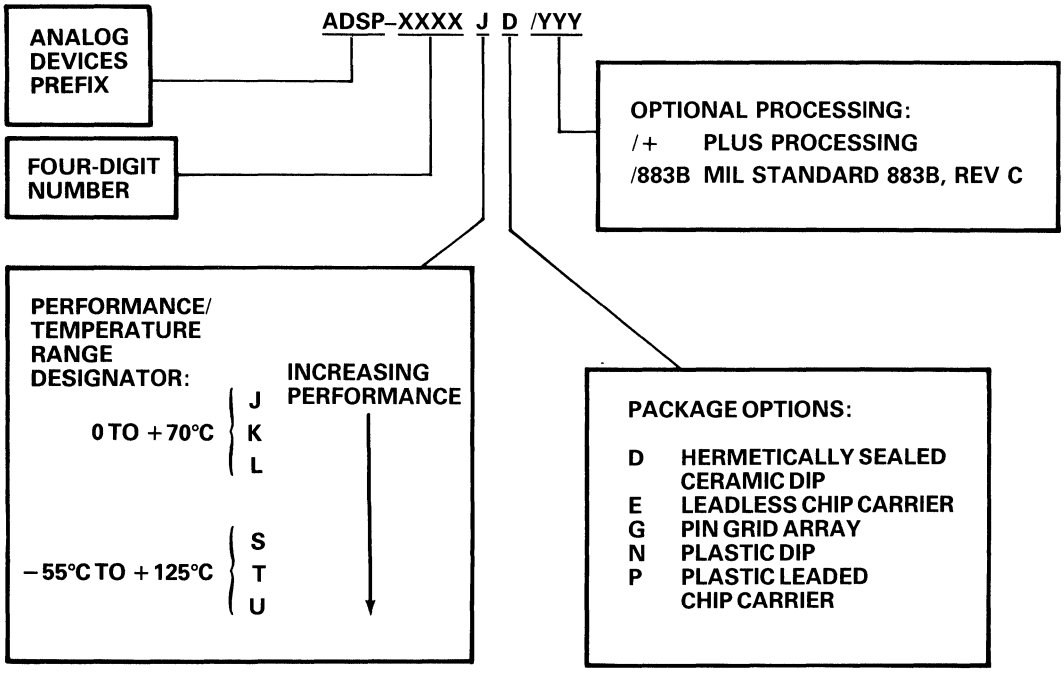
Payments for new accounts, where open-account credit has not yet been established, will be C.O.D. or prepaid. On all orders under fifty dollars ($50.00), a five-dollar ($5.00) processing charge is required.

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.


ANALOG DEVICES PREFIX
FOUR-DIGIT NUMBER
ADSP-XXXX J D /YYY
OPTIONAL PROCESSING:
/+ PLUS 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

Alabama
(205) 536-1506

Alaska
*(206) 251-9550
*(714) 641-9391

Arizona
(602) 949-0048
*(303) 590-9952

Arkansas
*(713) 664-6704
*(214) 231-5094

California
*(714) 641-9391
*(408) 559-2037
*(619) 268-4621

Colorado
(303) 443-5337
*(303) 590-9952

Connecticut
(516) 673-1900
*(617) 329-4700

Delaware
*(215) 643-7790

Florida
(305) 855-0843
(305) 724-6795
(813) 963-1076

Georgia
(404) 449-7662

Hawaii
*(714) 641-9391

Idaho
(303) 443-5337
*(303) 590-9952
*(206) 251-9550

Illinois
(312) 520-0710
*(312) 980-0300

Indiana
(317) 244-7867
*(312) 980-0300

Iowa
(319) 373-0200
*(312) 980-0300

Kansas
(913) 829-2800
*(312) 980-0300

Kentucky
(615) 459-0743
*(617) 329-4700

Louisiana
*(713) 664-6704

Maine
*(617) 329-4700

Maryland
*(301) 992-1994

Massachusetts
*(617) 329-4700

Michigan
(313) 559-9700
*(614) 764-8795

Minnesota
(612) 835-2414
*(312) 980-0300

Mississippi
(205) 536-1506

Missouri
(314) 521-2044
*(312) 980-0300

Montana
(801) 466-9336
*(714) 641-9391

Nebraska
(913) 829-2800
*(312) 980-0300

Nevada
(505) 883-9090
*(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) 883-9090
*(303) 590-9952

New York
(516) 673-1900
(716) 425-4101

North Carolina
(919) 373-0380

North Dakota
(612) 835-2414
*(312) 980-0300

Ohio
(216) 248-4995
*(614) 764-8795

Oklahoma
*(214) 231-5094

Oregon
*(206) 251-9550

Pennsylvania
*(215) 643-7790
*(614) 764-8795

Rhode Island
*(617) 329-4700

South Carolina
(919) 373-0380

South Dakota
(612) 835-2414
*(312) 980-0300

Tennessee
(205) 536-1506
(615) 459-0743

Texas
*(713) 664-6704
*(214) 231-5094

Utah
(801) 466-9336
*(303) 590-9952

Vermont
*(617) 329-4700

Virginia
*(301) 992-1994

Washington
*(206) 251-9550

West Virginia
*(614) 764-8795

Wisconsin
(414) 784-7736
*(312) 980-0300

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

Argentina
(1)379890

Australia
(03)5750222
(02)8888777

Austria
*(222)885504

Belgium
*(3)2371672

Brazil
(11)5339533

Denmark
*(2)845800

Finland
(0)7555133

France
*(1)46873411
*(76)222190
*(61)408562
*(99)535200
*(8)3516331

Holland
*(1620)81500

Hong Kong
(5)8339013

India
(212)53880
(11)312981
(812)569134

Ireland
*(932)232222
(United Kingdom)

Israel
*(052)28995
*(052)27536

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
*(617)329-4700

Mexico
(83)351721
(83)351661

New Zealand
(9)592629

Norway
(2)123600

People's Republic of China – Beijing
(1)890721, Ext. 120

Romania
*(222)885504
(Austria)

Singapore
*(617)329-4700

South Africa
(11)7863710

Spain
(1)7543001
(3)3007712

Sweden
*(8)282740

Switzerland
*(22)315760
(18)200102

Taiwan
(2)5018231

United Kingdom
*(932)232222
*(01)9411066
*(635)35335
*(506)30306
*(021)5011166
*(0279)418611

United States of America
*(617) 329-4700

West Germany
*(89)570050
*(4181)8051
*(721)616075
*(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: 174059
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: 174059, Cable: ANALOG NORWOODMASS

PRINTED IN U.S.A.

G1125–60–10/87