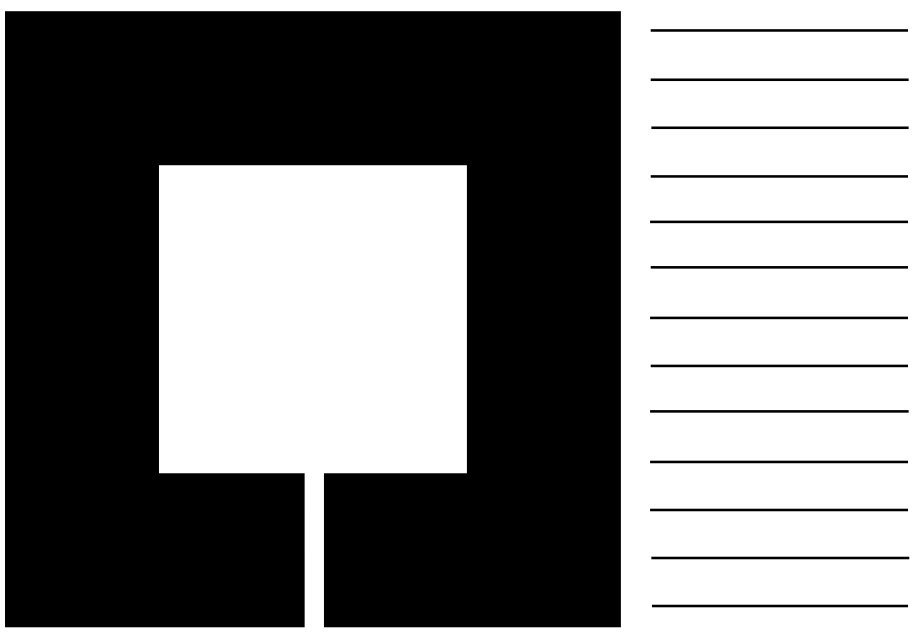
ADSP-2100 Family

EZ-KIT Lite Reference Manual



ANALOG DEVICES

ADSP-2100 Family EZ-KIT Lite Reference Manual

PRINTED IN U.S.A.

Printing History
FIRST EDITION 5/95

For marketing information or Applications Engineering assistance, contact your local Analog Devices sales office or authorized distributor.

If you have suggestions for how the ADSP-2100 Family EZ-KIT Lite or documentation can better serve your needs, or you need Applications Engineering assistance from Analog Devices, please contact:

Analog Devices, Inc.
DSP Applications Engineering
One Technology Way
Norwood, MA 02062-9106
Fax: (617) 461-3010
e-mail: dsp_applications@analog.com

The DSP Applications Engineering group runs a Bulletin Board Service that provides answers to many DSP questions and information on Analog Devices DSP products. The BBS can be reached at speeds up to 14,400 baud, no parity, 8 bits data, 1 stop bit, dialing (617) 461-4258. This BBS supports: V.32bis, error correction (V.42 and MNP classes 2, 3, and 4), and data compression (V.42bis and MNP class 5).

Please submit any technical questions or problems in writing and send it to the e-mail address listed, the fax number listed, or to the BBS.

The DSP Applications Group also maintains an Internet FTP site. Login as anonymous using your email address for your password. Type (from your UNIX prompt):

ftp ftp.analog.com (or type: ftp 137.71.23.11)

EZ-KIT Lite Hardware Warranty

Your EZ-KIT Lite hardware is warranted against defects in workmanship and materials under normal use and service for 90 days from the date of shipment by Analog Devices. This warranty does not extend to any units 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. Analog Devices, Inc. shall not be liable for consequential damages under any circumstances.

EZ-KIT Lite Hardware Service

Use the following procedure if you have a hardware problem with your EZ-KIT Lite:

- Describe the situation in written form and send it to the DSP Applications Group as described on the previous page. Make sure to include any source code examples or special circumstances that will help with problem diagnoses.
- You will receive a confirmation notice that explains a work-around to your problem. If it is determined that the problem lies in your EZ-KIT Lite, you will be directed to a sales representative to set up an EZ-KIT Lite product return.
- The Sales Representative will provide you with a Material Return Authorization number (MRA#) and the address to which you should send your EZ-KIT Lite Product. (See the following notes.)

All Returns: The MRA# must be written on the box for Analog Devices Receiving to accept shipment.

Warranty Returns: As mentioned in the warranty, the warranty period begins with the shipment date. This information is on the invoice for your EZ-KIT Lite Product. You must provide the Sales Representative with the shipment date information to obtain a warranty repair.

EZ-KIT Lite Returns: After the 90 day warranty, no repair is available for the EZ-KIT Lite.

Literature

The following is a list of related literature. Literature can be obtained / purchased from your local Analog Devices sales office or authorized distributor.

ADSP-2100 Family User's Manual (Prentice Hall)
Complete description of processor architectures and system interfaces.

ADSP-2171/81 User's Manual
Information specific to the ADSP-2181.

ADSP-2100 Family Assembler Tools & Simulator Manual
ADSP-2100 Family C Tools Manual
ADSP-2100 Family C Runtime Library Manual
Programmer's references.

APPLICATIONS INFORMATION

Digital Signal Processing Applications Using the ADSP-2100 Family, Volume 1 (Prentice Hall)
Topics include arithmetic, filters, FFTs, linear predictive coding, modem algorithms, graphics, pulse-code modulation, multirate filters, DTMF, multiprocessing, host interface and sonar.

Digital Signal Processing Applications Using the ADSP-2100 Family, Volume 2 (Prentice Hall)
Topics include modems, linear predictive coding, GSM codec, sub-band ADPCM, speech recognition, discrete cosine transform, digital tone detection, digital control system design, IIR biquad filters, software uart and hardware interfacing.

SPECIFICATION INFORMATION

ADSP-21xx Data Sheet
ADSP-2181 Data Sheet
AD1847 Data Sheet

Contents

CHAPTER 1 INTRODUCTION

INTRODUCTION 1–1
UNPACKING 1–1
EZ-KIT LITE 1–1
CONTENTS OF THIS MANUAL 1–2

CHAPTER 2 UPGRADE INFORMATION

OVERVIEW 2–1
UPGRADE DESCRIPTION 2–1

CHAPTER 3 GETTING STARTED

OVERVIEW 3–1
QUICK START SOFTWARE INSTALLATION 3–2
REQUIREMENTS 3–3
QUICK START HARDWARE INSTALLATION 3–4

CHAPTER 4 INSTALLATION PROCEDURES

SOFTWARE INSTALLATION 4–1
SOFTWARE INSTALLATION PROCEDURE 4–1
    Make Working Copies Of The Diskettes 4–1
    Modify Your CONFIG.SYS File 4–1
    Install The Software On Your Hard Disk 4–2
ENVIRONMENT VARIABLES 4–3
HARDWARE INSTALLATION 4–3

Contents

CHAPTER 5 DSP SYSTEM DEVELOPMENT

OVERVIEW 5–1
- System Requirements 5–1
- System Design 5–2
- Architecture Description File 5–2
- Code Development 5–3
- Running The Assembler 5–11
- Running The Linker 5–11
- Running The Simulator 5–12
- Programming An EPROM 5–12
- Running The ADSP-2181 EZ-KIT Lite Board 5–13
- Debugging 5–13

CHAPTER 6 EZ-KIT LITE HOST PROGRAM

PROGRAM OVERVIEW 6–1
COMMAND SUMMARY 6–2
DETAILED COMMANDS 6–3
- File Menu 6–3
- View Menu 6–4
  - Toolbar 6–4
  - Status Bar 6–6
- Demo Menu 6–6
  - DTMF 6–6
  - Filtering 6–8
  - Echo Cancellation 6–9
  - ADPCM 6–10
  - 7.8k LPC 6–11
  - 2.4k LPC 6–12
- Floating Menu 6–12
- Loading Menu 6–13
  - Download User Program and Go 6–13
  - Download User Program 6–15
  - Go 6–15
  - Upload Data Memory 6–16
  - Upload Program Memory 6–17
  - Download Data Memory 6–18
  - Download Program Memory 6–19
- Options Menu 6–20
  - Settings 6–21
  - List Of Demos 6–21

Contents

Help Menu 6–22
About EZ-KIT 6–22
User Configurable Settings 6–22
Error Messages & Troubleshooting 6–23
RUNNING DEMOS 6–26
CREATING YOUR OWN PROGRAMS 6–27

CHAPTER 7 EZ-KIT LITE MONITOR PROGRAM

PROGRAM OVERVIEW 7–1
MONITOR FEATURES 7–1
RESTRICTIONS 7–1
CREATING YOUR OWN PROGRAMS TO BE USED WITH THE MONITOR 7–2
DEBUGGING 7–5
DSP MEMORIES 7–6

CHAPTER 8 EZ-KIT LITE HARDWARE DESCRIPTION

DESIGN OVERVIEW 8–1
SPECIFICATIONS 8–3
CONNECTORS 8–3
SWITCHES 8–4
INDICATORS 8–5
HARDWARE OPERATION 8–5
HARDWARE EXPANSION 8–5
EXPANSION CONNECTORS 8–7
HARDWARE DEBUGGING 8–9

PROGRAMMER'S QUICK REFERENCE

SCHEMATICS

Introduction 1

INTRODUCTION

Congratulations! Your EZ-KIT Lite is one of the most cost effective, powerful development systems available on the market. The ADSP-2181 used in the EZ-KIT Lite offers the highest integration and performance in the 16-bit fixed-point DSP processor arena with 32K words of on-chip RAM, 30 ns instruction cycle time, DMA ports, and low power modes.

UNPACKING

Your EZ-KIT Lite should contain the following items.

- ADSP-2181 Development Board
- 3.5" Software Diskette
- EZ-KIT Lite Manual
- DB9 to DB9 RS-232 Cable
- Power Supply (US only)

If you are missing any items, please contact your Analog Devices sales office, authorized distributor, or reseller.

EZ-KIT LITE

EZ-KIT Lite provides an easy way for you to investigate the power of the ADSP-2100 Family of processors and develop your own applications based on these high-performance DSPs. It is a complete development system package with a price that makes it ideal for getting started in DSP and the performance to take you through all the phases of the development process. With EZ-KIT Lite you can:

- Evaluate Analog Devices' DSPs
- Learn About DSP Applications
- Develop DSP Applications
- Simulate & Debug Your Application
- Prototype Applications

1 Introduction

The EZ-KIT Lite consists of a small ADSP-2181 based development / demonstration board with full 16-bit stereo audio I / O capabilities. In this documentation, this board may be referred to as either the ADSP-2181 EZ-LAB® or the EZ-KIT Lite board. The terms are used interchangeably. The board's features include:

- ADSP-2181 33 MIPS DSP
- AD1847 Stereo SoundPort
- RS-232 Interface
- Socketed EPROM
- User Pushbuttons
- Power Supply Regulation
- Expansion Connectors
- User Configurable Jumpers

The board can run standalone or can simply connect to the RS-232 port of your PC. A monitor program running on the DSP in conjunction with a host program running on the PC lets you interactively download programs as well as interrogate the ADSP-2181. The board comes with a socketed EPROM so that you can run the monitor program and demonstrations provided or you can plug in an EPROM containing your own code.

The EZ-KIT Lite also comes with all the software you need to develop sophisticated, high-performance DSP applications. An Assembler, Linker, PROM Splitter utility, and Simulator are all included.

CONTENTS OF THIS MANUAL

This manual provides all the information you need to:

- Install the EZ-KIT Lite software on to an IBM compatible Personal Computer
- Connect your EZ-KIT Lite to your PC
- Connect your EZ-KIT Lite to the power supply
- Connect an input source (such as a microphone or CD player) to your EZ-KIT Lite
- Connect an output device (such as an amplified / powered speaker)
- Start the EZ-KIT Lite board
- Use the software provided
- Write your own ADSP-2181 programs to run on the EZ-KIT Lite board

Chapter 2 describes the general capabilities of the EZ-KIT Lite development software and tells you what other features are available by upgrading to the ADSP-2100 Family Development Software.

Chapter 3 contains basic information on how to get started. It is recommended that you read this manual completely and thoroughly, especially if you are new to programming a DSP processor. For those of you who can't wait and are anxious to power up the EZ-KIT Lite and begin tinkering, this chapter provides you with the basics to get up and running quickly.

Chapter 4 provides a detailed description for installing the various software modules on to an IBM compatible PC. These software modules include:

- ADSP-2181 Development Software
- EZ-KIT Lite Host Software
- Demonstration Programs
- Example Programs

This chapter also describes how to set up and use the EZ-KIT Lite board.

Chapter 5 provides an overview of the development process. This chapter should give you most of the information you need to write your own ADSP-2181 programs. Complete source code for an example program is listed and described. You should be able to use this program as a basis for writing your own program. Examples are also given for running the Assembler, Linker, and PROM Splitter.

Chapter 6 describes the operation of the host program. The host program runs under Windows™ on an IBM-compatible PC. The host program is used to communicate to the ADSP-2181 EZ-LAB.

Chapter 7 describes the DSP monitor program that is shipped with the EZ-KIT Lite. The EPROM installed on the EZ-KIT Lite board contains the monitor. This chapter also details the guidelines that you should follow to create your own DSP programs for use with the monitor.

Chapter 8 contains a detailed description of the EZ-KIT Lite hardware. Information concerning expanding the EZ-KIT Lite to better suit your needs is also provided.

1 Introduction

Programmer's Quick Reference provides a quick reference of all the assembly language instructions for the ADSP-2181. It also describes operation of the Assembler, Linker, PROM Splitter, and Simulator.

Schematics contains the schematic diagrams for the EZ-KIT Lite board.

Upgrade Information 2

OVERVIEW

EZ-KIT Lite is shipped with a version of software that is fully functional for basic DSP development operations such as assembling, linking, simulating / debugging, and PROM formatting. With this software, you can create a high performance ADSP-2181 based system with very sophisticated features. The EZ-KIT Lite software is more powerful and complete than the software you find in some of the other manufacturer's DSP starter kits.

The Lite version is powerful enough to fill the needs of most users, but those who want technical support and the full flexibility provided by a C compiler, C source-level debugger, a complete set of libraries, a librarian, and all of the simulators for the entire ADSP-2100 Family may want to upgrade to the standard development software for the ADSP-2100 Family.

UPGRADE DESCRIPTION

In addition to features found in the Lite version, the ADSP-2100 Family Development Software adds:

- **System Builder**
  Define your target system hardware in an architecture description file. The Linker and the Simulator use this information to know how much memory is in your system, which memory is RAM and which is ROM, which memory is internal to the processor and which is external, and what memory-mapped peripherals you have.

- **Simulators**
  Run an instruction level simulation of any ADSP-2100 Family processor. All of the ADSP-2100 Family Simulators provide an interactive, instruction-level simulation, displaying the cycle-by-cycle operation of different portions of the processor and system hardware through a window-based graphical user interface.

2 Upgrade Information

- **Librarian**
  Combine frequently used subroutines into a single library file to simplify the task of linking and streamline system software.

- **C Compiler**
  Code your applications in ANSI standard C. This compiler is based on the industry-standard GNU C Compiler of the Free Software Foundation.

- **C Runtime Library**
  Use this C callable library for ANSI standard and custom DSP functions. The C library includes functions for simplified interrupt handling with automatic save and restore of registers.

- **C Debugger**
  Simplify the process of debugging your programs with this C Source Level Debugger which is integrated within the simulator and emulator environments.

Compare and decide. A quick comparison of some of the features available in the EZ-KIT Lite software and the ADSP-2100 Family appears on the facing page.

Upgrade Information 2

Development Software Comparison

Features	EZ-KIT Lite Software	ADSP-2100 Family Software
System Builder (Create your own architecture description files)	*	√
Assembler	√	√
Linker	√	√
ADSP-2101 Simulator (Also simulates ADSP-2103, ADSP-2105 and ADSP-2115)		√
ADSP-2111 Simulator		√
ADSP-2171 Simulator		√
ADSP-2181 Simulator	√	√
Prom Splitter	√	√
C Compiler		√
C Runtime Libraries		√
Librarian		√
C Debugger		√
Full Set of Documentation		√
Technical Support		√

* EZ-KIT Lite is shipped with an architecture description file for the ADSP-2181

Getting Started 3

OVERVIEW

The EZ-KIT Lite contains the ADSP-2181 EZ-LAB evaluation board, ADSP-2181 Development Software, IBM PC compatible host software, example programs and demos, and documentation. You can obtain additional information and documentation from your local Analog Devices sales office or authorized distributor. You may find this documentation useful if you are planning to do a significant amount of ADSP-2181 code development. The following documents can be obtained or purchased:

ADSP-2100 Family User's Manual
ADSP-2100 Family Assembler Tools & Simulator Manual
ADSP-2181 User's Manual

To fully utilize all the functions of the EZ-KIT Lite you will need to install the provided software on to an IBM compatible PC and connect the EZ-KIT Lite board to the PC as well as connect it to a power supply, analog input source and amplified speakers.

3 Getting Started

QUICK START SOFTWARE INSTALLATION

This section contains a very brief description of the EZ-KIT Lite software installation procedure. It is written for experienced PC users familiar with installing software onto their PC.

The install utility for the EZ-KIT Lite should be run under Windows™.

1. Make sure Windows is running.

2. Insert EZ-KIT Lite Software disk into a floppy drive. (usually A: or B:)

3. Open the Program Manager window, if it is not already open.

4. From the File menu, select Run. The Run dialog box should appear.

5. In the "Command line" box type `A:\SETUP.EXE` (or `B:\SETUP.EXE`).

6. Click on OK or press enter.

7. Follow the instructions on the screen.

Note: You will need to edit your AUTOEXEC.BAT file in the following manner.

1. Add the environment variable `SET ADI_DSP=C:\ADI_DSP`

2. Append to the path variable `C:\ADI_DSP\21XX\BIN`

A detailed installation procedure is described in the next chapter.

REQUIREMENTS

To run the EZ-KIT Lite software, you will need:

- 386- (or higher) based PC with a hard disk, high-density floppy disk drive, color video card and VGA monitor, and a minimum of 2 MB extended RAM.
- 4 MB of free disk space to install the software.
- DOS 3.1 or higher.
- Microsoft Windows 3.1 or higher.

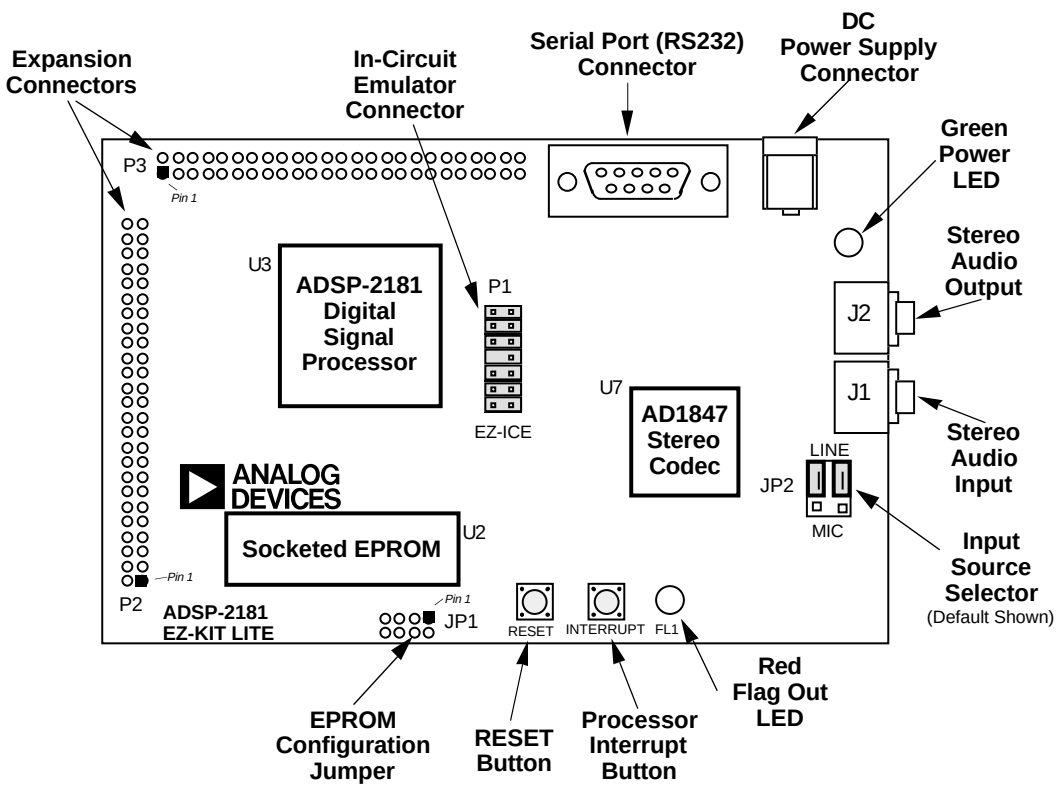
The setup program will create the following tree in the specified destination path:

21XX\BIN	Assembler, linker, simulator, and splitter DOS executables
21XX\ETC	Error messages and simulator on-line help
21XX\INCLUDE	2181 header file
21XX\LIB	2181 architecture file
21XX\EZKITL	Windows host program and demo .EXE files
21XX\EZKITL\2181\MONITOR	Source code and batch file for creating monitor program
21XX\EZKITL\2181\DSP	Source code and batch files for creating demo .EXE files
21XX\EZKITL\2181\DSP\ADPCM	
21XX\EZKITL\2181\DSP\DTMF	
21XX\EZKITL\2181\DSP\ECHO	
21XX\EZKITL\2181\DSP\FIRDEMO	
21XX\EZKITL\2181\DSP\LPC2K4	
21XX\EZKITL\2181\DSP\LPC7K8	

3 Getting Started

QUICK START HARDWARE INSTALLATION

The diagram below shows where connections for power, RS-232, audio input, and audio output are made. After all connections are made, apply power and press the reset button.



ADSP-2181 EZ-LAB Board Diagram

A detailed installation procedure is described in the next chapter.

Installation Procedures 4

SOFTWARE INSTALLATION

A diskette is included in your EZ-KIT Lite package. You should run the installation program contained on the diskette under Windows™. The installation program will create all the required directories and subdirectories and install the appropriate files.

SOFTWARE INSTALLATION PROCEDURE

The software requires:

- 386- (or higher based) PC with a hard disk, high-density floppy disk drive, color video card and VGA monitor, and a minimum of 2 MB extended RAM.
- 4 MB of free disk space to install the software.
- DOS 3.1 or higher.
- Microsoft Windows 3.1 or higher.

STEP 1: Make Working Copies Of The Diskettes

Before installing the software, you should copy the original diskette onto a working disk. This can be accomplished with either the XCOPY or COPY MS-DOS command (do not use the DISKCOPY command). The original disk should be stored in a safe place and used only to create a working disk.

STEP 2: Modify Your CONFIG.SYS File

To ensure proper operation of the development software, add (or modify) the directive `FILES=25` in the `CONFIG.SYS` file (`FILES` may also be set to a number greater than 25). You should also add (or modify) the directive `BUFFERS=30` in the `CONFIG.SYS` file. After modifying the `CONFIG.SYS` file, the PC must be rebooted for the change to take effect.

4 Installation Procedures

STEP 3: Install The Software On Your Hard Disk

Follow these directions to install the software on your hard disk.

1. Make sure Windows is running.

2. Insert EZ-KIT Lite Software disk into a floppy drive. (usually A: or B:)

3. Open the Program Manager window, if it is not already open.

4. From the File menu, select Run. The Run dialog box should appear.

5. In the "Command line" box type `A:\SETUP.EXE` (or `B:\SETUP.EXE`).

6. Click on OK or press enter.

7. Follow the instructions on the screen.

Note: You will need to edit your AUTOEXEC.BAT file in the following manner.

1. Add the environment variable `SET ADI_DSP=C:\ADI_DSP`

2. Append to the path variable `C:\ADI_DSP\21XX\BIN`

The setup program will create the following tree in the specified destination path:

21XX\BIN	Assembler, linker, simulator, and splitter DOS executables
21XX\ETC	Error messages and simulator help
21XX\INCLUDE	2181 header file
21XX\LIB	2181 architecture file
21XX\EZKITL	Windows host program and demo files
21XX\EZKITL\2181\MONITOR	Source code and batch file for monitor
21XX\EZKITL\2181\DSP	Source code and batch files for demos
21XX\EZKITL\2181\DSP\ADPCM	
21XX\EZKITL\2181\DSP\DTMF	
21XX\EZKITL\2181\DSP\ECHO	
21XX\EZKITL\2181\DSP\FIRDEMO	
21XX\EZKITL\2181\DSP\LPC2K4	
21XX\EZKITL\2181\DSP\LPC7K8	

Installation Procedures 4

ENVIRONMENT VARIABLES

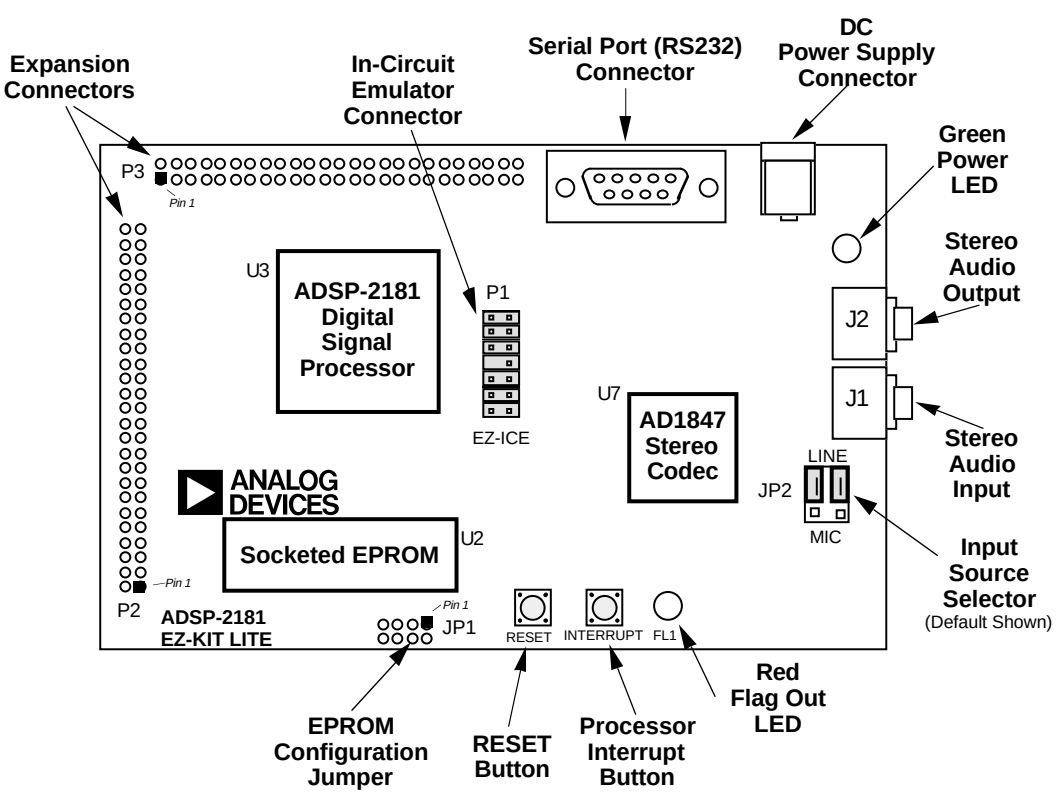
The following environment variables should be created and assigned default values (they should be copied into your `AUTOEXEC.BAT` file, to be invoked automatically when the PC is booted):

`ADI_DSP`	Path to the directory containing the installed files (default is `ADI_DSP= C:\ADI_DSP\`)

HARDWARE INSTALLATION

Place the board on a flat surface. Connect J3 to a serial port (COM port) on a PC. If the serial port on the PC is IBM compatible the cable should be a straight through DB9 male to DB9 female cable. The PC may have a 25 pin connector instead of a 9 pin connector. In this case, you can use a DB9 male to DB25 female adapter.

Connect amplified speakers to J2. Labtec part number CS-550 is an example of such speakers. Make sure that the amplifier for the speakers is on. Optionally a microphone or line level audio source may



be connected to J1. JP2 must be set to correspond to the type of input connected to J1. Two examples of suitable microphones are Radio Shack part number 33-1060 and Labtec part number AM-22.

Plug a 9 volt DC power supply capable of supplying at least 300 milli-amps into J4. The plug should be a P5 type connector with an outer diameter of 5.5 mm and an inner diameter of 2.1 mm. The outer sleeve of the plug must be positive polarity. Radio Shack catalog number 273-1455C is a suitable device. When power is applied the green LED labeled D2 will come on. When the DSP boots from the EPROM it will send out a sign-on message over the RS-232 connection, generate an audio signal which is sent to the speakers, and begin flashing the red LED (D1).

If the green LED fails to light, check your power connections. Verify that your power supply has the proper size connector and that the polarity is correct. The power supply voltage measured at the connector to the board should be in the range of 8 to 10 volts DC. Also, make sure that there are no objects beneath or on top of the board that may be causing a short circuit.

If the power connection is good and the green LED is lit yet the red LED does not flash and no audio signal was produced, make sure that the EPROM is properly seated in the socket.

Hit the reset button if the board appears to be operating improperly.

DSP System Development 5

OVERVIEW

If you'd like to develop your own programs, you can use the software development tools provided with the EZ-KIT Lite. If you have limited experience in developing code for a DSP based system you should review the following steps. Reading all the related product documentation would also be very useful.

The following development steps serve as a guideline for creating your own programs. Keep in mind that the development process varies depending upon the style of the particular developer. Follow this guideline as a starting point and feel free to modify it to suit your own work style. In many cases you will be able to skip a step because of the components shipped with EZ-KIT Lite. For example, the hardware development and architecture description process is described below, yet it is not necessary to follow this step since the ADSP-2181 development board is included in the EZ-KIT Lite package. There is also an architecture description file for the ADSP-2181 board included with the EZ-KIT Lite. This file is to be used with the development system software. The following sections explain these topics in more detail.

All commands that are mentioned throughout the following sections are to be typed at the DOS prompt (C:\>). In most cases, it does not matter in which actual directory you are.

Step 1: System Requirements

The first step in developing a DSP system is to determine what capabilities the system will need. These capabilities will depend on the types of algorithms being implemented, the types of signals being used and the types of I/O devices that need to be connected to the DSP processor. An evaluation of the size requirement for data memory and program memory is made based on the amount of data being acquired by the I/O and the amount of processing being performed. The estimated size of the program created by implementing the algorithms used also determines the required program memory.

For example, I may be building a speech processing system so I select an appropriate algorithm, such as LPC, for speech compression. This algorithm requires a certain amount of data storage as specified by the algorithm and a certain amount of program instruction storage determined by the amount of code needed to implement the algorithm. Of course, since at this stage the algorithm has not yet been implemented, these requirements are only approximations. For this example, let's say I need 4K words of data memory and 1K words of program memory. The internal memory of the ADSP-2181 is large enough. I would like to use an A/D and D/A for analog signal I/O. In this case I will connect an audio codec to the serial port of the DSP. I would like the DSP processor to be able to keep up with the speed of the data samples coming in from the I/O. The 33 MHz ADSP-2181 has more than enough speed.

Step 2: System Design

Once the system requirements are determined, a hardware system can be designed. In this case, an ADSP-2181 based system has been designed for you. This design utilizes an AD1847 Audio Codec to perform the A/D and D/A conversions. The AD1847 is connected to serial port 0 (SPORT0). The internal memory of the ADSP-2181 (16K words of program memory and 16K words of data memory) is all that is needed so that no external memory is connected. The signals of serial port 1 (SPORT1) are used to communicate via the RS-232 interface.

Step 3: Architecture Description File

When using the ADSP-2100 family development tools, the hardware system needs to be described in an architecture description file. Since the ADSP-2181 system is already defined, this step has been already done for you. A file called ADSP2181.ACH (file produced by the System Builder) is included in the EZ-KIT Lite software. The text file shown below contains the architecture description of a typical ADSP-2181 system and is used as input for the System Builder.

```
.system demo;
.adsp2181;
.mmap0;

.seg/pm/ram/abs=0/code/data      int_pm_lo[8192];
.seg/pm/ram/abs=8192/code/data   int_pm_hi[8192];
.seg/dm/ram/abs=0/data           int_dm_lo[8192];
.seg/dm/ram/abs=8192/data        int_dm_hi[8160];

.endsys;
```

The first three lines of the file defines the name of the system as `demo` using an ADSP-2181 with its EPROM boot feature enabled (MMAP pin tied low).

The 16K word internal program memory is described as two 8K word segments which can contain code as well as data. The first 8K word section will always be internal to the ADSP-2181. The second 8K word section can be programmed with an ADSP-2181 instruction (sets PMOVLAY modes) to be either internal or external overlay. Describing the architecture as two sections of memory will simplify memory management for applications that may use added external memory. The memory can also be described as a single 16K word section instead of the two 8K word sections.

Data memory is described as two sections as is the program memory. The 32 words at the end of data memory are reserved for the memory mapped control registers inside the ADSP-2181.

This text file (.SYS file) is used as input to the System Builder to create a .ACH file. The .ACH file is used by the Linker and the Simulator to flag inconsistencies between the software and use of the hardware. An architecture file called ADSP2181.ACH is supplied with the ADSP-2181 EZ-KIT Lite software. You would need to obtain the System Builder if you wish to define a different architecture. You can upgrade your system by purchasing the ADDS-21XX-SW-PC development system. This system contains the System Builder as well as a number of additional software tools and libraries.

Step 4: Code Development

Once the hardware is determined (or you feel comfortable with the approximation you have made of the hardware needs), you can begin to develop the software. First, determine all the memory requirements for variables and arrays along with all the needed interrupts for the ADSP-2181 system. Any hardware or registers that need to be initialized should also be planned out.

You will write a program by entering text (assembly language instructions) into a text file and then processing the text file with the assembler. The Assembler translates the processor's algebraic, easy-to-read instruction set from your source file into a relocatable object file. There are a few things that you need to know, however, about the

ADSP-2100 family assembly language. The algebraic syntax uses the '=' symbol to represent a data transfer. For example, the instruction

```
AX0 = MX0;
```

transfers data from the register called MX0 into the register called AX0. The arithmetic symbols that you are already familiar with (+, -, *) are used to denote arithmetic operations. For example, the instruction

```
AR = AX0 + AY0;
```

adds the contents of register AX0 to the contents of register AY0 and places the sum into register AR.

All instructions must end with a ';' and instructions can be made up of several “clauses” which are separated by a ','.

The following is an example of a multifunction instruction for the ADSP-21xx processors.

```
MR=MR+MX1*MY1(SU), MX1=DM(I0,M3), MY1=PM(I4,M5);
```

The first “clause” of the instruction (up to the first comma) is the multiply / accumulate (MAC) operation. The contents of the input registers MX1 and MY1 are multiplied together and added to the contents of the multiplier result register. The second clause loads the X input register from data memory (DM) and the third clause loads the Y input from program memory (PM). The I registers are used to hold addresses and the M registers hold the value used to modify the address after the fetch. All instructions can execute in a single processor cycle. Most instructions are also conditional.

Declaration of any variables, arrays or constants as well as the specification of any include files or ports are made in the front of the program using the assembler directives. Examples of some assembler directives are shown in the listing below.

After system reset, the ADSP-2181 will begin executing code from program memory location 0. As you can see in the listing below, location 0 contains an instruction `jump start`. This is used to jump over the interrupt vector table.

When an interrupt occurs during ADSP-2181 operation, the program flow will be redirected to the locations shown in the listing below. This jump is done automatically by the hardware when an interrupt is detected. The code segment shown below includes the interrupt vector table and will reside in the first 48 locations of program memory. The hexadecimal memory locations are shown for convenience in the comment field. The Linker will take care of memory address values for you.

You should fill unused interrupt locations with the `rti` instruction. This is done for safety. If for some reason program memory gets corrupted or program flow gets "lost" in the interrupt vector table, the return from interrupt instruction (`rti`) will bring program flow back into the program.

The following code example initializes all the EZ-KIT Lite hardware and will take the analog input samples and simply output them. The signal at the input connector is just passed to the output connector. You can take this code and insert a signal processing algorithm. The code is commented where the input samples are read and where the output samples are written.

```
.module/RAM/ABS=0  my_program;

{********  Constant Declarations ******}
{memory mapped ADSP-2181 control registers }
.const IDMA=                0x3fe0;
.const BDMA_BIAD=           0x3fe1;
.const BDMA_BEAD=           0x3fe2;
.const BDMA_BDMA_Ctrl=      0x3fe3;
.const BDMA_BWCOUNT=        0x3fe4;
.const PFDATA=              0x3fe5;
.const PFTYPE=              0x3fe6;
.const SPORT1_Autobuf=      0x3fef;
.const SPORT1_RFSDIV=       0x3ff0;
.const SPORT1_SCLKDIV=      0x3ff1;
.const SPORT1_Control_Reg=  0x3ff2;
.const SPORT0_Autobuf=      0x3ff3;
.const SPORT0_RFSDIV=       0x3ff4;
.const SPORT0_SCLKDIV=      0x3ff5;
.const SPORT0_Control_Reg=  0x3ff6;
.const SPORT0_TX_Channels0= 0x3ff7;
.const SPORT0_TX_Channels1= 0x3ff8;
.const SPORT0_RX_Channels0= 0x3ff9;
```

(listing continues on next page)

```
.const SPORT0_RX_Channels1= 0x3ffa;
.const TSCALE=              0x3ffb;
.const TCOUNT=              0x3ffc;
.const TPERIOD=             0x3ffd;
.const DM_Wait_Reg=         0x3ffe;
.const System_Control_Reg=  0x3fff;

{**** Variable and Buffer Declarations ******}
.var/dm/ram/circ rx_buf[3];  /* AD1847 receive buffer */
.var/dm/ram/circ tx_buf[3];  /* AD1847 transmit buffer */
.var/dm/ram/circ init_cmds[13];
.var/dm          stat_flag;

{***** Variable and buffer initialization ******}
.init tx_buf:  0xc000, 0x0000, 0x0000;
.init init_cmds:
    0xc003,   {AD1847 Left input control reg}
    0xc103,   {AD1847 Right input control reg}
    0xc288,   {AD1847 left aux 1 control reg}
    0xc388,   {AD1847 right aux 1 control reg}
    0xc488,   {AD1847 left aux 2 control reg}
    0xc588,   {AD1847 right aux 2 control reg}
    0xc680,   {AD1847 left DAC control reg}
    0xc780,   {AD1847 right DAC control reg}
    0xc85b,   {AD1847 data format register}
    0xc909,   {AD1847 interface configuration reg}
    0xca00,   {AD1847 pin control reg}
    0xcc40,   {AD1847 miscellaneous information reg}
    0xcd00;   {AD1847 digital mix control reg}

{*******  Interrupt Vector Table   ******}
        jump start; {Location 0000: reset }
         rti;
         rti;
         rti;

         rti;       {Location 0004: IRQ2 }
         rti;
         rti;
         rti;

         rti;       {Location 0008: IRQL1 }
         rti;
         rti;
         rti;

         rti;       {Location 000C: IRQL0 }
         rti;
         rti;
         rti;
```

```
   ar = dm(stat_flag);        {Location 0010: SPORT0 tx }
   ar = pass ar;
   if eq rti;
   jump next_cmd;

   jump input_samples;        {Location 0014: SPORT0 rx }
         rti;
         rti;
         rti;

         rti;                 {Location 0018: IRQE }
         rti;
         rti;
         rti;

         rti;                 {Location 001C: BDMA }
         rti;
         rti;
         rti;

         rti;                 {Location 0020: SPORT1 tx or IRQ1 }
         rti;
         rti;
         rti;

         rti;                 {Location 0024: SPORT1 rx or IRQ0 }
         rti;
         rti;
         rti;

         rti;                 {Location 0028: timer }
         rti;
         rti;
         rti;

         rti;                 {Location 002C: power down }
         rti;
         rti;
         rti;


{********  ADSP 2181 intialization    ********}
start:
        i0 = ^rx_buf;  {init. address pointer to start of buffer }
        l0 = %rx_buf;  {init. length register to size of buffer}
        i1 = ^tx_buf;
        l1 = %tx_buf;
        i3 = ^init_cmds;
        l3 = %init_cmds;
        m1 = 1;
```

(listing continues on next page)

```
{*****    Serial Port 0 (SPORT0) Set Up   *****}
   ax0 = b#0000001010000111;
   dm (SPORT0_Autobuf) = ax0;

   ax0 = 0;
   dm (SPORT0_RFSDIV) = ax0;
   dm (SPORT0_SCLKDIV) = ax0;
   ax0 = b#1000011000001111;
   dm (SPORT0_Control_Reg) = ax0;

   ax0 = b#0000000000000111;
   dm (SPORT0_TX_Channels0) = ax0;

   ax0 = b#0000000000000111;
   dm (SPORT0_TX_Channels1) = ax0;

   ax0 = b#0000000000000111;
   dm (SPORT0_RX_Channels0) = ax0;

   ax0 = b#0000000000000111;
   dm (SPORT0_RX_Channels1) = ax0;



{*****    Serial Port 0 (SPORT0) Set Up   *****}
   ax0=0;
   dm(SPORT1_Autobuf)=ax0;     { autobuffering disabled }
   dm(SPORT1_RFSDIV)=ax0;      { RFSDIV not used }
   dm(SPORT1_SCLKDIV)=ax0;     { SCLKDIV not used }
   dm(SPORT1_Control_Reg)=ax0; { ctrl functions disabled }


{******    Timer Setup    *****}
   ax0=0;
   dm(TSCALE)=ax0;             { timer not being used }
   dm(TCOUNT)=ax0;
   dm(TPERIOD)=ax0;

{******     System and Memory Set Up    ******}
   ax0 = b#0000000000000000;
   dm (DM_Wait_Reg) = ax0;

   ax0 = b#0001000000000000;   { enable SPORT0 }
   dm (System_Control_Reg) = ax0;

   ifc = b#00000011111111;     { clear pending interrupt }
   nop;

   icntl = b#00000;
   mstat = b#1000000;
```

```
{*****    AD1847 Codec intialization     *******}


        ax0 = 1;
        dm(stat_flag) = ax0;        { clear flag }
        imask = b#0001000000;       {enable transmit interrupt }
        ax0 = dm (i1, m1);          { start interrupt }
        tx0 = ax0;

check_init:
        ax0 = dm (stat_flag);       { wait for entire init }
        af = pass ax0;              { buffer to be sent to }
        if ne jump check_init;      { the codec            }

        ay0 = 2;

check_acih:
        ax0 = dm (rx_buf);          { once initialized, wait }
        ar = ax0 and ay0;           { for codec to come out  }
        if eq jump check_acih;      { of autocalibration     }

check_acil:
        ax0 = dm (rx_buf);          { once initialized, wait }
        ar = ax0 and ay0;           { for codec to come out  }
        if ne jump check_acil;      { of autocalibration     }

        idle;

        ay0 = 0xbf3f;               { unmute left DAC }
        ax0 = dm (init_cmds + 6);
        ar = ax0 AND ay0;
        dm (tx_buf) = ar;
        idle;

        ax0 = dm (init_cmds + 7);   { unmute right DAC }
        ar = ax0 AND ay0;
        dm (tx_buf) = ar;
        idle;

        ax0 = 0xc901;               {clear autocalibration request}
        dm (tx_buf) = ax0;
        idle;
        ax1 = 0x8000;               {control word to clear over-range flags}
        dm (tx_buf) = ax1;


        ifc = b#00000011111111;     {clear any pending interrupt}
        nop;

        imask = b#0000100000;       { enable rx0 interrupt }
```

(listing continues on next page)

```
{*****     wait for interrupt and loop forever    *****}

talkthru:  idle;
           jump talkthru;



{******  Interrupt service routines   *******}

{*****  receive interrupt used for loopback   *****}
input_samples:
        ena sec_reg;            {use shadow register bank}

        ax1 = dm (rx_buf + 1);  {get data from codec}
        mx1 = dm (rx_buf + 2);

{*****  Put your code here to process samples received from
        CODEC. Left channel data is in register ax1 and
        right channel data is in register mx1.   ********}


        dm (tx_buf + 1) = ax1;  {send data to codec}
        dm (tx_buf + 2) = mx1;

        rti;


{*****  transmit interrupt used for Codec initialization  *****}
next_cmd:
        ena sec_reg;
        ax0 = dm (i3, m1);    { fetch next control word and }
        dm (tx_buf) = ax0;    { place in transmit slot 0    }
        ax0 = i3;
        ay0 = ^init_cmds;
        ar = ax0 - ay0;
        if gt rti;            { rti if more control words }
        ax0 = 0x8000;         { else set done flag and }
        dm (tx_buf) = ax0;    { remove MCE if done with init }
        ax0 = 0;
        dm (stat_flag) = ax0; { reset status flag }
        rti;


.endmod;
```

Step 6: Running The Assembler

After you have finished creating the text file which contains the assembly language program, you can run the assembler with the following command.

```
asm21   my_prog  -2181
```

In this example, the text file I created is named `my_prog.dsp`. The `-2181` switch tells the assembler to accept ADSP-2181 specific instructions. The assembler creates the appropriate object code file or files. There are a number of assembler switches that can be optionally used for functions like list file creation, case sensitivity and object file naming. Here is another example which specifies the creation of a listing file which will be called `my_prog.lst`.

```
asm21   my_prog  -2181 -l
```

If you type the command `asm21` followed by a carriage return (no arguments used), the proper use of the command and all the switches will be listed on the screen. The assembler creates an object file with the extension `.obj` on the file name. The example command shown above creates the file `my_prog.obj`.

Step 7: Running The Linker

The linker creates an executable file from the object modules created by the assembler. The following example creates a file called `demo.exe`.

```
ld21   my_prog   -a adsp2181   -e demo
```

The architecture description file `adsp2181.ach` is specified with the `-a` switch and the executable file name is specified as `demo.exe` with the `-e` switch. There are a number of other switches that are used to create a symbol table, create a map file and specify the object files indirectly (specify a file name where the file contains a list of all the `.obj` files to be linked). If you type the command `ld21` followed by a carriage return (no arguments used), the proper use of the command and all the switches will be listed on the screen.

Step 8: Running The Simulator

The simulator lets you run your code in a simulation environment to test your software without using an actual hardware system. The simulation step is used to make sure your software works before you run it on your hardware.

Many time a problem can arise where you load your software on to the hardware and it doesn't work. Without verifying your program's operation on the simulator, you can not be sure whether the failure is related to hardware or software. If you have verified the operation of your software on the simulator then download the code to your hardware and it doesn't work, it most likely is due to faulty hardware. The simulator is invoked as follows:

```
sim2181  -a adsp2181  -e demo
```

This command starts the ADSP-2181 simulator, simulating the program `demo.exe` on the hardware described in the architecture description file `adsp2181.ach`. If the simulator can not find some of these files, you may want to check your path statement in the `autoexec.bat` or you can specify the full path as follows:

```
sim2181  -a c:\adi_dsp\21xx\lib\adsp2181  -e demo
```

Step 9: Programming An EPROM

Once you have verified that your software works you can format the executable so that it can be programmed into an EPROM. The EPROM can then be inserted into the EPROM socket on the board to run your program. The Prom Splitter can be invoked with the following command:

```
spl21    demo    demoprom    -loader    -2181
```

This will take the executable file `demo.exe` and create a PROM file called `demoprom.bnm`. The default PROM format is the Motorola S record format. You can specify the Intel Hex record format with the `-i` switch. The file created by the example command shown above is `demoprom.bnm`. This file can be downloaded to a PROM programmer to program an EPROM which can be inserted into the EPROM socket on the EZ-KIT Lite board.

Upon power up reset or when you hit the reset button on the board, the contents of the EPROM is automatically loaded into the internal program and data memories of the ADSP-2181 and coded execution begins.

Step 10: Running The ADSP-2181 EZ-KIT Lite Board

The EZ-KIT Lite board has an EPROM on it which contains ADSP-2181 code. When the board is powered up (or reset) the code is automatically transferred from the EPROM into the internal memory of the ADSP-2181. The code shipped with the EZ-KIT Lite includes a monitor program which allows the ADSP-2181 to communicate with the RS-232 interface. The code performs a self test and then sends an audio signal to the audio output connector. You will need to have a set of powered speakers attached to the audio output connector to hear the sound produced. The signal at this connector is a line level so you will need a set of speakers that have an amplifier in them such as the model CS-180 from Labtec.

Once you install the host software onto your PC you should be able to run the host program under Windows. You can then down load an executable file (.exe file produced by the linker). You should refer to the sections of this manual that describe installation in more detail.

If you prefer, you can program your own EPROM and insert it into the EPROM socket on the EZ-KIT Lite board to run your own program in stand alone mode.

Step 11: Debugging

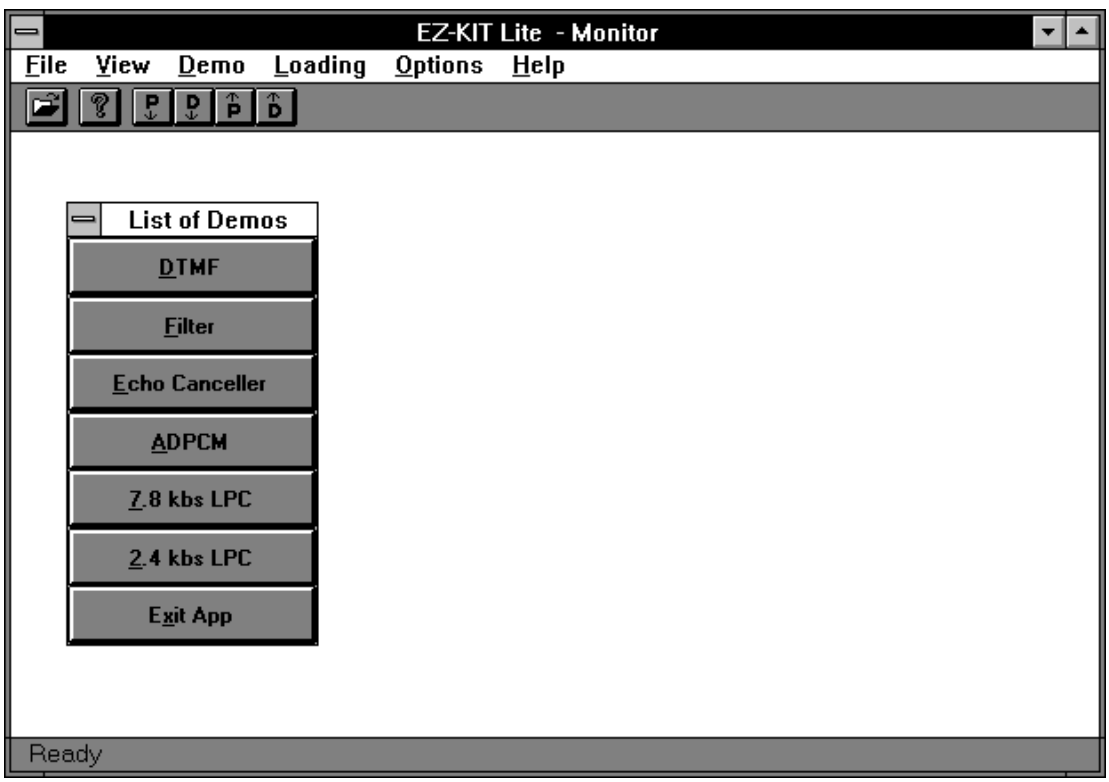
In general, if your EZ-KIT Lite board is not operating properly with the programs provided there is probably a problem with the power connection, the connection to the host PC via the RS-232 cable, or a problem with the components on the board. In general make sure that all connections are made tightly and that the power supply voltage is in the proper range (8 VDC to 10 VDC). Also make sure that the EPROM is seated properly in the socket. Finally, make sure that no objects are resting on the board or beneath the board causing a short circuit. When in doubt, press the reset button.

In general, if the software you develop does not work you should simulate it to try to find the problem. There are a number of example programs shipped with the EZ-KIT Lite. Use these as a basis to create your own programs. More detail on debugging is found throughout this manual.

EZ-KIT Lite Host Program 6

PROGRAM OVERVIEW

The EZ-KIT Lite Host Program is a Windows-based application program following standard Windows Graphical User Interface conventions. This is the application program you use to communicate with the EZ-LAB board. With it you can run EZ-KIT Lite demonstration programs, upload / download program and data memory contents, download user DSP programs, and execute user DSP programs. The following screen shows the main menu of the Host Program.



6 EZ-KIT Lite Host Program

COMMAND SUMMARY

By pointing and clicking on menu items, you can select from the many commands available. This chapter describes in detail each of these commands. The following commands are available.

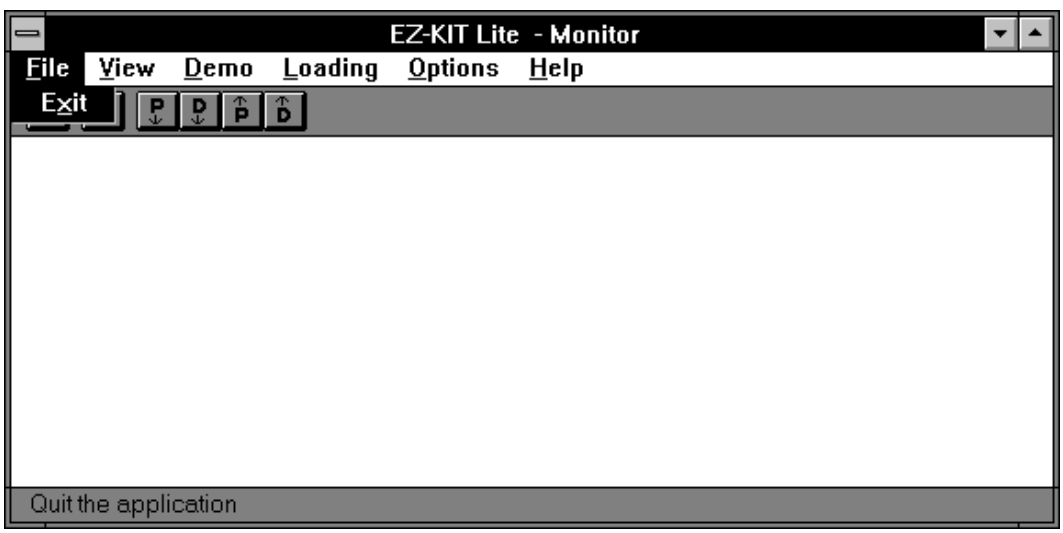
- **DTMF**
  run the DTMF demonstration program
- **Filtering**
  run the filtering demonstration program
- **Echo Cancellation**
  run the echo cancellation demonstration program
- **Speech Compression**
  run the LPC and ADPCM speech compression demonstration programs
- **Download DSP program & Go**
  download user programs and begin execution
- **Download DSP program**
  download user programs but do not start execution, also download memory image files
- **Go**
  begin execution from a given address
- **Upload data memory**
  upload a block of memory into a memory image file
- **Upload program memory**
  upload a block of memory into a memory image file
- **Download data memory**
  download a data memory image file to a given address
- **Download program memory**
  download a program memory image file to a given address
- **About**
  The last line in the dialog box indicates whether serial communication is established.

DETAILED COMMANDS

All commands are available via the standard menu bar selections and their drop-down menu items. You will notice that in addition to the standard Windows menu bar selections such as 'File' and 'Help,' there are some EZ-KIT Lite related items. All menu options are described in detail in the following sections.

File Menu

Since no information is retained in memory, the standard file save and file open commands are not applicable. There is only one command under the 'File' heading.



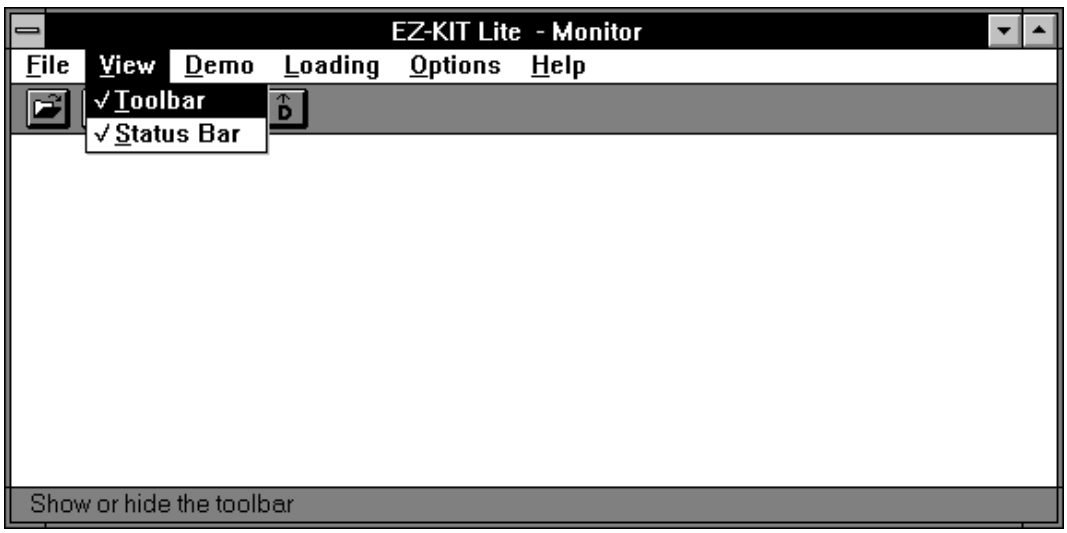
- *Exit*

Make this selection if you want to exit the EZ-KIT Host Program.

6 EZ-KIT Lite Host Program

View Menu

The View menu deals with the Toolbar and the Status Bar.



Toolbar

Selection of this feature will remove the check mark and remove the Toolbar near the top of the main window. Repeat to replace the check mark and bring back the Toolbar.

Toolbar is the gray bar immediately below the menu bar with several small square buttons on the left hand side. These buttons provide alternate quick access to useful commands. Simply click on these buttons with a mouse instead of selecting the drop down menu and select the corresponding menu items. A description follows.

- *Download user program & Go*

It is the first button from the left with the symbol of an opened folder. This button selects the corresponding menu selection under 'Loading' menu. It will download a user program and start user program execution. Select an ADSP-2181 executable file with a .exe extension. This is an executable file that is created by the ADSP-2181 linker.

- *About EzkitApp*

It is the second button from the left with a question mark. This button selects the corresponding menu selection under 'Help' menu. It displays the About dialog box.

The useful item in the About dialog box is the last line in the dialog box. If the Host Program is able to communicate with the EZ-KIT monitor program the last line will read 'EZ-KIT monitor is alive and well.' Otherwise the line will read 'EZ-KIT monitor is not running; try reset.' You may try to push the reset button on the EZ-KIT Lite board to re-activate the resident monitor.

- *Download program memory*

It is the third button from the left with the letter 'P' and a downward pointing arrow. This button selects the corresponding menu selection under 'Loading' menu. It will download a program memory image from a file to the DSP program memory at the specified starting address. You will need to select the Go command to run the program. Please see the corresponding section under 'Loading' menu for more detail.

- *Download data memory*

It is the fourth button from the left with the letter 'D' and a downward pointing arrow. This button selects the corresponding menu selection under 'Loading' menu. It will download a data memory image from a file to the DSP data memory starting from a user specified memory location. Please see the corresponding section under 'Loading' menu for more detail.

- *Upload program memory*

It is the fifth button from the left with the letter 'P' and a upward pointing arrow. This button selects the corresponding menu selection under 'Loading' menu. It will upload a program memory image from the DSP program memory to a file. You specify the specified starting address and the number of program memory locations to be transferred. Please see the corresponding section under 'Loading' menu for more detail.

- *Upload data memory*

It is the sixth button from the left with the letter 'D' and a upward pointing arrow. This button selects the corresponding menu selection under 'Loading' menu. It will upload a data memory image from the DSP data memory to a file. You specify the specified starting address and the number of data memory locations to be transferred. Please see the corresponding section under 'Loading' menu for more detail.

6 EZ-KIT Lite Host Program

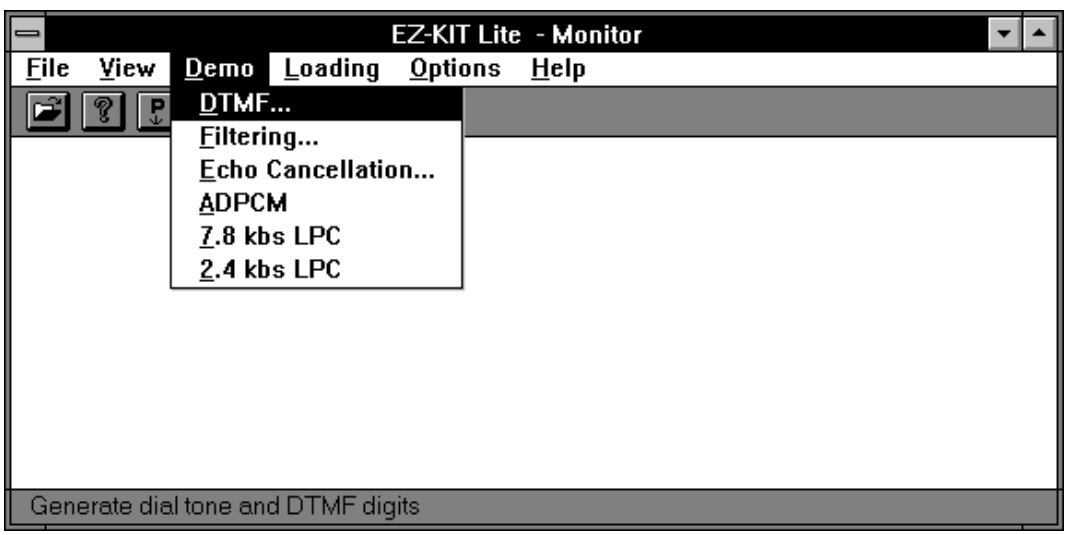
Status Bar

Selection of this option will remove the check mark and remove the Status Bar near the bottom of the main window. Repeat to replace the check mark and bring back the Status Bar.

When a menu bar selection is highlighted, a more detailed description of the currently highlighted menu selection is given in the Status Bar.

Demo Menu

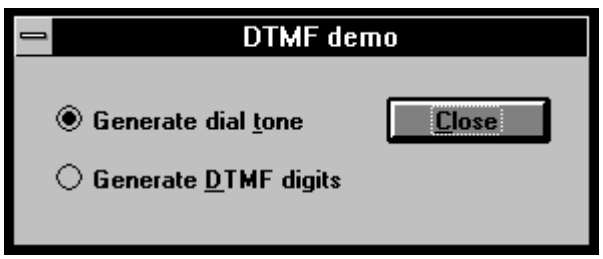
From the menu options, select one of the available demonstration programs to run on the EZ-KIT Lite. These demos may also be selected directly from the floating buttons. Please refer to the corresponding section for more detailed information.



DTMF

Selects the DTMF DSP demo program. This demonstration generates Dual-Tone Multi-Frequency (DTMF) tones, as used in the telephone network for push-button signaling. A DTMF tone is composed of two different single frequency tones - one of four row tones added to one of four column tones. Thus, a full implementation of a DTMF standard tone generator can generate 16 different tones (only 12 are commonly used on consumer handsets).

When selected the Windows Application Interface will download the necessary DSP program to the EZ-KIT LITE resulting in the following dialog box.



When it outputs a DTMF tone, EZ-KIT generates the two requisite sine waves, scales them and adds them, and then outputs the result to the codec for conversion to an analog signal. At the start of this program, EZ-KIT generates a dial tone to the output speaker. The following selections are possible.

- *Generate dial tone*

This selection may be selected by clicking the corresponding radio button or typing the letter ‘t.’ A standard dial tone which is a sum of continuous sine waves is generated.

- *Generate DTMF digits*

This option may be selected by clicking the corresponding radio button or typing the letter ‘d.’ The phone number of an Analog Devices DSP applications engineer is programmed into the demo program. Make this selection to generate the phone number. If you hold an off-hook telephone handset to the output speaker, the DTMF tones will tell the network call switching apparatus to connect you to the applications engineer. **Do this only if you want to make a call**.

- *Close*

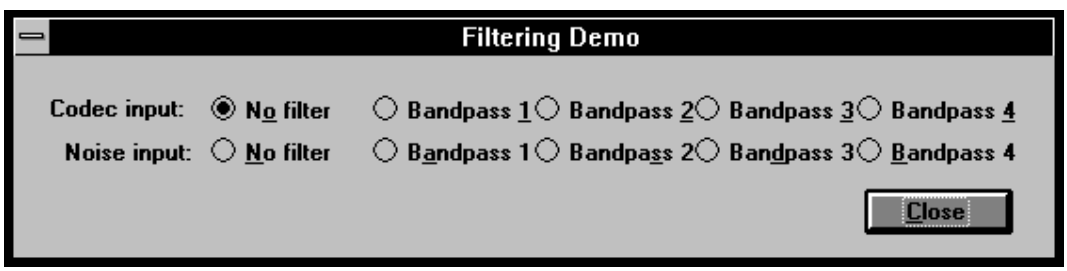
This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows a new demonstration to be selected.

6 EZ-KIT Lite Host Program

Filtering

Selects the FIR Filtering DSP demo program. This program demonstrates the effect of four bandpass filters against no filter on microphone input or an internally generated noise source.

When selected the Windows Application Interface will download the necessary DSP program to the EZ-KIT Lite and display the following dialog box.



This demonstration starts with a talk-through program. The AD1847 codec digitizes the analog microphone input and transmits the data to the DSP's serial port. The DSP reads data from the serial port and retransmits the data back to the codec. The codec converts the data to an analog signal that drives the speaker. No digital processing is performed on the data. When you speak into the microphone, you should hear your voice through the speaker.

The top row radio buttons use voice input from the microphone.

The filters have equivalent bandwidth and are evenly spaced on a logarithmic frequency axis. All FIR filters are 256 taps, and have been designed for 0.1 ripple.

FIR Filter	*Lower Stop Band*	*Pass Band*	*Upper Stop Band*
FIR1	0 - 269 Hz	328 - 448 Hz	547 - 4000 Hz
FIR2	0 - 426 Hz	521 - 710 Hz	866 - 4000 Hz
FIR3	0 - 675 Hz	825 - 1125 Hz	1375 - 4000 Hz
FIR4	0 - 1070 Hz	1308 - 1783 Hz	2179 - 4000 Hz

The bottom rows radio buttons use random noise as inputs.

- *Close*

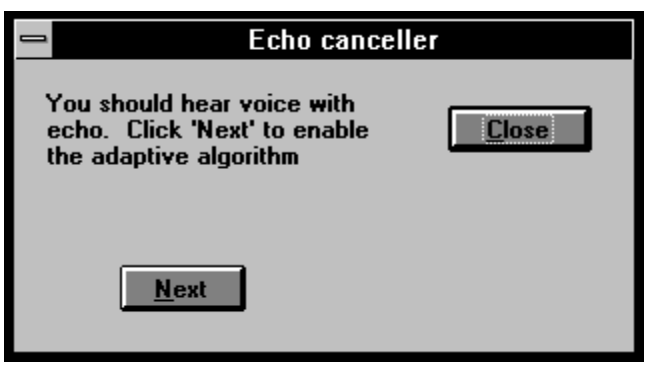
This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows a new demonstration to be selected.

Echo Cancellation

Selects the Echo Canceller DSP demo program. This program demonstrates echo cancellation on a simulated echoing channel.

The AD1847 codec digitizes the analog microphone input and transmits the data to the DSP's serial port. The DSP internally generates an echoed signal by summing the serial input with the output of a simulated echo channel. The echo channel consists of a linear delay implemented as a long FIR filter with zero-value tap weights and a 16-tap dispersive FIR filter. The echo canceled output is sent to the codec for reconstruction.

When selected the Windows Application Interface will download the necessary DSP program to the EZ-KIT Lite resulting in the following dialog box.

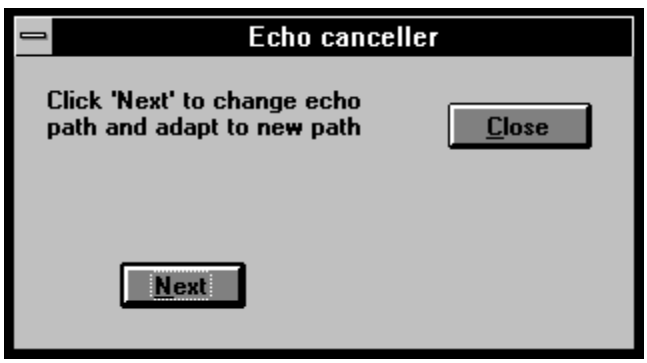


Initially the program sends the echoed output to the codec without echo cancellation. The following selections are possible.

- *Next*

When this button is depressed for the first time the program enters echo cancellation mode. The dialog box shown on the following page. There will be a significant reduction in echo. Further depression of the button will change the position of the dispersive filter and will force the filter to adapt to a new echo path.

6 EZ-KIT Lite Host Program



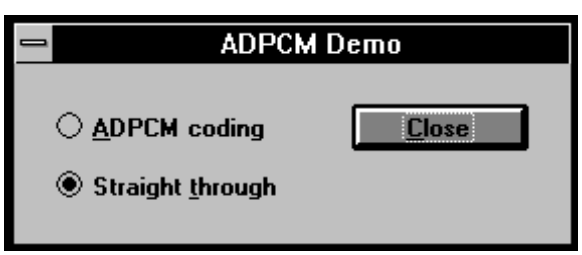
- *Close*

This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows a new demonstration to be selected.

ADPCM

Selects the ADPCM DSP demo program. This program demonstrates Adaptive Differential Pulse Code Modulation (ADPCM) capabilities. ADPCM consists of a number of real-time speech compression algorithms. For each sampling period it employs a linear predictive filter to generate a predicted output. The difference between the predicted output and actual sampled value is sent through the communication channel. Since the dynamic range of the differential error is significantly lower than the dynamic range of the voice signal, a lower bit rate results.

When selected the Windows Application Interface will download the necessary DSP program to the EZ-KIT Lite resulting in the following dialog box.



Initially the digitized microphone samples from the codec are sent directly to the codec for reconstruction. The following selections are possible.

- *ADPCM coding*

This option enables the ADPCM encoding / decoding feature. Each digitized microphone sample from the codec is first encoded and then decoded using ADPCM. After it is decoded, it is sent back to the codec for reconstruction. The red LED will light up when ADPCM encoding is in effect.

- *Straight through*

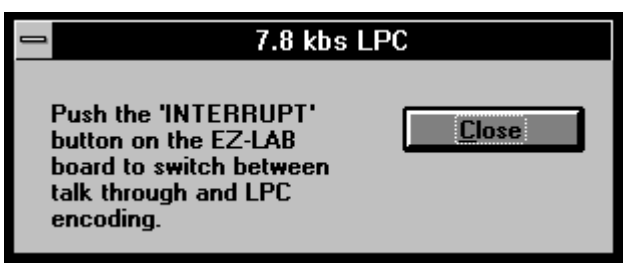
This option enables the straight through. The digitized microphone samples from the codec are sent directly back to the codec for reconstruction. The red LED will be off when straight through is in effect.

- *Close*

This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows a new demonstration to be selected.

7.8k LPC

Selects the 7.8k LPC DSP demo program.



- *Interrupt button on EZ-KIT Lite*

Push the Interrupt button on EZ-KIT Lite to toggle between talk through and 7.8k LPC encoding. The red LED will light up when LPC encoding is in effect.

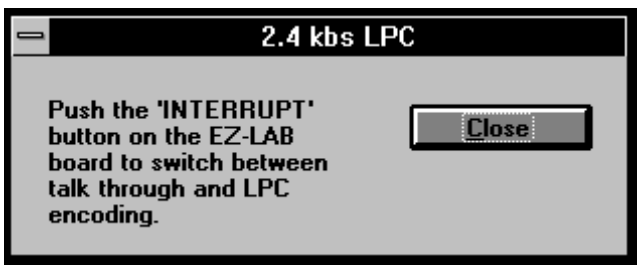
- *Close*

This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows new a demonstration to be selected.

6 EZ-KIT Lite Host Program

2.4k LPC

Selects the 2.4k LPC DSP demo program.



- *Interrupt button on EZ-KIT Lite*

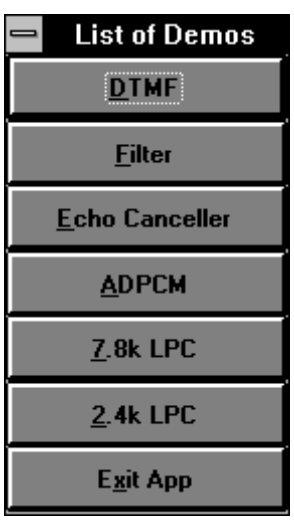
Push the Interrupt button on EZ-KIT Lite to toggle between talk through and 2.4k LPC encoding. The red LED will light up when LPC encoding is in effect.

- *Close*

This option terminates the demonstration program and restores the EZ-KIT resident monitor program before returning the user to the top menu bar selection. This allows a new demonstration to be selected.

Floating Menu

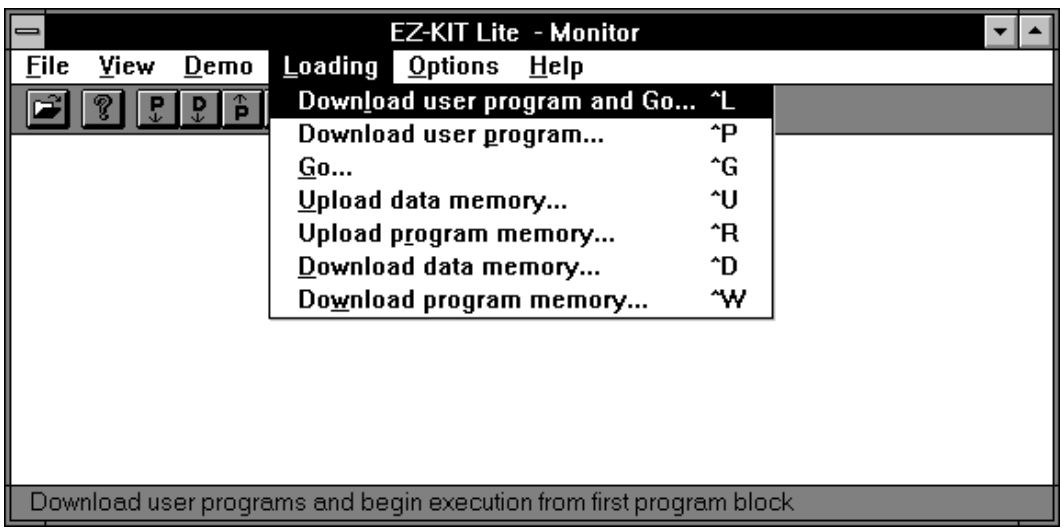
Alternatively all the demo programs can be selected by clicking the corresponding buttons in the following floating dialog box. This dialog box may be turned on or off via the options menu described later.



The extra button at the bottom of the dialog box provides an alternative and quick way of exiting the Window Application Interface program.

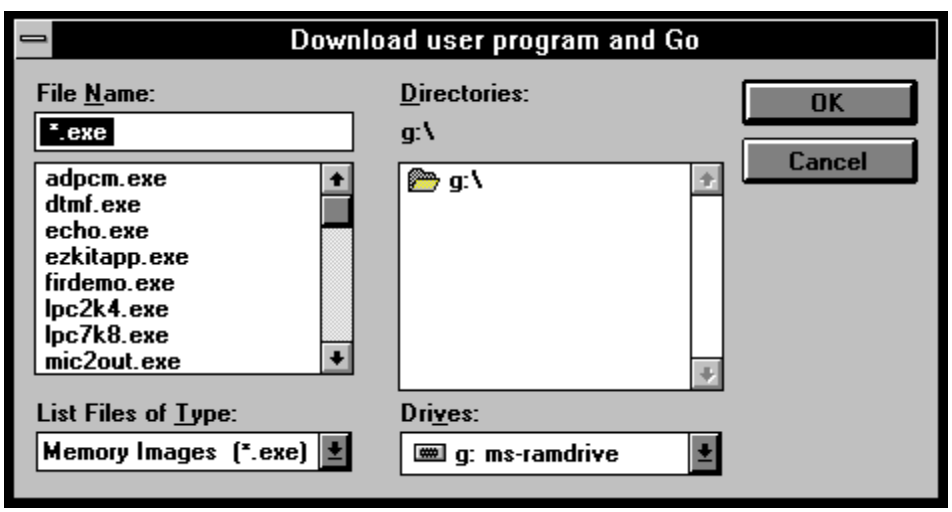
Loading Menu

These menu selections deal with the uploading and downloading of memory contents including user programs.



Download User Program and Go

When selected the following dialog box appears.



Selects a memory image file (*.exe) for downloading. Program memory and data memory images will be downloaded to locations specified in the memory image file kernel in the corresponding memory space. The monitor then transfers DSP execution to the newly downloaded program.

All user interrupt vectors will be directed to a special temporary buffer to enable the monitor to function. When the monitor is finally issued a command to execute user program, the redirected user interrupt vectors will be copied from the temporary buffer to the intended locations.

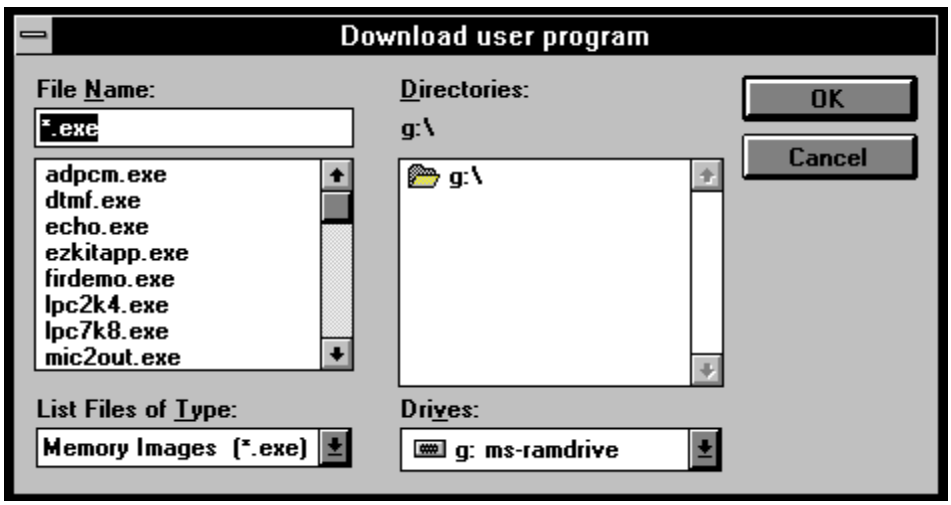
Interrupts are disabled before entering the user program. All user interrupts are masked using the DIS INTS instruction. The mask is also set to 0 and the timer is disabled.

The first location of the program memory image kernel encountered in the memory image file is used as the entry point to the user program. The monitor performs a subroutine call to that location. The user program should execute an RTS instruction to return to the monitor. Upon re-entering the monitor program all interrupt vectors are restored to the contents prior to the user program being downloaded to enable the monitor to continue to function. Alternatively, the user can set up the BDMA to simulate a power up BDMA boot load thus reloading the monitor program from the EPROM.

The monitor reserves certain memory locations. The user program should not use these monitor reserved memory locations. Please refer to chapter describing the EZ-KIT Lite Monitor Program for more details on reserved memory.

Download User Program

When selected the following dialog box appears.



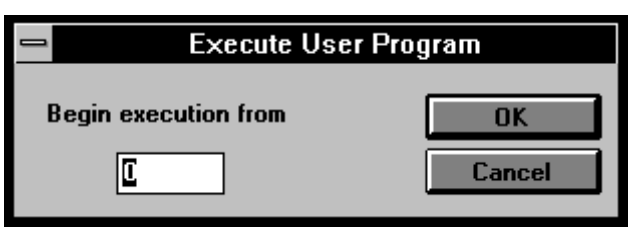
Selects a memory image file (*.exe) file for downloading. The difference between this option and the previous one is that this selection does not transfer DSP execution to the downloaded program upon download completion.

Upon download completion the monitor will continue to function. The program's entry point is reported in the acknowledgment dialog box. User program execution may be started by using the 'Go' command, described immediately below.

For all other details please refer to the previous section for 'Download user program and Go.'

Go

When selected the following dialog box appears.

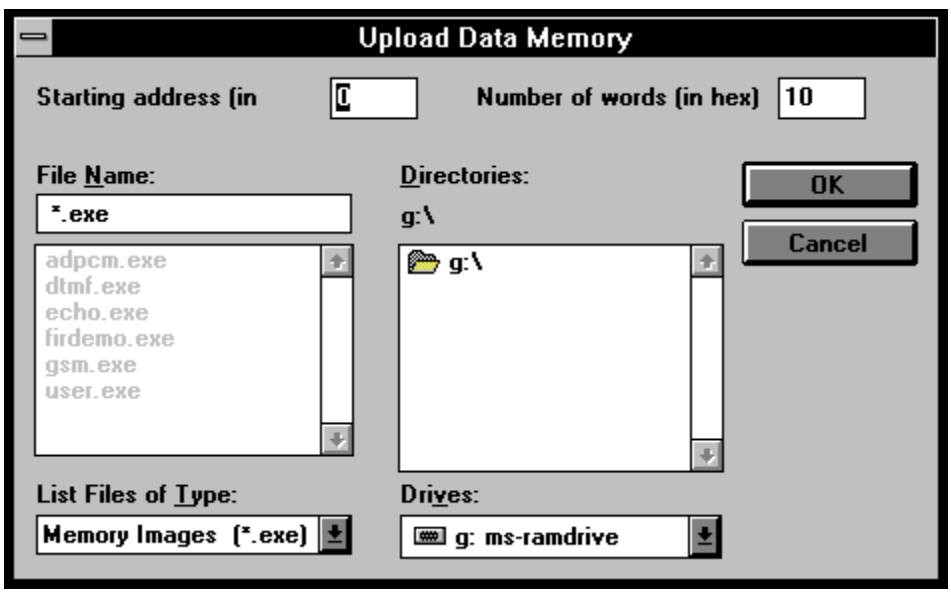


6 EZ-KIT Lite Host Program

Enter a hexadecimal address as the starting address of user programs. Valid address range is from 0x0000 to 0x37ff inclusive. Program memory locations 0x3800 to 0x3fff are used to store the monitor code. Loading into these locations will corrupt the monitor. For more details on user program restrictions please refer to the reference manual for EZ-KIT Lite.

Upload Data Memory

When selected the following dialog box appears.

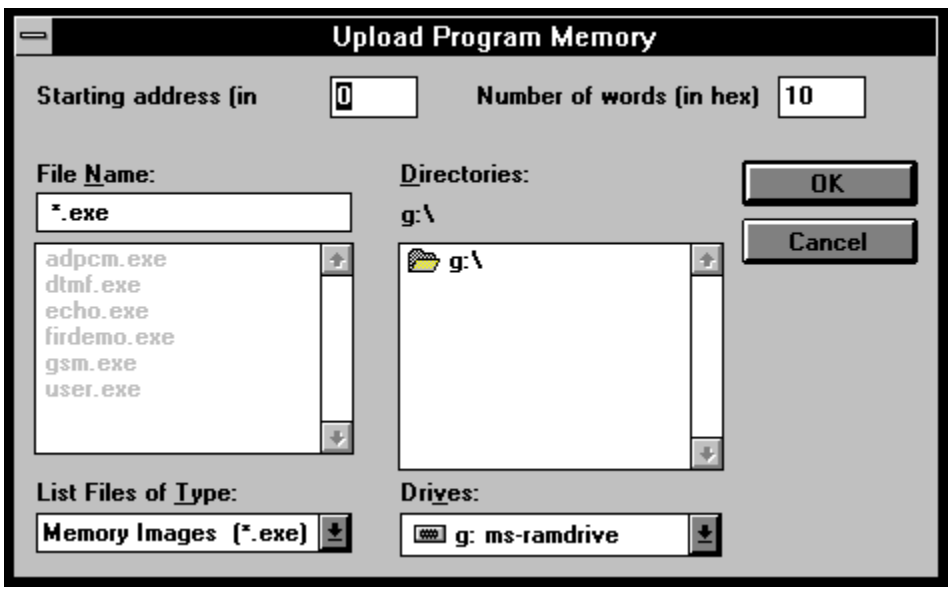


This command uploads a block of DSP program memory content to a memory image (*.exe) file on the host computer. Specify in hexadecimal the starting address and number of words. The resulting file may be downloaded back into the program memory at any specified non-restricted location. The file may also be examined by using any text editor.

You will get an error message if you entered an invalid hexadecimal number or specified an invalid range of memory, e.g. starting address plus number of words exceeds the memory range.

Upload Program Memory

When selected the following dialog box appears.



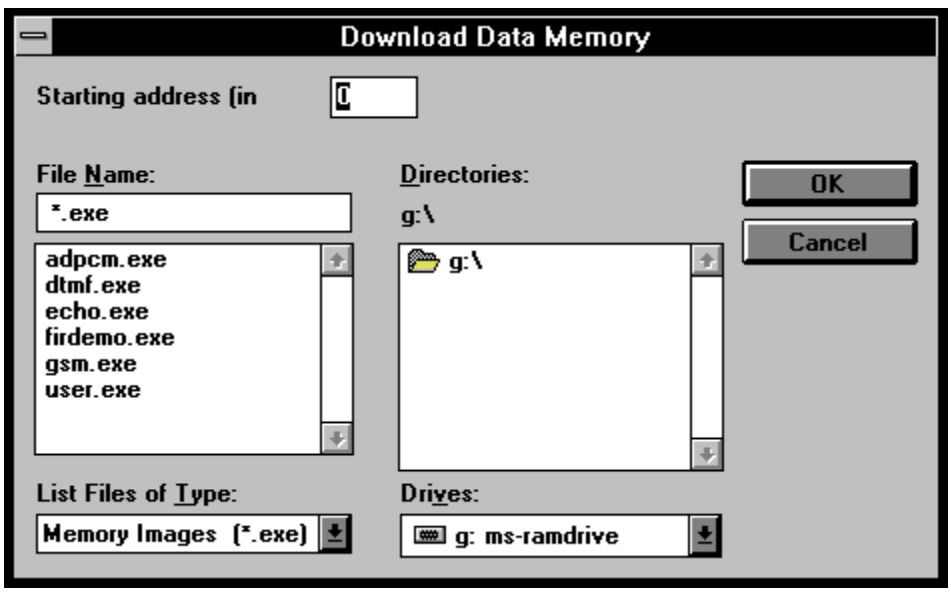
This command uploads a block of DSP data memory content to a memory image (*.exe) file on the host computer. Specify in hexadecimal the starting address and number of words. The resulting file may be downloaded back into the program memory at any specified non-restricted location. The file may also be examined by using any text editor.

You will get an error message if you entered an invalid hexadecimal number or specified an invalid range of memory, e.g. starting address plus number of words goes beyond the end of memory.

6 EZ-KIT Lite Host Program

Download Data Memory

When selected the following dialog box appears.



This command downloads a memory image (*.exe) file from the host computer into DSP data memory starting from the specified hexadecimal start address.

The first data memory image kernel in the image file will be downloaded. The memory image file may be memory image file generated (uploaded) by the Host Program or generated by the ADSP-21xx Linker. Since the Host Program generated file will have only one corresponding memory kernel, the entire file content will be downloaded. The linker generated image file may have multiple program memory and data memory kernels. Only the first data memory kernel will be downloaded and the rest will be ignored.

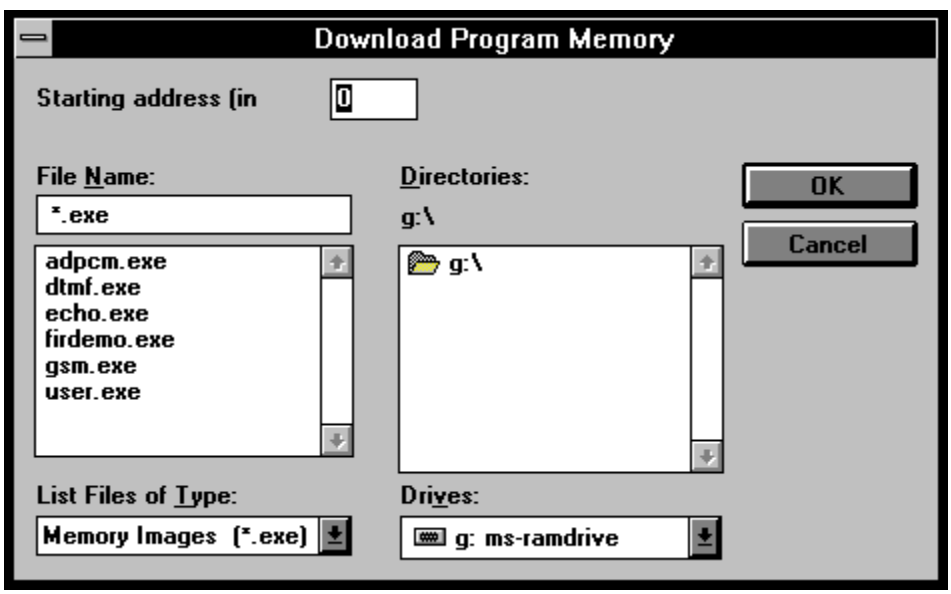
Since the memory image file format is an ASCII text file with no checksum information, it is possible to use an ASCII text editor to edit the memory image file and alter, delete, or add entries. Please refer to the ADSP-2100 Family Assembler & Simulator Manual for the format definition. However, as the format is rather straight forward, with a little inspection and experimentation you may be able to do the modification without referring to the manual.

You will get an error message if you entered an invalid hexadecimal number or specified an invalid range of memory, e.g. starting address plus number of words goes beyond the end of memory.

DO NOT overwrite the monitor reserved memory locations.

Download Program Memory

When selected the following dialog box appears.



This command downloads a memory image (*.exe) file from the host computer into DSP program memory starting from the specified hexadecimal start address.

The first program memory image kernel in the image file will be downloaded. The memory image file may be memory image file generated (uploaded) by the Host Program or generated by the ADSP-21xx Linker. Since the Host Program generated file will have only one corresponding memory kernel, the entire file content will be downloaded. The linker generated image file may have multiple program memory and data memory kernels. Only the first program memory kernel will be downloaded and the rest will be ignored.

6 EZ-KIT Lite Host Program

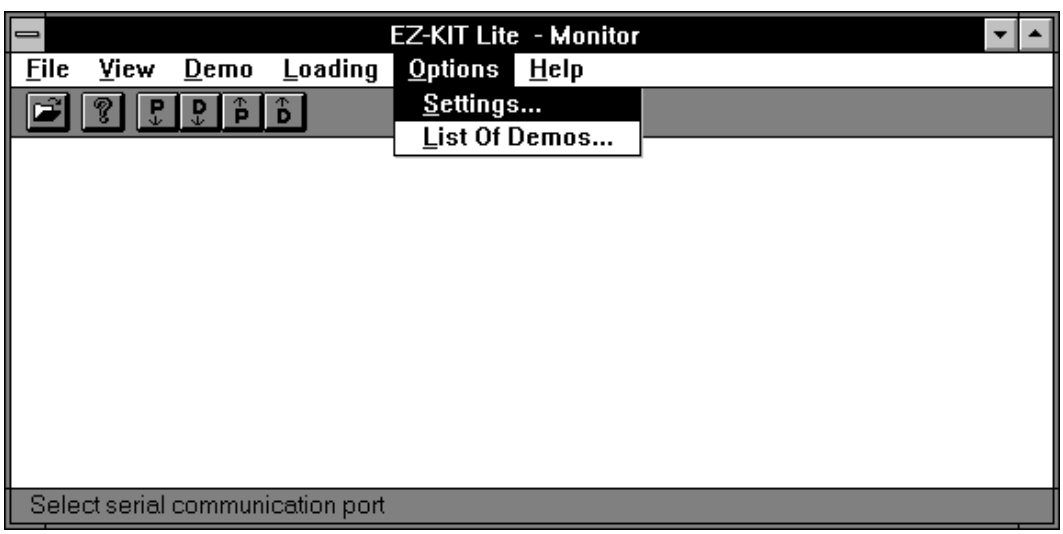
Since the memory image file format is an ASCII text file with no checksum information, it is possible to use an ASCII text editor to edit the memory image file and alter, delete, or add entries. Please refer to the *Cross-Software Manual* for the format definition. However, as the format is rather straight forward, with a little inspection and experimentation you may be able to do the modification without referring to the manual.

You will get an error message if you entered an invalid hexadecimal number or specified an invalid range of memory, e.g. starting address plus number of words goes beyond the end of memory.

DO NOT overwrite the monitor reserved memory locations.

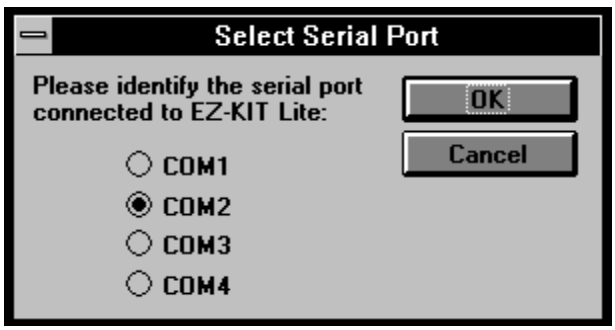
Options Menu

These selections determine the operation of the Host Program.



Settings

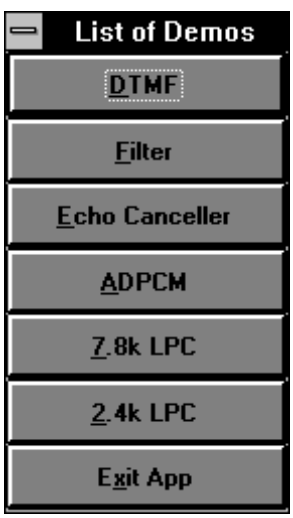
When selected the following dialog box appears.



Please select the corresponding serial communication port to which the EZ-KIT Lite board is connected. This selection is saved in the configuration file 'ezkitapp.ini.' However, it is not critical to have the 'ezkitapp.ini' file available; if the Host Program did not find the configuration file it will bring this dialog box up to allow the user to make a selection.

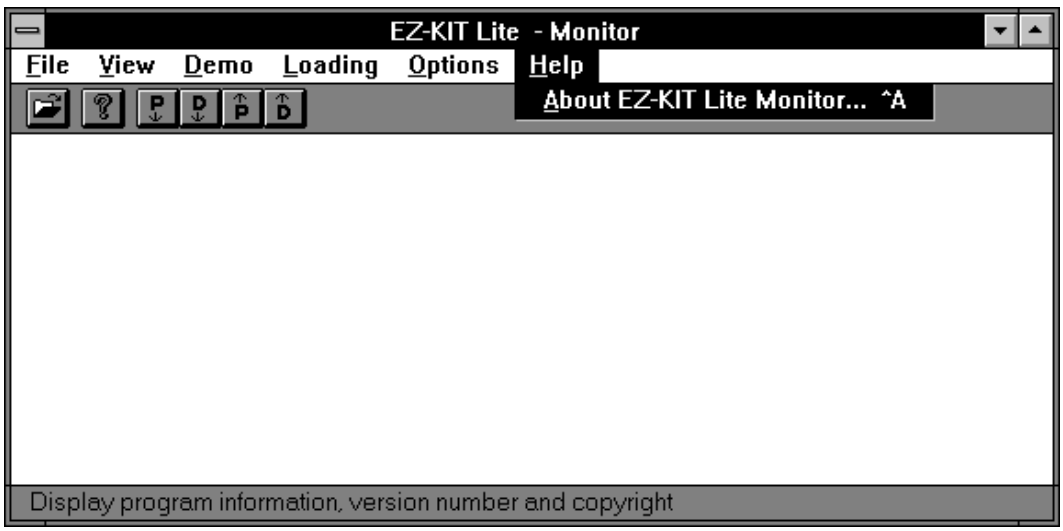
List Of Demos

When selected, the following dialog box appears.



6 EZ-KIT Lite Host Program

Help Menu



About EZ-KIT

When selected the following dialog box appears.

The third line will indicate whether the Host Program could communicate with the resident monitor. The line 'EZ-KIT monitor is alive and well' is displayed if the host to monitor communication is functioning; otherwise the line 'EZ-KIT monitor is not running; try reset' is displayed. Click 'OK' to remove the dialog box.

User Configurable Settings

Several operating aspects of the Host Program can be customized by changing the initialization string in 'EzkitApp.ini.' Two of these are automatically managed by the Host Program. The only one that may need to be manually modified is the time-out duration for the dialog box. The 'EzkitApp.ini' is not critical to the functioning of the Host Program. If the file is not found the Host Program simply uses default settings and prompts the user for the correct serial communication port number. Details follow.

• [EZ-KIT Interface App]
This is the heading for the section and it should not be changed.

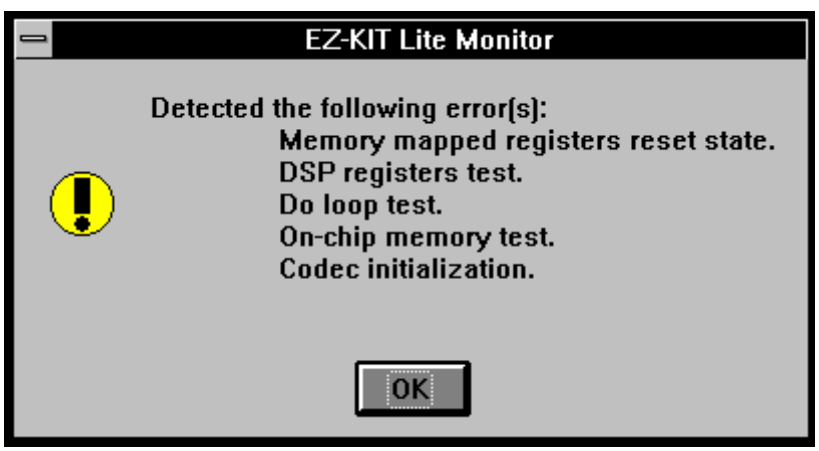
• PortNo=2
This is the serial communication port number. It is managed by the Host Program and should not be changed.

• MessageTimeOut=5
This is the dialog box time out duration. The default value is five seconds. Setting it to zero will disable the automatic time out feature. This is the only setting that may require direct user modification.

• ShowFloatingDialogBox=1
This is the visibility of the floating dialog box. If this is 1 the floating dialog will be visible when Host Program begins running. It is managed by the Host Program and should not be changed.

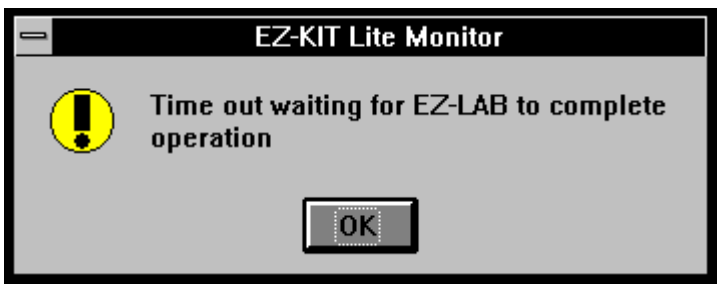
Error Messages & Troubleshooting

Several error messages may be reported should there be any problems encountered in communicating to the EZ-KIT Lite monitor program. These error messages are described below.



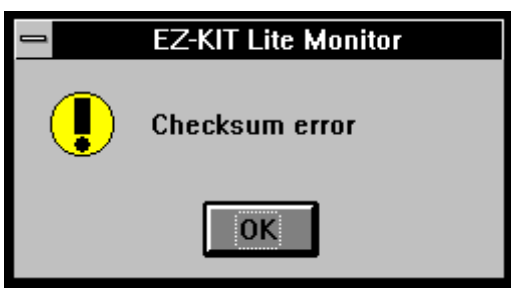
This indicates that the resident monitor detected the indicated error during its power up self test. Push the reset button on the EZ-KIT Lite board and try again. A persistent problem may indicate the board is defective.

6 EZ-KIT Lite Host Program



This indicates that the Host Program has timed out waiting for the monitor to respond. A possible cause is that the monitor reserved memory locations may have been corrupted. Push the reset button on the EZ-KIT Lite board and try again.

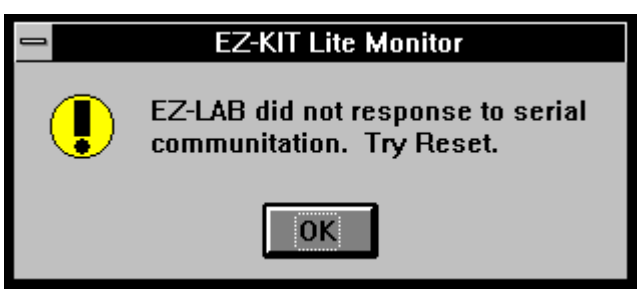
This indicates that the Windows Communication Module has reported an error to the Host Program. A possible cause is conflict between Windows applications. Close all other programs and try again. It will also help to verify that the serial communication port selected is functioning by temporarily reconfiguring the mouse driver to use this port.



This indicates that a checksum error has been detected. For all memory transfers a checksum is computed for the block of memory content being transferred by both the monitor and the Host Program. A possible cause is excessive line or environment noise. If the problem persists try shortening the length of the serial communication port cable. Also experiment with the location of the EZ-KIT Lite board and your compter.

This indicates that the Host Program is unable to ascertain the correct functioning of the EZ-KIT Lite resident monitor program. A possible cause is that the Host Program encountered problem with the Windows Communication Module. Exit Windows and try again.

This is the catch-all error message. All other unknown errors result in this message. A possible cause may be missing demo program images files, corrupted files, and internal programming error.



This indicates the Host Program did not get a response from the monitor program. A possible cause is that the EZ-KIT Lite board is not connected to the selected serial port, or that another Windows application is using the selected serial port. Close all other Windows applications and try again.

6 EZ-KIT Lite Host Program

This indicates that the User Program Image File being downloaded contains memory initializations to the resident monitor reserved memories. The download is aborted to protect the reserved memories. Remove the offending memory initializations and try again.

RUNNING DEMOS

To run the Demonstration Programs first start the Host Program under Windows.

The following instructions apply to the Demonstration Programs. Refer to the section on the DEMO MENU for more detailed information.

- DTMF
- Filtering
- Echo Cancellation
- ADPCM
- 7.8k LPC
- 2.4k LPC

You can select a demo by clicking one of the buttons on the Floating Menu or selecting the appropriate dropdown menu as described in the previous sections. Once the demo program is loaded onto the EZ-KIT Lite board, it begins to execute. A dialog box will appear with instructions on running the demo. (See Demo Menu section).

CREATING YOUR OWN PROGRAMS

The Host Program running on the PC communicates through the COM port to the EZ-KIT Lite board. The commands sent by the host software to the EZ-KIT Lite board and the response received by the host are described in this chapter. All values are in binary format. Use this information if you desire to create your own Host Program. The following are the serial commands that EZ-KIT Lite understands.

1. Beep
   sent: $$$
   received: ok[led count][alive called count][selftest hi][selftest lo]
   [cmds4demo hi][cmds4demo lo]

   [led count] is the lower 8 bits of a counter incremented at 28800 rate.
   [alive called count] is incremented by one each time alive is called.
   [selftest hi][selftest lo] is the selftest result
   [cmds4demo hi][cmds4demo lo] is demo command variable
   sing / generate 1 kHz tone.

2. Alive inquiry
   sent: $OK
   received: ok[led count][alive called count][selftest hi][selftest lo]
   [cmds4demo hi][cmds4demo lo]

   [led count] is the lower 8 bits of a counter incremented at 28800 rate.
   [alive called count] is incremented by one each time alive is called.
   [selftest hi][selftest lo] is the selftest result
   [cmds4demo hi][cmds4demo lo] is demo command variable

3. upload DM
   sent: $UD[hi start][lo start][hi len][lo len]
   received: [hi][lo]...[hi sum][lo sum]

4. upload PM
   sent: $UP[hi start][lo start][hi len][lo len]
   received: [hi][mi][lo]...[hi sum][mi sum][lo sum]

5. download DM
   sent: $DD[hi start][lo start][hi len][lo len][hi][lo]...
   received: ![hi sum][lo sum]

6. download PM
   sent: $DP[hi start][lo start][hi len][lo len][hi][mi][lo]...
   received: ![hi sum][mi sum][lo sum]

7. Call subroutine
   sent: $GO[hi address][loaddress]
   received: ![hi address][loaddress]

EZ-KIT Lite Monitor Program 7

PROGRAM OVERVIEW

The Monitor Program resides in an on-board EPROM. It is automatically loaded into the on-chip program and data memories at reset. As soon as the monitor begins execution it performs a self test of DSP registers, on-chip memories, and a reset and initialization of AD1847 codec. It then waits for commands via the RS-232 serial communication port. A codec talk through routine is also running in an interrupt routine.

MONITOR FEATURES

This monitor has the following features.

- Power up self test
- AD1847 codec initialization and talk through
- 9600 bps UART emulation
- upload and download of both program and data memory contents via serial link

RESTRICTIONS

In order to facilitate the function of the monitor, certain restrictions are imposed on all user programs. These are hard restrictions because violation will interfere with the proper downloading of the user programs. These restrictions include the following.

- Reserved program memory 0x3800 to 0x3fff
  This is where the monitor program resides. The user program should not download into these locations. Any violation will destroy the monitor resulting in incomplete user program download.

- Reserved data memory 0x3e00 to 0x3fdf
  These are the monitor operating variables. User programs downloading into these locations will interfere with the monitor execution resulting in an incomplete user program download.

- Returning to the monitor
  After the download of your program is complete, the monitor software calls your code as a subroutine. Your code should end with an `RTS` instruction to properly return to the monitor.

  For user programs that do not return to the monitor there is no restriction imposed after the program has been downloaded. In other words, while the memory image file should not attempt to initialize the reserved memories, the user program is nevertheless free to use these memories after its execution.

CREATING YOUR OWN PROGRAMS TO BE USED WITH THE MONITOR

You can edit one of the example programs supplied or you can create your own program using the development tools included in your EZ-KIT Lite. See chapter 5 for more information. There are a few things to note when developing programs that you want to run with the monitor.

User programs may use all non-reserved program memory and non-reserved data memory. These are locations 0x0000 to 0x37ff for program memory and locations 0x0000 to 0x3dff for data memory.

- Program entry point
  User programs to be downloaded by the Host Program should have their entry point at the beginning of the first program memory kernel. This can be achieved by making the entry point the first instruction in the module and specifying that module first during linking. In most cases, your program will start at location 0x0000. You can use the assembler directive `.module/ram/abs=0` to specify this (see examples).

- Interrupt enable
  All interrupts are masked (by IMASK = 0.) before execution is transferred to the user programs. This is so that the user program can safely carry out initializations before it re-enables interrupts.

- Interrupt vectors
  The user program must initialize all interrupt vectors it is using. Interrupt IRQ1 can not be used in your program since it is dedicated to the RS232 communications. You can place "dummy" `RTI` instructions in your code at this interrupt vector (location 0x0020 through 0x0023).

EZ-KIT Lite Monitor Program 7

- Monitor commands
  The source code for the monitor program is included in the directory 21XX\EZKITL\2181\MONITOR. You can edit the source code to tailor the monitor to your needs. If you are going to edit the monitor, you need to be aware of the commands the monitor responds to.

  The monitor can respond to the following commands via the serial communication link. It may be useful if you are interfacing the EZ-KIT Lite via serial link to other than the Host Program. The commands are the following.

0. This is the name of the command
   received: specifies the code and data received from the host(sent by the monitor). Unbracketed characters are sent exactly,; bracketed bytes are the high order and low order byte of a 16-bit quantity, or the high, middle, and lower order bytes of a 24-bit quantity.
   sent: specifies the code and data sent to the host by the monitor. Unbracketed characters are to be sent exactly as specified. Bracketed bytes are the high order and low order byte of a 16-bit quantity, or the high, middle, and lower order bytes of a 24-bit quantity.

1. Beep
   received: $$$
   sent: ok[led count][alive called count][selftest hi][selftest lo]

   [led count] is the lower 8 bits of a counter incremented at a rate of 28800 per second. A stalled count indicates the timer interrupt is probably not running.
   [alive called count] is incremented by one each time this command or the alive command is called.
   [selftest hi][selftest lo] is the selftest result
   A short tone will also be sent to the audio output by the monitor.

2. Alive inquiry
   received: $OK
   sent: ok[led count][alive called count][selftest hi][selftest lo]

   [led count] is the lower 8 bits of a counter incremented at a rate of 28800 per second. A stale count indicates the timer interrupt is probably not running.
   [alive called count] is incremented by one each time this command or the alive command is called.
   [selftest hi][selftest lo] is the selftest result
   This command is similar to the Beep command except no tone is generated.

3. upload data memory content
   received: $UD[hi start][lo start][hi len][lo len]
   sent: [hi][lo]...[hi sum][lo sum]$!

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][lo]... are the memory contents starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data sent by the monitor program.

4. upload program memory content
   received: $UP[hi start][lo start][hi len][lo len]
   sent: [hi][mi][lo]...[hi sum][mi sum][lo sum]$!

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][mi][lo]... are the memory contents starting from the first location.
   [hi sum][mi sum][lo sum] is a 24-bit checksum of the data sent by the monitor program.

5. download data memory content
   received: $DD[hi start][lo start][hi len][lo len][hi][lo]...
   sent: ![hi sum][lo sum]

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][lo]... are the memory contents starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data calculated and sent by the monitor program.

6. download PM
   receivedt: $DP[hi start][lo start][hi len][lo len][hi][mi][lo]...
   sent: ![hi sum][mi sum][lo sum]

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][mi][lo]... are the memory content starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data calculated and sent by the monitor program.
7. Begin user program execution
   received: $GO[hi address][loaddress]
   sent: ![hi address][loaddress]

   [hi address][loaddress] is the program entry point sent by the host to the monitor program. It is also the address sent by the monitor program back to the host for confirmation

DEBUGGING

The following techniques may be helpful in debugging user programs.

- SET FL1
- RESET FL1
- TOGGLE FL1

The FL1 LED is directly controlled by the DSP FL1 flag. These instructions may be placed anywhere in the users program. These may be used to provide a visual indication of the binary state of a variable, or whether a certain section is executing.

7 EZ-KIT Lite Monitor Program

For non time-critical programs, it is possible to provide information by programming the FL1 to blink at different rates. For example the following code blinks the flag at a certain rate. Change the counter value for other rates.

```
     cntr = 80;
     do lp1 until ce;
       cntr = 50;
       do lp2 until ce;
         cntr = 12000;
         do lp3 until ce;
lp3:       nop;
lp2:     nop;
lp1:   toggle fl1;
```

DSP MEMORIES

The user programs may store historical data in user program memory or user data memory. This data may be retrieved after user programs have returned control back to the Monitor program (by RTS).

The upload program memory or the upload data memory commands of the host program may be used to retrieve this data.

EZ-KIT Lite Monitor Program 7

PROGRAM OVERVIEW

The Monitor Program resides in an on-board EPROM. It is automatically loaded into the on-chip program and data memories at reset. As soon as the monitor begins execution it performs a self test of DSP registers, on-chip memories, and a reset and initialization of AD1847 codec. It then waits for commands via the RS-232 serial communication port. A codec talk through routine is also running in an interrupt routine.

MONITOR FEATURES

This monitor has the following features.

- Power up self test
- AD1847 codec initialization and talk through
- 9600 bps UART emulation
- upload and download of both program and data memory contents via serial link

RESTRICTIONS

In order to facilitate the function of the monitor, certain restrictions are imposed on all user programs. These are hard restrictions because violation will interfere with the proper downloading of the user programs. These restrictions include the following.

- Reserved program memory 0x3800 to 0x3fff
  This is where the monitor program resides. The user program should not download into these locations. Any violation will destroy the monitor resulting in incomplete user program download.

- Reserved data memory 0x3e00 to 0x3fdf
  These are the monitor operating variables. User programs downloading into these locations will interfere with the monitor execution resulting in an incomplete user program download.

- Returning to the monitor
  After the download of your program is complete, the monitor software calls your code as a subroutine. Your code should end with an `RTS` instruction to properly return to the monitor.

  For user programs that do not return to the monitor there is no restriction imposed after the program has been downloaded. In other words, while the memory image file should not attempt to initialize the reserved memories, the user program is nevertheless free to use these memories after its execution.

CREATING YOUR OWN PROGRAMS TO BE USED WITH THE MONITOR

You can edit one of the example programs supplied or you can create your own program using the development tools included in your EZ-KIT Lite. See chapter 5 for more information. There are a few things to note when developing programs that you want to run with the monitor.

User programs may use all non-reserved program memory and non-reserved data memory. These are locations 0x0000 to 0x37ff for program memory and locations 0x0000 to 0x3dff for data memory.

- Program entry point
  User programs to be downloaded by the Host Program should have their entry point at the beginning of the first program memory kernel. This can be achieved by making the entry point the first instruction in the module and specifying that module first during linking. In most cases, your program will start at location 0x0000. You can use the assembler directive `.module/ram/abs=0` to specify this (see examples).

- Interrupt enable
  All interrupts are masked (by IMASK = 0.) before execution is transferred to the user programs. This is so that the user program can safely carry out initializations before it re-enables interrupts.

- Interrupt vectors
  The user program must initialize all interrupt vectors it is using. Interrupt IRQ1 can not be used in your program since it is dedicated to the RS232 communications. You can place "dummy" `RTI` instructions in your code at this interrupt vector (location 0x0020 through 0x0023).

EZ-KIT Lite Monitor Program 7

- Monitor commands
  The source code for the monitor program is included in the directory 21XX\EZKITL\2181\MONITOR. You can edit the source code to tailor the monitor to your needs. If you are going to edit the monitor, you need to be aware of the commands the monitor responds to.

  The monitor can respond to the following commands via the serial communication link. It may be useful if you are interfacing the EZ-KIT Lite via serial link to other than the Host Program. The commands are the following.

0. This is the name of the command
   received: specifies the code and data received from the host(sent by the monitor). Unbracketed characters are sent exactly,; bracketed bytes are the high order and low order byte of a 16-bit quantity, or the high, middle, and lower order bytes of a 24-bit quantity.
   sent: specifies the code and data sent to the host by the monitor. Unbracketed characters are to be sent exactly as specified. Bracketed bytes are the high order and low order byte of a 16-bit quantity, or the high, middle, and lower order bytes of a 24-bit quantity.

1. Beep
   received: $$$
   sent: ok[led count][alive called count][selftest hi][selftest lo]

   [led count] is the lower 8 bits of a counter incremented at a rate of 28800 per second. A stalled count indicates the timer interrupt is probably not running.
   [alive called count] is incremented by one each time this command or the alive command is called.
   [selftest hi][selftest lo] is the selftest result
   A short tone will also be sent to the audio output by the monitor.

2. Alive inquiry
   received: $OK
   sent: ok[led count][alive called count][selftest hi][selftest lo]

   [led count] is the lower 8 bits of a counter incremented at a rate of 28800 per second. A stale count indicates the timer interrupt is probably not running.
   [alive called count] is incremented by one each time this command or the alive command is called.
   [selftest hi][selftest lo] is the selftest result
   This command is similar to the Beep command except no tone is generated.

3. upload data memory content
   received: $UD[hi start][lo start][hi len][lo len]
   sent: [hi][lo]...[hi sum][lo sum]$!

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][lo]... are the memory contents starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data sent by the monitor program.

4. upload program memory content
   received: $UP[hi start][lo start][hi len][lo len]
   sent: [hi][mi][lo]...[hi sum][mi sum][lo sum]$!

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][mi][lo]... are the memory contents starting from the first location.
   [hi sum][mi sum][lo sum] is a 24-bit checksum of the data sent by the monitor program.

5. download data memory content
   received: $DD[hi start][lo start][hi len][lo len][hi][lo]...
   sent: ![hi sum][lo sum]

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][lo]... are the memory contents starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data calculated and sent by the monitor program.

6. download PM
   receivedt: $DP[hi start][lo start][hi len][lo len][hi][mi][lo]...
   sent: ![hi sum][mi sum][lo sum]

   [hi start][lo start] is the starting memory location
   [hi len][lo len] is the number of words
   [hi][mi][lo]... are the memory content starting from the first location.
   [hi sum][lo sum] is a 16-bit checksum of the data calculated and sent by the monitor program.
7. Begin user program execution
   received: $GO[hi address][loaddress]
   sent: ![hi address][loaddress]

   [hi address][loaddress] is the program entry point sent by the host to the monitor program. It is also the address sent by the monitor program back to the host for confirmation

DEBUGGING

The following techniques may be helpful in debugging user programs.

- SET FL1
- RESET FL1
- TOGGLE FL1

The FL1 LED is directly controlled by the DSP FL1 flag. These instructions may be placed anywhere in the users program. These may be used to provide a visual indication of the binary state of a variable, or whether a certain section is executing.

7 EZ-KIT Lite Monitor Program

For non time-critical programs, it is possible to provide information by programming the FL1 to blink at different rates. For example the following code blinks the flag at a certain rate. Change the counter value for other rates.

```
     cntr = 80;
     do lp1 until ce;
       cntr = 50;
       do lp2 until ce;
         cntr = 12000;
         do lp3 until ce;
lp3:       nop;
lp2:     nop;
lp1:   toggle fl1;
```

DSP MEMORIES

The user programs may store historical data in user program memory or user data memory. This data may be retrieved after user programs have returned control back to the Monitor program (by RTS).

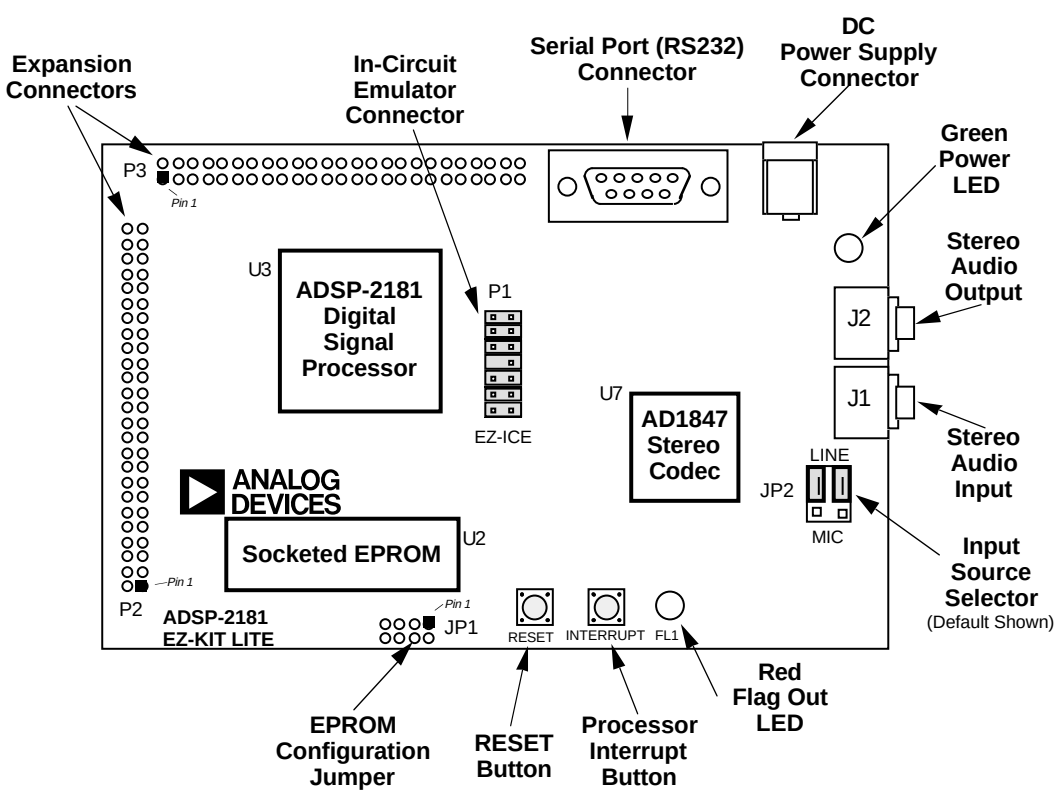
The upload program memory or the upload data memory commands of the host program may be used to retrieve this data.

EZ-KIT Lite Hardware Description

8

DESIGN OVERVIEW

The hardware consists of a printed circuit board measuring 3.5 inches by 5.5 inches. Assembled onto the printed circuit board are an ADSP-2181 digital signal processor, an EPROM, an AD1847 codec and various support circuits and connectors. The board is a complete signal processing system designed to demonstrate the capabilities of the ADSP-2181 digital signal processor. It can also be used as a platform to develop new applications for the ADSP-2181.



8 EZ-KIT Lite Hardware Description

The EZ-KIT Lite board is an example of a minimum implementation of an ADSP-2181 processor. The EPROM is connected to the processor via the Byte DMA Port. This interface uses only eight of the 24 data lines to carry data (D8 through D15). Eight of the spare data lines (D16 through D23) are used to provide additional address bits. This allows the ADSP-2181 to address up to 32 M bits (4 M bytes) of memory. On this board the EPROM socket is designed to accept EPROMs from 256K bits up to 8 M bits (the largest presently available). JP1 provides a way to adjust the function of pins 3 and 30 of the socket as required by the different size EPROMs. The DSP is configured to boot from the EPROM when reset is deasserted.

The AD1847 codec is connected to the DSP via SPORT0. This high speed synchronous serial port carries all of the data, control, and status information between the DSP and the codec. It is possible to disable the codec if the serial port is to be used for another purpose. The CODECDIS signal available on connector P3 can be used to disable the codec. When this signal is brought low, the codec is disabled and its signals are put in a high impedance state.

The SPORT1 pins are used to communicate with the host PC via the RS-232 interface (U5). The Flag In and Flag Out pins carry the receive and transmit data. The receive data also goes to IRQ1 so the DSP can detect activity without polling the Flag In pin. Software running on the DSP emulates a UART to provide the proper protocol for asynchronous serial communications at a data rate of 9600 bits per second.

U1 is a logic device containing six inverters. Two of these inverters are used to provide a power-on reset and to de-bounce the reset push button. Another two inverters are used to de-bounce the interrupt push button. A fifth inverter is used to drive the red LED (D1) because the FL1 pin cannot sink enough current to drive it directly. The sixth inverter is not used.

The IDMA port on the DSP is not used on the EZ-KIT Lite board. All of the IDMA signals are available on connector P3.

EZ-KIT Lite Hardware Description 8

SPECIFICATIONS

Processor:	ADSP-2181KS-133 operating at an instruction rate of 33 MHz (16.667 external clock)
Analog interface:	AD1847 stereo codec
Analog inputs:	One stereo pair of 2V RMS AC coupled line level inputs One stereo pair of 20mV RMS AC coupled microphone inputs
Analog outputs:	One stereo pair of 1V RMS AC coupled line level outputs
Power:	8 to 10V DC at 300 mA
Environment:	0 to 70° centigrade 10 to 90 percent relative humidity (non condensing)

CONNECTORS

J1 is a 1/8 inch (3.5 mm) stereo jack. This jack is used to bring either line level or microphone audio signals into the board.

J2 is also a 1/8 inch (3.5 mm) stereo jack. This jack is used to bring out line level audio signals from the board.

J3 is a female 9 pin D-Sub connector. It is used to communicate with a host computer using RS-232 signal levels and asynchronous serial protocols.

J4 is a jack for a 5.5 mm cylindrical plug. It is used to supply power to the board. The center pin of the jack is 2 mm diameter and should connect to the negative side of the power source. The outer sleeve of the mating plug must be positive.

JP1 is a site for an eight pin header. It can be used to configure the board for EPROM sizes other than the 1 Mbit (128K byte) EPROM (27C010) shipped with the board. Most users will not need this feature.

8 EZ-KIT Lite Hardware Description

JP2 is a six pin header. It is used to configure input jack J1 for either line level or microphone input. The center pin in each group of three is connected to one of the AD1847 codecs Line 1 Input pins. Jumpers (also known as shunts or shorting links) can be used to connect these pins to either the output of the microphone amplifier or to the output of the line level input filter.

P1 is a 14 pin header connector used to connect to an ADSP-2181 EZ-ICE® in-circuit emulator. Pin 7 should be removed for keying purposes.

P2 and P3 are sites for 50 pin header connectors. These connectors can be used to access the ADSP-2181 signals for expansion or test purposes.

U2 is a socket for an EPROM in a DIP package. As built the board will accept a 27C512 (64K byte) or 27C010 (128K byte) EPROM. Changing connections at JP1 allows the board to accept a 27C256 (32K byte), 27C020 (256K byte), 27C040 (512K byte), or 27C080 (1M byte) EPROM. This socket is connected to the ADSP-2181's byte-wide memory interface.

R28 is a site for a zero ohm resistor. If this resistor is installed the ADSP-2181 processor can reset the board under software control. The software would assert reset by configuring PF0 as an output and then setting it low.

R29 is another site for a zero ohm resistor. If this resistor is installed and X3 and C37 are removed the codec can operate off of the ADSP-2181's CLKOUT signal instead of its own 24.576 MHz clock. It will also be necessary to change X1 to a lower frequency value to stay within the codecs ratings.

SWITCHES

S1 is the reset push button switch. Pushing this button causes the ADSP-2181 processor and the AD1847 codec to go into the hardware reset state and remain there until it is released. The switches output is de-bounced electronically to prevent multiple transitions due to mechanical contact bounce.

S2 is the interrupt push button switch. Pushing this button causes the ADSP-2181 processor to receive an IRQE interrupt input. This may cause the processor to execute the current IRQE interrupt handler software if the interrupt is enabled and the IRQE interrupt vector is in place. The switches output is de-bounced electronically to prevent multiple interrupts due to mechanical contact bounce.

INDICATORS

D1 is a red light emitting diode which is controlled by the FL1 output of the ADSP-2181 processor. Software can control the state of this indicator by writing to an internal register. See the ADSP-2181 documentation for more details.

D2 is a green light emitting diode which is on whenever the board has power.

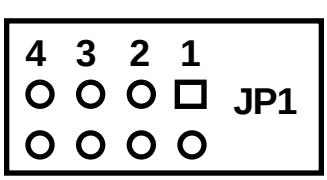
HARDWARE OPERATION

When power is applied to the board a reset circuit holds the processor in reset for approximately 30 ms. Reset is then deasserted and the processor begins the boot process. The BMODE and MMAP pins on the ADSP-2181 are grounded so the processor boots from the byte-wide memory interface which is connected to the EPROM socket. If the EPROM supplied with the board is installed in the socket the operation of the board will proceed as documented in the software section of this manual.

HARDWARE EXPANSION

CONFIGURING THE BOARD FOR DIFFERENT EPROMS

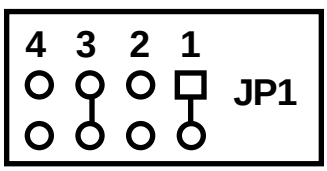
JP1 allows the ADSP-2181 EZ-KIT Lite board to be configured for any one of six different EPROM sizes. As the board is shipped it can accommodate either a 27C512 or 27C010. If some other size EPROM is installed in the socket at U2 it will be necessary to change the connections at JP1. JP1 looks like this.



8 EZ-KIT Lite Hardware Description

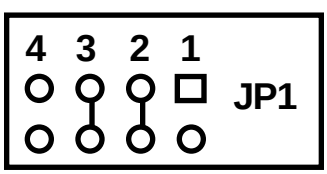
Connections are made vertically between pads. The pair of pads below each number constitutes the jumper position associated with that number. Connections can be made in several ways. If an eight pin header is installed and the etch connections on the back are cut then EPROM size changes can be accommodated easily by installing and removing shunts. If frequent size changes are not contemplated it may be sufficient to solder wires between the pads and so make the connections permanent.

For a 27C256 EPROM the connections should be as follows.



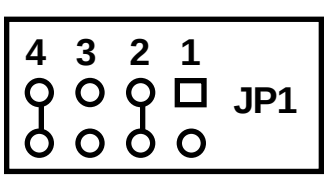
Note that this involves cutting the etch on the back of the board at jumper position 2 and adding a connection at jumper position 1.

For a 27C512 or 27C010 EPROM the connections should be as follows.



Note that this is how the connections are arranged when the board is manufactured.

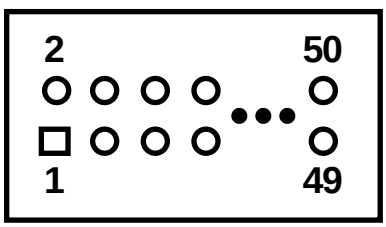
For a 27C020, 27C040 or 27C080 EPROM the connections should be as follows.



Note that this involves cutting the etch on the back of the board at jumper position 3 and adding a connection at jumper position 4.

EXPANSION CONNECTORS

P2 and P3 are sites for 50 pin header connectors which provide access to the ADSP-2181 signals for expansion or test purposes. The pin numbers on these connectors are arranged as follows.



The signals available on these pins are the following.

P2 Pin Number	Signal Name	P2 Pin Number	Signal Name
1	A0	2	A1
3	A2	4	A3
5	A4	6	A5
7	A6	8	A7
9	A8	10	A9
11	A10	12	A11
13	A12	14	A13
15	D0	16	D1
17	D2	18	D3
19	D4	20	D5
21	D6	22	D7
23	D8	24	D9
25	D10	26	D11
27	D12	28	D13
29	D14	30	D15
31	D16	32	D17
33	D18	34	D19
35	D20	36	D21
37	D22	38	D23
39	$\overline{WR}$	40	$\overline{RD}$
41	$\overline{IOMS}$	42	$\overline{BMS}$
43	$\overline{DMS}$	44	$\overline{CMS}$
45	$\overline{PMS}$	46	$\overline{BR}$
47	$\overline{BGH}$	48	$\overline{BG}$
49	VCC	50	GND

8 EZ-KIT Lite Hardware Description

P3 Pin Number	Signal Name	P3 Pin Number	Signal Name
1	GND	2	IAD0
3	IAD1	4	IAD2
5	IAD3	6	IAD4
7	IAD5	8	IAD6
9	IAD7	10	IAD8
11	IAD9	12	IAD10
13	IAD11	14	IAD12
15	IAD13	16	IAD14
17	IAD15	18	GND
19	IACK	20	IAL
21	IS	22	IWR
23	IRD	24	GND
25	PF0	26	PF1
27	PF2	28	PF3
29	PF4	30	PF5
31	PF6	32	PF7
33	FL0	34	FL1
35	FL2	36	CLKOUT
37	RESET	38	IRQL0
39	IRQL1	40	IRQ2
41	PWD	42	PWDACK
43	CODECDIS	44	TXD0
45	TFS0	46	RFS0
47	RXD0	48	SCK0
49	VCC	50	GND

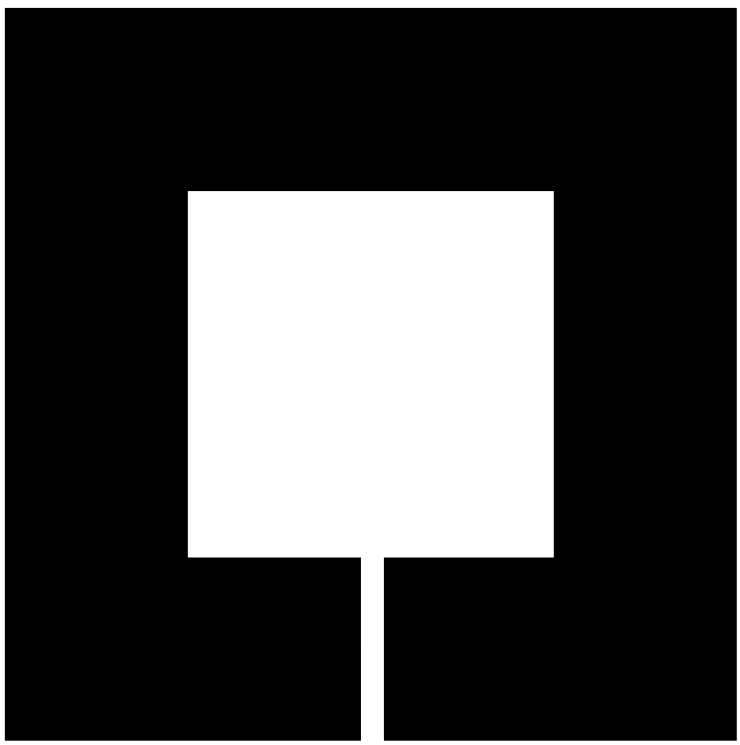
The functions of these signals are listed in the ADSP-2181 documentation with the exception of the signal CODECDIS. This pin may be grounded to disable the on-board AD1847 codec so the serial port can be used for some other purpose.

HARDWARE DEBUGGING

If the green LED fails to light, check your power connections. Verify that your power supply has the proper size connector and that the polarity is correct. The power supply voltage measured at the connector to the board should be in the range of 8 to 10 volts DC. Also, make sure that there are no objects beneath or on top of the board that may be causing a short circuit.

If the power connection is good and the green LED is lit yet the red LED does not flash and no audio signal was produced, make sure that the EPROM is properly seated in the socket.

Hit the reset button if the board appears to be operating improperly.



ADSP-2181
EZ-KIT Lite

Programmer's Quick Reference

Development Software Invocation Commands

Assembler `asm21 sourcefile [-switch …] -2181`

Switches

`-c`	Case-sensitivity for program symbols
`-l`	.LST list file generated
`-m [depth]`	Macros expanded in .LST file
`-i [depth]`	Show contents of INCLUDE files in .LST file
`-o filename`	Rename output files *(default: SOURCEFILE.OBJ)*
`-dident[=literal]`	Define identifier for assembler's C preprocessor
`-s`	No semantics checking on multifunction instructions
`-2181`	Use ADSP-2181 specific instructions

Linker `ld21 file1 [file2 …] [-switch …]`
or `ld21 -i file_all [-switch …]`

Switches

`-a archfile`	.ACH architecture file read by linker *(must specify ADSP2181.ACH)*
`-i file_all`	Files listed in indirect file *file_all* are linked
`-e executable`	Output files given filename *EXECUTABLE.EXE (default: 210x.EXE)*
`-dryrun`	Quick run to test for link errors *(no .EXE file generated)*
`-g`	.SYM symbol table file generated
`-x`	.MAP memory map file generated
`-user fastlibr`	Search library file generated by librarian
`-dir directory ;`	Specify directories to search for library routines
`-p`	Assign library routines to boot pages where called

Simulators `sim2181 [-switch …]`

Switches

`-a archfile`	.ACH architecture file read by simulator *(must specify ADSP2181.ACH)*
`-e exe_file`	.EXE program file loaded by simulator
`-c`	Case-sensitivity for program symbols
`-k keyfile`	Load and execute keystroke file
`-o msgfile`	Generate file containing error messages and Command Output Window messages
`-w winfile`	Simulator starts up with previously saved windows display file *(.WIN)*

PROM Splitter `spl21 exe_file promfile -pm [-switch …]`
or `spl21 exe_file promfile -dm [-switch …]`
or `spl21 exe_file promfile -loader -2181 [-s] [-i]`

Switches

`-pm`	Extract program memory ROM information
`-dm`	Extract data memory ROM information
`-s`	PROM file format: Motorola S record *(default)*
`-i`	PROM file format: Intel Hex record
`-us`	PROM file format: Motorola S record byte stream
`-us2`	PROM file format: Motorola S2 record byte stream
`-ui`	PROM file format: Intel Hex record byte stream
`-loader -2181`	Generate BDMA bootstrap loader in PROM file

Assembler Directives

```
.MODULE/qualifier/qualifier/ ...  module_name;
.PAGE;
.CONST constant_name=expression;
.VAR/qualifier/qualifier/ ...  buffer_name[length], ... ;
.INIT buffer_name: init_values;
.GLOBAL buffer_name, ... ;
.ENTRY program_label, ... ;
.EXTERNAL external_symbol, ... ;
.PORT port_name;
.INCLUDE <filename>;
.DMSEG dmseg_name;
.MACRO macro_name(param1,param2, ... );
.ENDMACRO;
.LOCAL macro_label, ... ;
.NEWPAGE;              Insert pagebreak in .LST file
.PAGELENGTH #lines;    Insert pagebreaks every #lines in .LST file
.LEFTMARGIN #columns;  Set left margin in .LST file
.INDENT #columns;      Indent code #columns in .LST file
.PAGEWIDTH #columns;   Set right margin in .LST file
.ENDMOD;
```

.MODULE qualifiers: RAM, ROM
ABS=*address (do not use with STATIC)*
SEG=*seg_name*
STATIC

.VAR qualifiers: PM, DM,
RAM, ROM
ABS=*address (do not use with STATIC)*
SEG=*seg_name*
CIRC
STATIC

.INIT init_values: constant, constant, ...
<filename>
^other_buffer
%other_buffer

Assembler C Preprocessor Directives

```
#define macro_name(param1, ... ) expression   Define a macro
#undef macro_name                             Undefine a macro
#include "filename "                          Include text from another source file

#if expression      Conditionally include and assemble, depending on the value of
                      an expression that evaluates to a constant
#else               Include and assemble if the previous  #if  test failed
#endif

#ifdef macro_name Conditionally include and assemble, if  macro_name  is defined (with #define or –d switch)
#ifndef macro_name      Conditionally include and assemble, if  macro_name  is not defined
#else               Include and assemble if the previous  #ifdef  or  #ifndef  test failed
#endif
```

Invoking The Simulator

```
> SIM2181  [-a filename] [-c] [-e filename] [-k filename]
           [-w filename] [-o filename] [-help]
```

-a *filename*	Specify .ACH architecture file
-c	Make simulator case-sensitive for assembly code symbols
-e *filename*	Load program (.EXE file)
-k *filename*	Load and run keystroke file
-w *filename*	Start simulator with .WIN windows configuration file
-o *filename*	Open ASCII output file to capture simulator error messages and command output window messages
-help	Display invocation syntax and command line switches; simulator is not invoked

example:

```
> sim2181  -a c:\ADI_DSP\21xx\LIB\ADSP2181
```

General-Purpose Registers	***Program Control Registers***
HOLDER1	PC
HOLDER2	CNTR
HOLDER3	CYCLE
HOLDER4	PM_ADDR
	DM_ADDR

Simulator-Maintained Software Registers

Command	***Definition***
`alias 'newname' 'cmd'`	Create command alias
`alias >'filename'`	Save aliases to file
`alias`	List aliases in command output window
`addsymbol+ 'symbol' addr` `{asym+}`	Add program symbol at address
`assemble addr instr` `{a}`	Assemble instruction at address
`break addr`	Set breakpoint
`break addr, n`	Set multi-breakpoint
`break` `{b}`	List all breaks in command output window
`breakchange expr` `{bc}`	Set break change expression
`breakdelete break#` `{bd}`	Delete break
`breakdelete ALL` `{bd}`	Delete all breaks
`breakexpression expr` `{be}`	Set break expression
`breakrange range` `{br}`	Set break address range
`chipreset` `{cr}`	Chip reset (with program boot)
`connect rfs# tfs#`	Connect RFS of SPORT*x* (*rfs#*) toTFS of SPORT*y* (*tfs#*)
`connect rfs# tfs# OPEN`	Disconnect RFS-to-TFS connection
`connect ALL OPEN`	Disconnect all RFS-to-TFS connections
`connect`	List current RFS-to-TFS connections
`delete 'newname'`	Delete alias
`dump range >'filename'` `{d}`	Dump memory to file
`execute instr` `{exe}`	Execute instruction
`find addr,addr expr` `{f}`	Find numeric value (*expr*) in memory range
`fill addr expr`	Set memory location to value (*expr*)
`fill range expr`	Fill memory range with value (*expr*)
`fill startaddr <'filename'`	Fill memory range from file
`go`	Start program execution
`go addr`	Start program execution, stop at *addr*

{g}

Command	***Definition***
`interrupt` *`irq# min`* [*`max`*][`OFFSET` *`#cycles`*]	Generate periodic interrupt signal
`interrupt` *`sig min`* [*`max`*][`OFFSET` *`#cycles`*]	Generate periodic serial port signal
`interrupt FI` *`period`* [`ONCE|RESET`]	Generate periodic Flag In signal
`interrupt ALL`	List settings for all signals
`interrupt ACTIVE`	List settings for all active signals
{i}	
`keyon` *`'keyfile'`*	Start recording keystroke file
{ko}	
`keyoff`	Stop recording keystrokes, close file
{kf}	
`<'keyfile'`	Playback keystroke file
`load` *`'filename'`*	Load program (.EXE file) into memory
{l}	
`loadrom` *`'filename'`*	Load ROM file (.BNM file) into boot memory
{lr}	
`loadsymbols` *`'filename'`*	Read new symbol table (.SYM file) into simulator
{ls}	
`loopback` *`sport#`*	Connect TFS to RFS and DT to DR of serial port
{lb}	
`loopback` *`sport#`* `OPEN`	Disconnect loopback connection
`loopback ALL OPEN`	Disconnect all loopback connections
`loopback`	List current loopback connections
{lb}	
`openport` *`addr`* [*`<'infile'`*][*`>'outfile'`*]	Open memory-mapped I/O port and assign data files
{op}	
`openport` *`addr  <'infile'`* `CIRC`	Open memory-mapped I/O port with circular input data file
{op}	
`openport` *`addr`*	Close memory-mapped I/O port
{op}	
`opensport` *`sport#`* [*`<'infile'`*][*`>'outfile'`*]	Open SPORT and assign data files
`opensport` *`sport# <'infile'`* `CIRC`	Open SPORT with circular input data file

`opensport` *`sport#`*	Close SPORT
`{os}`	
Command	***Definition***
`plot` *`range decimation`*	Plot memory data
`{pl}`	
`profileadd` *`range# addr,addr`*	Define (or redefine) profile address range
`{pa}`	
`profileclear`	Clear all profile data, delete all address ranges
`{pc}`	
`profiledelete` *`range#`*	Delete profile address range
`{pd}`	
`profilereset`	Clear all profile data, retain address ranges
`{pr}`	
`resetnoboot`	Reset chip, without program boot
`{re}`	
`shell`	Temporarily exit to operating system (type `"exit"` to return to simulator)
`{sh}`	
`state >'`*`file`*`'`	Save simulator state
`state <'`*`file`*`'`	Restore simulator state
`step`	Single step
`step` *`n`*	Multi-step *n* instructions
`{s}`	
`undefine` *`reg`*	Undefine contents of register
`undefine` *`addr`*	Undefine contents of memory location
`undefine` *`range`*	Undefine contents of memory range
`{u}`	
`watchdelete` *`watch#`*	Delete *watch#*
`{wd}`	
`watchexpression` *`expr`*	Set watch expression
`{we}`	
`watchpoint` *`addr`*	Set watchpoint on data variable or buffer
`watchpoint` *`range`*	Set watch address range
`watchpoint`	List all watches in command output window
`{w}`	
`win >'`*`file`*`'`	Save windows configuration (.WIN file)
`win <'`*`file`*`'`	Restore windows configuration (.WIN file)
`?` *`expr`*	Evaluate general expression
`?` *`reg`*	Evaluate contents of register
`?` *`addr`*	Evaluate contents of memory location
`?` *`symbol`*	Locate program symbol
`?+` *`expr`*	Add expression to expressions window
`?-` *`expr#`*	Delete *expr#* from expressions window
`reg = expr`	Set register equal to value of expression

(ex. AX0=0xFF)

Short form of command: *command* chipreset
{*cmd*} {cr}

To define an address range: *range = startaddr , endaddr*
range = startaddr / #locations

Command Window Commands

Ctrl-D Dump memory to file (*in any memory window*)
Ctrl-F Fill memory (*in any memory window*)
Ctrl-G Go to address (*in any memory window*)
Ctrl-L List open windows and choose one to bring to front
Ctrl-M Load memory (*in any memory window*)
Ctrl-R Resize trace length (*in Trace Window*)
Ctrl-T Toggle numeric format in window (decimal ↔ hexadecimal)
Ctrl-O Toggle tracking of data memory and program memory window

Control Key Sequences

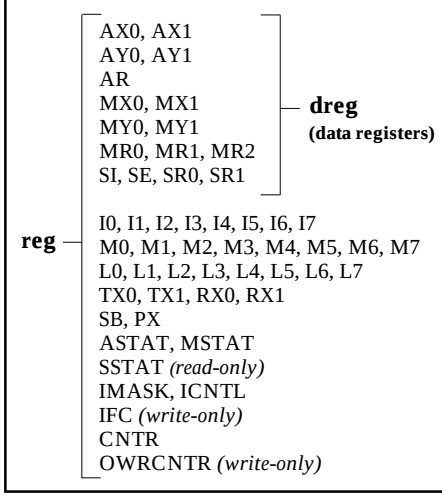
Control Key	^D	^F	^G	^M	^R	^T	^O
Window							
Boot Memory	✔	✔	✔	✔		✔	
Data Memory	✔	✔	✔	✔		✔	✔
Program Memory	✔	✔	✔	✔		✔	✔
Computational Registers						✔	
Control Registers						✔	
DAG Registers						✔	
Program Control Registers						✔	
SPORT Registers						✔	
Command Window						✔	
Expressions						✔	
Profile						✔	
Trace					✔	✔	

Window-to-Control Key Cross Reference

Function Key	*Function*
F1	Help
F2	Goto Next Window
F3	Goto Menu Bar
F4	Run (Go)
F5	Move Window Up ↑
F6	Move Window Down ↓
F7	Move Window Left ←
F8	Move Window Right →
F9	Set / Clear Breakpoint
F10	Single Step
Shift-F5	Enlarge Window Vertically ◊
Shift-F6	Reduce Window Vertically
Shift-F7	Enlarge Window Horizontally ←→
Shift-F8	Reduce Window Horizontally →←
Shift-F9	Set / Clear Multi-Breakpoint
Shift-F10	Multi-Step
Escape	Close window

Keyboard Function Keys (PC only)

Allowed Registers for Data Move & Multifunction Instructions



Circular Buffer Addressing

Next Address = (I + M – B) modulo(L) + B

I=current address M=modify value (signed) M≤L
B=base address L=buffer length

(Set L=0 for standard, non-circular indirect addressing: I+M=modified address)

Buffer Length & Base Address Operators

^`buffer_name` *Base address of* `buffer_name`
%`buffer_name` *Length (number of locations) of* `buffer_name`

Example: Setting Up DAG Registers for Circular Buffer & DO UNTIL Loop

```
      .VAR/DM/RAM/CIRC real_data[n];       {n=number of input samples}
       I5=^real_data;                      {buffer base address}
       L5=%real_data;                      {buffer length}
       M5=1;                               {post-modify I5 by 1}
       CNTR=%real_data;                    {loop counter = buffer length}
       DO loop UNTIL CE;
          AX0=DM(I5,M5);                   {get next sample}
          …
          {now process sample stored in AX0}
loop:     …
```

Instruction Set Summary

Notation Conventions

UPPERCASE	Explicit syntax—must be entered exactly as shown (either lowercase or uppercase may be used, however)
I0–I7	Index registers for indirect addressing
M0–M7	Modify registers for indirect addressing
L0–L7	Length registers for circular buffers (set to 0 for non-circular indirect addressing)
<data>	Immediate data value
<addr>	Immediate address value (absolute address or program label)
<exp>	Exponent (shift value) in shift immediate instructions (8-bit signed number)
cond	Condition code for conditional instruction
term	Termination code for DO UNTIL loop
dreg	Data register (of ALU, MAC, or Shifter)
reg	Any register (including dregs)
;	A semicolon terminates the instruction
,	Commas separate multiple operations of a single instruction
[ ]	Brackets enclose optional parts of instruction
[, ...]	Indicates multiple operations of an instruction that may be combined in any order, separated by commas.

option1 option2 option3	List of options; choose one.

ALU Instructions

[IF cond]	AR AF	=	xop	+	yop C yop + C constant	;	*Add/Add with Carry*
		=	xop	–	yop yop + C – 1 constant	;	*Subtract X–Y/Subtract X–Y with Borrow*
		=	yop	–	xop xop + C – 1 constant	;	*Subtract Y–X/Subtract Y–X with Borrow*

Permissible xops
AX0, AX1, AR, MR0, MR1, MR2, SR0, SR1

Permissible yops (base instruction set)
AY0, AY1, AF

Permissible yops and constants (extended instruction set)
AY0, AY1, AF, 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384,
32767, -2, -3, -5, -9, -17, -33, -65, -129, -257, -513, -1025, -2049, -4097, -8193, -16385, -32768

[IF cond]	AR AF	=	xop	AND OR XOR	yop	;	*AND, OR, XOR*

[IF cond] | AR / AF | = PASS | xop / yop / constant | ; *Pass, Clear*

Permissible yops (base instruction set)
AY0, AY1, AF

Permissible yops and constants (extended instruction set)
AY0, AY1, AF, 0, 1, 2, 3, 4, 5, 7, 8, 9, 15, 16, 17, 31, 32, 33, 63, 64, 65, 127, 128, 129, 255, 256, 257, 511, 512, 513, 1023, 1024, 1025, 2047, 2048, 2049, 4095, 4096, 4097, 8191, 8192, 8193, 16383, 16384, 16385, 32766, 32767, -1, -2, -3, -4, -5, -6, -8, -9, -10, -16, -17, -18, -32, -33, -34, -64, -65, -66, -128, -129, -130, -256, -257, -258, -512, -513, -514, -1024, -1025, -1026, -2048, -2049, -2050, -4096, -4097, -4098, -8192, -8193, -8194, -16384, -16385, -16386, -32767, -32768

[IF cond] | AR / AF | = – | xop / yop | ; *Negate*

= NOT | xop / yop / 0 | ; *NOT*

= ABS xop ; *Absolute Value*

= yop + 1 ; *Increment*

= yop – 1 ; *Decrement*

= DIVS yop, xop ; *Divide*
= DIVQ xop ;

= | TSTBIT n of xop / SETBIT n of xop / CLBIT n of xop / TGBIT n of xop | ; *Bit Operations*

Permissible xops
AX0, AX1, AR, MR0, MR1, MR2, SR0, SR1

Permissible n Values (0 = LSB)
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

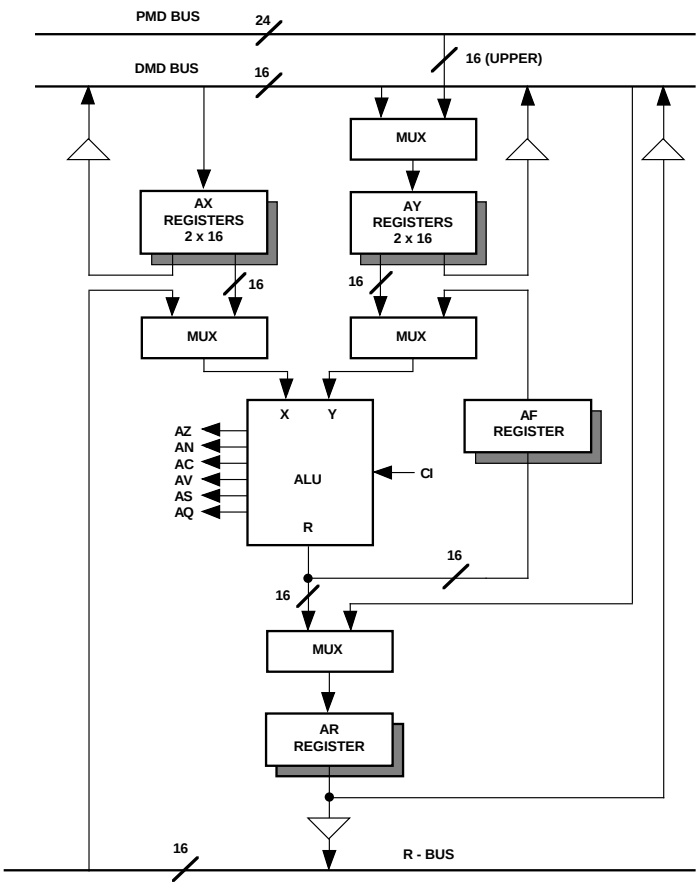
Definitions of Operations
TSTBIT is an AND operation with a 1 in the selected bit
SETBIT is an OR operation with a 1 in the selected bit
CLBIT is an AND operation with a 0 in the selected bit
TGBIT is an XOR operation with a 1 in the selected bit

NONE = <ALU>; *Result Free*

where <ALU> is any unconditional ALU operation of the 21xx base instruction set (except DIVS or DIVQ). (Note that the additional constant ALU operations of the ADSP-2171/2181 extended instruction set are not allowed.)

Description: Perform the designated ALU operation, set the condition flags, then discard the result value. This allows the testing of register values without disturbing the AR or AF register values.

ALU Block Diagram



IF Condition Codes

Cond	
EQ	Equal zero
NE	Not equal zero
LT	Less than zero
GE	Greater than or equal zero
LE	Less than or equal zero
GT	Greater than zero
AC	ALU carry
NOT AC	Not ALU carry
AV	ALU overflow
NOT AV	Not ALU overflow
MV	MAC overflow
NOT MV	Not MAC overflow
NEG	Xop input sign negative
POS	Xop input sign positive
NOT CE	Not counter expired
FLAG_IN *	FI pin=1
NOT FLAG_IN *	FI pin=0

* *Only for JUMP, CALL*

Allowed XOP, YOP Registers for ALU Instructions

xop	AX0, AX1 AR MR0, MR1, MR2 SR0, SR1
yop	AY0*, AY1 AF

* DIVS instruction may not use AY0 as YOP operand.

Instruction Set Summary

MAC Instructions

[IF cond]	MR / MF	=	xop *	yop / xop	(SS) / (SU) / (US) / (UU) / (RND)	;	*Multiply*
		=	MR + xop *	yop / xop	(SS) / (SU) / (US) / (UU) / (RND)	;	*Multiply/Accumulate*
		=	MR – xop *	yop / xop	(SS) / (SU) / (US) / (UU) / (RND)	;	*Multiply/Subtract*
		=	MR [(RND)] ;				*Transfer MR*
		=	0 ;				*Clear*

IF MV SAT MR ; *Conditional MR Saturation*

(S) Signed input (xop, yop)
(U) Unsigned input (xop, yop)
(RND) Rounded output

IF Condition Codes

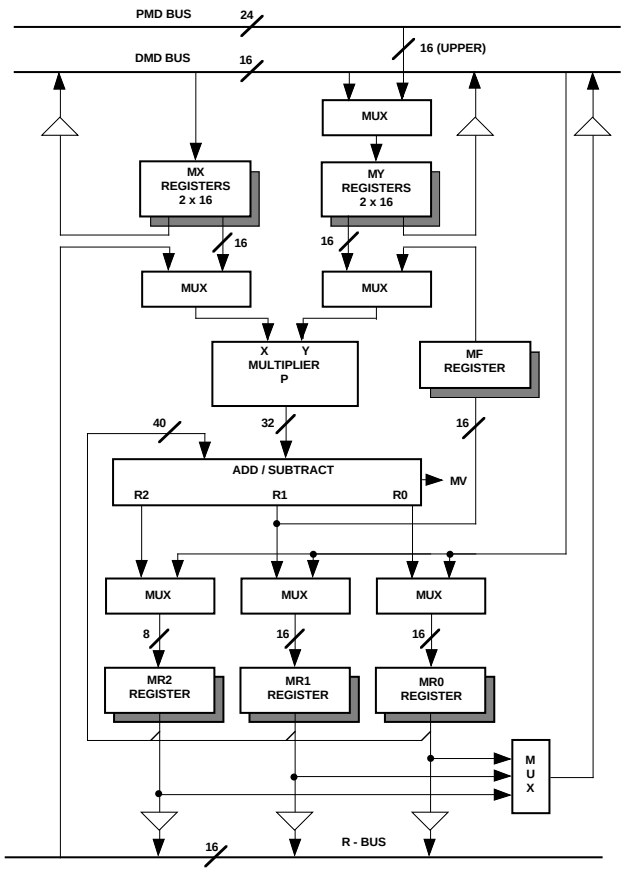
Cond	
EQ	Equal zero
NE	Not equal zero
LT	Less than zero
GE	Greater than or equal zero
LE	Less than or equal zero
GT	Greater than zero
AC	ALU carry
NOT AC	Not ALU carry
AV	ALU overflow
NOT AV	Not ALU overflow
MV	MAC overflow
NOT MV	Not MAC overflow
NEG	Xop input sign negative
POS	Xop input sign positive
NOT CE	Not counter expired
FLAG_IN *	FI pin=1
NOT FLAG_IN *	FI pin=0

** Only for JUMP, CALL*

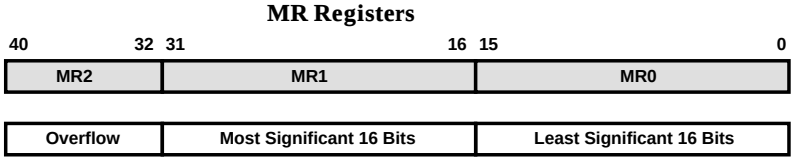
Allowed XOP, YOP Registers for MAC Instructions

xop	MX0, MX1 MR0, MR1, MR2 AR SR0, SR1
yop	MY0, MY1 MF

MAC Block Diagram



MR Registers



Instruction Set Summary

Shifter Instructions

[IF cond] SR = [SR OR] ASHIFT xop (HI)|(LO) ;		*Arithmetic Shift*
[IF cond] SR = [SR OR] LSHIFT xop (HI)|(LO) ;		*Logical Shift*
[IF cond] SR = [SR OR] NORM xop (HI)|(LO) ;		*Normalize*
[IF cond] SE = EXP xop (HI)|(LO)|(HIX) ;		*Derive Exponent*
[IF cond] SB = EXPADJ xop ;		*Block Exponent Adjust*
SR = [SR OR] ASHIFT xop BY <exp> (HI)|(LO) ;		*Arithmetic Shift Immediate*
SR = [SR OR] LSHIFT xop BY <exp> (HI)|(LO) ;		*Logical Shift Immediate*

(HI) Shift is referenced to SR1 (most significant 16 bits)
(LO) Shift is referenced to SR0 (least significant 16 bits)
(HIX) HI extend (AV overflow bit read by exponent detector)

IF Condition Codes

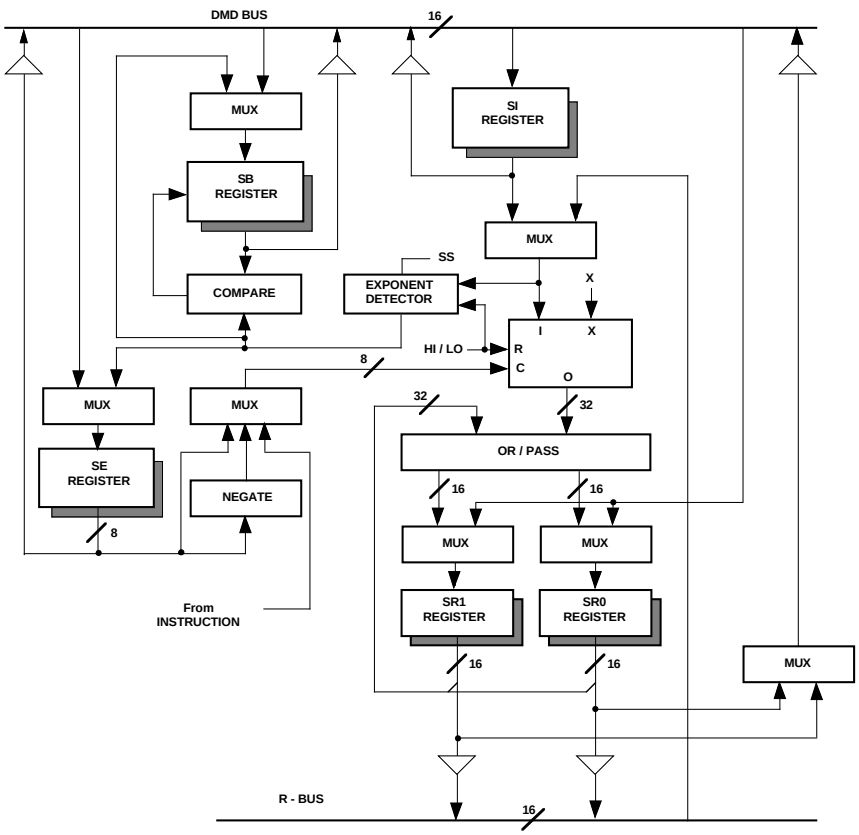
Cond	
EQ	Equal zero
NE	Not equal zero
LT	Less than zero
GE	Greater than or equal zero
LE	Less than or equal zero
GT	Greater than zero
AC	ALU carry
NOT AC	Not ALU carry
AV	ALU overflow
NOT AV	Not ALU overflow
MV	MAC overflow
NOT MV	Not MAC overflow
NEG	Xop input sign negative
POS	Xop input sign positive
NOT CE	Not counter expired
FLAG_IN *	FI pin=1
NOT FLAG_IN *	FI pin=0

** Only for JUMP, CALL*

Allowed XOP Registers for Shifter Instructions

xop	SI, SR0, SR1 AR MR0, MR1, MR2

Shifter Block Diagram



Instruction Set Summary

Data Move Instructions

reg = reg ;	*Register-to-Register Move*
reg = <data> ; dreg = <data> ;	*Load Register Immediate*
dreg = DMOVLAY	*Read Overlay Register*
DMOVLAY = dreg ;	*Write Overlay Register*
reg = DM (<addr>) ;	*Data Memory Read (Direct Address)*
dreg = IO (<addr>); *(See Note 1)*	*I/O Read (Direct Address)*
dreg = DM (I0 , M0) ; I1 M1 I2 M2 I3 M3 I4 M4 I5 M5 I6 M6 I7 M7	*Data Memory Read (Indirect Address)*
dreg = PM (I4 , M4) ; I5 M5 I6 M6 I7 M7	*Program Memory Read (Indirect Address*)
DM (<addr>) = reg ;	*Data Memory Write (Direct Address)*
DM (<addr>) = DMOVLAY ;	*Writes Contents of Overlay Registers to Data Memory*
IO (<addr>) = dreg ; *(See Note 1)*	*I/O Write (Direct Address)*
DM (I0 , M0) = dreg ; I1 M1 I2 M2 I3 M3 I4 M4 I5 M5 I6 M6 I7 M7	*Data Memory Write (Indirect Address)*
PM (I4 , M4) = dreg ; I5 M5 I6 M6 I7 M7	*Program Memory Write (Indirect Address)*

Note 1: <addr> is an address value between 0 and 2048.

Multifunction Instructions

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

Computation with Register-to-Register Move

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

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

Computation with Memory Read

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

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

Computation with Memory Write

```
|AX0 |
|AX1 |
|MX0 | = DM ( |I0| , |M0| )
|MX1 |        |I1|   |M1|
             |I2|   |M2|
             |I3|   |M3|
```

,

```
|AY0 |
|AY1 | = PM ( |I4| , |M4| ) ;
|MY0 |        |I5|   |M5|
|MY1 |        |I6|   |M6|
             |I7|   |M7|
```

Data & Program Memory Read

```
|<ALU>  |
|<MAC>  | ,
```

```
|AX0 |
|AX1 | = DM ( |I0| , |M0| )
|MX0 |        |I1|   |M1|
|MX1 |        |I2|   |M2|
             |I3|   |M3|
```

,

```
|AY0 |
|AY1 | = PM ( |I4| , |M4| ) ;
|MY0 |        |I5|   |M5|
|MY1 |        |I6|   |M6|
             |I7|   |M7|
```

ALU/MAC with Data & Program Memory Read

<ALU>*† Any ALU instruction (except DIVS, DIVQ)
<MAC>*† Any multiply/accumulate instruction
<SHIFT>* Any shifter instruction (except Shift Immediate)

** May not be conditional instructions*
† AR, MR result registers must be used—not AF, MF feedback registers

Instruction Set Summary

Program Flow Instructions

DO <addr> [UNTIL term] ;	*Do Until*
[IF cond] JUMP (I4) | (I5) | (I6) | (I7) | <addr> ;	*Jump*
[IF cond] CALL (I4) | (I5) | (I6) | (I7) | <addr> ;	*Call Subroutine*
IF FLAG_IN | NOT FLAG_IN JUMP | CALL <addr> ;	*Jump/Call on Flag In Pin*
[IF cond] SET | RESET | TOGGLE FLAG_OUT | FL0 | FL1 | FL2 [, ...] ;	*Modify Flag Out Pin*
[IF cond] RTS ;	*Return from Subroutine*
[IF cond] RTI ;	*Return from Interrupt Service Routine*
IDLE [(n)] ;	*Idle*

n=16, 32, 64, or 128

DO UNTIL Termination Codes

Term	
CE	Counter expired
EQ	Equal zero
NE	Not equal zero
LT	Less than zero
GE	Greater than or equal zero
LE	Less than or equal zero
GT	Greater than zero
AC	ALU carry
NOT AC	Not ALU carry
AV	ALU overflow
NOT AV	Not ALU overflow
MV	MAC overflow
NOT MV	Not MAC overflow
NEG	Xop input sign negative
POS	Xop input sign positive
FOREVER	Always

IF Condition Codes

Cond	
EQ	Equal zero
NE	Not equal zero
LT	Less than zero
GE	Greater than or equal zero
LE	Less than or equal zero
GT	Greater than zero
AC	ALU carry
NOT AC	Not ALU carry
AV	ALU overflow
NOT AV	Not ALU overflow
MV	MAC overflow
NOT MV	Not MAC overflow
NEG	Xop input sign negative
POS	Xop input sign positive
NOT CE	Not counter expired
FLAG_IN *	FI pin=1
NOT FLAG_IN *	FI pin=0

** Only for JUMP, CALL*

Miscellaneous Instructions

NOP ; — *NOP*

MODIFY (I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 , M0 | M1 | M2 | M3 | M4 | M5 | M6 | M7) ; — *Modify Address Register*

[PUSH | POP] STS] [, POP CNTR] [, POP PC] [, POP LOOP] ; — *Stack Control*

ENA | DIS SEC_REG | BIT_REV | AV_LATCH | AR_SAT | M_MODE | TIMER | G_MODE | INTS [, ...] ; — *Mode Control*

IDLE ; — *Put Processor In Idle State*

IDLE (n) ; — *Put Processor In Idle State And Slow Clock By a Factor Of n*

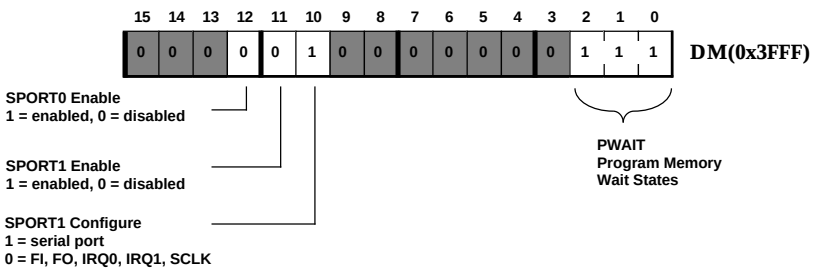
Permissible Values for n: 16, 32, 64, 128

Modes	
SEC_REG	Secondary register set
BIT_REV	Bit-reverse addressing in DAG1
AV_LATCH	ALU overflow (AV) status latch
AR_SAT	AR register saturation
M_MODE	MAC result placement mode
TIMER	Timer enable
G_MODE	Go mode enable
INTS	Interrupt enable

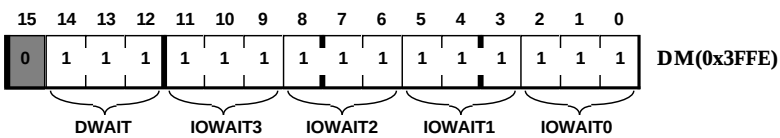
Control/Status Registers

Control register default bit values at reset are as shown; if no value is shown, the bit is undefined after reset. Reserved bits are shown on a gray field—these bits should always be written with zeros.

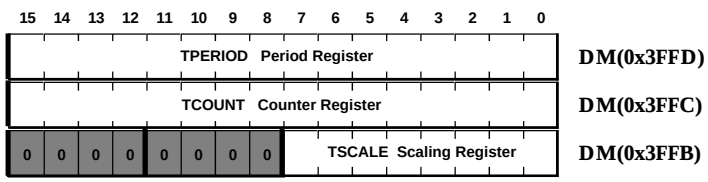
System Control Register



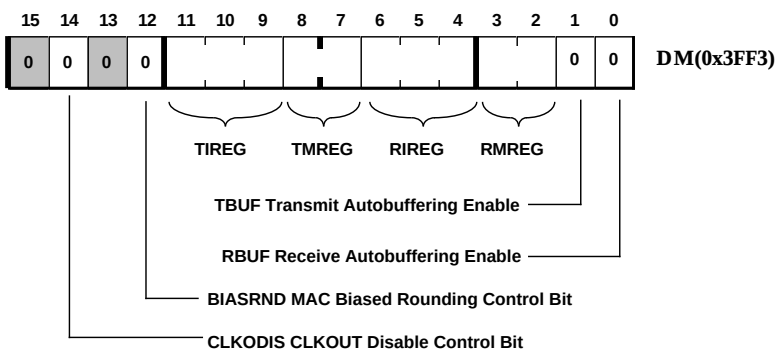
Data Memory Waitstate Register



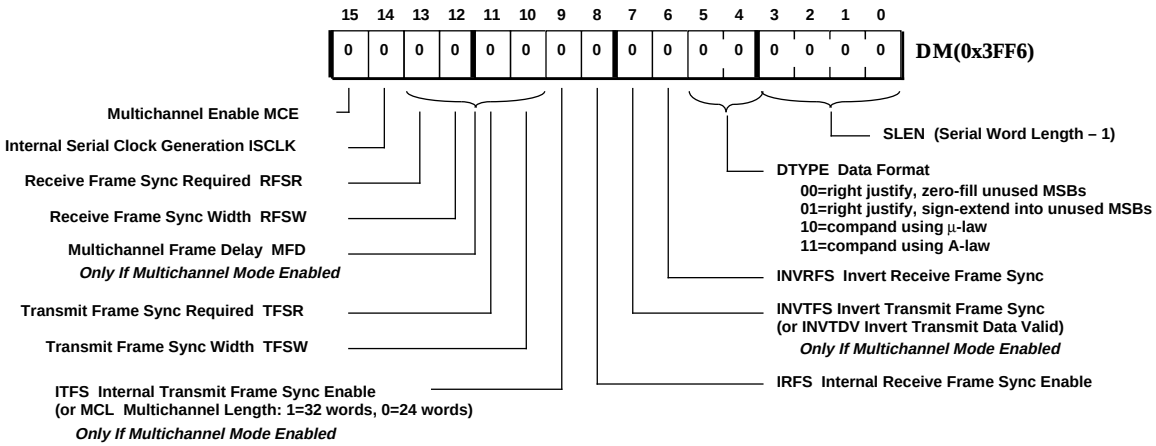
Timer Registers



SPORT0 Autobuffer Control

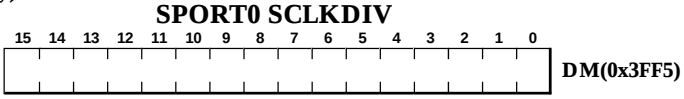


SPORT0 Control Register



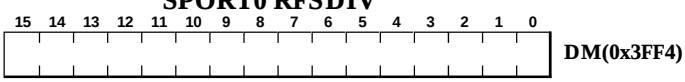
$$\text{SCLKDIV} = \frac{\text{CLKOUT frequency}}{2 * (\text{SCLK frequency})} - 1$$

SPORT0 SCLKDIV

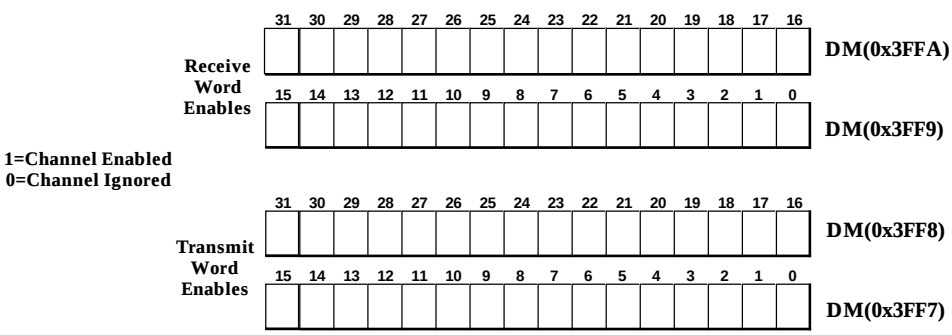


$$\text{RFSDIV} = \frac{\text{SCLK frequency}}{\text{RFS frequency}} - 1$$

SPORT0 RFSDIV

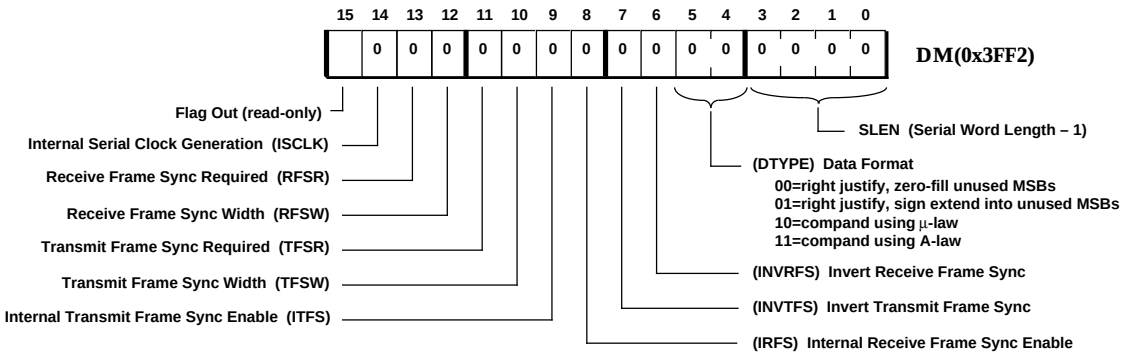


SPORT0 Multichannel Word Enables

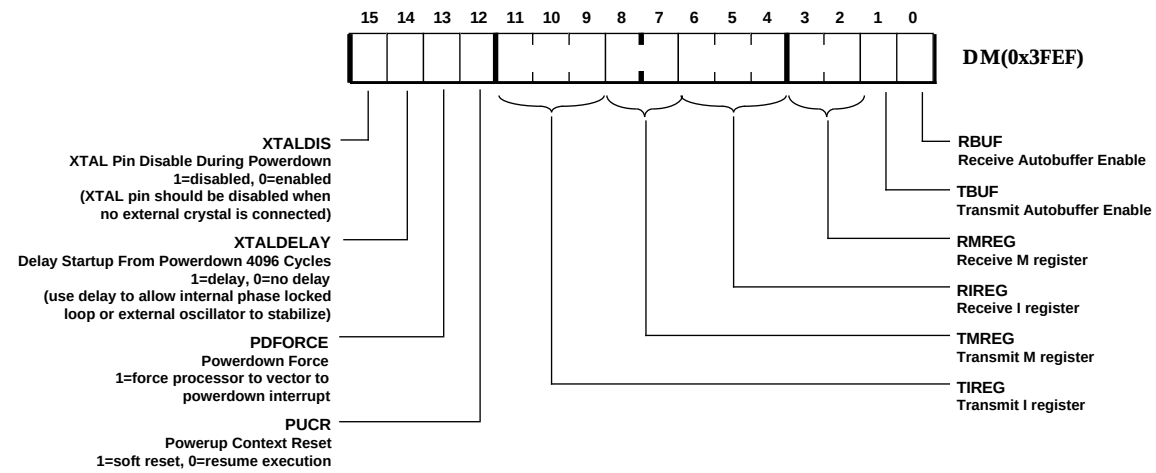


Control/Status Registers

SPORT1 Control Register

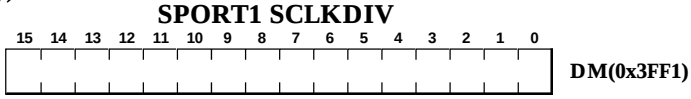


SPORT1 Autobuffer Control



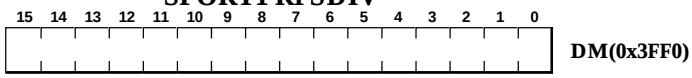
$$\text{SCLKDIV} = \frac{\text{CLKOUT frequency}}{2 * (\text{SCLK frequency})} - 1$$

SPORT1 SCLKDIV

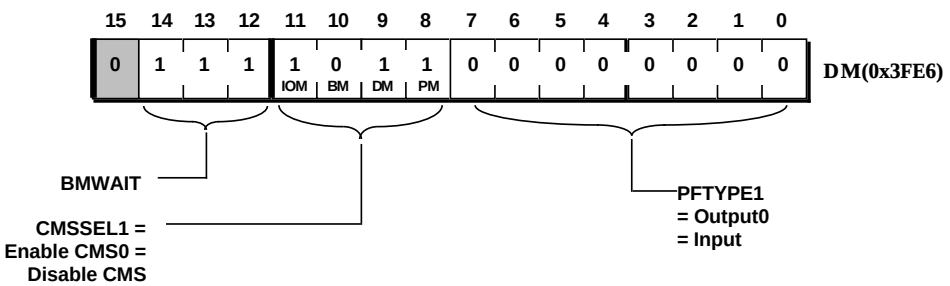


$$\text{RFSDIV} = \frac{\text{SCLK frequency}}{\text{RFS frequency}} - 1$$

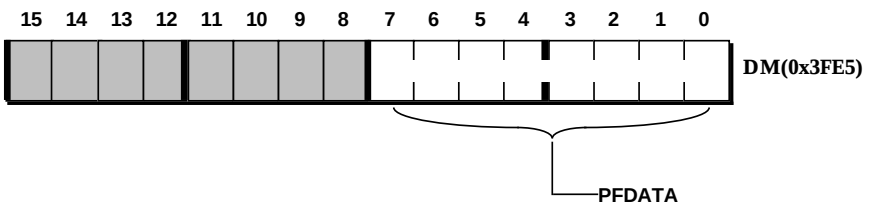
SPORT1 RFSDIV



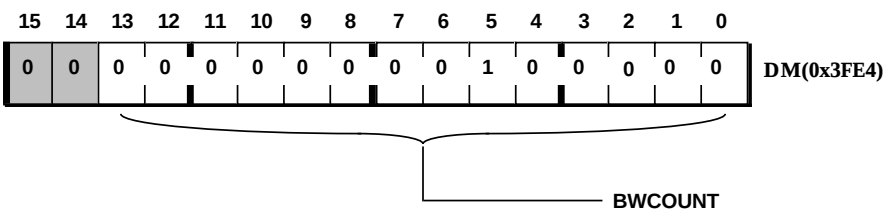
Programmable Flag & Composite Select Control



Programmable Flag Data

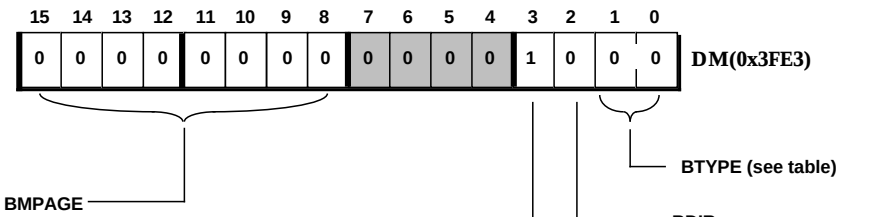


BDMA Word Count



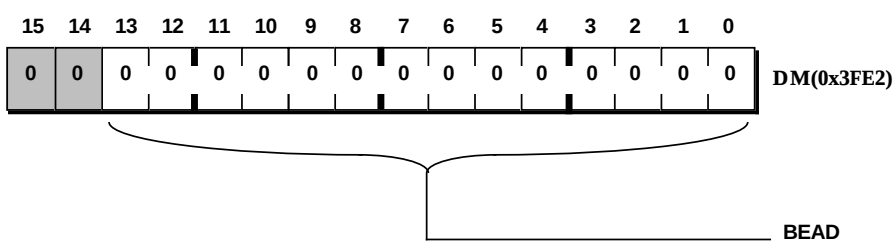
Control/Status Registers

BDMA Control

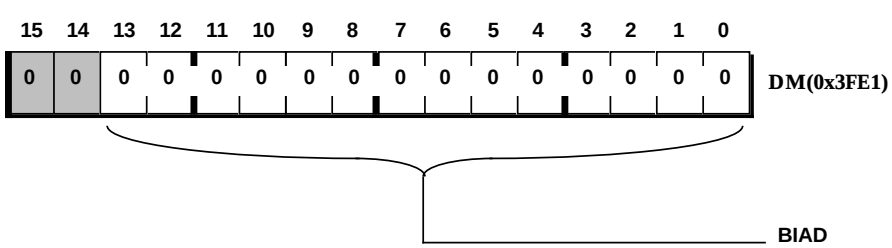


BTYPE	INTERNAL MEMORY SPACE	WORD SIZE	ALIGNMENT
00	PM	24	full word
01	DM	16	full word
10	DM	8	msb
11	DM	8	lsb

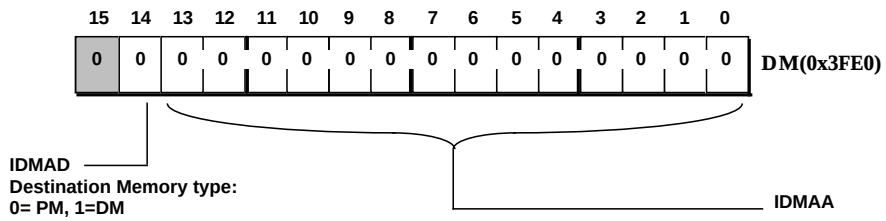
BDMA External Address



BDMA Internal Address



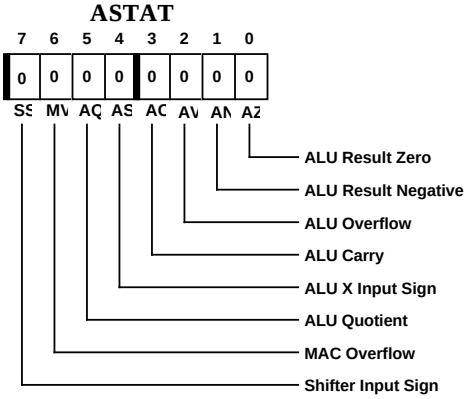
Programmable Flag & Composite Select Control



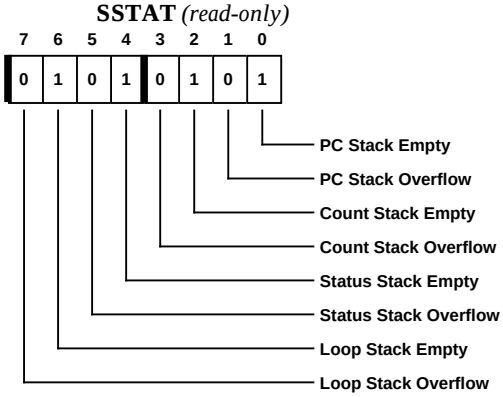
Status Registers

(Non-Memory-Mapped)

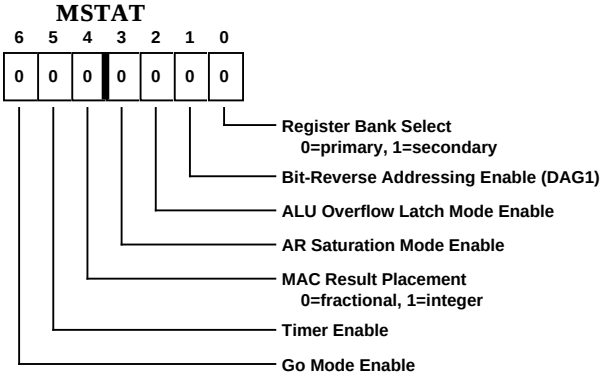
ASTAT



SSTAT *(read-only)*



MSTAT



Mode Names for Mode Control Instruction

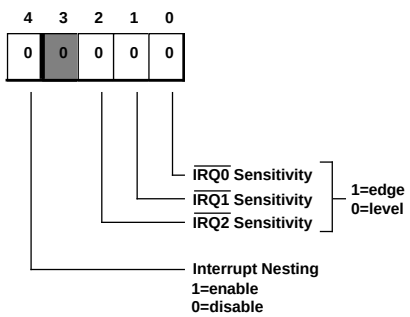
(see Miscellaneous Instructions on page 21)

Bit	*Mode Name*	
0	SEC_REG	Secondary register set
1	BIT_REV	Bit-reverse addressing in DAG1
2	AV_LATCH	ALU overflow (AV) status latch
3	AR_SAT	AR register saturation
4	M_MODE	MAC result placement mode
5	TIMER	Timer enable
6	G_MODE	Go mode enable
7	INTS	Interrupt enable

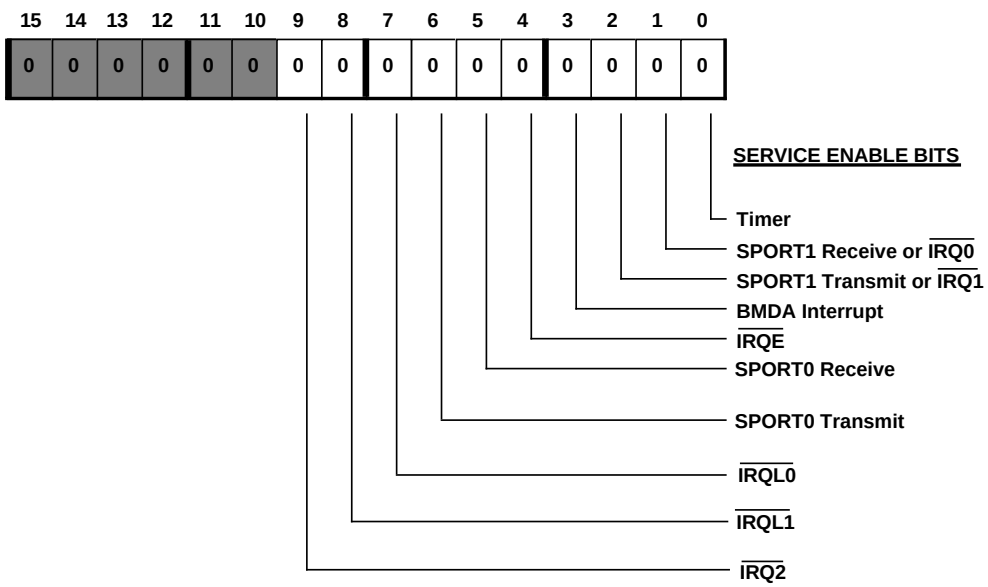
Interrupt Registers

(Non-Memory-Mapped)

ICNTL



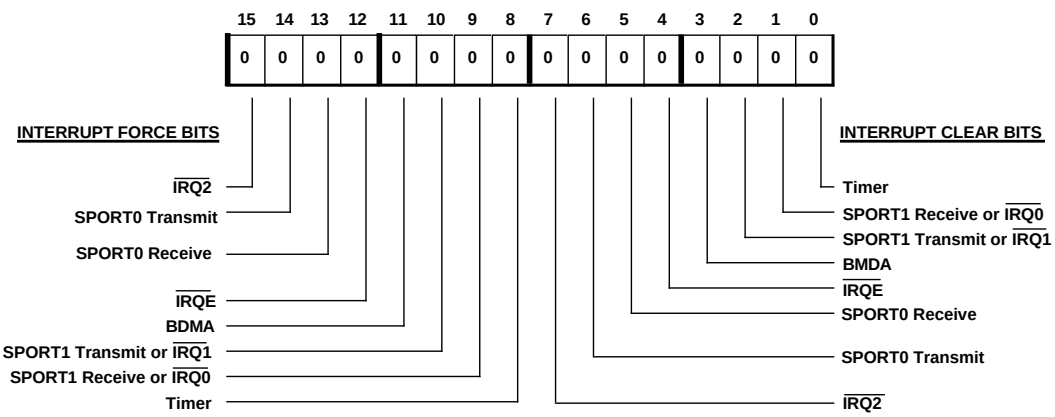
IMASK



Interrupt Registers

(Non-Memory-Mapped)

IFC



Memory Maps

Data Memory

Data Memory	*Address*
32 Memory mapped registers	0x3FFF 0x3FE0
Internal 8160 words	0x3FDF 0x2000
8K Internal (DMOVLAY=0) or External 8K (DMOVLAY=1,2)	0x1FFF 0x0000

Program Memory

Program Memory	*Address*
8K Internal (PMOVLAY = 0, MMAP = 0) or **External 8K** (PMOVLAY = 1 or 2, MMAP = 0)	0x3FFF 0x2000
8K Internal	0x1FFF 0x0000

MMAP = 0

Interrupt Vector Tables

ADSP-2181

Interrupt Source	*Interrupt Vector Address (Hex)*	
Reset (or Power up with PUCR=1)	0000	*(highest priority)*
Power Down (non-maskable)	002C	
IRQ2	0004	
IRQL1	0008	
IRQL0	000C	
SPORT0 Transmit	0010	
SPORT0 Receive	0014	
IRQE	0018	
BDMA Interrupt	001C	
SPORT1 Transmit or IRQ1	0020	
SPORT1 Receive or IRQ0	0024	
Timer	0028	*(lowest priority)*

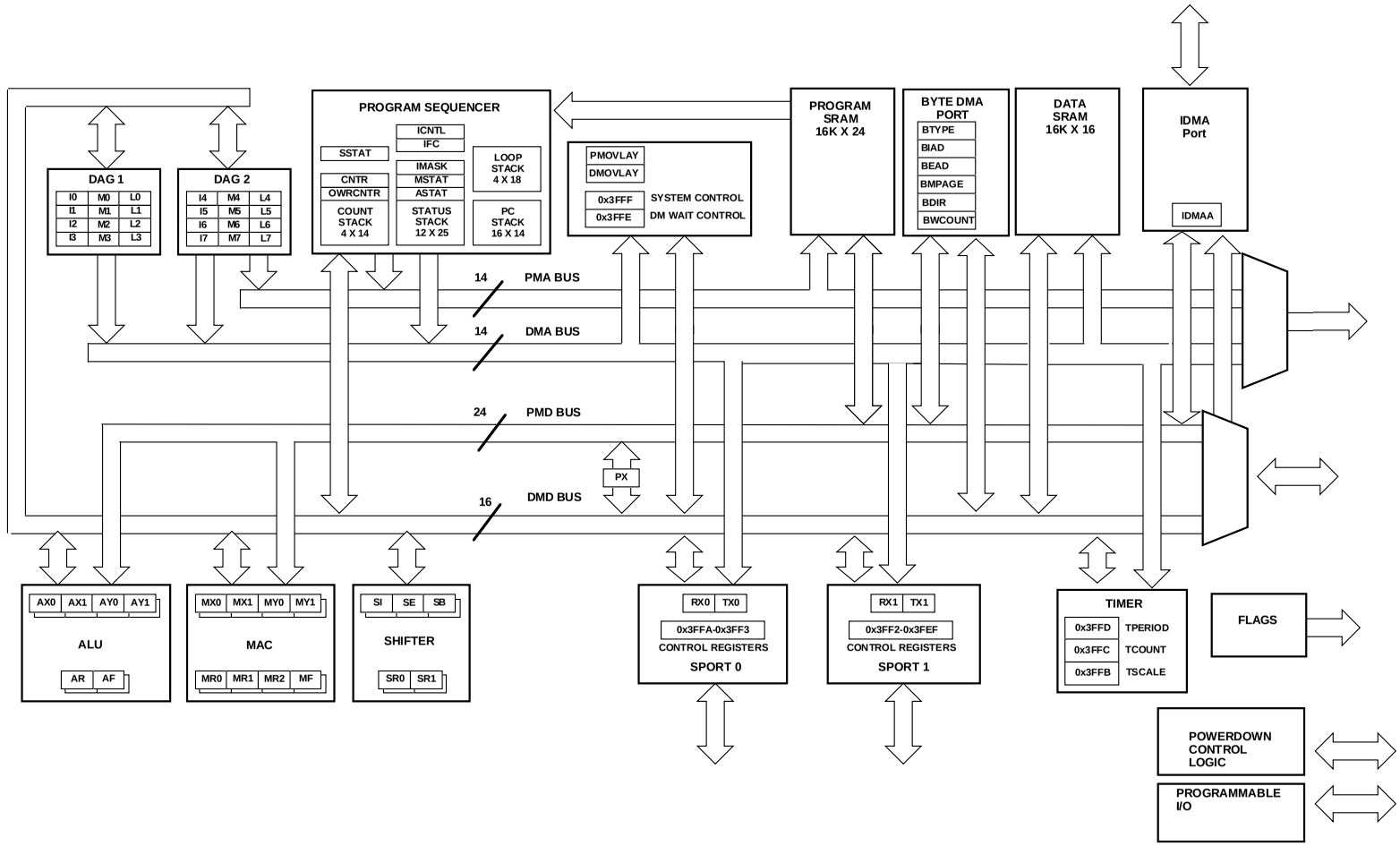
Control/Status Registers

Symbolic names for the memory-mapped control and status registers are provided in four files included with the development software. The symbols are defined as constants equal to the register addresses, and can be used for direct addressing. To use these symbols, include the appropriate file in the your source code files with the assembler's .INCLUDE directive:

Filename	*Include Directive To Use:*
DEF2181.H	`.INCLUDE <DEF2181.H>;`

Control/Status Register	*Data Memory Address*	*Assembly Code Symbol*
System Control Register	0x3FFF	`Sys_Ctrl_Reg`
Data Memory Wait State Control Register	0x3FFE	`Dm_Wait_Reg`
Timer Period	0x3FFD	`Tperiod_Reg`
Timer Count	0x3FFC	`Tcount_Reg`
Timer Scaling Factor	0x3FFB	`Tscale_Reg`
SPORT0 Multichannel Receive	0x3FFA	`Sport0_Rx_Words1`
Word Enable Register (32-bit)	0x3FF9	`Sport0_Rx_Words0`
SPORT0 Multichannel Transmit	0x3FF8	`Sport0_Tx_Words1`
Word Enable Register (32-bit)	0x3FF7	`Sport0_Tx_Words0`
SPORT0 Control Register	0x3FF6	`Sport0_Ctrl_Reg`
SPORT0 Serial Clock Divide Modulus	0x3FF5	`Sport0_Sclkdiv`
SPORT0 Rcv Frame Sync Divide Modulus	0x3FF4	`Sport0_Rfsdiv`
SPORT0 Autobuffer Control Register	0x3FF3	`Sport0_Autobuf_Ctrl`
SPORT1 Control Register	0x3FF2	`Sport1_Ctrl_Reg`
SPORT1 Serial Clock Divide Modulus	0x3FF1	`Sport1_Sclkdiv`
SPORT1 Rcv Frame Sync Divide Modulus	0x3FF0	`Sport1_Rfsdiv`
SPORT1 Autobuffer Control Register	0x3FEF	`Sport1_Autobuf_Ctrl`
Programmable Flag & Composite Select	0x3FE6	`Prog_Flag_Comp_Sel_Ctrl`
Programmable Flag Data	0x3FE5	`Prog_Flag_Data`
BDMA Word Count	0x3FE4	`BDMA_Word_Count`
BDMA Control	0x3FE3	`BDMA_Control`
BDMA External Address	0x3FE2	`BDMA_External_Adress`
BDMA Internal Address	0x3FE1	`BDMA_Internal_Address`
IDMA Control	0x3FE0	`IDMA_Control`

ADSP-2181 Registers



Serial-Port 16-Bit SoundPort Stereo Codec

AD1847

FEATURES

Single-Chip Integrated ΣΔ Digital Audio Stereo Codec
Supports the Microsoft Windows Sound System*
Multiple Channels of Stereo Input
Analog and Digital Signal Mixing
Programmable Gain and Attenuation
On-Chip Signal Filters
Digital Interpolation and Decimation
Analog Output Low-Pass
Sample Rates from 5.5 kHz to 48 kHz
44-Lead PLCC and TQFP Packages
Operation from +5 V Supplies
Serial Digital Interface Compatible with ADSP-21xx Fixed-Point DSP

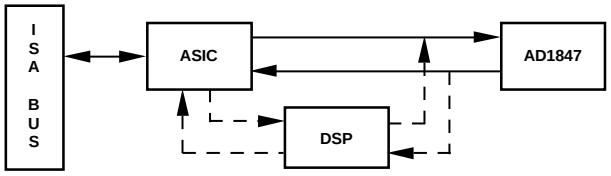


Figure 1. Example System Diagram

PRODUCT OVERVIEW

The AD1847 SoundPort® Stereo Codec integrates key audio data conversion and control functions into a single integrated circuit. The AD1847 is intended to provide a complete, low cost, single-chip solution for business, game audio and multimedia applications requiring operation from a single +5 V supply. It provides a serial interface for implementation on a computer motherboard, add-in or PCMCIA card. See Figure 1 for an example system diagram.

External circuit requirements are limited to a minimal number of low cost support components. Anti-imaging DAC output filters are incorporated on-chip. Dynamic range exceeds 70 dB over the 20 kHz audio band. Sample rates from 5.5 kHz to 48 kHz are supported from external crystals.

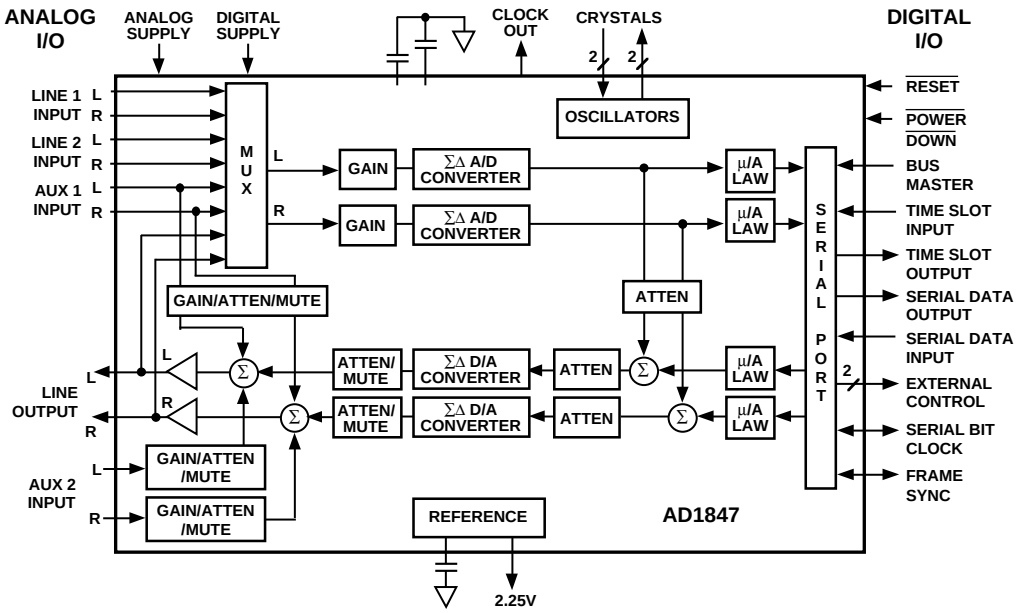
The Codec includes a stereo pair of ∑Δ analog-to-digital converters (ADCs) and a stereo pair of ∑Δ digital-to-analog converters (DACs). Inputs to the ADC can be selected from four stereo pairs of analog signals: line 1, line 2, auxiliary ("aux") line #1, and post-mixed DAC output. A software-controlled programmable gain stage allows independent gain for each channel going into the ADC. The ADCs' output can be digitally mixed with the DACs' input.

The pair of 16-bit outputs from the ADCs is available over a serial interface that also supports 16-bit digital input to the DACs and control/status information. The AD1847 can accept and generate 16-bit twos-complement PCM linear digital data, 8-bit unsigned magnitude PCM linear data, and 8-bit μ-law or A-law companded digital data.

(Continued on page 7)

*Windows Sound System is a registered trademark of Microsoft Corp.
SoundPort is a registered trademark of Analog Devices, Inc.

FUNCTIONAL BLOCK DIAGRAM



REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 617/329-4700 **Fax: 617/326-8703**

AD1847–SPECIFICATIONS

STANDARD TEST CONDITIONS UNLESS OTHERWISE NOTED

Temperature	25	°C	*DAC Output Conditions*
Digital Supply (V_{DD})	5.0	V	0 dB Attenuation
Analog Supply (V_{CC})	5.0	V	Full-Scale Digital Inputs
Word Rate (F_S)	48	kHz	16-Bit Linear Mode
Input Signal	1007	Hz	No Output Load
Analog Output Passband	20	Hz to 20 kHz	Mute Off
FFT Size	4096		*ADC Input Conditions*
V_{IH}	2.4	V	0 dB Gain
V_{IL}	0.8	V	–3.0 dB Relative to Full Scale
V_{OH}	2.4	V	Line Input
V_{OL}	0.4	V	16-Bit Linear Mode

ANALOG INPUT

	Min	Typ	Max	Units
Full-Scale Input Voltage (RMS Values Assume Sine Wave Input)				
Line1, Line2, AUX1, AUX2		1		V rms
	2.54	2.8	3.10	V p-p
Input Impedance				
Line1, Line2, AUX1, AUX2†	10			kΩ
Input Capacitance†		15		pF

PROGRAMMABLE GAIN AMPLIFIER—ADC

	Min	Typ	Max	Units
Step Size (All Steps Tested, –30 dB Input)	1.10	1.5	1.90	dB
PGA Gain Range Span†	21.0		24.0	dB

AUXILIARY INPUT ANALOG AMPLIFIERS/ATTENUATORS

	Min	Typ	Max	Units
Step Size (+12 dB to –28.5 dB, Referenced to DAC Full Scale)	1.3	1.5	1.7	dB
(–30 dB to –34.5 dB, Referenced to DAC Full Scale)	1.1	1.5	1.9	dB
Input Gain/Attenuation Range Span†	45.5		47.5	dB
AUX Input Impedance†	10			kΩ

DIGITAL DECIMATION AND INTERPOLATION FILTERS†

	Min	Max	Units
Passband	0	$0.4 \times F_S$	Hz
Passband Ripple	–0.1	+0.1	dB
Transition Band	$0.4 \times F_S$	$0.6 \times F_S$	Hz
Stopband	$0.6 \times F_S$	∞	Hz
Stopband Rejection	74		dB
Group Delay		$30/F_S$	
Group Delay Variation Over Passband		0	µs

ANALOG-TO-DIGITAL CONVERTERS

	Min	Typ	Max	Units
Resolution		16		Bits
Dynamic Range (–60 dB Input, THD+N Referenced to Full Scale, A-Weighted)	70			dB
THD+N (Referenced to Full Scale)			0.040	%
			–68	dB
Signal-to-Intermodulation Distortion†		83		dB
ADC Crosstalk†				
Line Inputs (Input L, Ground R, Read R; Input R, Ground L, Read L)			–80	dB
Line1 to Line2 (Input Line1, Ground and Select Line2, Read Both Channels)			–80	dB
Line to AUX1			–80	dB
Line to AUX2			–80	dB
Line to DAC			–80	dB
Gain Error (Full-Scale Span Relative to V_{REFI})			±10	%
Interchannel Gain Mismatch (Difference of Gain Errors)			±0.2	dB
DC Offset			±55	LSB

DIGITAL-TO-ANALOG CONVERTERS

	Min	Typ	Max	Units
Resolution		16		Bits
Dynamic Range (–60 dB Input, THD+N Referenced to Full Scale, A-Weighted)	76			dB
THD+N (Referenced to Full Scale)			0.025	%
			–72	dB
Signal-to-Intermodulation Distortion†		86		dB
Gain Error (Full-Scale Span Relative to V_{REFI})			±10	%
Interchannel Gain Mismatch (Difference of Gain Errors)			±0.2	dB
DAC Crosstalk† (Input L, Zero R, Measure R_OUT; Input R, Zero L, Measure L_OUT)			–80	dB
Total Out-of-Band Energy† (Measured from 0.6 × F_S to 100 kHz)			–50	dB
Audible Out-of-Band Energy (Measured from 0.6 × F_S to 22 kHz, Tested at F_S = 5.5 kHz)			–55	dB

DAC ATTENUATOR

	Min	Typ	Max	Units
Step Size (0 dB to –22.5 dB) (Tested at Steps 0 dB, –19.5)	1.3	1.5	1.7	dB
Step Size (–24 dB to –94 dB)	1.0	1.5	2.0	dB
Output Attenuation Range Span†	–93		95	dB

DIGITAL MIX ATTENUATOR

	Min	Typ	Max	Units
Step Size (0 dB to –22.5 dB) (Tested at Steps 0 dB, –19.5)	1.3	1.5	1.7	dB
Step Size (–24 dB to –94 dB)	1.0	1.5	2.0	dB
Output Attenuation Range Span†	–93.5		95.5	dB

ANALOG OUTPUT

	Min	Typ	Max	Units
Full-Scale Line Output Voltage		0.707		V rms
V_{REFI} = 2.35*	1.80	2	2.20	V p-p
Line Output Impedance†			600	Ω
External Load Impedance	10			kΩ
Output Capacitance†			15	pF
External Load Capacitance			100	pF
V_{REF} (Clock Running)	2.00		2.50	V
V_{REF} Current Drive		100		µA
V_{REFI}		2.35		V
Mute Attenuation of 0 dB Fundamental† (LOUT)			–80	dB
Mute Click† (|Muted Output Minus Unmuted Midscale DAC Output|)			8	mV

*Full-scale line output voltage scales with V_{REF} (e.g., V_{OUT} (typ) – 2.0 V × (V_{REF}/2.35)).
†Guaranteed, Not Tested.

SYSTEM SPECIFICATIONS

	Min	Typ	Max	Units
System Frequency Response† (Line In to Line Out, 20 Hz to 20 kHz)		±0.3		dB
Differential Nonlinearity†			±1/2	Bit
Phase Linearity Deviation†		1		Degrees

STATIC DIGITAL SPECIFICATIONS

	Min	Max	Units
High Level Input Voltage (V_{IH})			
Digital Inputs	2.0		V
XTAL1/2I	2.4		V
Low Level Input Voltage (V_{IL})		0.8	V
High Level Output Voltage (V_{OH}) I_{OH} = 1 mA	2.4	V_{DD}	V
Low Level Output Voltage (V_{OL}) I_{OL} = 4 mA		0.4	V
Input Leakage Current (GO/NOGO Tested)	–10	+10	µA
Output Leakage Current (GO/NOGO Tested)	–10	+10	µA

TIMING PARAMETERS (Guaranteed Over Operating Temperature Range)

	Min	Typ	Max	Units
Serial Frame Sync Period (t_1)		1/0.5 F_S		µs
Clock to Frame Sync [SDFS] Propagation Delay (t_{PD1})			20	ns
Data Input Setup Time (t_S)	15			ns
Data Input Hold Time (t_H)	15			ns
Clock to Output Data Valid (t_{DV})			25	ns
Clock to Output Three-State [High-Z] (t_{HZ})			20	ns
Clock to Time Slot Output [TSO] Propagation Delay (t_{PD2})			20	ns
$\overline{\text{RESET}}$ and $\overline{\text{PWRDOWN}}$ Lo Pulse Width (t_{RPWL})	100			ns

POWER SUPPLY

	Min	Max	Units
Power Supply Range – Digital & Analog	4.75	5.25	V
Power Supply Current – Operating (10 kΩ Line Out Load)		140	mA
Analog Supply Current – Operating (10 kΩ Line Out Load)		70	mA
Digital Supply Current – Operating (10 kΩ Line Out Load)		70	mA
Analog Power Supply Current – Power Down		400	µA
Digital Power Supply Current – Power Down		400	µA
Power Dissipation – Operating (Current × Nominal Supply)		750	mW
Power Dissipation – Power Down (Current × Nominal Supply)		4	mW
Power Supply Rejection (@ 1 kHz)†			
(At Both Analog and Digital Supply Pins, ADCs)	45		dB
(At Both Analog and Digital Supply Pins, DACs)	55		dB

CLOCK SPECIFICATIONS†

	Min	Max	Units
Input Clock Frequency		27	MHz
Recommended Clock Duty Cycle		±10	%
Initialization/Sample Rate Change Time			
16.9344 MHz Crystal Selected at Power-Up		171	ms
24.576 MHz Crystal Selected at Power-Up		171	ms
16.9344 MHz Crystal Selected Subsequently		6	ms
24.576 MHz Crystal Selected Subsequently		6	ms

†Guaranteed, not tested.
Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

	Min	Max	Units
Power Supplies			
Digital (V_{DD})	–0.3	6.0	V
Analog (V_{CC})	–0.3	6.0	V
Input Current			
(Except Supply Pins)		±10.0	mA
Analog Input Voltage (Signal Pins)	–0.3	(VA+) + 0.3	V
Digital Input Voltage (Signal Pins)	–0.3	(VD+) + 0.3	V
Ambient Temperature (Operating)	0	+70	°C
Storage Temperature	–65	+150	°C

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

ORDERING GUIDE

Model	Temperature Range	Package Description	Package Option*
AD1847JP	0°C to +70°C	44-Lead PLCC	P-44A
AD1847JST	0°C to +70°C	44-Lead TQFP	ST-44

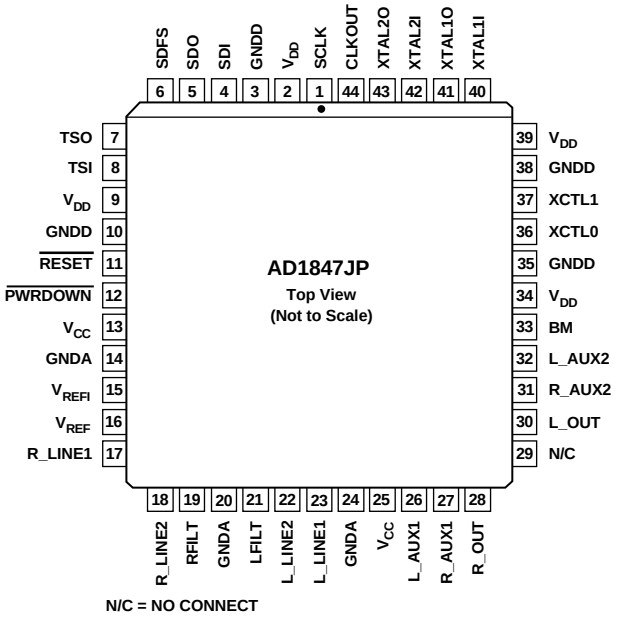
*P = PLCC; ST = TQFP.

CAUTION

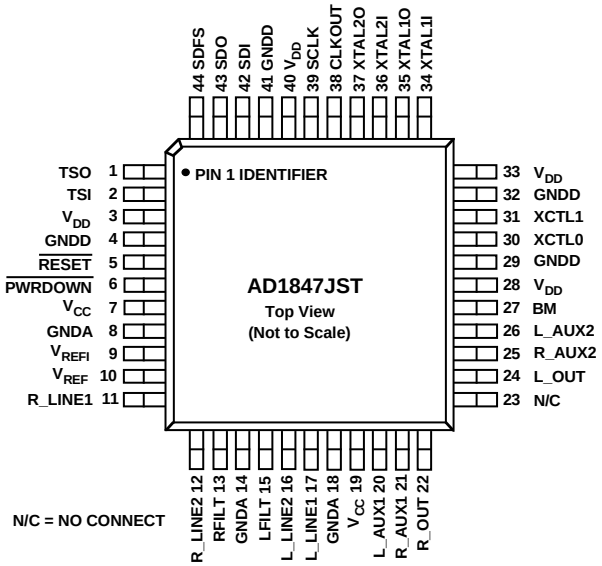
ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD1847 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PINOUTS

44-Lead PLCC



44-Lead TQFP



PIN DESCRIPTIONS

Parallel Interface

Pin Name	PLCC	TQFP	I/O	Description
SCLK	1	39	I/O	Serial Clock. SCLK is a bidirectional signal that supplies the clock as an output to the serial bus when the Bus Master (BM) pin is driven HI and accepts the clock as an input when the BM pin is driven LO. The serial clock output is fixed at 12.288 MHz when XTAL1 is selected, and 11.2896 MHz when XTAL2 is selected. SCLK runs continuously. An AD1847 should always be configured as the serial bus master unless it is a slave in a daisy-chained multiple codec system.
SDFS	6	44	I/O	Serial Data Frame Sync. SDFS is a bidirectional signal that supplies the frame synchronization signal as an output to the serial bus when the Bus Master (BM) pin is driven HI and accepts the frame synchronization signal as an input when the BM pin is driven LO. The SDFS frequency powers up at one half of the AD1847 sample rate (i.e., FRS bit = 0) with two samples per frame and can be programmed to match the sample rate (i.e., FRS bit = 1) with one sample per frame. An AD1847 should always be configured as the serial bus master unless it is a slave in a daisy-chained multiple codec system.
SDI	4	42	I	Serial Data Input. SDI is used by peripheral devices such as the host CPU or a DSP to supply control and playback data information to the AD1847. All control and playback transfers are 16 bits long, MSB first.
SDO	5	43	O	Serial Data Output. SDO is used to supply status/index readback and capture data information to peripheral devices such as the host CPU or a DSP. All status/index readback and capture data transfers are 16 bits long, MSB first. Three-state output driver.
$\overline{\text{RESET}}$	11	5	I	Reset. The $\overline{\text{RESET}}$ signal is active LO. The assertion of this signal will initialize the on-chip registers to their default values. See the "Control Register Definitions" section for a description of the contents of the control registers after $\overline{\text{RESET}}$ is deasserted.
$\overline{\text{PWRDOWN}}$	12	6	I	Powerdown. The $\overline{\text{PWRDOWN}}$ signal is active LO. The assertion of this signal will reset the on-chip control registers (identically to the $\overline{\text{RESET}}$ signal) and will also place the AD1847 in a low power consumption mode. V_{REF} and all analog circuitry are disabled.
BM	33	27	I	Bus Master. The assertion (HI) of this signal indicates that the AD1847 is the serial bus master. The AD1847 will then supply the serial clock (SCLK) and the frame sync (SDFS) signals for the serial bus. One and only one AD1847 should always be configured as the serial bus master. If BM is connected to logic LO, the AD1847 is configured as a bus slave, and will accept the SCLK and SDFS signals as inputs. An AD1847 should only be configured as a serial bus slave when an AD1847 serial bus master already exists, in daisy-chained multiple codec systems.
TSO	7	1	O	Time Slot Output. This signal is asserted HI by the AD1847 coincidentally with the LSB of the last time slot used by the AD1847. Used in daisy-chained multiple codec systems.
TSI	8	2	I	Time Slot Input. The assertion of this signal indicates that the AD1847 should immediately use the next three time slots (TSSEL = 1) or the next six time slots (TSSEL = 0) and then activate the TSO pin to enable the next device down the TDM chain. TSI should be driven LO when the AD1847 is the bus master or in single codec systems. Used in daisy-chained multiple codec systems.
CLKOUT	44	38	O	Clock Output. This signal is the buffered version of the crystal clock output and the frequency is dependent on which crystal is selected. This pin can be three-stated by driving the BM pin LO or by programming the CLKTS bit in the Pin Control Register. See the "Control Registers" section for more details. The CLKOUT frequency is 12.288 MHz when XTAL1 is selected and 16.9344 MHz when XTAL2 is selected.

Analog Signals

Pin Name	PLCC	TQFP	I/O	Description
L_LINE1	23	17	I	Left Line Input #1. Line level input for the #1 left channel.
R_LINE1	17	11	I	Right Line Input #1. Line level input for the #1 right channel.
L_LINE2	22	16	I	Left Line Input #2. Line level input for the #2 left channel.
R_LINE2	18	12	I	Right Line Input #2. Line level input for the #2 right channel.
L_AUX1	26	20	I	Left Auxiliary Input #1. Line level input for the AUX1 left channel.
R_AUX1	27	21	I	Right Auxiliary Input #1. Line level input for the AUX1 right channel.
L_AUX2	32	26	I	Left Auxiliary Input #2. Line level input for the AUX2 left channel.
R_AUX2	31	25	I	Right Auxiliary Input #2. Line level input for the AUX2 right channel.
L_OUT	30	24	O	Left Line Output. Line level output for the left channel.
R_OUT	28	22	O	Right Line Output. Line level output for the right channel.

Miscellaneous

Pin Name	PLCC	TQFP	I/O	Description
XTAL1I	40	34	I	24.576 MHz Crystal #1 Input.
XTAL1O	41	35	O	24.576 MHz Crystal #1 Output.
XTAL2I	42	36	I	16.9344 MHz Crystal #2 Input.
XTAL2O	43	37	O	16.9344 MHz Crystal #2 Output.
XCTL1:O	37 & 36	31 & 30	O	External Control. These TTL signals reflect the current status of register bits inside the AD1847. They can be used for signaling or to control external logic.
V_{REF}	16	10	O	Voltage Reference. Nominal 2.25 volt reference available externally as a voltage datum for dc-coupling and level-shifting. V_{REF} should not have any signal dependent load.
V_{REFI}	15	9	I	Voltage Reference Internal. Voltage reference filter point for external bypassing only.
L_FILT	21	15	I	Left Channel Filter Capacitor. This pin requires a 1.0 µF capacitor to analog ground for proper operation.
R_FILT	19	13	I	Right Channel Filter Capacitor. This pin requires a 1.0 µF capacitor to analog ground for proper operation.
NC	29	23		No Connect. Do not connect.

Power Supplies

Pin Name	PLCC	TQFP	I/O	Description
V_{CC}	13 & 25	7 & 19	I	Analog Supply Voltage (+5 V).
GNDA	14, 20, 24	8, 14, 18	I	Analog Ground.
V_{DD}	2, 9, 34, 39	40, 3, 28, 33	I	Digital Supply Voltage (+5 V).
GNDD	3, 10, 35, 38	41, 4, 29, 32	I	Digital Ground.

(Continued from page 1)

The ∑Δ DACs are preceded by a digital interpolation filter. An attenuator provides independent user volume control over each DAC channel. Nyquist images are removed from the DACs' analog stereo output by on-chip switched-capacitor and continuous-time filters. Two stereo pairs of auxiliary line-level inputs can also be mixed in the analog domain with the DAC output.

The AD1847 serial data interface uses a Time Division Multiplex (TDM) scheme that is compatible with DSP serial ports configured in Multi-Channel Mode with 32 16-bit time slots (i.e., SPORT0 on the ADSP-2101, ADSP-2115, etc.).

AUDIO FUNCTIONAL DESCRIPTION

This section overviews the functionality of the AD1847 and is intended as a general introduction to the capabilities of the device. As much as possible, detailed reference information has been placed in "Control Registers" and other sections. The user is not expected to refer repeatedly to this section.

Analog Inputs

The AD1847 SoundPort Stereo Codec accepts stereo line-level inputs. All inputs should be capacitively coupled (ac-coupled) to the AD1847. LINE1, LINE2, and AUX1, and post-mixed DAC output analog stereo signals are multiplexed to the internal programmable gain amplifier (PGA) stage.

The PGA following the input multiplexer allows independent selectable gains for each channel from 0 to 22.5 dB in +1.5 dB steps. The Codec can operate either in a global stereo mode or in a global mono mode with left-channel inputs appearing at both channel outputs.

Analog Mixing

AUX1 and AUX2 analog stereo signals can be mixed in the analog domain with the DAC output. Each channel of each auxiliary analog input can be independently gained/attenuated from +12 dB to –34.5 dB in –1.5 dB steps or completely muted. The post-mixed DAC output is available on L_OUT and R_OUT externally and as an input to the ADCs.

Even if the AD1847 is not playing back data from its DACs, the analog mix function can still be active.

Analog-to-Digital Datapath

The ∑Δ ADCs incorporate a proprietary fourth-order modulator. A single pole of passive filtering is all that is required for antialiasing the analog input because of the ADC's high 64 times oversampling ratio. The ADCs include digital decimation filters that low-pass filter the input to $0.4 \times F_S$. ("F_S" is the word rate or "sampling frequency.") ADC input overrange conditions will cause status bits to be set that can be read.

Digital-to-Analog Datapath

The ∑Δ DACs contain a programmable attenuator and a low-pass digital interpolation filter. The anti-imaging interpolation filter oversamples and digitally filters the higher frequency images. The attenuator allows independent control of each DAC channel from 0 dB to –94.5 dB in 1.5 dB steps plus full mute. The DACs' ∑Δ noise shapers also oversample and convert the signal to a single-bit stream. The DAC outputs are then filtered in the analog domain by a combination of switched-capacitor and continuous-time filters. These filters remove the very high frequency components of the DAC bitstream output. No external components are required.

Changes in DAC output attenuation take effect only on zero crossings of the digital signal, thereby eliminating "zipper" noise on playback. Each channel has its own independent zero-crossing detector and attenuator change control circuitry. A timer guarantees that requested volume changes will occur even in the absence of an input signal that changes sign. The time-out period is 8 milliseconds at a 48 kHz sampling rate and 48 milliseconds at an 8 kHz sampling rate. (Time-out [ms] ≈ 384/F_S [kHz]).

Digital Mixing

Stereo digital output from the ADCs can be mixed digitally with the input to the DACs. Digital output from the ADCs going out of the serial data port is unaffected by the digital mix. Along the digital mix datapath, the 16-bit linear output from the ADCs is attenuated by an amount specified with control bits. Both channels of the monitor data are attenuated by the same amount. (Note that internally the AD1847 always works with 16-bit PCM linear data, digital mixing included; format conversions take place at the input and output.)

Sixty-four steps of –1.5 dB attenuation are supported to –94.5 dB. The digital mix datapath can also be completely muted, preventing any mixing of the digital input with the digital output. Note that the level of the mixed signal is also a function of the input PGA settings, since they affect the ADCs' output.

The attenuated digital mix data is digitally summed with the DAC input data prior to the DACs' datapath attenuators. The digital sum of digital mix data and DAC input data is clipped at plus or minus full scale and does not wrap around. Because both stereo signals are mixed before the output attenuators, mix data is attenuated a second time by the DACs' datapath attenuators.

Analog Outputs

A stereo line-level output is available at external pins. Other output types such as headphone and speaker must be implemented in external circuitry. The stereo line-level outputs should be capacitively coupled (ac-coupled) to the external circuitry. Each channel of this output can be independently muted. When muted, the outputs will settle to a dc value near V_{REF}, the midscale reference voltage.

Digital Data Types

The AD1847 supports four global data types: 16-bit twos-complement linear PCM, 8-bit unsigned linear PCM, companded μ-law, and 8-bit companded A-law, as specified by control register bits. Eight-bit data is always left-justified in 16-bit fields; in other words, the MSBs of all data types are always aligned; in yet other words, full-scale representations in all four formats correspond to equivalent full-scale signals. The eight least significant bit positions of 8-bit data in 16-bit fields are ignored on digital input and zoned on digital output (i.e., truncated).

The 16-bit PCM data format is capable of representing 96 dB of dynamic range. Eight-bit PCM can represent 48 dB of dynamic range. Companded μ-law and A-law data formats use nonlinear coding with less precision for large-amplitude signals. The loss of precision is compensated for by an increase in dynamic range to 64 dB and 72 dB, respectively.

On input, 8-bit companded data is expanded to an internal linear representation, according to whether μ-law or A-law was specified in the Codec's internal registers. Note that when μ-law compressed data is expanded to a linear format, it requires 14 bits. A-law data expanded requires 13 bits.

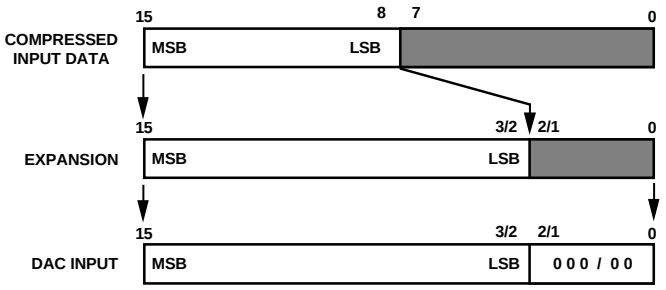


Figure 2. A-Law or μ-Law Expansion

When 8-bit companding is specified, the ADCs' linear output is compressed to the format specified.

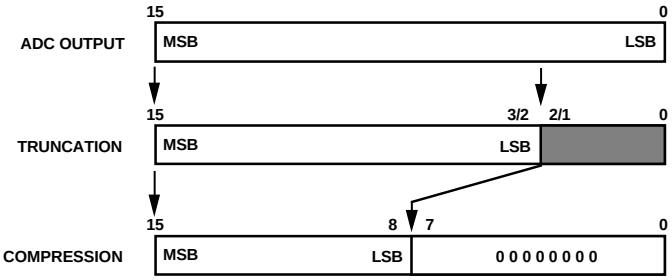


Figure 3. A-Law or μ-Law Compression

Note that all format conversions take place at input or output. Internally, the AD1847 always uses 16-bit linear PCM representations to maintain maximum precision.

Power Supplies and Voltage Reference

The AD1847 operates from +5 V power supplies. Independent analog and digital supplies are recommended for optimal performance though excellent results can be obtained in single-supply systems. A voltage reference is included on the Codec and its 2.25 V buffered output is available on an external pin (V_{REF}). The reference output can be used for biasing op amps used in dc coupling. The internal reference must be externally bypassed to analog ground at the V_{REFI} pin, and must not be used to bias external circuitry.

Clocks and Sample Rates

The AD1847 operates from two external crystals, XTAL1 and XTAL2. The two crystal inputs are provided to generate a wide range of sample rates. The oscillators for these crystals are on the AD1847, as is a multiplexer for selecting between them. They can be overdriven with external clocks by the user, if so desired. At a minimum, XTAL1 must be provided since it is selected as the reset default. If XTAL2 is not used, the XTAL2 input pin should be connected to ground. The recommended crystal frequencies are 16.9344 MHz and 24.576 MHz. From them, the following sample rates can be selected: 5.5125, 6.615, 8, 9.6, 11.025, 16, 18.9, 22.05, 27.42857, 32, 33.075, 37.8, 44.1, 48 kHz.

CONTROL REGISTERS

Control Register Mapping

The AD1847 has six 16-bit and thirteen 8-bit on-chip user-accessible control registers. Control information is sent to the AD1847 in the 16-bit Control Word. Status information is sent from the AD1847 in the 16-bit Status Word. Playback Data and Capture Data each have two 16-bit registers for the right and left channels. Additional 8-bit Index Registers are accessed via indirect addressing in the AD1847 Control Word. [Index Registers are reached with indirect addressing.] The contents of an indirect addressed Index Register may be readback by the host CPU or DSP (during the Status Word/Index Readback time slot) by setting the Read Request (RREQ) bit in the Control Word. Note that each 16-bit register is assigned its own time slot, so that the AD1847 always consumes six 16-bit time slots. Figure 4 shows the mapping of the Control Word, Status Word/ Index Readback and Data registers to time slots when TSSEL = 0. TSSEL = 0 is used when the SDI and SDO pins are tied together (i.e., "1-wire" system). This configuration is efficient in terms of component interconnect (one bidirectional wire for serial data input and output), but inefficient in terms of time slot usage (six slots consumed on single bidirectional Time Division Multiplexed [TDM] serial bus). When TSSEL = 0, serial data input to the AD1847 occurs sequentially with serial data output from the AD1847 (i.e., Control Word, Left Playback and Right Playback data is received on the SDI pin, then the Status Word/ lndex Readback, Left Capture and Right Capture data is transmitted on the SDO pin).

Slot	Register Name (16-Bit)
0	Control Word Input
1	Left Playback Data Input
2	Right Playback Data Input
3	Status Word/Index Readback Output
4	Left Capture Data Output
5	Right Capture Data Output

Figure 4. Control Register Mapping with TSSEL = 0

Figure 5 shows the mapping of the Control Word, Status Word/ Index Readback and Data registers to time slots when TSSEL = 1. Note that the six 16-bit registers "share" three time slots. TSSEL = 1 is used when the SDI and SDO pins are independent inputs and output (i.e., "2-wire" system). This configuration is inefficient in terms of component interconnect (two unidirectional wires for serial data input and output), but efficient in terms of time slot usage (three slots consumed on each of two unidirectional TDM serial buses). When TSSEL = 1, serial data input to the AD1847 occurs concurrently with serial data output from the AD1847 (i.e., Control Word reception on the SDI pin occurs simultaneously with Status Word/lndex Readback transmission on the SDO pin).

Slot	Register Name (16-Bit)
0	Control Word Input
1	Left Playback Data Input
2	Right Playback Data Input
0	Status Word/Index Readback Output
1	Left Capture Data Output
2	Right Capture Data Output

Figure 5. Control Register Mapping with TSSEL = 1

An Index Register readback request to an invalid index address (11, 14 and 15) will return the contents of the Status Word. Attempts to write to an invalid index address (11, 14 and 15) will have no effect on the AD1847. As mentioned above, the RREQ bit of the Control Word is used to request Status Word output or Index Register readback output during either time slot 3 (TSSEL = 0) or time slot 0 (TSSEL = 1). RREQ is set for Index Register readback output, and reset for Status Word output. When Index Register readback is requested, the Index Readback bit format is the same as the Control Word bit format. All status bits are updated by the AD1847 before a new Control Word is received (i.e., at frame boundaries). Thus, if TSSEL = 0 and the Control Word written at slot 0 causes some status bits to change, the change will show up in the Status Word transmitted at slot 3 of the same sample.

Control Word (16-Bit)

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
CLOR	MCE	RREQ	res	IA3	IA2	IA1	IA0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

DATA7:0 Index Register Data. These bits are the data for the desired AD1847 Index Register referenced by the Index Address. Written by the host CPU or DSP to the AD1847.

IA3:0 Index Register Address. These bits define the indirect address of the desired AD1847 Index Register. Written by the host CPU or DSP to the AD1847.

RREQ Read Request. Setting this bit indicates that the current transfer is a request by the host CPU or DSP for readback of the contents of the indirect addressed Index Register. When this bit is set (RREQ = HI), the AD1847 will not transmit its Status Word in the following Status Word Index readback slot, but will instead transmit the data in the Index Register specified by the Index Address. Although the Index Readback is transmitted in the following Status Word/Index Readback time slot, the format of the Control Word is used (i.e., CLOR, MCE, RREQ and the Index Register Address in the most significant byte, and the readback Index Register Data in the least significant byte). When this bit is reset (RREQ = LO), the AD1847 will transmit its Status Word in the following Status Word Index Readback time slot.

A read request is serviced in the next available Index Readback time slot. If TSSEL = 0, the Index Register readback data is transmitted in slot 3 of the same sample. If TSSEL = 1, Index Register readback data is transmitted in slot 0 of the next sample. If TSSEL changes from 0 to 1, Index Register readback will occur twice, in slot 3 of the current sample, and slot 0 of the next. If TSSEL changes from 1 to 0, the last read request is lost.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

MCE Mode Change Enable. This bit must be set (MCE = HI) whenever protected control register bits of the AD1847 are changed. The Data Format register, the Miscellaneous Information register, and the ACAL bit of the Interface Configuration register can NOT be changed unless this bit is set. The DAC outputs will be muted when MCE is set. The user must mute the AUX1 and AUX2 channels when this bit is set (no audio activity should occur). Written by the host CPU or DSP to the AD1847. This bit is HI after reset.

CLOR Clear Overrange. When this bit is set (CLOR = HI), the overrange bits in the Status Word are updated every sample. When this bit is reset (CLOR = LO), the overrange bits in the Status Word will record the largest overrange value. The largest overrange value is sticky until the CLOR bit is set. Written by the host CPU or DSP to the AD1847. Since there can be up to 2 samples in the data pipeline, a change to CLOR may take up to 2 samples periods to take effect. This bit is HI after reset.

Immediately after reset, the contents of this register is: 1100 0000 0000 0000 (C000h).

Left/Right Playback/Capture Data (16-Bit)

The data formats for Left Playback, Right Playback, Left Capture and Right Capture are all identical.

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

DATA15:0 Data Bits. These registers contain the 16-bit, MSB first data for capture and playback. The host CPU or DSP reads the capture data from the AD1847. The host CPU or DSP writes the playback data to the AD1847. For 8-bit linear or 8-bit companded modes, only DATA15:8 contain valid data; DATA7:0 are ignored during capture, and are zeroed during playback. Mono mode plays back the same audio sample on both left and right channels. Mono capture only captures data from the left audio channel. See “Serial Data Format” Timing Diagram.

Immediately after reset, the content of these registers is: 0000 0000 0000 0000 (0000h).

Status Word (16-Bit)

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
res	res	RREQ	res	ID3	ID2	ID1	ID0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
res	res	ORR1	ORR0	ORL1	ORL0	ACI	INIT

INIT Initialization. This bit is an indication to the host that frame syncs will stop and the serial bus will be shut down. INIT is set HI on the last valid frame. It is reset LO for all other frames. Read by the host CPU or DSP from the AD1847.

The INIT bit is set HI on the last sample before the serial interface is inactivated. The only condition under which the INIT bit is set is when a different sample rate is programmed. If FRS = 0 (32 slots per frame, two samples per frame) and the sample rate is changed in the first sample of the 32 slot frame (i.e., during slots 0 through 15), the INIT bit will be set on the second sample of that frame (i.e., during slots 16 through 31). If FRS = 0 and the sample rate is changed in the second sample of the 32 slot frame, the INIT bit will be set on the second sample of the following frame.

ACI Autocalibrate In-Progress. This bit indicates that autocalibration is in progress or the Mode Change Enable (MCE) state has been recently exited. When exiting the MCE state with the ACAL bit set, the ACI bit will be set HI for 384 sample periods. When exiting the MCE state with the ACAL bit reset, the ACAL bit will be set HI for 128 sample periods, indicating that offset and filter values are being restored. Read by the host CPU or DSP from the AD1847.

0 Autocalibration not in progress

1 Autocalibration is in progress

ACI clear (i.e., reset or LO) should be recognized by first polling for a HI on the sample after the MCE bit is reset, and then polling for a LO. Note that it is important not to start polling until one sample after MCE is reset, because if MCE is set while ACI is HI, an ACI LO on the following sample will suggest a false clear of ACI.

ORL1:0 Overrange Left Detect. These bits indicate the overrange on the left input channel. Read by the host CPU or DSP from the AD1847.

0 Greater than –1.0 dB underrange

1 Between –1.0 dB and 0 dB underrange

2 Between 0 dB and 1.0 dB overrange

3 Greater than 1.0 dB overrange

ORR1:0 Overrange Right Detect. These bits indicate the overrange on the right input channel. Read by the host CPU or DSP from the AD1847.

0 Greater than –1.0 dB underrange

1 Between –1.0 dB and 0 dB underrange

2 Between 0 dB and 1.0 dB overrange

3 Greater than 1.0 dB overrange

ID3:0 AD1847 Revision ID. These four bits define the revision level of the AD1847. The first version of the AD1847 is designated ID = 0001. Read by the host CPU or DSP from the AD1847.

RREQ This bit is reset LO for the Status Word, echoing the RREQ state written by the host CPU or DSP in the previous Control Word. Read by the host CPU or DSP from the AD1847.

res Reserved for future expansion. All reserved bits read zero (LO).

Immediately after reset, the contents of this register is: 0000 0001 0000 0000 (0100h).

Index Readback (16-Bit)

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
CLOR	MCE	RREQ	res	IA3	IA2	IA1	IA0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

DATA7:0 Index Register Data. These bits are the readback data from the desired AD1847 Index Register referenced by the Index Address from the previous Control Word (with the RREQ bit set). Read by the host CPU or DSP from the AD1847.

IA3:0 Index Register Address. These bits echo the indirect address (written during the previous Control Word (with the RREQ bit set) of the desired AD1847 Index Register to be readback. Read by the host CPU or DSP from the AD1847.

RREQ Read Request. This bit is set HI for Index Readback, echoing the RREQ state written by the host CPU or DSP in the previous Control Word. Read by the host CPU or DSP from the AD1847.

res Reserved for future expansion. All reserved bits read zero (LO).

MCE Mode Change Enable. This bit echoes the MCE state written by the host CPU or DSP during the previous* Control Word (with the RREQ bit set). Read by the host CPU or DSP from the AD1847.

CLOR Clear Overrange. This bit echoes the CLOR state written by the host CPU or DSP during the previous Control Word (with the RREQ bit set). Read by the host CPU or DSP from the AD1847.

Immediately after reset, the contents of this register is: 1110 0000 0000 0000 (E000h).

Indirect Mapped Registers

Following in Figure 6 is a table defining the mapping of AD1847 8-bit Index Registers to Index Address. These registers are accessed by writing the appropriate 4-bit Index Address in the Control Word.

Index	Register Name
0	Left Input Control
1	Right Input Control
2	Left Aux #1 Input Control
3	Right Aux #l Input Control
4	Left Aux #2 Input Control
5	Right Aux #2 Input Control
6	Left DAC Control
7	Right DAC Control
8	Data Format
9	Interface Configuration
10	Pin Control
11	Invalid Address
12	Miscellaneous Information
13	Digital Mix Control
14	Invalid Address
15	Invalid Address

Figure 6. Index Register Mapping

A detailed description of each of the Index Registers is given below.

Left Input Control Register (Index Address 0)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0000	LSS1	LSS0	res	res	LIG3	LIG2	LIG1	LIG0

LIG3:0 Left Input Gain Select. The least significant bit of this 16-level gain select represents +1.5 dB. Maximum gain is +22.5 dB.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

LSS1:0 Left Input Source Select. These bits select the input source for the left gain stage preceding the left ADC.

0 Left Line 1 Source Selected
1 Left Auxiliary 1 Source Selected
2 Left Line 2 Source Selected
3 Left Line 1 Post-Mixed Output Loopback Source Selected

This register's initial state after reset is: 0000 0000 (00h).

Right Input Control Register (Index Address 1)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0001	RSS1	RSS0	res	res	RIG3	RIG2	RIG1	RIG0

RIG3:0 Right Input Gain Select. The least significant bit of this 16-level gain select represents +1.5 dB. Maximum gain is +22.5 dB.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

RSS1:0 Right Input Source Select. These bits select the input source for the right gain stage preceding the right ADC.

0 Right Line 1 Source Selected
1 Right Auxiliary 1 Source Selected
2 Right Line 2 Source Selected
3 Right Line 1 Post-Mixed Output Loopback Source Selected

This register's initial state after reset is: 0000 0000 (00h).

Left Auxiliary #1 Input Control Register (Index Address 2)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0010	LMX1	res	res	LX1G4	LX1G3	LX1G2	LX1G1	LX1G0

LX1G4:0 Left Auxiliary Input #1 Gain Select. The least significant bit of this 32-level gain/attenuate select represents –1.5 dB. LX1G4:0 = 0 produces a +12 dB gain. LX1G4:0 = "01000" (8 decimal) produces 0 dB gain. Maximum attenuation is –34.5 dB. Gains referred to 2.0 V p-p full-scale output level.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

LMX1 Left Auxiliary #1 Mute. This bit, when set HI, will mute the left channel of the Auxiliary #1 input source. This bit is set HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Right Auxiliary #1 Input Control Register (Index Address 3)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0011	RMX1	res	res	RX1G4	RX1G3	RX1G2	RX1G1	RX1G0

RX1G4:0 Right Auxiliary Input #1 Gain Select. The least significant bit of this 32-level gain/attenuate select represents –1.5 dB. RX1G4:0 = 0 produces a +12 dB gain. RX1G4:0 = "01000" (8 decimal) produces 0 dB gain. Maximum attenuation is –34.5 dB. Gains referred to 2.0 V p-p full-scale output level.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

RMX1 Right Auxiliary #1 Mute. This bit, when set to HI, will mute the right channel of the Auxiliary #1 input source. This bit is set to HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Left Auxiliary #2 Input Control Register (Index Address 4)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0100	LMX2	res	res	LX2G4	LX2G3	LX2G2	LX2G1	LX2G0

LX2G4:0 Left Auxiliary #2 Gain Select. The least significant bit of this 32-level gain/attenuate select represents –1.5 dB. LX2G4:0 = 0 produces a +12 dB gain. LX2G4:0 = "01000" (8 decimal) produces 0 dB gain. Maximum attenuation is –34.5 dB. Gains referred to 2.0 V p-p full-scale output level.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

LMX2 Left Auxiliary #2 Mute. This bit, when set HI, will mute the left channel of the Auxiliary #2 input source. This bit is HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Right Auxiliary #2 Input Control Register (Index Address 5)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0101	RMX2	res	res	RX2G4	RX2G3	RX2G2	RX2G1	RX2G0

RX2G4:0 Right Auxiliary #2 Gain Select. The least significant bit of this 32-level gain/attenuate select represents –1.5 dB. RX2G4:0 = 0 produces a +12 dB gain. RX2G4:0 = "01000" (8 decimal) produces 0 dB gain. Maximum attenuation is –34.5 dB. Gains referred to 2.0 V p-p full-scale output level.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

RMX2 Right Auxiliary #2 Mute. This bit, when set HI, will mute the right channel of the Auxiliary #2 input source. This bit is HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Left DAC Control Register (Index Address 6)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0110	LDM	res	LDA5	LDA4	LDA3	LDA2	LDA1	LDA0

LDA5:0 Left DAC Attenuate Select. The least significant bit of this 64-level attenuate select represents –1.5 dB. LDA5:0 = 0 produces a 0 dB attenuation. Maximum attenuation is –94.5 dB.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

LDM Left DAC Mute. This bit, when set HI, will mute the left channel output. Auxiliary inputs are muted independently with the Left Auxiliary Input Control Registers. This bit is HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Right DAC Control Register (Index Address 7)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0111	RDM	res	RDA5	RDA4	RDA3	RDA2	RDA1	RDA0

RDA5:0 Right DAC Attenuate Select. The least significant bit of this 64-level attenuate select represents –1.5 dB. RDA5:0 = 0 produces a 0 dB attenuation. Maximum attenuation must be at least –94.5 dB.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

RDM Right DAC Mute. This bit, when set HI, will mute the right DAC output. Auxiliary inputs are muted independently with the Right Auxiliary Input Control Registers. This bit is HI after reset.

This register's initial state after reset is: 1000 0000 (80h).

Data Format Register (Index Address 8)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1000	res	FMT	C/L	S/M	CFS2	CFS1	CFS0	CSL

The contents of this register can NOT be changed except when the AD1847 is in the Mode Change Enable (MCE) state (i.e., the MCE bit in the Control Word is HI). Write attempts to this register when the AD1847 is not in the MCE state will not be successful.

CSL Clock Source Select. This bit selects the clock source to be used for the audio sample rate.

0 XTAL1 (24.576 MHz)

1 XTAL2 (16.9344 MHz)

CFS2:0 Clock Frequency Divide Select. These bits select the audio sample rate frequency. The audio sample rate depends on which clock source is selected and the frequency of the clock source.

CFS2:0	Divide Factor	XTAL1 24.576 MHz	XTAL2 16.9344 MHz
0	3072	8.0 kHz	5.5125 kHz
1	1536	16.0 kHz	11.025 kHz
2	896	27.42857 kHz	18.9 kHz
3	768	32.0 kHz	22.05 kHz
4	448	Not Supported	37.8 kHz
5	384	Not Supported	44.1 kHz
6	512	48.0 kHz	33.075 kHz
7	2560	9.6 kHz	6.615 kHz

Note that the AD1847's internal oscillators can be overdriven by external clock sources at the crystal inputs. This is the configuration used by serial bus slave codecs in daisy-chained multiple codec systems. If an external clock source is applied, it will be divided down by the selected Divide Factor. The external clock need not be at the recommended crystal frequencies.

S/M Stereo/Mono Select. This bit determines how the audio data streams are formatted. Selecting stereo will result with alternating samples representing left and right audio channels. Mono playback plays the same audio sample on both channels. Mono capture only captures data from the left audio channel.

0 Mono
1 Stereo

C/L Companded/Linear Select. This bit selects between a linear digital representation of the audio signal or a nonlinear, companded format for all input and output data. The type of linear PCM or the type of companded format is defined by the FMT bits.

0 Linear PCM
1 Companded

FMT Format Select. This bit defines the format for all digital audio input and output based on the state of the C/L bit.

	Linear PCM (C/L = 0)	Companded (C/L = 1)
0	8-bit unsigned linear PCM	8-bit μ-law companded
1	16-bit signed linear PCM	8-bit A-law companded

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

This register's initial state after reset is: 0000 0000 (00h).

Interface Configuration Register (Index Address 9)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1001	res	res	res	res	ACAL	res	res	PEN

PEN Playback Enable. This bit will enable the playback of data in the format selected. PEN may be set and reset without setting the MCE bit.

0 Playback Disabled
1 Playback Enabled

ACAL Autocalibrate Enable. This bit determines whether the AD1847 performs an autocalibrate when exiting from the Mode Change Enable (MCE) state. If the ACAL bit is not set, the previous autocalibration values are used when returning from the Mode Change Enable (MCE) state and no autocalibration takes place. Autocalibration must be preformed after initial power-up for proper operation. This bit is HI after reset.

0 No autocalibration
1 Autocalibration allowed
NOTE: The ACAL bit can only be changed when the AD1847 is in the Mode Change Enable (MCE) state.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

This register's initial state after reset is: 0000 1000 (08h).

Pin Control Register (Index Address 10)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1010	XCTL1	XCTL0	CLKTS	res	res	res	res	res

CLKTS Clock Three-State. If the BM bit is HI, and the CLKTS bit is HI, then the CLKOUT pin will be three-stated. If the BM bit is HI, and the bit CLKTS is LO, then the CLKOUT pin is not three-stated. If the BM bit is LO, then the CLKOUT pin is always three-stated.

XCTL1:0 External Control. The state of these independent bits is reflected on the respective XCTL1 and XCTL0 pins of the AD1847.

0 TTL logic LO on XCTL1, XCTL0 pins

1 TTL logic HI on XCTL1, XCTL0 pins

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

This register's initial state after reset is: 0000 0000 (00h).

Invalid Address (Index Address 11)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1011	inval	inval	inval	inval	inval	inval	inval	inval

inval Writes to this index address are ignored. Index readback of this index address will return the Status Word.

Miscellaneous Information Register (Index Address 12)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1100	FRS	TSSEL	res	res	res	res	res	res

The Miscellaneous Information Register can only be changed when the AD1847 is in the Mode Change Enable (MCE) state. Changes to this register are updated at the next Serial Data Frame Sync (SDFS) boundary. If FRS is LO (i.e., 32 slots per frame), and either TSSEL or FRS change in the first sample of a frame, the change is not updated at the second sample of the same frame, but at the first sample of the next frame.

TSSEL Transmit Slot Select. This bit determines which TDM time slots the AD1847 should transmit on.
- 0 Transmit on time slots 3, 4 and 5. Used when SDI and SDO are tied together (i.e., "1-wire" system).
- 1 Transmit on slots 0, 1 and 2. Used when SDI and SDO are independent inputs and outputs (i.e., "2-wire" system).

FRS Frame Size. This bit selects the number of time slots per frame.
- 0 Selects 32 slots per frame (two samples per frame sync or frame sync at half the sample rate).
- 1 Selects 16 slots per frame (one sample per frame sync or frame sync at the sample rate).

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

This register's initial state after reset is: 0000 0000 (00h).

Digital Mix Control Register (Index Address 13)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1101	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0	res	DME

DME Digital Mix Enable. This bit enables the digital mix of the ADCs' output with the DACs' input. When enabled, the data from the ADCs is digitally mixed with other data being delivered to the DACs (regardless of whether or not playback [PEN] is enabled, i.e., set). If there is a capture overrun, then the last sample captured before overrun will be used for the digital mix. If playback is enabled (PEN set) and there is a playback underrun, then a midscale zero will be added to the digital mix data.
- 0 Digital mix disabled (muted)
- 1 Digital mix enabled

DMA5:0 Digital Mix Attenuation. These bits determine the attenuation of the ADC output data mixed with the DAC input data. The least significant bit of this 64-level attenuate select represents –1.5 dB. Maximum attenuation is –94.5 dB.

res Reserved for future expansion. Write zeros (LO) to all reserved bits.

This register's initial state after reset is: 0000 0000 (00h).

Invalid Address (Index Address 14)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1110	inval	inval	inval	inval	inval	inval	inval	inval

inval Writes to this index address are ignored. Index readback of this index address will return the Status Word.

Invalid Address (Index Address 15)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1111	inval	inval	inval	inval	inval	inval	inval	inval

inval Writes to this index address are ignored. Index readback of this index address will return the Status Word.

Serial Data Interface

The AD1847 serial data interface uses a Time Division Multiplex (TDM) scheme that is compatible with DSP serial ports configured in Multi-Channel Mode with either 32 or 16 16-bit time slots. An AD1847 is always the serial bus master, transmitting the serial clock (SCLK) and the serial data frame sync (SDFS). The AD1847 always receives control and playback data in time slots 0, 1 and 2. The AD1847 will transmit status or index register readback and capture data in time slots 0, 1 and 2 if TSSEL = 1, and will transmit status or index register readback and capture data in time slots 3, 4 and 5 if TSSEL = 0. The following table in Figure 7 shows an example of how the time slots might be assigned.

In this example design, which uses the ADSP-21xx DSP, each frame is divided into 32 time slots of 16-bits each (FRS = 0). Two audio samples are contained in the 32 time slots, with a single frame sync (SDFS) at the beginning of the frame. The ADSP-21xx serial port (SPORT0) supports 32 time slots. The format of the first 16 time slots (sample N) is the same as the format of the second 16 time slots (sample N+1). In this example, 24 time slots are used, as indicated below. Note that time slots 12 through 15 and 28 through 31 are unused in this example, and that Figure 7 presumes that TSSEL = 0 ("1-wire" system).

Slot Number	Source	Destination	Format
0, 16			AD1847 Control Word
1, 17	ASIC	AD1847	Left Playback Data
2, 18			Right Playback Data
3, 19			AD1847 Status Word/ Index Readback
4, 20	AD1847	ASIC	Left Capture Data
5, 21			Right Capture Data
0, 16			AD1847 Control Word
1, 17	DSP	AD1847	Left Playback Data
2, 18			Right Playback Data
3, 19			AD1847 Status Word/ Index Readback
4, 20	AD1847	DSP	Left Capture Data
5, 21			Right Capture Data
6, 22			DSP Control
7, 23	ASIC	DSP	Left Processed Playback Data
8, 24			Right Processed Playback Data
9, 25			DSP Status
10, 26	DSP	ASIC	Left Processed Capture Data
11, 27			Right Processed Capture Data

Figure 7. Time Slot Assignment Example

Note that in this "1-wire" system example, the Digital Signal Processor (DSP) and ISA Bus Interface ASIC (ASIC) use the same slots to communicate to the AD1847. This reduces the number of total time slots required and eliminates the need for the AD1847 to distinguish between DSP data and ASIC data. Also, in this example the ASIC and the DSP do not send data to the AD1847 at the same time, so separate slots are unnecessary.

The digital data in the serial interface is pipelined up to 2 samples deep. This pipelining is required to properly resolve the interface between the relatively fast fixed SCLK rate, and the relatively slow sample rates (and therefore frame sync rates) at which the AD1847 is capable of running. At low sample rates, two samples of data can be serviced in a fraction of a sample period. For example, at an 8 kHz sample rate, 32 time slots only consume $32 \times 16 \times (1/12.288\ \text{MHz}) = 41.67\ \mu\text{s}$ out of a 125 µs period. The two-deep data pipeline thus allows sample overrun (capture) and sample underrun (playback) to be avoided.

Figure 8 represents a logical view of the slot utilization between devices.

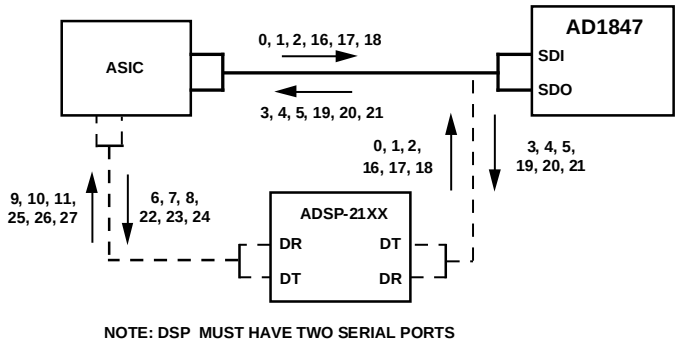


Figure 8. Time Slot Allocation Example

Note that this is a system specific 1-wire example. For non-DSP operation, the DSP is either not present or disabled. If the DSP is present, the ASIC configures the DSP through slot 6 (and slot 22) to three-state its outputs in time slots 0, 1 and 2 (and slots 16, 17 and 18). The ASIC can then enable its drivers for time slots 0, 1 and 2 (and slots 16, 17 and 18). For DSP operation, the ASIC three-states its outputs for time slots 0, 1 and 2 (and slots 16, 17 and 18) and enables the DSP drivers for slots 0, 1 and 2 (and slots 16, 17, and 18).

An application note is available from Analog Devices with additional information on interfacing to the AD1847 serial port. This application note can be obtained through your local Analog Devices representative, or downloaded from the DSP Bulletin Board Service at (617) 461-4258 (8 data bits, no parity, 1 stop bit, 300/1200/2400/4600 baud).

Control Word

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
CLOR	MCE	RREQ	res	IA3	IA2	IA1	IA0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Left Playback Data

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Right Playback Data

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Status Word

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
res	res	RREQ	res	ID3	ID2	ID1	ID0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
res	res	ORR1	ORR0	ORL1	ORL0	ACI	INIT

Index Readback

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
CLOR	MCE	RREQ	res	IA3	IA2	IA1	IA0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Left Capture Data

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA 1	DATA0

Right Capture Data

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0	Index
0000	LSS1	LSS0	res	res	LIG3	LIG2	LIG1	LIG0	0
0001	RSS1	RSS0	res	res	RIG3	RIG2	RIG1	RIG0	1
0010	LMX1	res	res	LX1G4	LX1G3	LX1G2	LX1G1	LX1G0	2
0011	RMX1	res	res	RX1G4	RX1G3	RX1G2	RX1G1	RX1G0	3
0100	LMX2	res	res	LX2G4	LX2G3	LX2G2	LX2G1	LX2G0	4
0101	RMX2	res	res	RX2G4	RX2G3	RX2G2	RX2G1	RX2G0	5
0110	LDM	res	LDA5	LDA4	LDA3	LDA2	LDA1	LDA0	6
0111	RDM	res	RDA5	RDA4	RDA3	RDA2	RDA1	RDA0	7
1000	res	FMT	C/L	S/M	CFS2	CFS1	CFS0	CSL	8
1001	res	res	res	res	ACAL	res	res	PEN	9
1010	XCTL1	XCTL0	CLKTS	res	res	res	res	res	10
1011	inval	inval	inval	inval	inval	inval	inval	inval	11
1100	FRS	TSSEL	res	res	res	res	res	res	12
1101	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0	res	DME	13
1110	inval	inval	inval	inval	inval	inval	inval	inval	14
1111	inval	inval	inval	inval	inval	inval	inval	inval	15

Figure 9. Register Map Summary

Control Register Mapping Summary
A detailed map of the control register bit assignments is summarized for reference in Figure 9.

Daisy-Chained Multiple Codecs
Multiple AD1847s can be configured in a daisy-chain system with a single master Codec and one or more slave Codecs. Codecs in a daisy-chained configuration are synchronized at the sample level.

The master and slave AD1847s should be powered-up together. If this is not possible, the slave(s) should power-up before the master Codec, such that the slave(s) are ready when the master starts to drive the serial interface, and a serial data frame sync (SDFS) can synchronize the master and slave(s).

The sample rate for the master and slave(s) should be programmed together. If this is not possible, the slave(s) should be programmed before the master AD1847. A slave AD1847 enters a time-out period after a new sample rate has been selected. During this time-out period, a slave will ignore any activity on the SDFS signal (i.e., frame syncs). There is no software means to determine when a slave has exited from this time-out period and is ready to respond to frame syncs. However, as long as the AD1847 master is driving the serial interface, a frame sync will not occur before the slave Codec(s) are ready.

Note that the time slots for all slave AD1847s must be assigned to those slots which immediately follow the time slots consumed by the master AD1847 so that the TSO (Time Slot Output)/TSI (Time Slot Input) signaling operates properly. For example, in a 2-wire system with one master and one slave, the time slot assignment should be 0, 1, 2 (16, 17, 18) for the master AD1847, and 3, 4, 5 (19, 20, 21) for the slave AD1847.

Figure 10 illustrates the connection between master and slave(s) in a daisy-chained, multiple Codec system. Note that the TSI pin of the master Codec should be tied to digital ground. The XTAL1I pin of the slaves should be connected to digital ground, and XTAL1O pin should be left unconnected, while the XTAL2I pin should be connected to the CLKOUT pin of the AD1847 master, and the XTAL2O pin generates a driven version of the CLKOUT signal applied to the XTAL2I pin.

INITIALIZATION AND PROCEDURES

Reset and Power Down
A total reset of the AD1847 is defined as any event which requires both the digital and analog section of the AD1847 to return to a known and stable state. Total reset mode, as well as power down, occurs when the $\overline{\text{PWRDOWN}}$ pin of the AD1847 has been asserted low for minimum power consumption. When the $\overline{\text{PWRDOWN}}$ signal is deasserted, the AD1847 must be calibrated by setting the ACAL bit and exiting from the Mode Change Enable (MCE) state.

The reset occurs, and only resets the digital section of the AD1847, when the $\overline{\text{RESET}}$ pin of the AD1847 has been asserted LO to initialize all registers to known values. See the register definitions for the exact values initialized. The register reset defaults include TSSEL = 0 (1-wire system) and FRS = 0 (32 slots per frame). If the target application requires a 2-wire system design or 16 slots per frame, the AD1847 can be bootstrapped into these configurations.

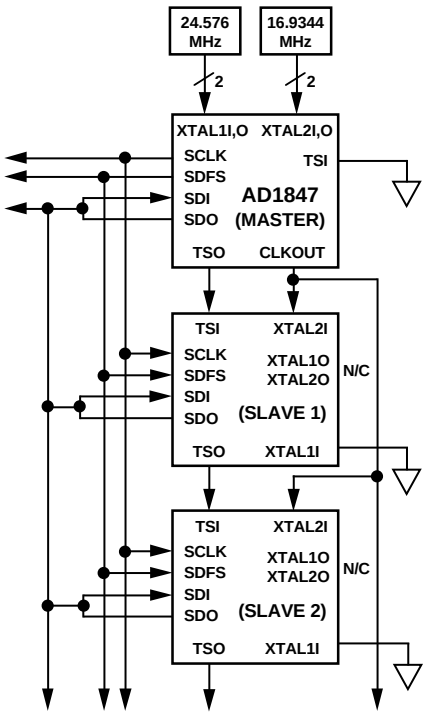


Figure 10a. One-Wire Daisy-Chained Codec Interconnect

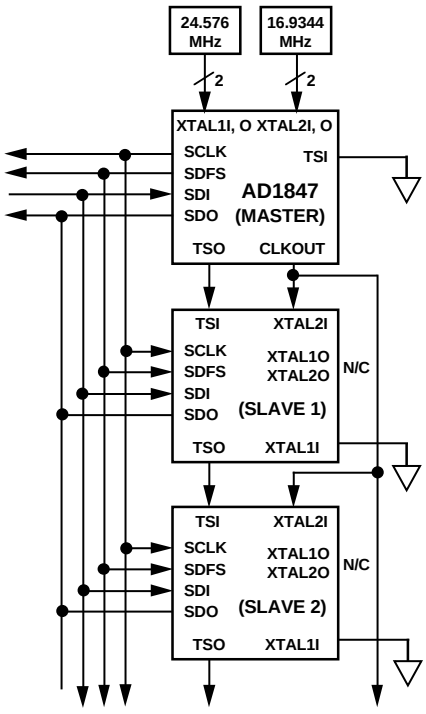


Figure 10b. Two-Wire Daisy-Chained Codec Interconnect

To bootstrap into TSSEL = 1 (i.e., 2-wire system design), the host CPU or DSP must transmit to the AD1847 in slot 0 a Control Word with the MCE bit set HI, IA3:0 = "1100" to address the Miscellaneous Information Index Register, and DATA7:0 = "X100 000" to set the TSSEL bit HI. To bootstrap into FRS = 1 (i.e., 16 slots per frame), the host CPU or DSP must transmit to the AD1847 in slot 0 a Control Word with the MCE bit set HI, IA3:0 = "1100" to address the Miscellaneous Information Index Register, and DATA7:0 = "1X00 0000" to set the FRS bit HI.

The host CPU or DSP must maintain the MCE bit set HI in slot 16, which is the Control Word of the second sample of the frame, so that the AD1847 does not initiate autocalibration prematurely. At the next frame sync, the AD1847 will be reconfigured.

The AD1847 must be reset after power up. When the $\overline{\text{RESET}}$ signal is deasserted, the AD1847 will autocalibrate when the MCE bit is reset LO (i.e., when exiting the Mode Change Enable state) only if the ACAL bit is set. If the ACAL bit is not set, the previous autocalibration values will be used.

The AD1847 will not function properly unless an autocalibration is performed after power up.

During power down, the serial port digital output pins and the analog output pins take the following states:

SCLK–LO if BM is HI (i.e., bus master), input pin if BM is LO (i.e., bus slave)
SDFS–LO if BM is HI, input pin if BM is LO
SDO–three-state
TSO–three-state
CLKOUT–LO if BM HI, three-state if BM is LO
V_{REF}–pulled to analog ground
L_OUT, R_ OUT– pulled to analog ground

Clock Connections and Clock Rates

When the AD1847 is configured as a bus slave (BM = LO), the XTAL1I pin should be connected to digital ground, and the XTAL2I pin should be tied to the CLKOUT of the AD1847 bus master. The XTAL1O and the XTAL2O pins should be left unconnected. When the AD1847 is configured as a bus master (BM = HI), the XTAL1I and the XTAL1O pin should be connected to a 24.576 MHz crystal, and the XTAL2I and XTAL2O pin should be connected to a 16.9344 MHz crystal.

When XTAL1 is selected (by resetting the CSL bit LO in the Data Format Register) as the clock source, the SCLK pin will generated a serial clock at 12.288 MHz (or one half of the crystal frequency applied at XTAL1), and the CLKOUT pin will also generate a clock output at 12.288 MHz when the AD1847 is in bus master mode (BM = HI). When XTAL2 is selected (by setting the CSL bit HI in the Data Format Register) as the clock source, the SCLK pin will generate a serial clock at 11.2896 MHz (or two thirds of the crystal frequency applied at XTAL2), and the CLKOUT pin will generate a clock output at 16.9344 MHz when the AD1847 is in bus master mode (BM = HI). The CLKOUT pin will be three-stated when the AD1847 is placed in bus slave mode (BM = LO).

When the selected frame size is 32 slots per frame (by resetting the FRS bit LO in the Miscellaneous Information Register), the SDFS pin will generate a serial data frame sync at the frequency of the selected sample rate divided by two, when the AD1847 is in bus master mode (BM = HI). When the selected frame size is 16 slots per frame (by setting the FRS bit HI in the Miscellaneous Information Register), the SDFS pin will generate a serial data frame sync at the frequency of the selected sample rate, when the AD1847 is in bus master mode (BM = HI).

When the AD1847 is in bus slave mode (BM = LO), the TSI pin should be connected to the TSO pin of the AD1847 master or slave which has been assigned to the preceding time slots. The signal on the TSO pin is essentially the signal received on the TSI pin, but delayed by 3 or 6 time slots from TSI (depending on the state of TSSEL). The frequency of the transitions on the TSI and TSO lines is equivalent to the frequency on the SDFS pin.

When the AD1847 is in bus master mode (BM = HI), the TSI pin should be connected to digital ground. The signal on the TSO pin is essentially the same as the signal output on the SDFS pin, but delayed by 3 or 6 time slots from SDFS (again, depending on the state of TSSEL).

Mode Change Enable State
The AD1847 must be in the Mode Change Enable (MCE) state before any changes to the ACAL bit of the Interface Configuration Register, the Data Format Register, or the Miscellaneous Information Register are allowed. Note that the MCE bit does not have to be reset LO in order for changes to take effect.

Digital Mix
Digital mix is enabled via the DME bit in the Digital Mix Control Register. The digital mix routes the digital data from the ADCs to the DACs. The mix can be digitally attenuated via bits also in the Digital Mix Control Register. The ADC data is summed with the DAC data supplied at the digital bus interface. When digital mix is enabled and the PEN bit is not set, ADC data is summed with zeros to produce the DAC output.

If the sum of the digital mix (ADC output and DAC input from the serial bus interface) is greater than full scale, the AD1847 will send a positive or negative full scale value to the DACs, whichever is appropriate (clipping).

Autocalibration
The AD1847 has the ability to calibrate its ADCs and DACs for greater accuracy by minimizing dc offsets. Autocalibration occurs whenever the AD1847 exits from the Mode Change Enable (MCE) state AND the ACAL bit in the Interface Configuration Register has been set.

The completion of the autocalibration sequence can be determined by polling the Autocalibration In-Progress (ACI) bit in the Status Word. This bit will be HI while the autocalibration is in progress and LO once autocalibration has completed. The autocalibration sequence will take at least 384 sample periods.

The autocalibration procedure is as follows:

1. Mute both left and right AUX1 and AUX2 inputs via the Left Auxiliary Input and Right Auxiliary Input Control Registers.
2. Place the AD1847 in the Mode Change Enable (MCE) state using the MCE bit of the AD1847 Control Word. Set the ACAL bit in the Interface Configuration Register.
3. Exit from the Mode Change Enable state by resetting the MCE bit.
4. Poll the ACI bit in the AD1847 Status Word for a HI (autocalibration in progress), then poll the ACI bit for a LO (autocalibration complete).
5. Unmute the AUX inputs, if used.

If ACAL is not set, the AD1847 is muted for 128 sample periods after resetting the MCE bit, and the ACI bit in the Status Word is set HI during this 128 sample periods. Autocalibration must be performed after power-up to ensure proper operation of the AD1847.

Exiting from the MCE state always causes ACI to go HI. If the ACAL bit is set when MCE state is exited, then the ACI bit will be HI for 384 sample periods. If the ACAL bit is reset when MCE is exited, then the ACI bit will be HI for 128 sample periods.

Changing Sample Rates
The internal states of the AD1847 are synchronized by the selected sample frequency defined in the Data Format Register. The changing of either the clock source or the clock frequency divide requires a special sequence for proper AD1847 operation.

1. Mute the outputs of the AD1847 and enter the Mode Change Enable (MCE) state by setting the MCE bit of the AD1847 Control Word.
2. During a single atomic or nondivisible write cycle, change the Clock Frequency Divide Select (CFS) and/or the Clock Source Select (CSL) bits of the Data Format Register to the desired values. CFS and CSL can be programmed in the same Control Word as MCE.
3. The INIT bit in the Status Word will be set HI at the last sample of the next frame to indicate that the serial port will be disabled for a timeout period.
4. The AD1847 requires a period of time to resynchronize its internal states to the newly selected clock. During this time, the AD1847 will be unable to respond at its serial interface port (i.e., no frame syncs will be generated). The time-out period is $2^{21} \times$ SCLK $\approx$ 170 ms after power-up, and $\approx$ 5 ms for subsequent changes of sample rate.
5. Exit the Mode Change Enable state by resetting the MCE bit. Upon exiting the MCE state, an autocalibration of duration 384 sample periods or an output mute of duration 128 sample periods occurs, depending on the state of the ACAL bit.
6. Poll the ACI bit in the AD1847 Status Word for a HI (indicating that autocalibration is in progress) then poll the ACI bit for a LO (indicating that autocalibration has completed). Once the ACI bit has been read back LO, normal operation of the Codec can resume.

The CSL and CFS bits cannot be changed unless the AD1847 is in the Mode Change Enable state (i.e., the MCE bit in the AD1847 Control Word is set). Attempts to change the contents of the Data Format Register without MCE set will result in the write cycle not being recognized (the bits will not be updated).

The MCE bit should not be reset until after the INIT bit in the AD1847 Status Word is detected HI. After the INIT bit is detected HI, the serial port is disabled. When the next frame sync arrives (after the time-out period), all internal clocks are stable and the serial port is ready for normal operation.

DATA FORMAT DEFINITIONS

There are four data formats supported by the AD1847: 16-bit signed, 8-bit unsigned, 8-bit companded μ-law, and 8-bit companded A-law. The AD1847 supports these four formats because each of them have found wide use in important applications.

16-Bit Signed Format

The 16-bit signed format (also called 16-bit twos-complement) is the standard method of representing 16-bit digital audio. This format yields 96 dB of dynamic range and is common in consumer compact disk audio players. This format uses the value – 32768 (8000h) to represent minimum analog amplitude while 32767 (7FFFh) represents maximum analog amplitude. Intermediate values are a linear interpolation between minimum and maximum amplitude values.

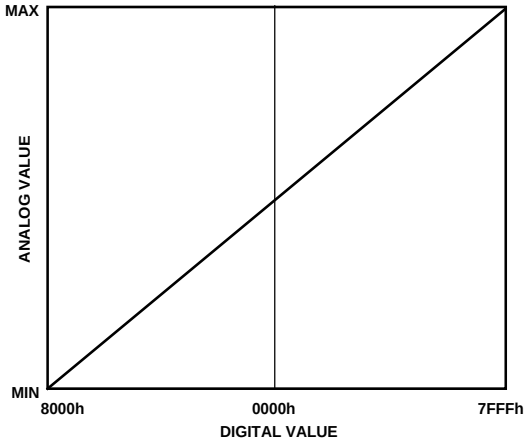


Figure 11. 16-Bit Signed Format

8-Bit Unsigned Format

The 8-bit unsigned format is commonly used in the personal computer industry. This format delivers 48 dB of dynamic range. The value 0 (00h) is used to represent minimum analog amplitude while 255 (FFh) is used to represent maximum analog amplitude. Intermediate values are a linear interpolation between minimum and maximum amplitude values. The least significant byte of the 16-bit internal data is truncated to create the 8-bit output samples.

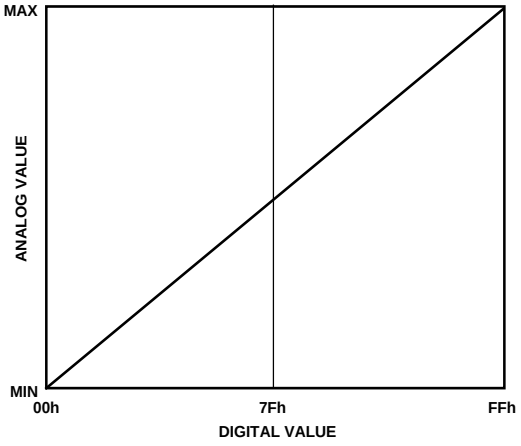


Figure 12. 8-Bit Unsigned Format

8-Bit Companded Formats

The 8-bit companded formats (μ-law and A-law) are used in the telecommunications industry. Both of these formats are used in ISDN communications and workstations; μ-law is the standard for the United States and Japan while A-law is used in Europe. Companded audio allows either 64 dB or 72 dB of dynamic range using only 8-bits per sample. This is accomplished using a nonlinear formula which assigns more digital codes to lower amplitude analog signals at the expense of resolution of higher amplitude signals. The μ-law format of the AD1847 conforms to the Bell System μ = 255 companding law while the A-law format conforms to CCITT "A" law models. Figure 13 shows approximately how both the μ-law and A-law companding schemes behave. Refer to the standards mentioned above for an exact definition.

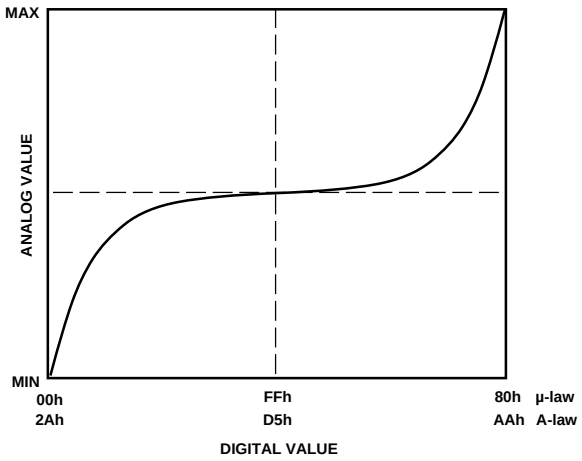


Figure 13. 8-Bit Companded Format

APPLICATIONS CIRCUITS

The AD1847 Stereo Codec has been designed to require a minimum of external circuitry. The recommended circuits are shown in Figures 14 through 22. Analog Devices estimates that the total cost of all the components shown in these Figures, including crystals, to be less than $3 in 10,000 quantities.

Industry-standard compact disc "line-levels" are 2 V_{rms} centered around analog ground. (For other audio equipment, "line level" is much more loosely defined.) The AD1847 SoundPort is a +5 V only powered device. Line level voltage swings for the AD1847 are defined to be 1 V_{rms} for a sine wave ADC input and 0.707 V_{rms} for a sine wave DAC output. Thus, 2 V_{rms} input analog signals must be attenuated and either centered around the reference voltage intermediate between 0 V and +5 V or ac-coupled. The V_{REF} pin will be at this intermediate voltage, nominally 2.25 V. It has limited drive but can be used as a voltage datum to an op amp input. Note, however, that dc-coupled inputs are not recommended, as they provide no performance benefits with the AD1847 architecture. Furthermore, dc offset differences between multiple dc-coupled inputs create the potential for "clicks" when changing the input mux selection.

Circuits for 2 V_{rms} line-level inputs and auxiliaries are shown in Figure 14 and Figure 15. Note that these are divide-by-two resistive dividers. The input resistor and 560 pF (1000 pF) capacitor provide the single-pole of antialias filtering required for the ADCs. If line-level inputs are already at the 1 V_{rms} levels expected by the AD1847, the resistors in parallel with the 560 pF (1000 pF) capacitors can be omitted. If the application does not route the AUX2 inputs to the ADCs, then no antialias filtering is required (only the 1 µF ac coupling capacitor).

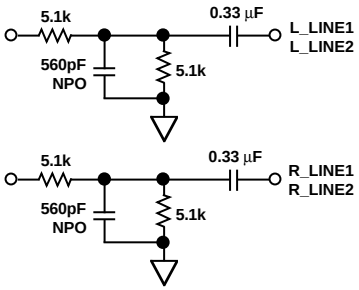


Figure 14. 2 V_{rms} Line-Level Input Circuit for Line Inputs

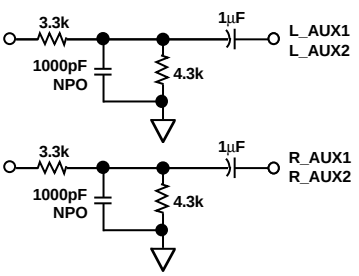


Figure 15. 2 V_{rms} Line-Level Input Circuit for AUX Inputs

Figure 16 illustrates one example of how an electret condenser microphone requiring phantom power could be connected to the AD1847. V_{REF} is shown buffered by an op amp; a transistor like a 2N4124 will also work well for this purpose. Note that if a battery-powered microphone is used, the buffer and R2s are not needed. The values of R1, R2, and C should be chosen in light of the mic characteristics and intended gain. Typical values for these might be R1 = 20 kΩ, R2 = 2 kΩ, and C = 220 pF.

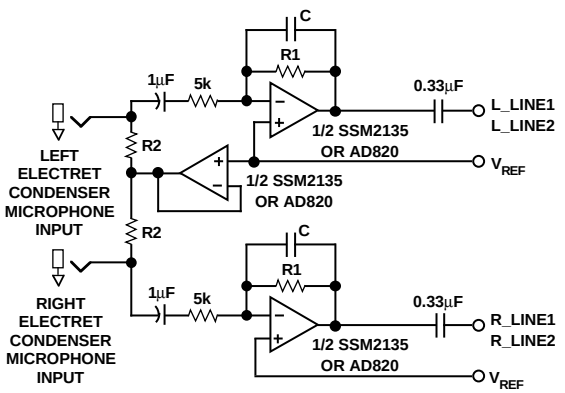


Figure 16. "Phantom-Powered" Microphone Input Circuit

Figure 17 shows ac-coupled line outputs. The resistors are used to center the output signals around analog ground. If dc-coupling is desired, V_{REF} could be used with op amps as mentioned previously.

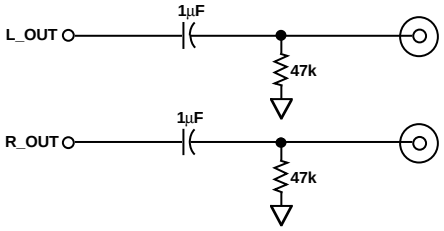


Figure 17. Line Output Connections

A circuit for headphone drive is illustrated in Figure 18. Drive is supplied by +5 V operational amps. The circuit shown ac couples the headphones to the line output.

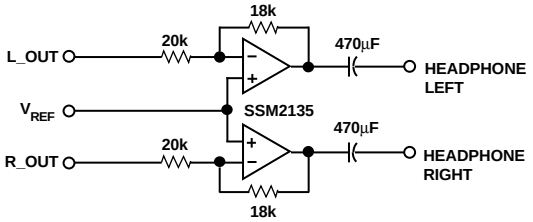


Figure 18. Headphone Drive Connections

Figure 19 illustrates reference bypassing. V_{REFI} should only be connected to its bypass capacitors.

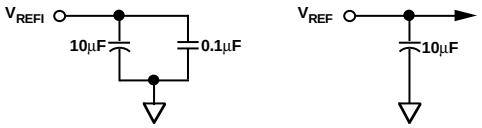


Figure 19. Voltage Reference Bypassing

Figure 20 illustrates signal-path filtering capacitors, L_FILT and R_FILT. The AD1847 must use 1.0 µF capacitors. The 1.0 µF capacitors required by the AD1847 can be of any type.

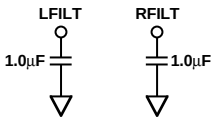


Figure 20. External Filter Capacitor Connections

The crystals shown in the crystal connection circuitry of Figure 21 should be fundamental-mode and parallel-tuned. Two sources for the exact crystals specified are Component Marketing Services in Massachusetts, U.S. at 617/762-4339 and Cardinal Components in New Jersey, U.S. at 201/746-0333. Note that using the exact data sheet frequencies is not required and that external clock sources can be used to overdrive the AD1847s internal oscillators. (See the description of the CFS2:0 control bits above.) If using an external clock source, apply it to the crystal input pins while leaving the crystal output pins unconnected. Attention should be paid to providing low-jitter external input clocks .

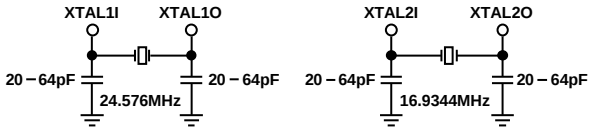


Figure 21. Crystal Connections

Analog Devices also recommends a pull-down resistor on the $\overline{\text{PWRDOWN}}$ signal.

Good, standard engineering practices should be applied for power-supply decoupling. Decoupling capacitors should be placed as close as possible to package pins. If a separate analog power supply is not available, the circuit shown in Figure 22 is recommended when using a single +5 V supply. Ferrite beads suffice for the inductors shown. This circuitry should be as close to the supply pins as is practical.

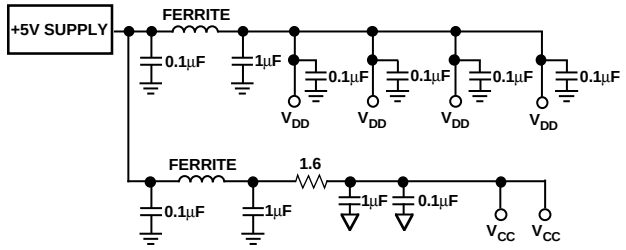


Figure 22. Recommended Power Supply Bypassing

Analog Devices recommends a split ground plane as shown in Figure 23. The analog plane and the digital plane are connected directly under the AD1847. Splitting the ground plane directly under the SoundPort Codec is optimal because analog pins will be located directly above the analog ground plane and digital pins will be located directly above the digital ground plane for the best isolation. The digital and analog grounds should be tied together in the vicinity of the AD1847. Other schemes may also yield satisfactory results. If the split ground plane recommended here is not possible, the AD1847 should be entirely over the analog ground plane with the ASIC and DSP over the digital plane.

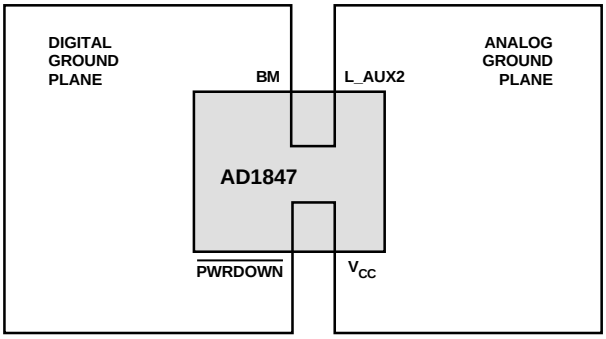


Figure 23. Recommended Ground Plane

FREQUENCY RESPONSE PLOTS

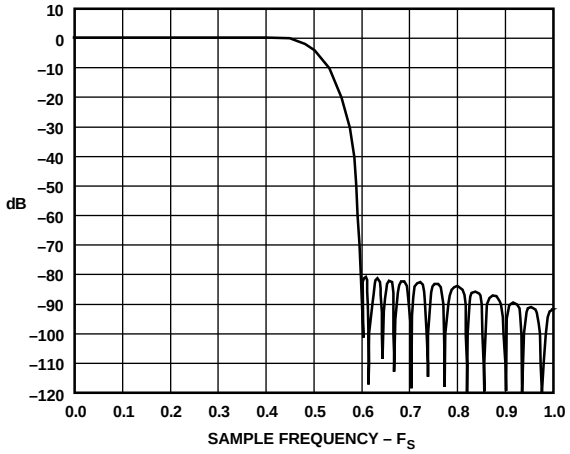


Figure 24. AD1847 Analog-to-Digital Frequency Response (Full-Scale Line-Level Inputs, 0 dB Gain)

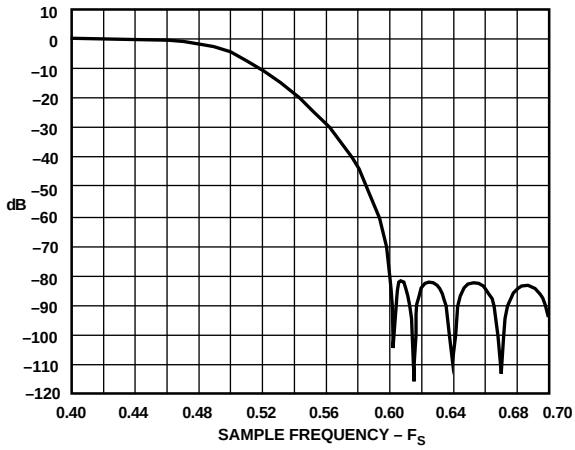


Figure 25. AD1847 Analog-to-Digital Frequency Response –Transition Band (Full-Scale Line-Level Inputs, 0 dB Gain)

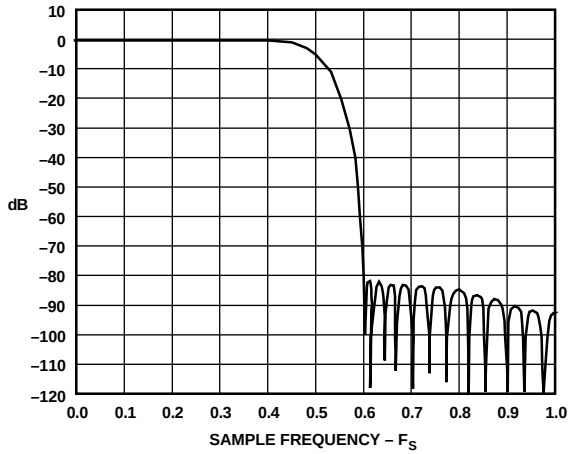


Figure 26. AD1847 Digital-to-Analog Frequency Response (Full-Scale Inputs, 0 dB Attenuation)

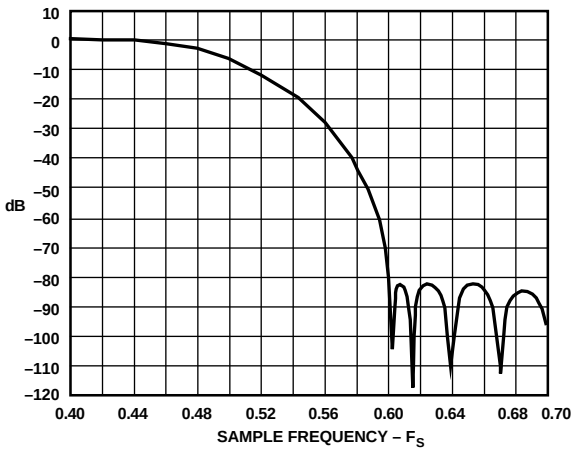


Figure 27. AD1847 Digital-to-Analog Frequency Response –Transition Band (Full-Scale Inputs, 0 dB Attenuation)

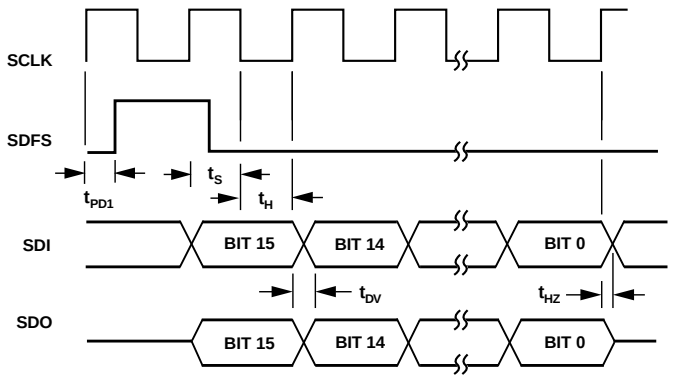


Figure 28. Time Slot Timing Diagram

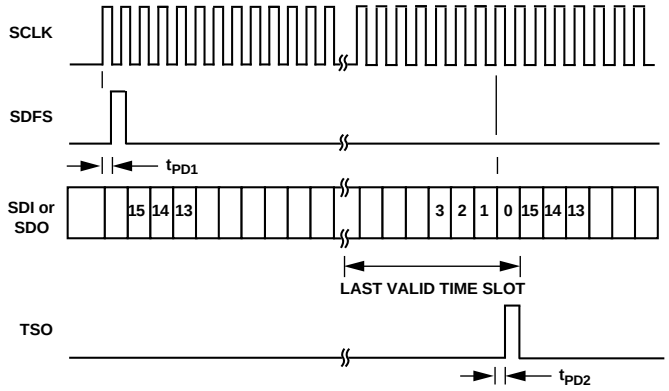


Figure 29. TSO Timing Diagram

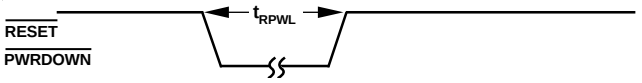


Figure 30. Reset and Power Down Timing Diagram

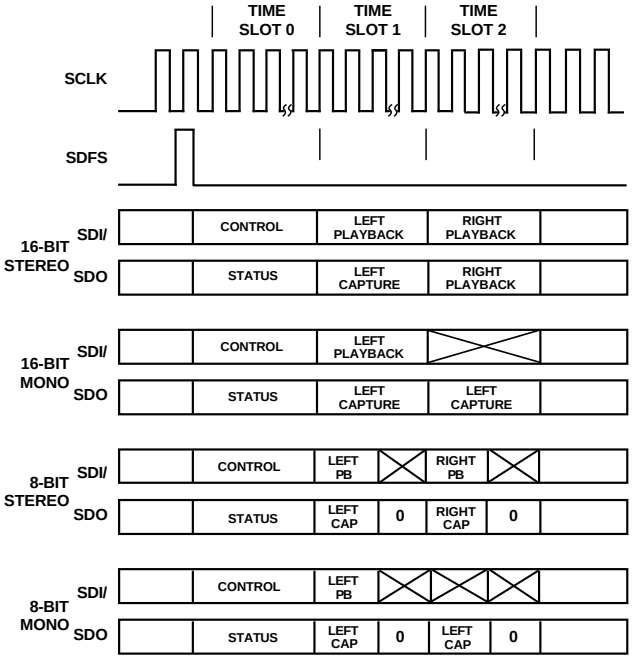


Figure 31. Serial Data Format, 2-Wire System (TSSEL = 1)

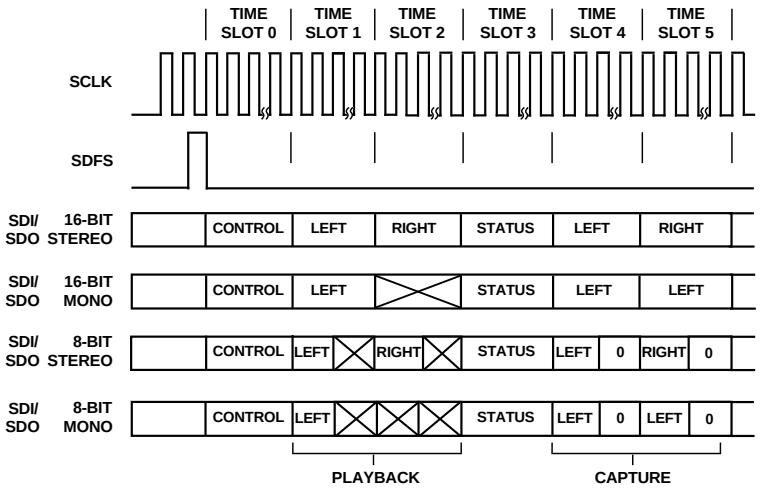
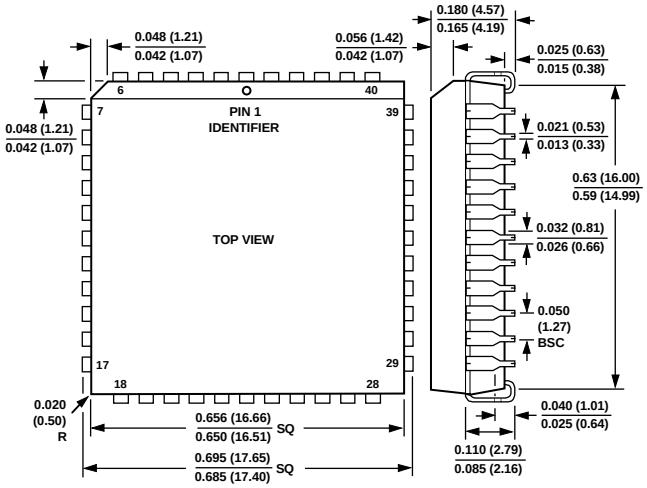


Figure 32. Serial Data Format, 1-Wire System (TSSEL = 0)

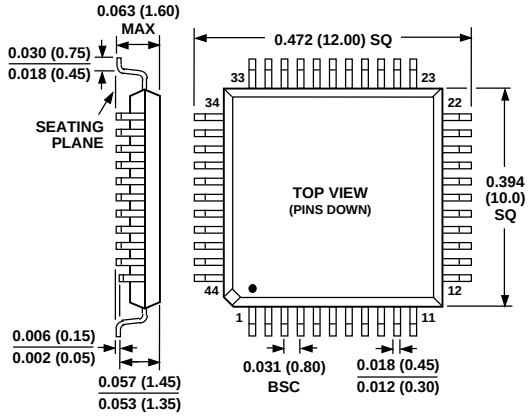
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

44-Lead PLCC (P-44A)



44-Terminal Plastic Thin Quad Flatpack (TQFP) (ST-44)



INDEX **PAGE**

PRODUCT OVERVIEW 1
AD1847 SPECIFICATIONS 2
ORDERING GUIDE 5
PINOUTS 5
PIN DESCRIPTIONS 6
AUDIO FUNCTIONAL DESCRIPTION 7
- Analog Inputs 7
- Analog Mixing 7
- Analog-to-Digital Datapath 7
- Digital-to-Analog Datapath 7
- Digital Mixing 8
- Analog Outputs 8
- Digital Data Types 8
- Power Supplies and Voltage Reference 8
- Clocks and Sample Rates 8

CONTROL REGISTERS 9
- Control Register Mapping 9
- Control Word 10
- Left/Right Playback/Capture Data 10
- Status Word 11
- Index Readback 12
- Indirect Mapped Registers 12
- Left Input Control Register 13
- Right Input Control Register 13
- Left Auxiliary #1 Input Control Register 13
- Right Auxiliary #1 Input Control Register 13
- Left Auxiliary #2 Input Control Register 14
- Right Auxiliary #2 Input Control Register 14
- Left DAC Control Register 14
- Right DAC Control Register 14
- Data Format Register 15
- Interface Configuration Register 16
- Pin Control Register 16
- Invalid Address 16
- Miscellaneous Information Register 17
- Digital Mix Control Register 17
- Invalid Address 17
- Serial Data Interface 18
- Control Register Mapping Summary 20
- Daisy-Chained Multiple Codecs 20

INITIALIZATION AND PROCEDURES 21
- Reset and Power Down 21
- Clock Connections and Clock Rates 21
- Mode Change Enable State 22
- Digital Mix 22
- Autocalibration 22
- Changing Sample Rates 22

DATA FORMAT DEFINITIONS 23
- 16-Bit Signed Format 23
- 8-Bit Unsigned Format 23
- 8-Bit Companded Formats 23

APPLICATIONS CIRCUITS 23
FREQUENCY RESPONSE PLOTS 26
TIMING DIAGRAMS 27
OUTLINE DIMENSIONS 28

C1871b–2–8/96

PRINTED IN U.S.A.