Introduction 1

1.1 OVERVIEW

This book presents a compilation of routines for a variety of common digital signal processing applications based on the ADSP-2100 DSP microprocessor family. These routines may be used as is or they may serve as a jumping-off point for the development of routines tailored to your particular application. Each routine is prefaced by a discussion of the algorithm or data formats underlying the code.

Besides showing the specific applications, the set of routines demonstrates a variety of programming tactics for getting the most performance out of the ADSP-2100 family processors, for example, the proper way to segment loops to utilize the ADSP-2100's cache memory. We believe that readers will benefit from reading every chapter, even if their present application interests concern only a single topic.

The material in this book was originally published as Volumes 1, 2 and 3 of the *ADSP-2100 Family Applications Handbook*. The information in those volumes has been updated and integrated into this book; it supersedes those earlier publications.

1.2 ADSP-2100 FAMILY OF PROCESSORS

This section briefly describes the ADSP-2100 family of processors. For more complete information, refer to the *ADSP-2100 User's Manual*, *ADSP-2101 User's Manual*, and *ADSP-2111 User's Manual*, available from Analog Devices or the *ADSP-2100 Family User's Manual*, available from Prentice Hall and Analog Devices. For the applications in this book, "ADSP-2100" refers to *any* processor in the ADSP-2100 family unless otherwise noted.

The ADSP-2100 is a programmable single-chip *microprocessor* optimized for digital signal processing (DSP) and other high-speed numeric processing applications. The ADSP-2100 chip contains an ALU, a multiplier/accumulator, a barrel shifter, two data address generators and a program sequencer; data and program memories are external. The ADSP-2100A is a pin- and code-compatible version of the original ADSP-2100 fabricated, in 1.0-µm CMOS. It can operate at a faster clock rate than the ADSP-2100.

1 Introduction

The ADSP-2101 is a programmable single-chip *microcomputer* based on the ADSP-2100. Like the ADSP-2100, the ADSP-2101 contains computational units, as well as a program sequencer and dual address generators. Additionally, there are 1K words of data memory and 2K words of program memory on chip, two serial ports, a timer, boot circuitry (for loading on-chip program memory at reset), and enhanced interrupt capabilities. Because the ADSP-2101 is code-compatible with the ADSP-2100, the programs in this book can be executed on these chips as well (some modifications for interrupt vectors may be necessary), although not all programs are designed to make use of the extra features and functions of the ADSP-2101.

1.2.1 ADSP-2100 Architecture

This section gives a broad overview of the ADSP-2100 internal architecture, using Figure 1.1 to show the architecture of the ADSP-2100 processor.

The ADSP-2100 processor contains three full-function and independent computational units: an arithmetic/logic unit, a multiplier/accumulator and a barrel shifter. The computational units process 16-bit data directly and provide for multiprecision computation.

Two dedicated data address generators and a complete program sequencer supply addresses. The sequencer supports single-cycle conditional branching and executes program loops with zero overhead. Dual address generators allow the processor to output simultaneous addresses for dual operand fetches. Together the sequencer and data address generators allow computational operations to execute with maximum efficiency. The ADSP-2100 family uses a modified Harvard architecture in which data memory stores data, and program memory stores both instructions and data. Able to store data in both program and data memory, ADSP-2100 processors are capable of fetching two operands on the same instruction cycle.

The internal components are supported by 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 (which interconnects the computational units)

Introduction 1

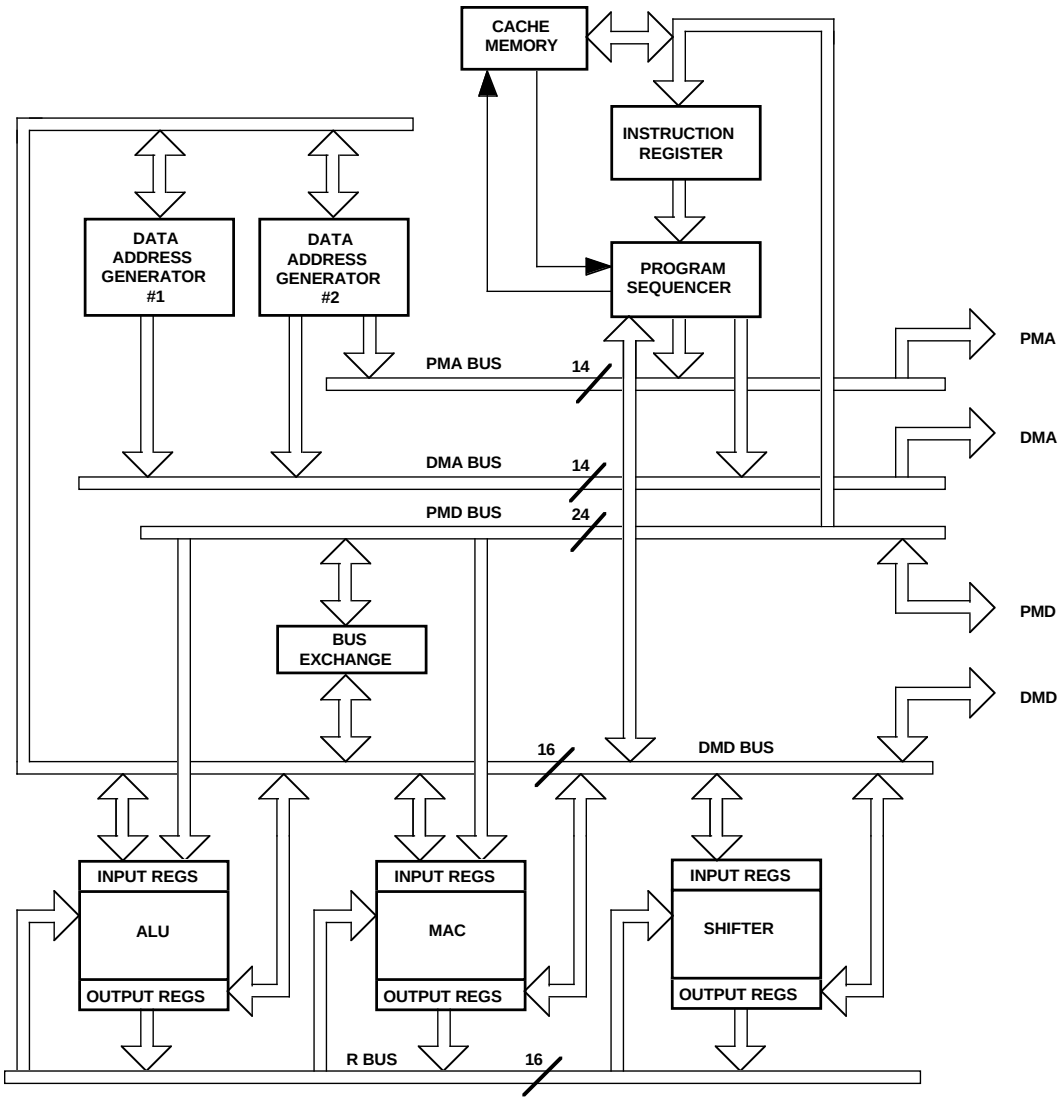

Figure 1.1 ADSP-2100 Internal Architecture

On the ADSP-2100, the four memory buses are extended off-chip for direct connection to external memories.

The program memory data (PMD) bus serves primarily to transfer instructions from off-chip memory to the internal instruction register. Instructions are fetched and loaded into the instruction register during one processor cycle; they execute during the following cycle while the

1 Introduction

next instruction is being fetched. The instruction register introduces a single level of pipelining in the program flow. Instructions loaded into the instruction register are also written into the cache memory, to be described below.

The next instruction address is generated by the program sequencer depending on the current instruction and internal processor status. This address is placed on the program memory address (PMA) bus. The program sequencer uses features such as conditional branching, loop counters and zero-overhead looping to minimize program flow overhead. The program memory address (PMA) bus is 14 bits wide, allowing direct access to up to 16K words of instruction code and 16K words of data. The state of the PMDA pin distinguishes between code and data access of program memory. The program memory data (PMD) bus, like the processor's instruction words, is 24 bits wide.

The data memory address (DMA) bus is 14 bits wide allowing direct access of up to 16K words of data. The data memory data (DMD) bus is 16 bits wide. The data memory data (DMD) bus provides a path for the contents of any register in the processor to be transferred to any other register, or to any external data memory location, *in a single cycle*. The data memory address can come from two sources: an absolute value specified in the instruction code (direct addressing) or the output of a data address generator (indirect addressing). Only indirect addressing is supported for data fetches via the program memory bus.

The program memory data (PMD) bus can also be used to transfer data to and from the computational units through direct paths or via the PMD-DMD bus exchange unit. The PMD-DMD bus exchange unit permits data to be passed from one bus to the other. It contains hardware to overcome the 8-bit width discrepancy between the two buses when necessary.

Each computational unit contains a set of dedicated input and output registers. Computational operations generally take their operands from input registers and load the result into an output register. The registers act as a stopover point for data between the external memory and the computational circuitry, effectively introducing one pipeline level on input and one level on output. The computational units are arranged side by side rather than in cascade. To avoid excessive pipeline delays when a series of different operations are performed, the internal result (R) bus allows any of the output registers to be used directly (without delay) as the input to another computation.

Introduction 1

For a wide variety of calculations, it is desirable to fetch two operands at the same time—one from data memory and one from program memory. Fetching data from program memory, however, makes it impossible to fetch the next instruction from program memory on the same cycle; an additional cycle would be required. To avoid this overhead, the ADSP-2100 incorporates an instruction cache which holds sixteen words. The benefit of the cache architecture is most apparent when executing a program loop that can be totally contained in the cache memory. In this situation, the ADSP-2100 works like a three-bus system with an instruction fetch and two operand fetches taking place at the same time. Many algorithms are readily coded in loops of sixteen instructions or less because of the parallelism and high-level syntax of the ADSP-2100 assembly language.

Here's how the cache functions: Every instruction loaded into the instruction register is also written into cache memory. As additional instructions are fetched, they overwrite the current contents of cache in a circular fashion. When the current instruction does a program memory data access, the cache automatically sources the instruction register if its contents are valid. Operation of the cache is completely transparent to user.

There are two independent data address generators (DAGs). As a pair, they allow the simultaneous fetch of data stored in program and in data memory for executing dual-operand instructions in a single cycle. One data address generator (DAG1) can supply addresses to the data memory only; the other (DAG2) can supply addresses to either the data memory or the program memory. Each DAG can handle linear addressing as well as modulo addressing for circular buffers.

With its multiple bus structure, the ADSP-2100 supports a high degree of operational parallelism. In a single cycle, the ADSP-2100 can fetch an instruction, compute the next instruction address, perform one or two data transfers, update one or two data address pointers and perform a computation. Every instruction executes in a single cycle.

Figure 1.2, on the next page, is a simplified representation of the ADSP-2100 in a system context. The figure shows the two external memories used by the processor. Program memory stores instructions and is also used to store data. Data memory stores only data. The data memory address space may be shared with memory-mapped peripherals, if desired. Both memories may be accessed by external devices, such as a

1 Introduction

system host, if desired. Figure 1.2 also shows the processor control interface signals, (RESET, HALT and TRAP) the four interrupt request lines, the bus request and bus grant lines (BR and BG) and the clock input (CLKIN) and output (CLKOUT).

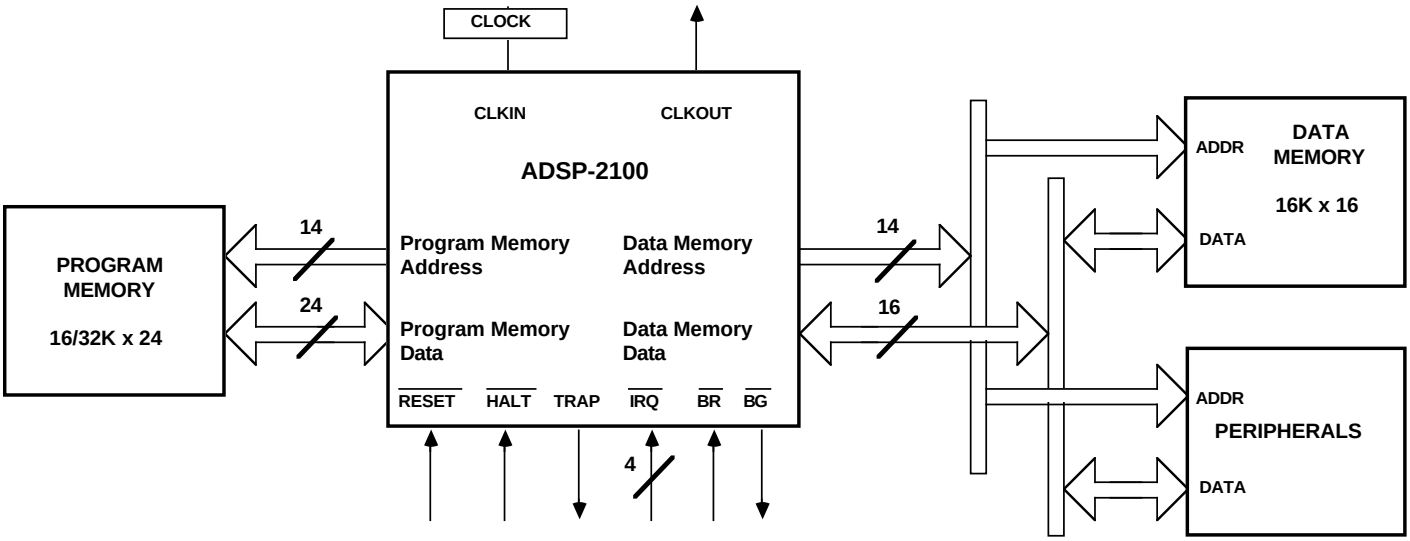

Figure 1.2 ADSP-2100 System

1.2.2 ADSP-2101 Architecture

Figure 1.3 shows the architecture of the ADSP-2101 processor. Like the ADSP-2100, the ADSP-2101 contains an arithmetic/logic unit, a multiplier/accumulator, and a barrel shifter—plus two data address generators and a program sequencer.

The ADSP-2101 has 1K words of 16-bit data memory on-chip and 2K words of 24-bit program memory on-chip. The processor can fetch an operand from on-chip data memory, an operand from on-chip program memory and the next instruction from on-chip program memory in a single cycle. (The speed of on-board memory access makes this possible and eliminates the need for cache memory as on the ADSP-2100.)

This scheme is extended off-chip via a single external memory address bus and data bus which may be used for either program or data memory access and for booting. Consequently, the processor can access external memory once in any cycle.

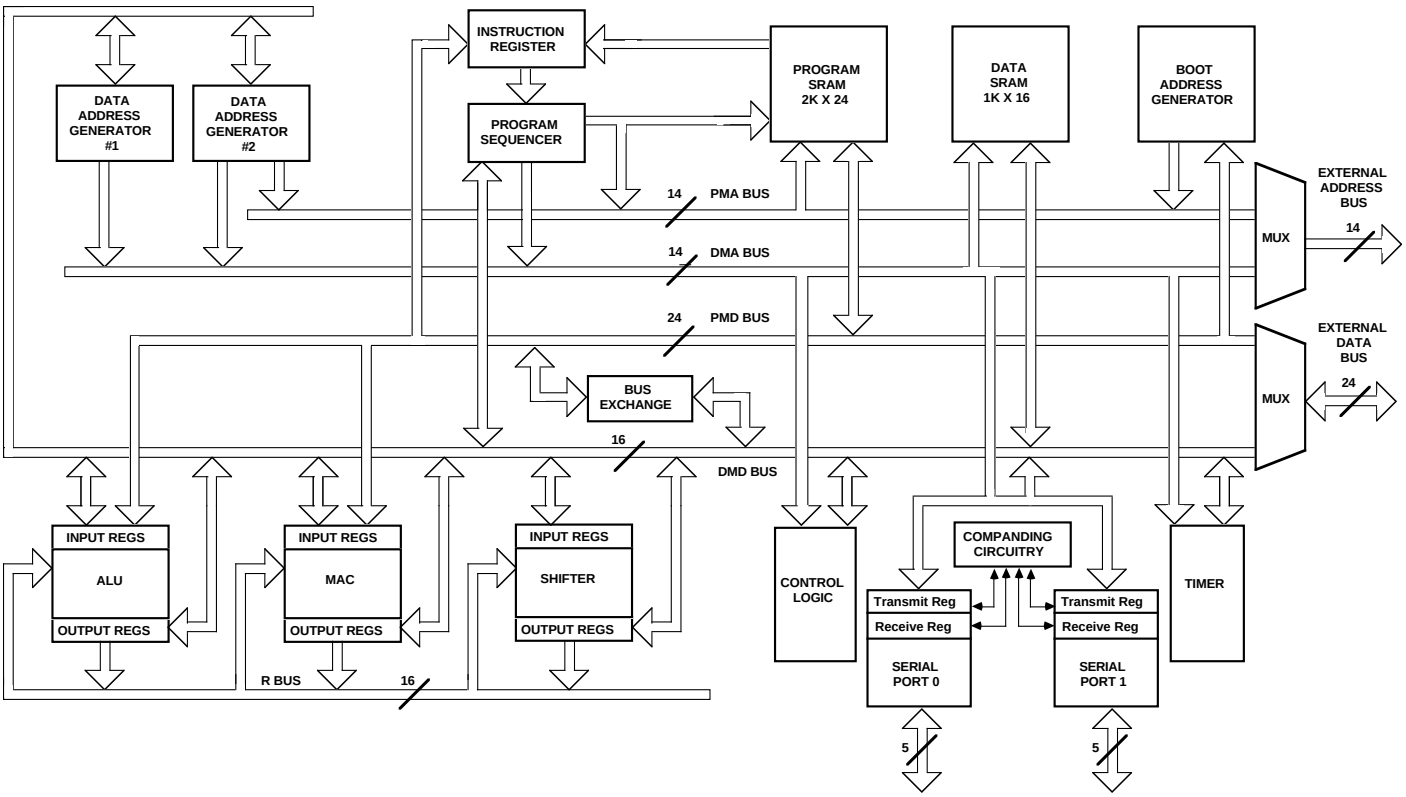

Figure 1.3 ADSP-2101 Internal Architecture

Boot circuitry provides for loading on-chip program memory automatically after reset. Wait states are generated automatically for interfacing to a single low-cost EPROM. Multiple programs can be selected and loaded from the EPROM with no additional hardware.

The memory interface supports memory-mapped peripherals with programmable wait-state generation. External devices can gain control of buses with bus request/grant signals (BR and BG). An optional execution mode allows the ADSP-2101 to continue running while the buses are granted to another master as long as an external memory operation is not required.

The ADSP-2101 can respond to six user interrupts. There can be up to three external interrupts, configured as edge- or level-sensitive. Internal interrupts can be generated from the timer and the serial ports. There is also a master RESET signal.

1 Introduction

The two serial ports (“SPORTs”) provide a complete serial interface; they interface easily and directly to a wide variety of popular serial devices. They have inherent hardware companding (data compression and expansion) with both μ-law and A-law available. Each port can generate an internal programmable clock or accept an external clock.

The SPORTs are synchronous and use framing signals to control data flow. Each SPORT can generate its serial clock internally or use an external clock. The framing synchronization signals may be generated internally or by an external device. Word lengths may vary from three to sixteen bits. One SPORT (SPORT0) has a multichannel capability which allows the receiving or transmitting of arbitrary data words from a 24-word or 32-word bitstream. The SPORT1 pins have alternate functions and can be configured as two additional external interrupt pins and the Flag Out (FO) and Flag In (FI) pins.

The programmable interval timer provides periodic interrupt generation. An 8-bit prescaler register allows the timer to decrement a 16-bit count register over a range from each cycle to every 256 cycles. An interrupt is generated when this count register reaches zero. The count register is automatically reloaded from a 16-bit period register, and the count resumes immediately.

1.3 ASSEMBLY LANGUAGE OVERVIEW

The ADSP-2100 family’s 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. Each assembly statement, however, corresponds to a single 24-bit instruction, executable in one cycle. Register mnemonics, listed below, are concise and easy to remember.

Mnemonic	*Definition*
AX0, AX1, AY0, AY1	ALU inputs
AR	ALU result
AF	ALU feedback
MX0, MX1, MY0, MY1	Multiplier inputs
MR0, MR1, MR2	Multiplier result (3 parts)
MF	Multiplier feedback
SI	Shifter input
SE	Shifter exponent
SR0, SR1	Shifter result (2 parts)
SB	Shifter block (for block floating-point format)
PX	PMD-DMD bus exchange
I0 - I7	DAG index registers
M0 - M7	DAG modify registers
L0 - L7	DAG length registers (for circular buffers)
PC	Program counter
CNTR	Counter for loops
ASTAT	Arithmetic status
MSTAT	Mode status
SSTAT	Stack status
IMASK	Interrupt mask
ICNTL	Interrupt control modes
RX0, RX1	Receive data registers (not on ADSP-2100)
TX0, TX1	Transmit data registers (not on ADSP-2100)

The ADSP-2101 instruction set is an upward-compatible superset of the ADSP-2100 instruction set; thus, programs written for the ADSP-2100 can be executed on the ADSP-2101 with minimal changes.

Here are some examples of the ADSP-2100 assembly language. The statement

```
MR = MR + MX1*MY1;
```

performs a multiply/accumulate operation. It multiplies the input values in registers MX1 and MY1, adds that product to the current value of the MR register (the result of the previous multiplication) and then writes the new result to MR.

The statement

```
DM(buffer1) = AX0;
```

writes the value of register AX0 to data memory at the location which is the value of the variable *buffer1*.

1 Introduction

1.4 DEVELOPMENT SYSTEM

The ADSP-2100 family is supported with a complete set of software and hardware development tools. The ADSP-2100 Family Development System consists of Development Software to aid in software design and in-circuit emulators to facilitate the debug cycle. EZ-ICE™ evaluation boards are available for evaluating the processors. Additional development tool capabilities continue to be added as new members of the processor family are introduced.

The Development Software includes:

- System Builder

This module allows the designer to specify the amount of RAM and ROM available, the allocation of program and data memory and any memory-mapped I/O ports for the target hardware environment. It uses high-level constructs to simplify this task. This specification is used by the linker, simulators, and emulators.

- Assembler

This module assembles a user's source code and data modules. It supports the high-level syntax of the instruction set. To support modular code development, the Assembler provides flexible macro processing and "include" files. It provides a full range of diagnostics.

- Linker

The Linker links separately assembled modules. It maps the linked code and data output to the target system hardware, as specified by the System Builder output.

- Simulator

This module performs an instruction-level simulated execution of ADSP-2100 family assembly code. The interactive user interface supports full symbolic assembly and disassembly of simulated instructions. The Simulator fully simulates the hardware configuration described by the System Builder module. It flags illegal operations and provides several displays of the internal operations of the processor.

- PROM Splitter

This module reads the Linker output and generates PROM-programmer-compatible files.

- C Compiler

The C Compiler reads ANSI C source and outputs source code ready to be assembled. It also supports inline assembler code.

In-circuit emulators provide stand-alone, real-time, in-circuit emulation. The emulators provide program execution with little or no degradation in processor performance. The emulators duplicate the simulators' interactive and symbolic user interface.

Complete information on development tools is available from Analog Devices.

1.5 CONVENTIONS OF NOTATION

The following conventions are used throughout this book:

- All listings begin with a comment block that summarizes the calling parameters, the return values, the registers that are altered, and the computation time of the routine (in terms of the routine's parameters, in some cases).

- In listings, all keywords are uppercase; user-defined names (such as labels, variables, and data buffers) are lowercase. In text, keywords are uppercase and user-defined names are lowercase italics. Note that this convention is for readability only.

- In comments, register values are indicated by "=" if the register contains the value or by "—>" if the register points to the value in memory.

- All numbers are decimal unless otherwise specified. In listings, constant values are specified in binary, octal, decimal, or hexadecimal by the prefixes B#, O#, D#, and H#, respectively.

1 Introduction

1.6 PROGRAMS ON DISK

This book includes an IBM PC 5¼ inch high-density diskette containing the routines that appear in this book. As with the printed routines, we cannot guarantee suitability for your application. The diskette also contains a demonstration version of the ADSP-2101 Simulator. This demonstration is self-running and documented on-line.

1.7 FOR FURTHER SUPPORT

You can reach Analog Devices in the following ways:

- By contacting your local Analog Devices Sales Representative
- For information on DSP product features, availability, and pricing, call (617) 461-3881 in Norwood, Massachusetts, USA
- For Applications Engineering information, call either the DSP applications group at (617) 461-3672 in Norwood, Massachusetts, USA, or the Linear applications group at (617) 461-2628 in Wilmington, Massachusetts, USA
- The DSP Norwood office Fax number is (617) 461-3010
- The factory may also be reached by
  Telex: 924491
  TWX: 710/394-6577
  Cables: ANALOGNORWOODMASS
- Through the DSP Group's Bulletin Board Service; it can be reached at 300, 1200, or 2400 baud, no parity, 8 bits data, 1 stop bit by dialing:
  (617) 461-4258
- By writing to:
  Analog Devices
  SPD-DSP
  One Technology Way
  P.O. Box 9106
  Norwood, MA 02062-9106
  USA