

**AdLib®**  
**GOLD**

# Developer Toolkit

Reproduction

# **Ad Lib Gold Developer Toolkit Version 1.01**

## **Release Notes**

### **RL2DRV.EXE**

The current version of the ROL2 playback TSR does not check for the presence of other drivers. The drivers used by RL2DRV.EXE must be loaded prior to executing this TSR.

When loading the ROL2 playback driver, the current directory must be the directory where the .SMP files are located.

### **SAMPL.EXE**

The current version of the Sample Editor does not use the AD Lib Gold drivers. It uses linkable libraries that conflict with the drivers.

In order to execute the Sample Editor, all drivers must be removed from memory, otherwise the program may hang or display an "Insufficient Memory" message.

### **Disabling Interrupts when accessing the hardware**

In order to avoid possible conflicts between applications that try to access the same hardware at the same time, it is recommended that interrupts be disabled when accessing the OPL3, the Control Chip or the MMA. This will avoid conflicts between applications, TSR programs and drivers that will be supplied with the Gold card in the future.

This procedure should be strictly adhered to for all software developed for the Gold card.

To insure that the interrupt flag status is not destroyed when re-enabling interrupts, the following procedure is recommended:

To disable interrupts:

```
pushf      ; push flags, include interrupt flags  
cli       ; clear interrupts
```

To re-enable interrupts:

```
popf      ; pop flag, includes interrupt flags
```

### **TSR Hotkey reconfiguration**

The Mixer Panel TSR and ROL2 Playback TSR hotkeys can now be reconfigured from the SETUP.EXE application. Please the README.TXT file for more details on the changes that were made to SETUP.EXE, RL2DRV.EXE and MIXER.EXE

### **MIDI Driver, SCSI Driver, Windows DLLs**

To be released



# **Ad Lib Gold**

## **Developer Toolkit**

**Version 1.01**

## **Copyright**

This manual is protected by copyright law and may not be reproduced in whole or in part, whether for sale or not, without written consent from Ad Lib Inc. Under the copyright laws, copying includes translation into another language or format.

Ad Lib is a registered trademark of Ad Lib Inc.

## **Licensing Policy**

Developers are authorized to incorporate the example source code provided with the Developer Toolkit into their products.

The no cost right to distribute Ad Lib Gold drivers provided with the Developer's Toolkit must be obtained through the Ad Lib Developer Support Department, at (418) 529-9676.

In order to ensure that third party applications operate in a consistent environment, we strongly recommend that developers contact the Ad Lib Developer Support Department.

Ad Lib Inc.  
220 Grande-Allée East, Suite 850  
Québec, QC, Canada G1R 2J1

50 Staniford Street, Suite 800  
Boston, MA 02114

Copyright ©1992, Ad Lib Inc.

1st edition

ISBN 2-920858-32-7

Printed in Canada

# **Table of Contents**

---

---

<b>1. Introduction</b>	<b>1-1</b>
<b>2. Quick Start and Evaluation Software</b>	<b>2-i</b>
Installing the Hardware	2-1
Installing the Software	2-1
Using the Gold Card Evaluation Software	2-2
Adjusting the Volume	2-3
<b>3. Gold Hardware</b>	<b>3-i</b>
3.1 Description of the Hardware	3-1
3.2 Getting Installed	3-9
3.3 Surround Sound Module	3-15
<b>4. Software Applications</b>	<b>4-i</b>
4.1 Software Installation and Configuration	4-1
4.2 Test Program	4-5
4.3 Mixer Panel TSR	4-7
4.4 Juke Box Gold Music Playback Program	4-13
4.5 Instrument Maker Gold	4-17
4.6 Sample Maker	4-19
4.7 Surround Sound Editor	4-23
4.8 Batch File Utilities	4-29
4.9 ROL2 Playback TSR	4-31

---

## **Table of Contents**

---

---

### **5. DOS Software Drivers** 5-i

---

5.1 Interfacing DOS Drivers with Applications	5-1
5.2 DOS Control Features Driver	5-3
5.3 DOS FM Synthesis Driver	5-57
5.4 DOS Wave Driver	5-71
5.5 DOS Timer Driver	5-93
5.6 DOS MIDI Driver (To be released)	
5.7 DOS SCSI CD-ROM Driver (To be released)	

---

### **6. Windows DLLs**

---

(To be released)

---

### **7. Low-level Programming** 7-i

---

7.1 Mixer and Setup Features	7-1
7.2 FM Synthesis	7-15
7.3 Digital Input and Output (Digital Audio and MIDI)	7-37

---

## **Appendices**

---

Appendix A: Gold Sound Standard	A-i
Appendix B: YM7128 - Surround Processor	B-i
Appendix C: Pin Out for Joystick-MIDI Connector	C-1
Appendix D: List of Installed Files	D-1

# **Introduction**

---

The Ad Lib Gold Developer Toolkit is a set of software applications, libraries, documentation and other information that will accelerate application support for the Ad Lib Gold series of cards.

The Developer Toolkit covers the following areas:

- Quick Start and Evaluation Software**

This section is for developers who want to try the software provided with the Gold Card and get a quick look at all programming possibilities. It summarizes the installation of the Gold card and briefly describes some of the available applications. The evaluation software provides tools for a variety of sound quality tests.

- Gold Hardware**

This section describes the physical layout of the Gold card, and the procedure for installing the card in your computer. It also provides a short description of the Surround Sound Module.

- Software Applications**

Describes the applications and TSRs that you can use with the Gold card. Some of these applications can be used to create FM sounds, digitized sounds, and Surround presets, that the developer can use in his/her applications.

- Software Drivers**

This section explains how your applications can interface with the memory-resident drivers. It also contains a complete function directory for each of the drivers. Sample source code is supplied on diskette to provide a better understanding of the use of the drivers and associated interface libraries. The MIDI driver, SCSI driver and Windows DLLs will be included with the next version.

- Low-Level Programming**

Details the I/O map of each of the hardware sections of the Gold Card. This section is intended for programmers who want to directly access the hardware, instead of using the software drivers.

- Appendices**

The appendices provide additional information on the Yamaha Gold Sound Standard, the Surround Processor chip (YM7128), a diagram covering the MIDI / Joystick connector, and a list of files copied to the hard drive during installation.

Please read the README.TXT file on DISK 1 for more information.



# **Chapter 2 - Quick Start and Evaluation Software**

---

<b><u>Installing the Hardware</u></b>	<b>1</b>
Installing the Gold Card	1
Connect the Other Peripherals	1
<b><u>Installing the Software</u></b>	<b>1</b>
Read the README.TXT File	1
Install Gold Applications and Resources	2
Test Hardware	2
<b><u>Using the Gold Card Evaluation Software</u></b>	<b>2</b>
Running Gold DOS Applications	2
<b><u>Adjusting the Volume</u></b>	<b>3</b>



# Quick Start and Evaluation Software

---

The Quick Start is intended for developers who want to have quick access to some of the evaluation programs provided with the Gold card.

## Installing the Hardware

### **Installing the Gold card**

1. Make sure that the on-board jumpers, the "Game port enable jumper", the "Port address jumper" and the "Dual joystick selector jumpers", are in the desired position.
2. Plug the Gold card into the computer in a free slot as far as possible from the video adapter card.

\* *NOTE: Certain cards, such as video adapters, produce high-frequency signals which can interfere with the sound quality of the sound card.*

See "3.2: Getting Installed".

### **Connect the Other Peripherals**

- Plug headphones or external speakers into the main audio output of the card, or connect the output to the input of a stereo system.
- Connect your microphone to the microphone input of the card.

- Connect the output of your stereo source (CD player, CD-ROM drive, synthesizer or cassette player) to the stereo auxiliary input of the card, using a stereo cable.
  - Connect your joystick to the DB-15 game port of the card. If you plan to use the MIDI interface, connect your MIDI device with the Ad Lib adapter cable.
- See "3.2: Getting Installed".

## Installing the Software

### **Read the README.TXT File**

We suggest that you examine the README.TXT file prior to installing the software. This file contains information on the latest program updates, and other necessary information.

- Insert Ad Lib diskette No. 1 into the floppy drive, set the current drive to A (or B, depending on which drive you are using), and type the following command:

**A :>type readme.txt**

**Installing the Software**

---

**Install Gold Applications and Resources**

- Run the Gold Setup Program by typing the following commands:

**A:\>ctrldrv**

**A:\>setup**

- See "4.1: Software Installation and Configuration".

**Test Hardware**

Once installation is complete, run the Test program to verify that the Ad Lib Gold card is functioning properly.

- Go to the directory where you placed the Gold Test Program at installation and load this program by typing the following command:

**testgold**

- See "4.2: Test Program".

**Using the Gold Card Evaluation Software**

---

**Running Gold DOS Applications**

Once the Gold hardware and software are installed, you can run any Ad Lib Gold application by proceeding as follows:

1. Set the current directory to the one where you placed the Gold programs during the installation process.

2. Load the Mixer Panel TSR program first, which serves to control the different sound parameters (balance, tone, volume, etc.), by typing the following command:

**mixer**

3. Load the program you want by typing the corresponding command:

**jukegold** Juke Box Gold Music Playback Program

**insgold** Instrument Maker Gold Program

**samp1** Sample Maker Program

**surround** Juke Box Gold Music Playback Program including the Surround Sound Editor

Note that the Juke Box Gold Music Playback Program offers on-line Help containing summarized information on how to operate the program and how to use the various features.

- See "Chapter 4 - Software Applications".

### Adjusting the Volume

When running a program with the Gold card, you can adjust output volume at any time, without opening the Mixer Panel, using the following shortcuts:

**[Alt]-[Up]-[U]** For increasing output volume.

**[Alt]-[Up]-[D]** For decreasing output volume.

See "4.3: Mixer Panel TSR".



# **Chapter 3 - Gold Hardware**

---

---

<b>3.1 Description of the Hardware</b>	<b>1</b>
Functionality	1
Digital Recording	1
Digitized and Synthesized Sound Playback	1
MIDI Recording and Playback	2
Game Port	2
SCSI Interface	2
Layout of the Card	2
Bracket Connectors	2
On-board Connectors and Main Components	3
The On-board Jumpers	6
Available Interrupt Lines and DMA Channels	7
<b>3.2 Getting Installed</b>	<b>9</b>
System Requirements	9
Installing the Hardware	9
Hardware Configuration Settings	9
Removing the Computer Cover	11
Removing the Slot Cover	11
Installing the Gold Card	11
Connecting Other Peripherals	12
<b>3.3 Surround Sound Module</b>	<b>15</b>
Required Equipment	15
Installing the Surround Sound Module	15
Remove the Computer Cover	15
Remove the Sound Card	16
Attach the Surround Sound Module	17
Reinstall the Sound Card	17

## **Chapter 3 - Gold Hardware**

---

### **Table of Contents**

---

Using the Surround Sound Option	18
Surround-based Applications	19
Using Surround Sound with Other Sound Sources	19
Programming the Surround Sound Module	20

## Functionality

Your Ad Lib Gold Stereo Sound Adapter is a multifunction card with digital recording, playback of digitized and synthesized sounds, analog audio mixing, MIDI recording and playback, game port, and (for model Gold 2000) SCSI/CD-ROM interface.

### Digital Recording

With the Ad Lib Gold card, you can record from:

- A microphone, using Voice Pad or third party software;
- An audio tape or a compact disk, using Voice Pad or third party software;
- A telephone, using the optional add-on board contained in the Ad Lib PC Telephone Answering System.

### Digitized and Synthesized Sound Playback

With the Gold card, you can play back:

- Digitized sounds: The Gold card has two channels for digitized sounds. These channels can be used in a variety of ways, such as for voice notes with the Voice Pad program, percussion sounds in the Juke Box Gold songs, or third party software using stereo music or music with voiceover.
- Synthesized sounds: The Gold card has a 20-voice FM synthesizer which is used for Juke Box Gold songs and third party software music.

The sound capability of the Gold card also features:

- Audio mixing: The internal analog mixer of the Gold card controls the volume of various audio sources, through programming, such as within third party software, or manually using the Mixer Panel (see "Mixer Panel TSR" section).
- Volume control: The output volume of the Gold card is software controlled (see "Mixer Panel TSR").

## Description of the Hardware

### Functionality

- Tone control: Bass and treble controls are software controlled (see "Mixer Panel TSR").

### MIDI Recording and Playback

The MIDI (Musical Instrument Digital Interface) interface of the Gold card allows MIDI files to be recorded and played back using a MIDI adapter cable, any external MIDI instrument and an Ad Lib Gold supporting sequencer program.

### Game Port

The Ad Lib Gold card allows a standard IBM compatible joystick to be connected.

### SCSI Interface

The Gold card SCSI interface (optional with Gold 1000 and included on-board with Gold 2000) allows a CD-ROM drive or any SCSI type peripheral to be connected.

### Layout of the Card

#### Bracket Connectors

Ad Lib Gold Stereo Sound Adapters have three 1/8" connectors and one DB-15 connector on the support brackets, as shown in the diagrams (Figures 1 and 2).

These connectors are:

- The microphone input (No. 7), for sampling and/or mixing with other audio sources.
- The stereo auxiliary input (No. 8), for connecting an external source such as a CD or cassette player, a synthesizer or any audio source, in order to sample and/or mix with other audio sources.
- The main stereo audio output (No. 9), for connecting to standard headphones, bookshelf speakers or a stereo system.
- The Game Port/MIDI connector (No. 10), for using a standard PC joystick and/or a MIDI device. This requires an optional MIDI cable adapter available from Ad Lib.

## On-Board Connectors and Main Components

Ad Lib Gold Stereo Sound Adapters have on-board connectors to support many different options and internal/external devices. These connectors are shown in the diagrams (Figures 1 and 2).

These connectors are:

- On model Gold 2000, the SCSI port connector (Figure 2: No. 22), to connect a SCSI device, such as a CD-ROM, hard disk or tape backup drive. A 50-pin flat cable is provided for connecting an internal device. Cabling for connecting an external device is optional.
- On model Gold 1000, the SCSI option connectors (Figure 2: No. 20), to snap on a SCSI piggyback board.
- The Surround Sound option connectors (No. 1), to connect a Surround Sound Module. This module is used to provide stereo and depth enhancements.
- The telephone option connector (No. 2), to connect a telephone line interface add-on board. This board allows the Ad Lib Gold card to be connected to a standard telephone line and provides access to various functions, such as creating a completely digital telephone answering system capable of leaving

personalized messages for callers and recording and playing back messages left by callers directly to and from a hard disk, or creating an interactive automated telephone routing and database navigation station. It is also capable of automated dialing.

- The internal stereo auxiliary input (No. 4), to connect a PC internal audio device (such as an internal CD-ROM drive) for direct input. This connector is in parallel with and has exactly the same functions as the external stereo auxiliary input connector on the support bracket.

\* *NOTE: It is not recommended to use both the external and internal stereo auxiliary input at the same time, because this will decrease the volume of the auxiliary audio source.*

- The PC speaker connector (No. 3), to connect the signal of the PC's internal speaker directly to the Ad Lib Gold card, so that it is mixed with the other audio signals on the card and can be heard through the headphones or speakers.

## Description of the Hardware

### Layout of the Card

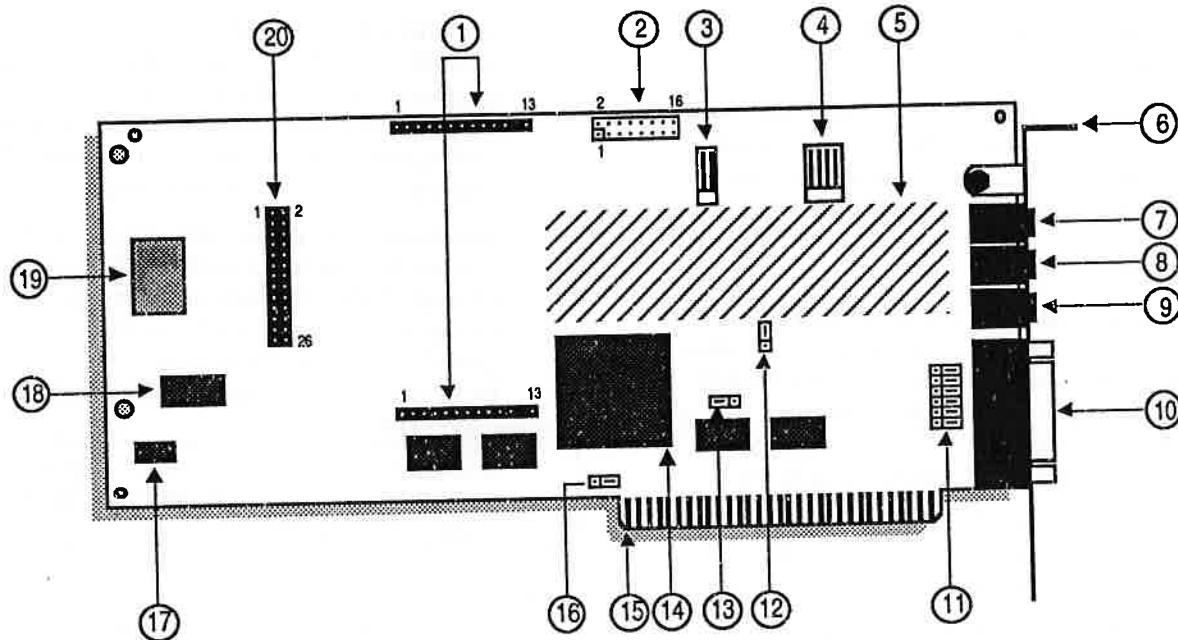


Figure 1: Gold 1000 diagram

- |                                     |                                            |                                       |
|-------------------------------------|--------------------------------------------|---------------------------------------|
| 1. Surround Sound option connectors | 8. Stereo aux. input                       | 15. Bus connector                     |
| 2. Telephone option connector       | 9. Main audio output                       | 16. Game port enable jumper (JP1)     |
| 3. PC speaker connector             | 10. Game port/MIDI DB-15 connector         | 17. 16-bit FM DAC                     |
| 4. Internal stereo aux. input       | 11. Dual joystick selector jumpers (JP2-7) | 18. Professional FM synthesis chip    |
| 5. Power amp and analog mixer       | 12. Port address jumper (JP8)              | 19. Sampling 12-bit DAC and MIDI chip |
| 6. Bracket                          | 13. Control chip reset jumper (JP9)        | 20. SCSI option connector             |
| 7. Microphone input (mono)          | 14. Custom control VLSI chip               |                                       |

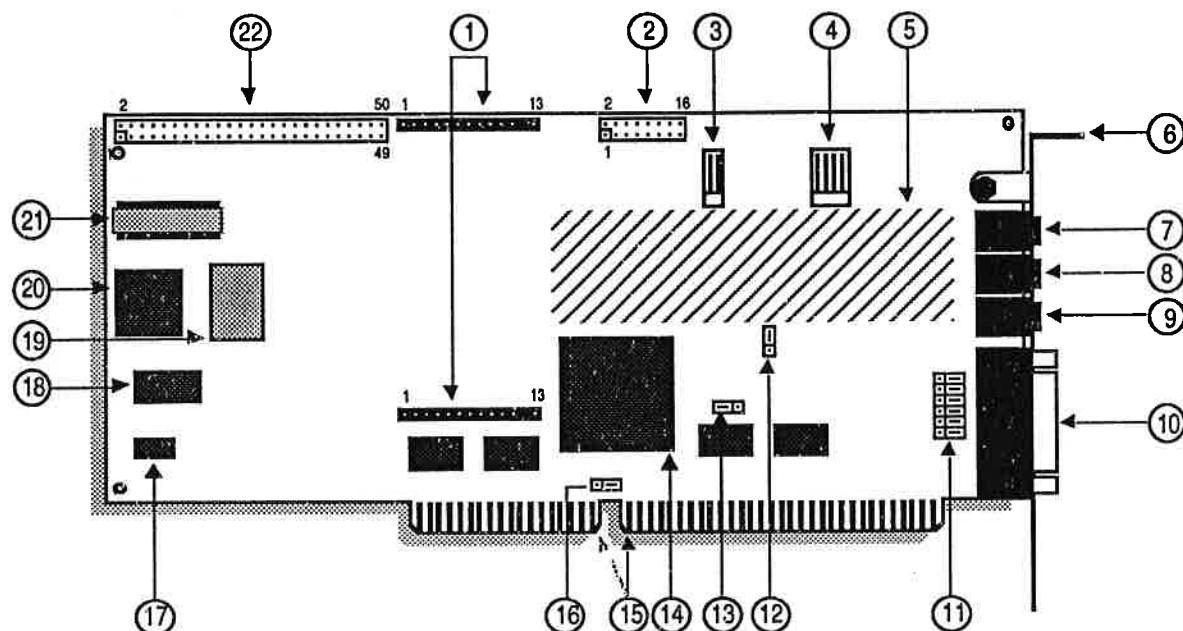


Figure 2: Gold 2000 diagram

- |                                     |                                            |                                       |
|-------------------------------------|--------------------------------------------|---------------------------------------|
| 1. Surround Sound option connectors | 9. Main audio output                       | 17. 16-bit FM DAC                     |
| 2. Telephone option connector       | 10. Game port/MIDI DB-15 connector         | 18. Professional FM synthesis chip    |
| 3. PC speaker connector             | 11. Dual joystick selector jumpers (JP2-7) | 19. Sampling 12-bit DAC and MIDI chip |
| 4. Internal stereo aux. input       | 12. Port address jumper (JP8)              | 20. SCSI chip                         |
| 5. Power amp and analog mixer       | 13. Control chip reset jumper (JP9)        | 21. SCSI terminator resistor          |
| 6. Bracket                          | 14. Custom control VLSI chip               |                                       |
| 7. Microphone input (mono)          | 15. Bus connector                          |                                       |
| 8. Stereo aux. input                | 16. Game port enable jumper (JP1)          |                                       |

### The On-board Jumpers

To make it easier to configure the Ad Lib Gold card, we have made the Interrupt lines (IRQ) and DMA channels software selectable, thereby keeping the amount of jumpers to a minimum. The four remaining jumper sets are the game port enable jumper (JP1), the dual joystick selector jumpers (JP2-7), the port address change jumper (JP8), and the Control chip reset jumper (JP9).

The Gold card jumpers are the following:

- **Game port enable jumper (No. 16)**  
This jumper lets the user enable/disable Ad Lib Gold's on-board game port interface. The interface should be disabled in cases where the user already has a standard PC game port interface inside his/her PC, to avoid conflicts.

Jumper setting is shown here:



Game port enabled

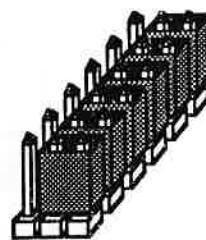


Game port disabled

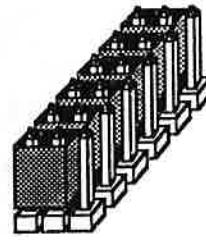
Figure 3: Game port jumper enabling

\* NOTE: The jumper is factory set to the game port enabled position.

- **Dual joystick selector jumpers (No. 11)**  
These jumpers let the user change the factory-set "joystick plus MIDI" option (all jumpers on the bracket side) to the "two joysticks without MIDI" option (all jumpers on the opposite side). All jumpers in this selector must be changed to the same position, as shown in the following illustrations.



Single joystick with MIDI option (factory-set)



Dual joystick option

Figure 4: Dual joystick jumper selection

- *Port address jumper* (No. 12)

This jumper lets the user choose a single or double port address for the Gold card. The Ad Lib Gold card addresses can be assigned by software programming. The default port address of the Gold card is 388H and can be changed by software in cases where another card inside the PC uses the same address, in order to avoid conflicts. In the case where the software cannot recognize the programmed address, the port address jumper is used to force the Gold card into answering at both the programmed address and at the default factory address 388H.

\* *NOTE: The port address jumper is factory set to single port address position (jumper plugged on the two pins on the bracket side) which enables only one port address to be used at a time.*

The port address jumper can be changed to double port address position (jumper plugged on the side opposing the bracket) which forces the default address 388H to be used in conjunction with any other user-defined one.

- *Control chip reset jumper* (No. 13)

This jumper is used where the programmed configuration of the control chip is lost. In some cases, losing the configuration can cause the card to use addresses that are already in use by other hardware. Removing

the control chip reset jumper disables certain functions of the Gold card that could cause hardware conflicts. Once the jumper is removed, reconfigure the Gold card to the factory preset values by issuing the following command:

**setup /R**

Once the Gold card is reconfigured, replace the control chip reset jumper.

#### Available Interrupt Lines and DMA Channels

- There are four software selectable interrupt lines (IRQ 3, 4, 5 and 7) on the Gold 1000, and four additional choices on the Gold 2000 (IRQ 10, 11, 12 and 15).
- DMA channels 1,2 and 3 are software selectable on the Gold 1000, and DMA channels 0, 1, 2, and 3 are software selectable on the Gold 2000.



## System Requirements

To use the Ad Lib Gold card and the Gold software, you need the following:

1. For the Gold 1000: an IBM PC, XT, AT (286), 386, 486 compatible computer, PS/2 Model 25 and 30, or Tandy 1000 (except EX/HX), a disk drive (1.2 MB 5 1/4" or 720 KB 3 1/2") and 640K of RAM.  
For the Gold 2000: an IBM AT (286), 386 and 486 compatible computer, a disk drive (1.2 MB 5 1/4" or 720 KB 3 1/2") and 640K of RAM.
2. A hard disk.
3. Graphics adapter, any model.
4. PC/MS-DOS 3.0 or higher.
5. Headphones, an external speaker or a home stereo system.
6. A microphone.

## Installing the Hardware

We suggest that you read this section thoroughly before you begin. This will familiarize you with the standard installation procedure.

These instructions are for installing your Ad Lib Gold card in your computer. We recommend that you read the owner's manual supplied with your computer for instructions specific to your model of computer.

## **Hardware Configuration Settings**

To install the Ad Lib Gold card, there are two types of configuration settings: hardware settings and software settings. The Gold card uses software for most of the configuration settings (see the sections on installation and configuration below). Only three hardware settings, made with jumpers, are necessary: game port enabling/disabling, dual joystick selection and port address. These should be made before installing the Gold card in your computer.

**To Set Game Port Enable/Disable Jumper**

The game port enable/disable jumper lets the user enable or disable Ad Lib Gold's on-board game port interface. The interface should be disabled in cases where the user already has a standard PC game port interface inside his/her PC, in order to avoid conflicts. To obtain the desired setting:

1. Locate the jumpers for the game port enable/disable setting (refer to Figure 1 or 2, No. 15).
2. If your computer does not have a game port, make sure that the jumper is over the two left pins as shown in Figure 3. This jumper is factory set in the game port enabled position.
3. If your computer has a game port, unplug the jumper from the two left pins and replug it onto the two right pins as shown in Figure 3.

**To Set Dual Joystick Selector Jumpers**

The dual joystick selector jumpers let the user change the factory-set "joystick plus MIDI" option to the "two joysticks without MIDI" option. All jumpers in this selector must be changed to the same position, as shown in Figure 4. To obtain the desired setting:

1. Locate the dual joystick selection jumpers (refer to Figure 1 or 2, No. 11).
2. If you wish to use the Gold card's game port in the "single joystick with MIDI option", leave the jumpers in the factory-set position (i.e. plugged into the bracket side of the card) as shown in Figure 4.
3. If you want to use the Gold card's game port in the "dual joystick option", unplug all six jumpers and replug them onto the jumpers, as shown in Figure 4.

**To Set Port Address Jumper**

The port address jumper lets the user choose a single or double port address for the Gold card. The default port address of the Gold card is 388H. It can be changed with the Setup program in cases where another card inside the PC uses the same address as the Ad Lib card.

The port address jumper is factory set to the single port address position which enables only one port address to be used at a time. These port addresses can be modified by software. It can be changed to double port address position, which forces the default address 388H to be used in conjunction with any other.

To obtain the desired setting:

1. Locate the jumpers for the port address setting (refer to Figure 1 or 2, No. 12).
2. If you wish to use only one port address at a time (388H or any other), make sure that the jumper is over the two upper pins. This jumper is factory set in the single port address position.
3. If you wish to force the default port address 388H to be used in conjunction with another one, unplug the jumper from the two upper pins and replug it onto the two lower pins.

#### Removing the Computer Cover

1. Switch off the computer.
2. Disconnect the power cord and all peripheral devices and cables.
3. Set the computer on a flat, clear surface.
4. Remove the mounting screws that hold the computer cover.
5. Remove the computer cover.

#### Removing the Slot Cover

1. Choose a free slot as far as possible from the video adapter card.

\* *NOTE: Certain cards, such as video adapters, produce high-frequency signals which can interfere with the sound quality of the sound card.*

2. Remove the screw that holds the slot cover in place.
3. Lift the slot cover to remove it.

! *WARNING: If a screw falls into the computer, you absolutely must remove it before switching your system back on. If a metal object is left loose inside the casing of your computer, it may cause a short circuit that will damage your system.*

#### Installing the Gold Card

1. Place the card immediately above the slot without inserting it into the socket.
2. Make sure that the bracket is inserted in the groove previously occupied by the slot cover.
3. Press the card down into the socket.

4. Put the card's bracket screw back on and tighten it.
5. Put the computer cover back on and tighten the screws.
6. Reconnect the power cord and other cables.

## Connecting Other Peripherals

The card is equipped with jacks and plugs for connecting various peripherals. These allow devices to be connected to: stereo audio output, microphone input, line-level stereo audio input, MIDI/game port, PC internal speaker and SCSI port (Gold 2000 model only).

### Audio Output

The Gold card is equipped with three 1/8" jacks for connecting audio equipment. The main audio output is the lowermost of the three jacks, located above the DB-15 connector (refer to Figure 3 or 4). This jack can be connected to headphones, external speakers or a stereo system using stereo adapters and cables. Model Gold 2000 comes with a cable. To avoid distortion when connecting to a stereo system, connect the card to an auxiliary-type input.

### Microphone Input

The microphone input is the uppermost of the three audio jacks on the card's bracket (refer to Figure 3 or 4). This connector lets the audio signal from a standard microphone be mixed with other audio sources or to be used as a source for sampling sounds.

### Stereo Auxiliary Input – External Connector

The external stereo auxiliary input connector is located in the center of the three audio jacks on the card's bracket (refer to Figure 3 or 4). This connector lets audio signals from a stereo source (such as a CD player, CD-ROM drive, synthesizer or cassette player) be mixed with the other audio sources or to be used as a source for sampling sounds.

! **WARNING:** To avoid distortion, it is important to keep the audio level of the device you are connecting to this input jack at low volume and to adjust the volume using the software controls explained in the next section. Also, make sure that you are using the auxiliary output of the device you are connecting to the card, instead of using the speaker output which would overload the card's amplifier.

connect one of the following three options:

1. An IBM compatible joystick.
2. A MIDI device. (This requires an adapter cable.)
3. Dual joystick. (This requires a special adapter, which is usually supplied by the joystick manufacturer.—See Figure 4 for related jumper settings.)

#### Stereo Auxiliary Input – Internal Connector

As mentioned in the "Description of the Hardware: Layout of the Card" section, there is an internal connector (see No. 4 in Figure 1 or 2) for connecting the audio output of an internal CD-ROM drive. This connector is electrically in parallel with the external stereo auxiliary input connector. Thus, to obtain good sound results, you may only use one of these at a time.

#### MIDI/Game Port

The Ad Lib Gold card features a standard DB-15 connector at the bottom of its supporting bracket (refer to Figure 1 or 2). This connector lets the user



Ad Lib Gold Stereo Sound Adapters have an expansion connector for an optional plug-in module capable of adding a "surround" sound effect to the audio output of the card. This effect can range from stereo depth simulation to artificial reverberation and echo.

The Ad Lib Surround Sound Module is a piggyback card and so does not require its own dedicated slot.

### Required Equipment

To use the Ad Lib Surround Sound Module, you need the following equipment:

1. An Ad Lib Gold Stereo Sound Adapter: Gold 1000 or Gold 2000.
2. For the Gold 1000: an IBM PC, XT, AT (286), 386, 486 compatible computer, PS/2 Model 25 and 30, or Tandy 1000 (except EX/HX), a disk drive (1.2 MB 5 1/4" or 720 KB 3 1/2") and 640K of RAM.  
For the Gold 2000: an IBM AT (286), 386 and 486 compatible computer, a disk drive (1.2 MB 5 1/4" or 720 KB 3 1/2") and 640K of RAM.
3. A hard disk.

4. An operating system: PC/MS-DOS 3.0 or later.
5. A graphics adapter (monitor).
6. Headphones, external speaker(s) or home stereo system.

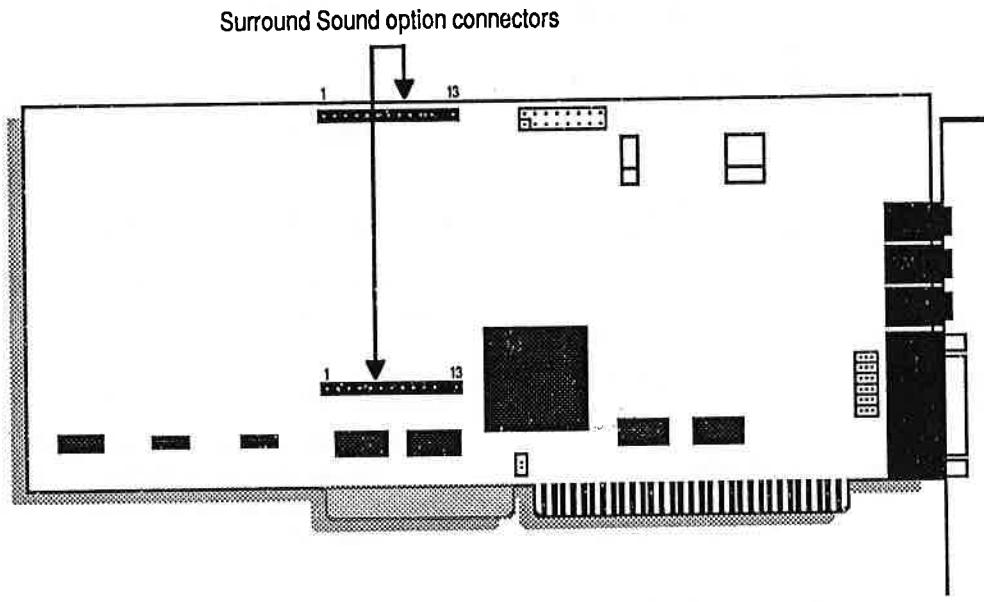
### Installing the Surround Sound Module

---

To install the Surround Module onto the Ad Lib Gold card, proceed as follows:

#### **Remove the Computer Cover**

1. Switch off the computer, disconnect the power cord, and disconnect all peripheral devices and cables connected to the computer.
2. Set the computer on a flat, clear surface.
3. Remove the mounting screws at the back of the computer (consult your hardware user's guide).
4. Remove the computer cover (consult your hardware user's guide).



**Figure 1:** Location of the Surround Sound option sockets on the Ad Lib Gold 1000 and 2000 cards

**Remove the Sound Card**

! **WARNING:** *Users should ground themselves before handling the card. Please read the manual before beginning installation.*

1. Remove the sound card bracket screw.
2. Take the sound card out of the computer and place it on a flat surface so that it is positioned as in Figure 1.

**Attach the Surround Sound Module**

1. Locate the Surround Sound option sockets on the sound card (see Figure 1).
2. Place the module connector pins immediately above the socket holes on the sound card.
3. Make sure that the No. 1 pins of the module line up with the No. 1 holes of the sound card. (If properly aligned, the Ad Lib logo will be in an upright position.)
4. Simultaneously press both ends of the module firmly into the card sockets (see Figure 2).

!

***WARNING: The module only fits in one way, since both No. 12 pins are missing and the No. 12 socket holes are stoppered. Do not force the module in if you feel resistance; it may be incorrectly positioned.***

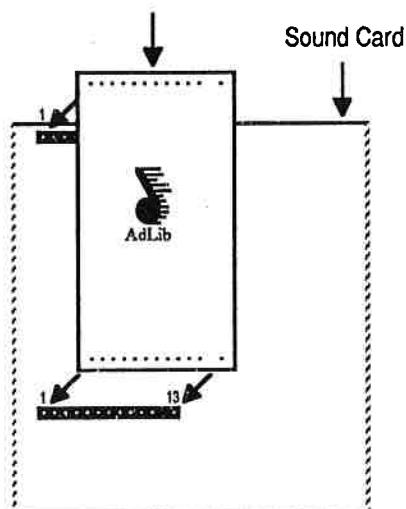
**Surround Sound Module**

Figure 2: Attaching the Surround Sound Module

**Reinstall the Sound Card**

1. Place the card immediately above the slot without inserting it into the socket.
2. Make sure that the bracket is inserted in the groove previously occupied by the slot cover.
3. Press the card down into the socket.

## **Surround Sound Module**

### **Installing the Surround Sound Module**

4. Put the card's bracket screw back on and tighten it.
5. Put the computer cover back on and tighten the screws.
6. Reconnect the power cord and other cables.

### **Using the Surround Sound Option**

Once your module is connected to the sound card, the Surround Sound option is ready to use. Ad Lib's control software offers the user a choice of various pre-programmed audio enhancements that create totally new, compelling effects. To use the Surround Sound option, simply proceed as follows:

1. Load the Ad Lib Gold Mixer Panel Utility (see the Gold card user guide for complete information on the Mixer Panel).
2. Activate the Mixer Panel. **Alt**-**Shift**-**M** are the default activation keys, but you may change this combination with the "Keys" window of the Mixer Panel.
3. Press the **F3** key when in the Mixer Panel main window to open the Surround Features control window (see Figure 3).

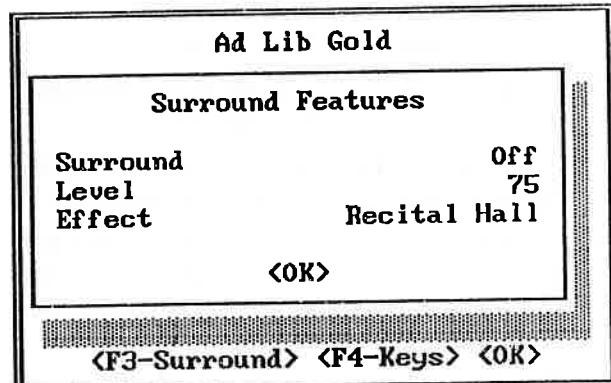


Figure 3: Mixer Panel Surround Features window

- \* *NOTE: If the Surround Sound Module is not installed, or not correctly installed, the program will display the message: "OPTION NOT INSTALLED" and changes you make to the parameters will have no effect.*
- 4. Using the arrow keys, select (with **↓** and **↑**) and set (with **→** and **←**) the Surround Sound option parameters:

**Surround**

A toggle On/Off allows the Surround Sound option to be enabled or disabled. The default setting is Surround Off. The Mixer Panel also allows you to use a combination of keys for enabling and disabling the surround sound effect. The default combination is **[Alt]-[Space]-[S]**, but you may change this combination with the "Keys" window of the Mixer Panel.

**Level**

Sets the volume level of the surround sound effect.

! **WARNING:** *Do not set the level of the surround sound effect too high, because it may result in distortion in some cases. It is advisable to increase the level by only a few units at a time.*

**Effect**

Sets the type of surround sound effect you want from among the presets.

**<OK>** (**[F1]** or **[Esc]**)

Closes the Surround Features control window, then the main Mixer Panel window with the changes you made in the parameter settings.

**Surround-based Applications**

Software does not have to be specially written to take advantage of the Ad Lib Surround Sound module. The module will instantly enhance any music and sound program written with Ad Lib sound support. Nevertheless, software developers can program surround sound effect changes within their application for contrast and drama.

**Using Surround Sound with Other Sound Sources**

Besides sound and music software, any audio source mixed with the Gold card can take advantage of Surround effects. The internal analog mixer allows you to blend FM and sampling sounds, with live mike sounds, a CD or cassette player, a synthesizer, a CD-ROM drive, etc. Simply connect your instrument, choose the surround effect you want and set the output balance for these sound sources with the Ad Lib Gold Mixer Panel Utility. Please refer to your *Ad Lib Gold Stereo Sound Adapter and DOS Software* user guide for details on the mixer and Mixer Panel.

## **Surround Sound Module**

---

### **Using the Surround Sound Option**

---

It is also possible to add surround sound effects to regular tape recordings. To do so, simply connect the audio source (microphone, synthesizer, etc.) to the input connector of the Gold card, and the output of the card to the tape input of your tape recorder.

### **Programming the Surround Sound Module**

If you do your own programming, it is possible to program the Surround Sound Module with the Ad Lib Gold Programmer's Manual. It shows you how to program your own presets and dynamic surround changes, to add extra contrast and drama to your Ad Lib Gold Sound applications.

---

3-20

Ad Lib Gold

© Ad Lib Inc. 1992

Confidential

Mon, Mar 23, 1992

# **Chapter 4 - Software Applications**

<b><u>4.1 Software Installation and Configuration</u></b>	<b>1</b>
Using the Setup Program	1
Configuration Environment Variable	3
<b><u>4.2 Test Program</u></b>	<b>5</b>
Testing the Hardware	5
Preparing the Test Program	5
Loading the Test Program	5
To Continue the Test	5
To Cancel the Test and Exit the Program	6
Choosing Test Options	6
<b><u>4.3 Mixer Panel TSR</u></b>	<b>7</b>
Loading the Mixer Panel TSR	7
Activating the Mixer Panel	8
Using the Mixer Panel	8
Sound Parameters	8
Sub Mixer	9
Surround Features	10
ALT-SHIFT Keys (Short Cuts)	11
Closing the Mixer Panel	12
Removing the Mixer Panel TSR	12
<b><u>4.4 Juke Box Gold Music Playback Program</u></b>	<b>13</b>
Loading Juke Box Gold	13
Selecting Songs	14
Creating a Selection of Songs	14
To Remove Songs from the Selection	14

## **Chapter 4 - Software Applications**

---

### **Table of Contents**

---

Playing Music	14
To Play Songs	14
To Stop Music Playback	14
To Pause and Resume Music Playback	14
To Scan Songs	15
Adjusting the Sound	15
To Adjust the Volume, Bass or Treble	15
To Set the Stereo/Mono Option	15
Asking for Help	15
Exiting the Program	16
Using the ROL2 Playback TSR	16
<b>4.5 Instrument Maker Gold</b>	<b>17</b>
Loading Instrument Maker Gold	17
Using Menu Commands	17
<input type="checkbox"/> F5 File	17
<input type="checkbox"/> F6 Options	17
<input type="checkbox"/> F7 Document	18
Editing FM Instrument Sounds	18
<b>4.6 Sample Maker</b>	<b>19</b>
Loading Sample Maker	19
Using Menu Commands	19
<input type="checkbox"/> F5 File	19
<input type="checkbox"/> F6 Edit	19
<input type="checkbox"/> F7 Sampling	20
<input type="checkbox"/> F8 Options	20

Important Warnings for this Development Version of Sample Maker	21
Sampling Rate Limitations	21
Sampling Length Limitation	21
Scope Mode	21
Graphic Display in Different PCM Format	21
<b>4.7 Surround Sound Editor</b>	<b>23</b>
Ad Lib Surround Sound Editor	23
Technical Features	23
Opening the Surround Sound Editor	23
Using the Surround Sound Editor	24
Channel Line Attenuation Sections	25
Global Level and Feedback Parameter Section	25
Filter Parameter Section	25
Global Delay Line Parameter Section	25
Editing Surround Sound Presets	26
Using Menu Commands	26
File Menu	26
Panel Menu	27
Closing the Surround Sound Editor	27
<b>4.8 Batch File Utilities</b>	<b>29</b>
ROL2 Playback Utility	29
Digitized Sound Playback Utility	29
<b>4.9 ROL2 Playback TSR</b>	<b>31</b>
Using the ROL2 Playback TSR	31
ROL2 Playback TSR Data Files	31
ROL2 Playback TSR Options	32



The running and using of the Ad Lib Gold card drivers and programs require hard disk space of 3 megabytes. They must be installed onto the hard disk following a precise procedure. For this purpose, the Gold software package includes a special Setup Program. This program enables you to install the drivers and all associated programs, and to configure your Ad Lib Gold card.

## Using the Setup Program

The Ad Lib diskettes are not copy-protected. We recommend that you make a back-up copy before installing Gold software. Put the originals away in a safe place. This way, if a diskette is lost or damaged, you will have a replacement. We suggest that you use the DISKCOPY command. (For all details concerning the copy commands, refer to your DOS manual.)

### To Load the Setup Program

To load the Ad Lib Gold Setup Program:

1. Insert the first Ad Lib diskette into the floppy drive.
2. Set the current drive to A (or B, depending on the drive you are using).
3. Type the following commands:

A : \>ctrldrv

A : \>setup

When the program opens, a window entitled "Installation Notes" introduces you to the Setup Program and its basic commands. Two main buttons are displayed at the bottom of each screen: <Cancel> and <Continue>.

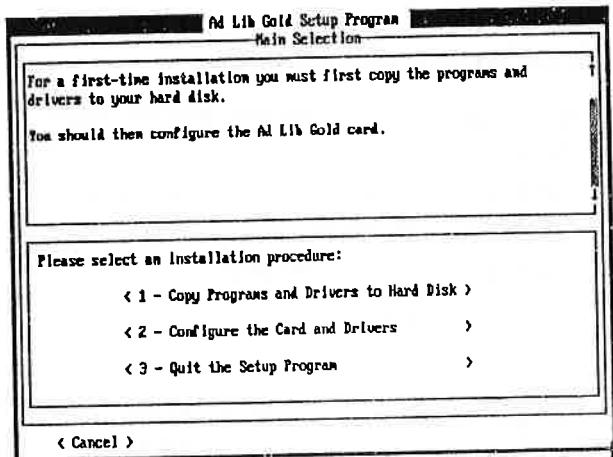
Activating the <Continue> button in the Installation Notes introductory screen makes the Setup Program open the Main Selection menu (see Figure 5).

This menu lets you choose and access the three following submenus:

1. Copy programs and drivers to hard disk
2. Configure the card and drivers
3. Leave the Installation Program

## **Software Installation and Configuration**

### **Using the Setup Program**



**Figure 5:** Setup Program's Main Selection menu

#### To Cancel the Setup Process

At any moment, you can interrupt the setup process by clicking on the <Cancel> button at the lower left corner of the screen, or by pressing the **Esc** key. Doing this will abort the setup and cancel the steps you have made. The changes you made are not saved in permanent memory on the card. When you reboot your system, these changes will not be restored. So, if you have a problem after making a change, just reboot your system.

\* **NOTE:** Certain elements cannot be reversed and will remain installed, such as copied files. To cancel the entire operation, it is necessary to re-run the Setup Program and reverse the corresponding steps, or delete the copied files. See Appendix A for a list of the installed files.

#### To Continue the Setup Process

In the setup process, activating the <Continue> button lets you close the current dialog box and access the next step. Doing this will initiate the changes you made in this dialog box, if there were any.

#### To Answer Program Questions

During each step of the setup procedure, you will be asked to choose between different options or to enter answers in edit fields. The program suggests an answer that will be correct in most situations. You can use this default, or enter your own answer.

### Configuration Environment Variable

A special environment variable, called "GOLD" is used by the software to recognize the base address of the Gold card. When the card is relocated by the Setup program, the program automatically modifies the environment variable in the AUTOEXEC.BAT file.

To change the GOLD environment variable, type the following command, which should be preferably put in the AUTOEXEC.BAT file:

```
SET GOLD = xxxx
```

Where **xxxx** is the hexadecimal value of the Gold card base address (Control chip address).

When the Gold environment variable is not defined, the programs assume a default base address of 388H.



**Testing the Hardware**

The Ad Lib Gold Test Program, which is supplied with the Gold software, enables you to verify that the Gold card is functioning properly. These tests are not only used to test the Ad Lib hardware, but also to test the connections to all associated peripherals (MIDI ports, joystick, SCSI, etc.).

**Preparing the Test Program**

Prior to running the Test Program:

1. Make sure that the Gold card is properly installed. If necessary, refer to the previous section, "Installing the Hardware".
2. Connect headphones, a speaker or stereo system to the audio jack.
3. Connect the peripherals you plan to use with the Gold card.
4. Turn on your computer. If it is already on, we recommend resetting it.

**Loading the Test Program**

To load the Test Program:

1. Make the directory where you placed the Gold software the current directory. For example:

C:\>cd gold

2. Load the TEST program by typing the following command at the DOS prompt:

C:\GOLD>test

When the program opens, a first window entitled "Installation Notes" introduces you to the Test Program and its basic commands. Two main buttons are displayed at the bottom of each screen: <Cancel> and <Continue>.

Before each test, the program will explain what the test does. It will also point out the procedure to follow to complete the test. This information is shown at the top of each test screen. To see the whole text, click on the scroll bar with the mouse, or select the scroll bar with the **Tab** key and use the vertical arrow keys (**↓** and **↑**).

If a test does not succeed, a message will appear giving probable causes and solutions.

**To Continue the Test**

The <Continue> button lets the user access the next step of the test.

## Test Program

### Testing the Hardware

#### To Cancel the Test and Exit the Program

When all tests are finished, or anytime within the test procedure, you can exit the Test Program and return to DOS by activating the <Cancel> button.

#### Choosing Test Options

Clicking on the <Continue> button in the Installation Notes introductory screen opens the Selection Panel dialog box (see Figure 6). This dialog box presents the list of the tests you can execute:

- Configuration
- Joystick
- Audio
- MIDI Interface
- SCSI Interface
- Sampling and Playback
- Timers
- Telephone
- FM Sound
- Mixer

Some of these options may be grayed to indicate that they are disabled depending on the available hardware. Checking off any of these check boxes will let you access the corresponding tests.

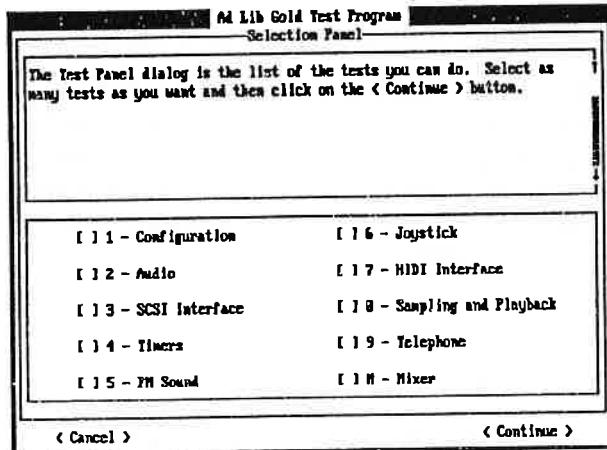


Figure 6: Test Program's Selection Panel

The options can be selected by using either the mouse, the Tab key (**Tab**) or (**Shift**-**Tab**) or the arrow keys (**→** or **←**). Select the option you wish to test and then activate the <Continue> button. You can also test several options in a row by selecting the options you want and activating the <Continue> button. Any test can be executed more than once if desired.

Each test panel displays information on the test currently being performed and describes the problems and solutions which may be encountered during the test.

The Ad Lib Gold cards (models 1000 and 2000) have an on-board analog mixer that permits the volume of different audio sources to be controlled, as well as overall output volume, balance and tone. These features can be accessed using the Mixer Panel TSR.

The Mixer Panel TSR is a program that allows you to set the different sound parameters of the Ad Lib Gold card at anytime and from within any application. This memory resident program includes four different control windows:

1. Sound Parameters
2. Sub Mixer
3. Surround Features
4. Activation and Volume Keys

? *TSR stands for Terminate-and-Stay Resident program, which is also called memory resident program. It is a utility program designed to remain in the computer's memory at all times after loading so the user can activate it with a keystroke at any time, even while running another program. For more information on TSRs, see Appendix D.*

---

#### Loading the Mixer Panel TSR

---

To load the Mixer Panel TSR, set the current directory to the one where you placed the Mixer Panel at installation and type the following command:

**mixer**

\* *NOTE: This command can be placed in a batch file so that it is loaded automatically. See your DOS user guide for details.*

When the program is loaded, it will display a message indicating that the program has been successfully loaded. It will also indicate which keys must be used to activate the Mixer Panel.

The Mixer Panel window will not open upon loading and has to be activated as explained in the next section. If you want the Mixer Panel window open upon loading the program, you can use the option "/a" with the loading command. To do this, go to the appropriate directory and type the following at the DOS prompt:

**mixer /a**

**Activating the Mixer Panel**

**Alt** - **Space** - **M** are the default activation keys, but you may change this combination, as explained in the "Activation and Volume Keys" section. To activate the Mixer Panel, press all of the activation keys down at the same time and release them. Upon releasing the keys, the main Mixer Panel window will appear as shown in Figure 7.

This TSR can be activated at any time with applications supporting the Ad Lib Gold. The screen will be returned to its original state and mode when you exit the Mixer Panel window.

\* *NOTE: If you use a Hercules card with a MGA monitor, the Mixer Panel can be activated only in text mode. If you activate the Mixer Panel while in graphics mode, this may cause problems with your system.*

**Using the Mixer Panel**

Each window of the Mixer Panel displays the different parameters and options of the Gold card (see Figure 7). To set one of these:

1. Select the item you want using the vertical arrow keys (**↓** and **↑**).
2. After this, you can modify the chosen item in one of the following ways:

- increase or decrease the numerical parameters and option words one step at a time using the horizontal arrow keys (**→** and **←**);
- increase or decrease the numerical parameters ten steps at a time using Shift with the horizontal arrow keys (**Shift**-**→** and **Shift**-**←**);
- toggle On/Off parameters using the Space bar.

**Sound Parameters**

When the program is activated, you will see a window appear for setting the basic sound parameters of the Gold card.

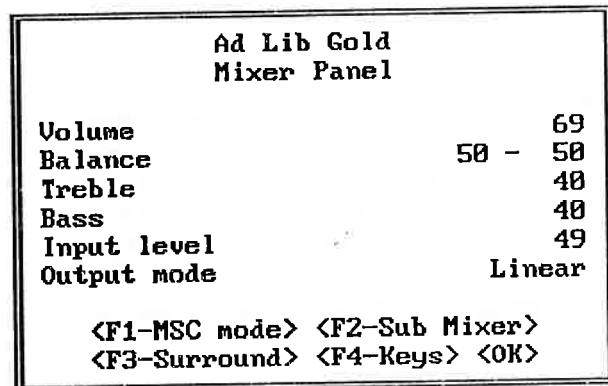


Figure 7: The main Mixer Panel window

**volume**

Sets the master output volume of the board.

**Balance**

Sets the relative volume of the two stereo channels. Setting the right channel will automatically set the left channel; when you increase the right by one unit, the left channel decreases by one unit, and vice versa.

**Treble**

Sets the relative loudness of the high frequencies of the sound.

**Bass**

Sets the relative loudness of the low frequencies of the sound.

**Input level**

Sets the gain (input level) of the external auxiliary source and of the microphone.

**Output mode**

Sets the output mode of the audio source to one of four options:

- Linear: without any effect on the audio source;
- Pseudo: pseudo stereo effect that can be applied when the source is mono;
- Mono: forced mono effect that can be applied when the source is stereo;
- Spatial: light surround sound effect that can

be applied when the source is stereo.

The default setting is Linear.

**<F1-MSC mode/Gold mode>**

Resets the Gold card so it is compatible with the original Ad Lib Music Synthesizer Card. This might be necessary in cases where a third party application that does not properly put the Gold card in its default mode for full compatibility with the original Ad Lib card.

**<F2-Sub Mixer>**

Opens the Sub Mixer window.

**<F3-Surround>**

Opens the Surround Features window (when using the add-on Surround Sound Module).

**<F4-Keys>**

Opens the ALT-SHIFT Keys (short cuts) window.

**<OK> ( or )**

Closes the Mixer Panel main window and returns to the current application saving the changes you made in the settings.

**Sub Mixer**

Activating the **F2** key when in the Mixer Panel main window will open the Sub Mixer control window. Sub Mixer parameters allow the output volume from the five different audio sources to be controlled:

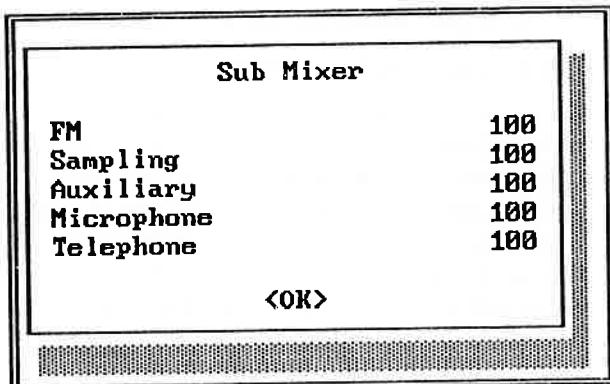


Figure 8: The Sub Mixer control window

**FM**

Sets the output volume of the FM source.

**Sampling**

Sets the output volume of the Sampling source.

**Auxiliary**

Sets the output volume of the auxiliary source (external or internal).

**Microphone**

Sets the output volume of the microphone.

**Telephone**

Sets the output volume of the telephone (when using the add-on Telephone Module).

<OK> ( or )

Closes the Sub Mixer control window and returns to the main Mixer Panel window saving the changes you made in the Sub Mixer parameter settings.

**Surround Features**

Activating the  key when in the Mixer Panel main window will open the Surround Features control window.

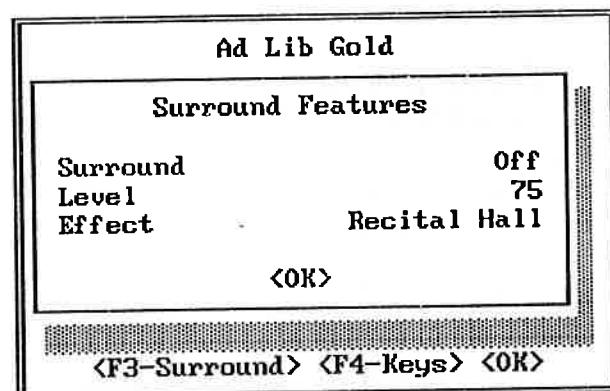


Figure 9: The Surround control window

\* NOTE: If the Surround Sound Module is not installed, the program will display the message "OPTION NOT INSTALLED" and the changes you make to the parameters will have no effect (Fig. 10).

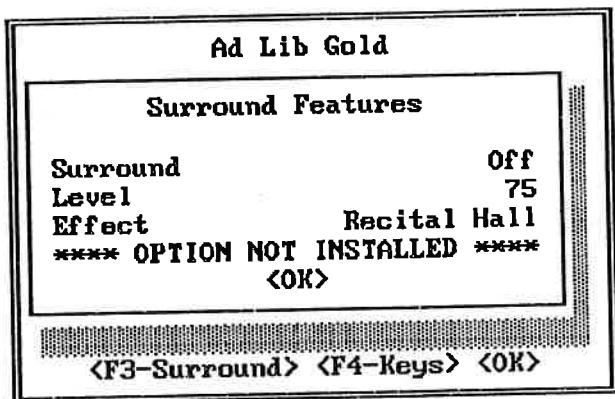


Figure 10: The Surround control window when option not installed

#### **Surround**

A toggle On/Off allows the Surround Sound option to be enabled or disabled. The default setting is Surround Off.

#### **Level**

Sets the level of the surround sound effect produced by the Surround Sound Module.

#### **Effect**

Sets the type of surround sound effect you want to enhance the sound ambience selected from a variety of presets.

<OK> (<Esc> or <Esc>)

Closes the Surround Features control window and returns to the main Mixer Panel window saving the changes you made in the parameter settings.

#### **ALT-SHIFT Keys (Short Cuts)**

In order to avoid conflicts with other programs, you may change the last key in the combination of keys used to activate the Mixer Panel, to set the master volume and to turn On and Off the Surround Sound. Activating the **F4** key when in the Mixer Panel main window will open the Keys window.

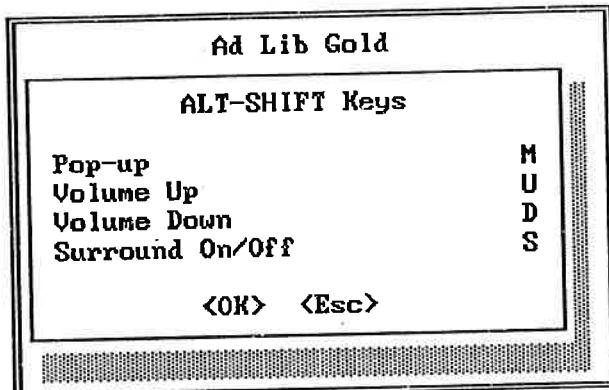


Figure 11: The ALT-SHIFT Keys window

**Pop-up**

Sets the key combination used to activate the main Mixer Panel window. The default combination is **Alt**-**Shift**-**M**.

**Volume Up**

Sets the key combination used to increase the master volume one unit at a time. The default combination is **Alt**-**Shift**-**U**.

**Volume Down**

Sets the key combination used to decrease the master volume one unit at a time. The default combination is **Alt**-**Shift**-**D**.

**Surround on/off**

Sets the key combination used to enable and disable the surround sound effect. The default combination is **Alt**-**Shift**-**S**.

**<OK> (**P**)**

Closes the Keys window and returns to the main Mixer Panel window saving the changes you made in the control keys.

**<Esc> (**Esc**)**

Closes the Keys window and returns to the main Mixer Panel window canceling the changes you made in the control keys.

**Closing the Mixer Panel**

When in the main window of the Mixer Panel, press the **Q** or **Esc** key to leave the program and return to where you were when the Mixer Panel was activated.

**Removing the Mixer Panel TSR**

When the Mixer Panel TSR is already installed but you do not wish to use it, you can unload the program with the option "/r". This option removes the Mixer Panel TSR from the computer's memory. To remove the Mixer Panel TSR, go to the appropriate directory and type the following command:

**mixer /r**

Once this command is entered, the program will display a message indicating that the Mixer Panel has been removed.

Juke Box Gold is a music playback program specially designed to demonstrate the sound capabilities of the Ad Lib Gold card itself. It enables you to play pre-programmed songs, or those you create yourself using the Visual Composer Gold music composition program (sold separately). Selected songs can also be played at any time while other applications are running, with the use of the ROL2 Playback TSR commands.

#### Loading Juke Box Gold

To load Juke Box Gold, set the current directory to the one in which you placed Juke Box Gold at installation and type:

```
jukegold
```

Once the program is loaded, the main Juke Box Gold window will appear as shown in Figure 12.

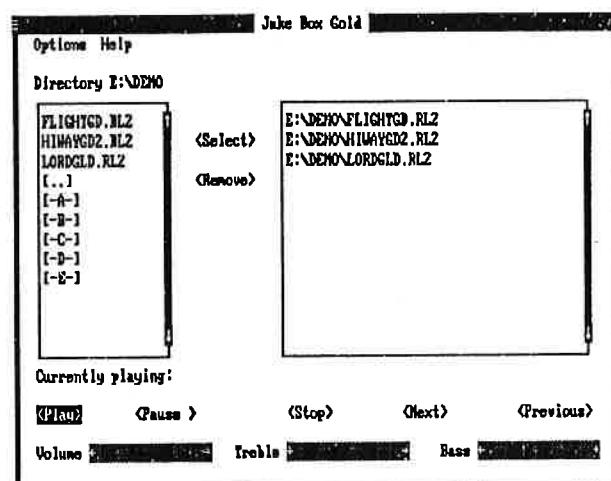


Figure 12: The main Juke Box Gold window

This window displays the various menu titles and command buttons (see Figure 12). Other possible options are contained in the menus. To activate a menu, use one of the following methods:

1. Using the Tab key: Scroll and choose the command you want with the **Tab** key.
2. Using the keyboard shortcuts: To activate the menu or command you want, press the **Alt** key and the letter highlighted in its name. To activate a command in an open menu, press the highlighted letter.

3. Using a mouse: To activate the menu or command you want, click on the menu or button command with the mouse.

## Selecting Songs

---

### **Creating a Selection of Songs**

The main Juke Box Gold window contains two large boxes for song selection. The box located at the left of the screen displays the contents of the current directory. This is a list of files, subdirectories and drives through which you can navigate by selecting a name and pressing the key, or by double clicking with the mouse. The name of the current directory is displayed above the box. When the program is loaded, the default directory is the directory where you placed Juke Box Gold.

To create a selection of songs, go to any directory containing ROL2 files, highlight the file and activate the Select command, or press , or double click with the mouse, for each song you wish to add to the selection. As each song is selected, its DOS file name will be displayed in the Selection box at the right of the screen in its order of selection. You may select as many songs as you wish (depending on memory capacity), but only from a single directory.

### **To Remove Songs from the Selection**

The box at the right of the screen displays the list of songs contained in the selection you have made. To remove a song from the selection, highlight its name and activate the Remove command.

## Playing Music

---

### **To Play Songs**

To play your selection of songs, activate the Play command. Each song in the list will be played in order.

Once the music begins playing, the name of the song currently playing is displayed at the bottom of the window.

### **To Stop Music Playback**

To stop music playback, activate the Stop command.

### **To Pause and Resume Music Playback**

To pause music playback, activate the Pause command. When the music pauses, the Pause button toggles to Resume.

To continue music playback at the exact place where it stopped, activate the Resume command. Once the music starts up again, the Resume button switches back to Pause.

#### To Scan Songs

To skip to the next song during playback, activate the Next command. This will immediately start the next song if there are any left in the song selection list.

To return to the previous song during playback, activate the Previous command. This will immediately start the previous song.

#### Adjusting the Sound

---

The Ad Lib Gold card has an on-board analog mixer that allows volume and tone controls to be adjusted. These features can be accessed from any application by using the Mixer Panel TSR program (see the section "Mixer Panel TSR"). But you can also adjust the sound directly inside the Juke Box Gold program. Three sliders located along the bottom of the window allow you to adjust volume, bass and treble controls while listening to Juke Box songs.

#### To Adjust the Volume, Bass or Treble

To adjust one of these three parameters, activate the slider you want and use the left and right arrow keys ( $\leftarrow$  and  $\rightarrow$ ) to raise or lower the value of the chosen parameter. You can also scroll the indicator inside a slider with a mouse.

#### To Set the Stereo/Mono Option

The Gold card output can be set either to stereo sound (distinctive signals for the left and right channels) or mono sound (identical signals from both channels). To change from one to the other, choose the Stereo or the Mono option from the Options menu.

#### Asking for Help

---

When you choose the Help command, a window opens up on the screen containing summarized information on how to operate Juke Box Gold and how to use the various features.

Exiting the Program

Exiting the Program

To leave the program and return to DOS, use one of the following methods:

- Activate the Exit command from the Options menu.
- OR
- Click on the System menu box at the upper left corner of the window and activate the Close command.

Using the ROL2 Playback TSR

The music files played by Juke Box Gold are called ROL files (.ROL or .RL2). In order to play these music files, the application uses a TSR driver, which we refer to as ROL2 Playback TSR.

Since TSRs stay in memory while we use other programs, the ROL2 Playback TSR allows you to play the songs previously selected in Juke Box Gold, while using other applications.

For details on the loading options of the ROL2 Playback TSR, see the section 4.9 "ROL2 Playback TSR".

The playback commands of this TSR can be used at any time by the following key combinations:

<b>[Alt]</b> - <b>[Space]</b> - <b>P</b>	Plays the selected songs.
<b>[Alt]</b> - <b>[Space]</b> - <b>R</b>	Pauses and resumes the music playback.
<b>[Alt]</b> - <b>[Space]</b> - <b>T</b>	Stops the music playback.
<b>[Alt]</b> - <b>[Space]</b> - <b>N</b>	Skips to the next song from the selection.
<b>[Alt]</b> - <b>[Space]</b> - <b>V</b>	Returns to the previous song from the selection.

In order to avoid conflicts with other programs, you may change the last key in the combination of keys used to activate the above commands by using the Setup program.

**!** *WARNING: Do not use the ROL2 Playback TSR while running other music applications, as this will cause conflicts with the ROL2 Playback driver.*

**Loading Instrument Maker Gold**

To load Instrument Maker Gold, type the following command at the DOS prompt:

**INSGOLD**

OR

**ED /BBANKNAME.BNK**

Where "BANKNAME" is the name of the bank.

**Using Menu Commands****[F5] File****New**

Opens a new empty sound patch.

**Open...**

Opens an existing sound patch.

**Close**

Closes the current sound patch.

**Save**

Automatically saves changes made to sound patch.

**Save As...**

Saves the current sound patch under a new name (maximum of 11 characters).

**Delete...**

Deletes an existing sound patch.

**Read "opl3.txt"****Save "opl3.txt"****Debug...**

You do not have to use these commands.

They were implemented for development and will be removed for the final version of Instrument Maker Gold.

**Quit**

Closes all opened sound patches, quits the Instrument Maker Gold application and returns to DOS.

**[F6] Options****Note Select**

You do not have to use this option. It was implemented for development and will be removed for the final version of Instrument Maker Gold.

**AM Depth**

Amplitude Modulation Depth: When this option is checked, it increases the LFO volume modulation (tremolo).

**PM Depth**

Pitch Modulation Depth: when this option is checked, it increases the LFO frequency modulation (vibrato).

**Octave Up**

This command makes the screen keyboard and computer keyboard play an octave higher.

**Octave Down**

This command makes the screen keyboard and computer keyboard play an octave lower.

**[F7] Document**

This menu gives you a fast way to switch between open sound patch documents. It lists all the sound patch documents you have open, to a maximum of ten (including the untitled document). The checkmarked document is the active one on which you can work. An asterisk (\*) placed beside a document name indicates that the document has been modified.

---

**Editing FM Instrument Sounds**

To select a parameter:

1. Click on the chosen parameter with the mouse.
2. Use the arrows to navigate between the

different parameters.

To modify a parameter:

1. Use the Space Bar to increase the value of the chosen parameter one unit at a time.
2. Use Shift-Space Bar to decrease the value of the chosen parameter one unit at a time.
3. Use the Plus Key on the numeric keyboard to increase the value of the chosen parameter one unit at a time.
4. Use the Minus Key on the numeric keyboard to decrease the value of the chosen parameter one unit at a time.

To mute operators:

- The Mute function allows you to disable (turn off) any instrument sound operator, thus making it possible to work on individual operators and listen separately to each as you change the parameters. Use the [F1], [F2], [F3], and [F4] keys to mute operators 1, 2, 3, and 4 respectively.

NOTE: In the supplied sound bank, the two-operator sound names begin with a capital letter and end with a "#", while four-operator sound names begin with a lower case letter.

Loading Sample Maker

To load Sample Maker, go to the appropriate directory and type the following command at the DOS prompt:

SAMPL

Using Menu Commands**F5 File****New**

Opens a new empty sampled sound.

**Open from Bank...**

Opens an existing sampled sound in ADPCM format from the bank.

**Save to Bank As...**

Saves the current sampled sound in ADPCM format under a new name in the bank.

**Delete from Bank...**

Deletes an existing sampled sound in ADPCM format from the bank.

**WARNING:** All sampled sounds saved to bank are temporarily limited to 64 K. Furthermore, the format of sampled sounds saved to bank will change in the next development versions. For these reasons, we recommend that you do not use the commands related to a bank.

**Compact Bank**

You do not have to use this command. It was implemented for development and will be removed for the final version of Sample Maker.

**Open File...**

Opens an existing sampled sound in PCM format (.SMP file) from the current directory.

**Save File As...**

Saves the current sampled sound in PCM format (as .SMP file) in the current directory.

**Open Sample Vision File...****Open Lyre File...**

You do not have to use these commands.

They were implemented for development and will be removed for the final version of Sample Maker.

**Quit**

Closes the displayed sampled sound, quits the Sample Maker application and returns to DOS.

**F6 Edit****Copy**

Copies the selected section of the sampled sound and puts it into a memory buffer.

## **Sample Maker**

### **Using Menu Commands**

#### **Cut**

Deletes the selected section from the sampled sound and puts it into a memory buffer.

#### **Paste**

Inserts a copy of the buffer's contents at the point where the cursor is positioned in the displayed sampled sound.

#### **Clear**

Deletes the selected section from the sampled sound, but, unlike the command Cut, does not put it into the buffer.

#### **Clear All**

Deletes the entire sampled sound from the screen.

#### **F1 Sampling**

##### **Record**

This command enables any audio signals mixed with the Gold card to be recorded and sampled with Sample Maker. The sampling process will follow the sampling parameters defined within the Sampling Params... dialog.

##### **Play**

This command lets you listen to the opened sample sound. When a section of the sampled sound is selected, you will only hear that section played.

#### **Sampling Params...**

Use this command to determine the value of the various parameters related to the sampled sound you are working on.

#### **F8 Options**

##### **Scope Mode**

This command makes Sample Maker's screen display the sampled signal received at the input as a scope.

**NOTE:** When in Scope Mode, only the sampling frequency (in PCM) is an effective parameter in the "Sampling Params..." dialog.

##### **Scale Up**

Each time you use this command, Sample Maker zooms the displayed sampled sound out horizontally by a ratio of 1 to 1/2.

##### **Scale Down**

Each time you use this command, Sample Maker zooms the displayed sampled sound in horizontally by a ratio of 1 to 2.

##### **Scale Reset**

Resets a zoomed out sampled sound to the original 1 to 1 scale.

**ADPCM File Format**

This command is grayed out because you do not have to use it. The PCM to ADPCM file format converter is not completely implemented at this time.

**Gen. Example**

This command automatically generates a sampled sin wave and pastes it onto the screen. You should not use this command. It was implemented for development and will be removed for the final version of Sample Maker.

---

**Important Warnings for this Development Version of Sample Maker**

---

**Sampling Rate Limitations**

1. Due to a limitation of the sampling chip, the sampling rate of 5.5125 kHz does not work in 4-bit PCM format. Do not choose this option, because Sample Maker will then set another sampling rate, which will give unexpected results.
2. Due to a limitation of the sampling chip, the sampling rate of 44.1 kHz does not work in 4-bit ADPCM format. Do not choose this option, because Sample Maker will then set another sampling rate, which will give unexpected results.

- See "Digital Input and Output".

**Sampling Length Limitation**

- Sampled sounds are limited to 256 K. If a recording goes over 256 K, it will clip at 256 K.

**Scope Mode**

- When in Scope Mode, choosing the Record command will make the program lock up.

**Graphic Display In Different PCM Format**

- Even though every PCM format makes Sample Maker record and play correctly, only PCM 8-bit format is displayed correctly on screen.



## Ad Lib Surround Sound Editor

Ad Lib is providing a special application program, the Surround Sound Editor, which allows you to program your own presets for the Surround Sound Module. This program is included within a special version of Juke Box Gold so that you may play back Juke Box songs and listen to changes you make while working in the editor.

## Technical Features

The underlying technology of the Surround Sound Module is a circuit designed as a general-purpose digital processing element. The board's main component is an LSI chip which has quality digital surround sound capabilities made possible through Yamaha's digital audio technology. Each of its eight digital delay lines may provide a delay time of up to 100 milliseconds, and combining delay line signals for two-channel output assures a wide range of applications.

## Opening the Surround Sound Editor

As stated above, the Surround Sound Editor is at this moment included within a special version of Juke Box Gold. So, to open the editor, you first have to load Juke Box Gold.

To load Juke Box Gold with the Surround Sound Editor, set the current directory to the one in which you placed Juke Box Gold during installation and type the following command:

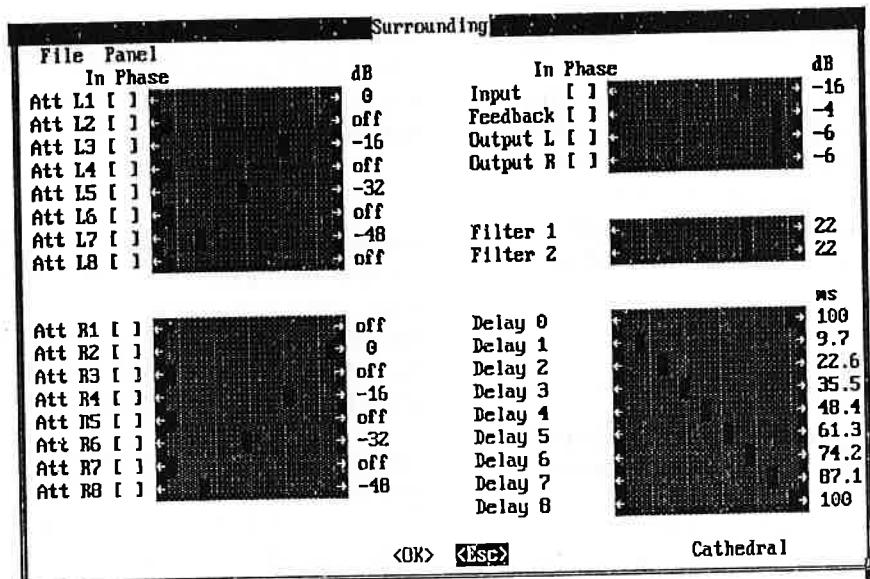
**surround**

Once the program is loaded, the main Juke Box Gold window will appear. You can then select and play back songs as you wish. For complete information on using Juke Box Gold, refer to the "Juke Box Gold Music Playback Program" section of *Ad Lib Gold Pre-Release Evaluation Kit*.

When ready, open the Surround Sound Editor by choosing Surround from the Options menu. Upon opening, the Surround Sound Editor window will appear as shown in the following figure.

## Surround Sound Editor

### Opening the Surround Sound Editor



The Surround Sound Editor window

This window displays the various parameters used to construct a surround sound effect.

### Using the Surround Sound Editor

The Surround Sound Editor window contains five main parts:

- The left channel line attenuation section, located

at the upper left corner of the screen.

- The right channel line attenuation section, located at the lower left corner of the screen.
- The global level and feedback parameter section, located at the upper right corner of the screen.
- The filter parameter section, located in the middle of the right side of the screen.

- The global delay line parameter section, located at the lower right corner of the screen.

### Channel Line Attenuation Sections

These two sections display the two delay line attenuation parameters related to left and right channels.

#### In Phase

When this check box is checked off, means that the delay line output signal is in phase with the input signal. When this check box is not checked off, means that the delay line output signal is phase reversed with the input signal.

#### dB

Displays the attenuation value setting of the delay line, which ranges from -60 decibels to 0 decibels, in steps of 2 dB. A delay line can also be turned off (-∞).

### Global Level and Feedback Parameter Section

This section displays the two global attenuation parameters related to global signal input, feedback output, and left and right channel global outputs.

#### In Phase

When this check box is checked off, means

that the output signal is in phase with the input signal. When this check box is not checked off, means that the output signal is phase reversed with the input signal.

#### dB

Displays the attenuation value setting of the signal, which ranges from -60 decibels to 0 decibels, in steps of 2 dB. A signal can also be turned off (-∞).

### Filter Parameter Section

This section displays the value setting of the two low pass filters for the feedback loop, which ranges from 0 to 31 units, in steps of 1 unit.

### Global Delay Line Parameter Section

#### ms

Displays the time value setting for each of the 8 delay lines (Delay 1 to Delay 8) and the feedback loop delay line (Delay 0), which range from 0 to 100 milliseconds, in steps of approximately 3.2 milliseconds.

**Editing Surround Sound Presets**

To modify a parameter, use one of the following methods:

1. Click on the slide bar indicator of the chosen parameter with the mouse and drag it to the desired value.
2. Click on the gray zone of a slide bar to move the indicator and to decrease or increase the value of the chosen parameter several steps at a time.
3. Click on the left or right arrows at the end of a slide bar to decrease or increase the value of the chosen parameter one step at a time.
4. Click on a check box to turn it On or Off.

**Using Menu Commands**

To activate menu commands, use one of the following methods:

1. Using a mouse: To activate a menu command, click on the menu with the mouse, drag to the command you want and release the mouse button.
2. Using the keyboard shortcuts: To activate a menu command, press the letter highlighted in the menu's name and the **[Alt]** key at the same

time. Then, activate the command you want by pressing the letter highlighted in its name.

**File Menu**

**New**

Opens a new empty surround sound preset with no name.

**Open**

Opens an existing surround sound preset from the bank entitled STANDARD . SRD.

**Save**

Opens a dialog box which allows the current surround sound preset to be saved under a chosen name (maximum of 8 characters) in the bank entitled STANDARD . SRD.

**Text**

This command will save the current preset in text form, in both C and Assembler formats, in the file PRESET . TXT. If PRESET . TXT exists, the text will be appended

**Delete**

Deletes an existing surround sound preset from the bank entitled STANDARD . SRD.

**Panel Menu****Open**

Opens an existing surround sound preset from the Control Panel executable file (CONTROL.EXE).

**Save**

Saves the current surround sound preset in the Control Panel executable file (CONTROL.EXE).

NOTE: This command does not allow the name of the current surround sound preset to be changed.

**Closing the Surround Sound Editor****<OK> (F1)**

Closes the Surround Sound Editor window and returns to Juke Box Gold, temporarily keeping the changes you have just made to the current preset for a further work session.

**<Esc> (Esc)**

Closes the Surround Sound Editor window and returns to Juke Box Gold, without keeping the changes you have just made to the current preset.



**ROL2 Playback Utility**

The ROL2 Playback utility is a small program that allows the user to play RL2 music files from the DOS command line or from a batch file.

The format of the command running the ROL2 Playback utility is the following:

```
playrl2 fileName [/Q]
```

Where *fileName* is the name of the ROL2 music file (.RL2) to be played.

The optional "/Q" parameter can be used to start the playback of the RL2 song file and immediately returns control to DOS. The playback of the song will be taken in charge by the ROL2 Playback memory resident driver.

If you enter "playrl2" alone or with the option "/?" ("playrl2 /?"), the program displays help lines giving summarized information on program parameters.

The ROL2 Playback utility uses the following five drivers, which have to be loaded before running it:

- Control driver (CTRLDRV.EXE)
- FM driver (FMDRV.EXE)
- Wave driver WAVEDRV.EXE)

- Timer driver (TIMERDRV.EXE)
- ROL2 Playback driver (RL2DRV.EXE)

**Digitized Sound Playback Utility**

The Digitized Sound Playback utility is a small program that allows to play back digitized sound files (recorded in the .SMP format) from the DOS command line or from a batch file.

The format of the command running the Digitized Sound Playback utility is the following:

```
playdigi fileName [/p] [/n]
```

Where *fileName* is the name of the digitized sound file (.SMP) to be played.

Where "p" in the option "/p", may be "c" (center), "R" (right), or "L" (left), indicating the stereo position you want for the playback of the digitized sound file.

Where "n" in the option "/n", may be a number from 0 to 100, indicating the volume you want for the playback of the digitized sound file.

If you enter "playdigi" alone or with the option "/?" ("playdigi /?"), the program displays help lines giving summarized information on program parameters.

**Batch File Utilities**

---

**Digitized Sound Playback Utility**

---

The Digitized Sound Playback utility uses the following two drivers, which have to be loaded before running it:

- Control driver (CTRLDRV.EXE)
- Wave driver (WAVEDRV.EXE)

---

4-30

Ad Lib Gold

© Ad Lib Inc. 1992

Confidential

Mon, Mar 30, 1992

Using the ROL2 Playback TSR

The music files played by Juke Box Gold are called ROL files (.ROL or .RL2). In order to play these music files, the application uses a TSR driver, which we refer to as ROL2 Playback TSR.

The ROL2 Playback TSR is used by various applications to control the playback of Ad Lib music files. It is a powerful utility that uses the services of the other underlying drivers to simplify the task of integrating music and digitized sound to applications. (See below for the options supported with this driver.)

Since TSRs stay in memory while we use other programs, the ROL2 Playback TSR allows you to play the songs previously selected in Juke Box Gold, while using other applications.

The playback commands of the ROL2 Playback TSR can be used at any time by the following key combinations:

- [Alt] - [Space] - [P]** Plays the selected songs.
- [Alt] - [Space] - [R]** Pauses and resumes the music playback.
- [Alt] - [Space] - [T]** Stops the music playback.
- [Alt] - [Space] - [N]** Skips to the next song from the selection.



Returns to the previous song from the selection.

In order to avoid conflicts with other programs, you may change the last key in the combination of keys used to activate the above commands by using the Setup program.



**WARNING: Do not use the ROL2 Playback TSR while running other music applications, as this will cause conflicts with the ROL2 Playback driver.**

ROL2 Playback TSR Data Files

The ROL2 Playback TSR uses the a number of data files. All the data files must be in the same directory. Unless the path for these files is specified as a command line argument, the directory containing those files must be the current directory when RL2DRV.EXE is loaded. The data files used by the ROL2 Playback TSR are:

- SAMPLES.BNK and OPL3.BNK: instrument description files (for digitized and FM sounds, respectively).
- SAMPLBNK.EQU: Digitized sounds name translation table.
- \*.SMP: Digitized sound files used in the songs.

**ROL2 Playback TSR Options**

---

The following command-line options can be used with the ROL2 Playback TSR:

**rl2drv /r**

This option removes the ROL2 Playback TSR from the computer's memory when it is installed but you do not wish to use it. Once this command is entered, the program will display a message indicating that the ROL2 Playback TSR has been removed and is no longer loaded.

**rl2drv**

Loads the ROL2 Playback TSR into the computer's memory (RAM).

**rl2drv /vn**

This option disables the specified sampling voice "n", which can be "1" or "2". Use two options consecutively, "/v1 /v2", to disable the two sampling voices.

The playback of sampled voices consumes a lot of computer resources. On slower PCs, or when the ROL2 Playback TSR is used in conjunction with more demanding applications, this option can be used to ensure a proper functioning of all parts involved.

**rl2drv /spath**

This option is used to specify a path for the data files used by ROL2 Playback TSR, if the data files are not in the default directory.

**drivers**

This batch file command loads all Ad Lib Gold drivers.

# **Chapter 5 - DOS Software Drivers**

---

---

<b>Introduction</b>	<b>iii</b>
<hr/>	<hr/>
<b>5.1 Interfacing DOS Drivers with Applications</b>	<b>1</b>
<hr/>	<hr/>
<b>5.2 DOS Control Features Driver</b>	<b>3</b>
Function Directory	4
<hr/>	<hr/>
<b>5.3 DOS FM Synthesis Driver</b>	<b>57</b>
Voice Allocation Structure	57
Function Directory	58
<hr/>	<hr/>
<b>5.4 DOS Wave Driver</b>	<b>71</b>
DOS Wave Driver Functions	71
<hr/>	<hr/>
<b>5.5 DOS Timer Driver</b>	<b>93</b>
Function Directory	95
<hr/>	<hr/>
<b>5.6 DOS MIDI Driver</b>	
(To be released)	
<hr/>	<hr/>
<b>5.7 DOS SCSI CD-ROM Driver</b>	
(To be released)	



# Introduction

Ad Lib supplies memory-resident drivers as part of its end-user software packages. Developers should, when possible, use the services provided by those drivers. There are a number of advantages to using memory-resident drivers:

- A lot of the applications supplied for the Gold Card are TSR applications. Memory-resident drivers provide applications with a common software core for managing shared resources. Digital playback and recording, MIDI input and output and the timers available on the card, for instance, share a same interrupt request line.
- Memory-resident drivers can easily be maintained and updated, independently of the application code.

The method used by applications to interface with the Ad Lib Gold memory-resident drivers minimizes the overhead in calling the driver services. For most applications, calling the drivers services will not introduce a noticeable overhead.

The following drivers are available as part of the developer toolkit.

## DOS Control Features Driver

The DOS Control Features driver supports the mixer features defined in Gold Sound Standard architecture

It also controls the configuration options of the Gold card, such as interrupt line selection, DMA channel allocation and address relocation.

Finally, it acts as a main management layer for all other drivers. It manages interrupt redirection to other drivers and keeps track of the location of the drivers.

For this reason, the Control features driver should always be the first one loaded.

## DOS FM Driver

The DOS FM driver gives access to the FM sound generation features of the YM262 chip.

## **DOS Software Drivers**

---

### **Introduction**

---

### **DOS Wave Driver**

---

The DOS Wave Driver supports the digitized sound playback and recording features of the YMZ263 chip.

### **DOS Timer Driver**

---

The DOS Timer driver supplies routines to control the hardware timers on both the YMF262 and the YMZ263 chips. The timers can be used for high-precision synchronization of events.

### **DOS MIDI Driver**

---

The DOS MIDI driver offers services to input and output data through the YMZ263 MIDI FIFO buffers.

## 5.1

# Interfacing DOS Drivers with Applications

---

Drivers load themselves in memory and hook themselves to the DOS multiplex interrupt 2FH. Once a driver is loaded in memory, it registers itself to the Control features driver. It transmits to the Control features driver the address for an entry point to be used by applications, and an address for an entry point to a routine that will handle interrupts from the Gold card.

There are two ways an application can interface with a driver. By issuing commands through int 2FH, or by directly calling the driver entry-point function, used for command dispatching. The second method is much more efficient.

To directly call the driver entry-point function, an application that wants to use the services of a driver first needs to issue an interrupt 2FH with register AH equal to ADLIB\_MULTIPLEX\_DRIVER\_ID and register AL equal to the GET\_ALL\_ENTRY command (defined in ctrldrv.h). This will return a table containing the entry points for all Gold drivers present in memory.

The application can then communicate with a specific driver just by issuing a FAR call to a specific driver. This call will take as an argument a far pointer to an argument-passing structure which is specific to each driver.

The Developer Toolkit supplies a set of linkable modules that are used to ease the interfacing to the drivers, using the second method of interfacing. The Link modules hide all the complexity of interfacing to the drivers. The application just needs to call the drivers functions as if they were part of a linkable library.

The modules can use the second method of communicating with the drivers. In order to do this, they have to call an initialization function, *InitxxxLink()*. These functions will build up a table of function pointer to accelerate the calling of driver routines.

<b>Driver link module</b>	<b>Description</b>	<b>Link Initialization routine</b>
CTRLLNK	Control Features Driver	InitCtrlLink()
FMLNK	FM Synthesis Driver	InitFMLink()
WAVELNK	Wave Driver	InitWaveLink()
TIMERLNK	Timer Driver	InitTimerLink()
MIDILNK	MIDI Driver	InitMidiLink()

Once the *InitxxxLink()* function is called, applications just need to call the routines described in the following sections.

The source code for the Link modules has been supplied as part of the Developer Toolkit. Developers can use this source to customize Link modules to their version of the C compiler. The source code can also be used as a reference in debugging environments.

---

### **Initialization Sequence**

---

Developers using the Gold drivers should use the following initialization steps in order to insure that their applications do not try to access drivers that are not loaded in memory.

Applications should first check for the presence of the Gold card. They should then make sure that the driver is present by calling the appropriate function.

Once the application has made verified that the driver is loaded in memory, it can call the appropriate *InitxxxLink()* function.

<b>Driver or service</b>	<b>Detection function</b>	<b>Returns</b>
Gold Card Presence	CtGetGoldCardPresence()	0 if the Gold card is not found. 1 If the Gold card is found
Control Features Driver	CtGetDriverPresence()	0 if the driver is not present. 1 if the driver is loaded
FM Synthesis Driver	GetFMDriverStatus()	0xFF if the driver is present
WAVELNK	GetWaveDriverStatus()	0xFF if the driver is present
TIMERLNK	GetTimerDriverStatus()	0xFF if the driver is present
MIDILNK	GetMIDI DriverStatus()	0xFF if the driver is present

## SetControlRegister

### Syntax

```
int SetControlRegister(int reg, WORD val)
```

Sets register 'reg' of Ad Lib Control Chip to 'val'.

### Parameters

**int** reg

Which register to write to.

**WORD** val

Which value to write in register.

### Return value

If no error 0, otherwise 1.

### Comments

This low-level routine handles the details related to accessing the Control Chip, like interrupt disabling and reenabling. It also verifies that no access is made while the Control Chip's RB & SB bits are set.

# **CtStoreConfigInPermMem**

## Syntax

**WORD      CtStoreConfigInPermMem()**

This causes all control chip registers, in their current state, to be written to permanent memory.

## Parameters

None

## Return value

1 if ok. 0 if a problem occurred.

## Comments

None

# **CtRestoreConfigFromPermMem**

## Syntax

**WORD      CtRestoreConfigFromPermMem()**

Restores the Gold crd configuration from permanent memory.

## Parameters

None

## Return value

1 if ok. 0 if a problem occurred

## Comments

None

# CtSetChannel0SampGain

CtSetChannel1SampGain  
CtGetChannel0SampGain  
CtGetChannel1SampGain

## Syntax

<b>WORD</b>	<b>CtSetChannel0SampGain(WORD value)</b>
<b>WORD</b>	<b>CtSetChannel1SampGain(WORD value)</b>
<b>WORD</b>	<b>CtGetChannel0SampGain(WORD value)</b>
<b>WORD</b>	<b>CtGetChannel1SampGain(WORD value)</b>

Sets the gain of sampling channels.

## Parameters

**WORD**      value

Gain value from 0 to 255.

256 different values possible giving a range from approximately 0.04 to 10 times the input value. The exact gain is given by the equation:

Gain = (registerValue \* 10) / 256 Linear gain.

## Return value

1 if ok.

## Comments

None

# CtSetChannelFilter0Mode

CtSetChannel1FilterMode

## Syntax

**WORD**      **CtSetChannel0FilterMode(WORD value)**  
**WORD**      **CtSetChannel1FilterMode(WORD value)**

Sets the antialiasing filters in the proper mode for the channel.

## Parameters

**WORD**      **value**

0 = playback mode, 1 = sample mode

## Return Value

1 if ok.

## Comments

This filter MUST be set in sample mode before sampling.

This filter MUST be set in playback mode before playback.

The Ad Lib Gold card uses the same antialiasing filters during sampling and playback. The appropriate filter mode must be set before any sampling or playback operation.

# **CtGetChannelFilter0Mode**

**CtGetChannel1FilterMode**

## Syntax

<b>WORD</b>	<b>CtGetChannel0FilterMode(void)</b>
<b>WORD</b>	<b>CtGetChannel1FilterMode(void)</b>

Returns the current antialising filter mode for the channel.

## Parameters

None

## Return Value

0: playback mode. 1: Sampling mode

## Comments

None

# **CtStereoMonoAuxSamp**

## Syntax

**WORD      CtStereoMonoAuxSamp(WORD value)**

Forces auxiliary inputs to work monophonically or sterophonically.

## Parameters

**WORD      value**

0 = auxiliary input is stereo, 1 = auxiliary input is mono

## Return Value

1 if ok.

## Comments

The microphone and telephone inputs are monophonic sources and can only be sampled monophonically on channel 0. However, the auxiliary inputs are normally sampled in stereo on both channel 0 and 1 at the same time. This stereo audio input can be turned monophonic and sampled on channel 0 using this function.

# **CtGetStereoMonoAuxSamp**

## Syntax

**WORD      CtGetStereoMonoAuxSamp(void)**

Returns whether the auxiliary inputs are used for monophonic sampling or stereophonic sampling.

## Parameters

None

## Return Value

0 = auxiliary input is stereo, 1 = auxiliary input is mono

## Comments

None

# CtEnabDisabMicroOutput

## Syntax

**WORD      CtEnabDisabMicroOutput(WORD value)**

Enables/disables microphone output.

## Parameters

**WORD      value**

0 = Microphone output enabled, 1 = Microphone output disabled

## Return Value

1 if ok.

## Comments

When using the microphone input and the normal loudspeaker outputs of the audio card, audio feedback could result. In normal mode, microphone output is enabled. When disabled, the microphone signal is cut from the output of the card but sent to the telephone output, eliminating possible causes of feedback.

# **CtGetEnabDisabMicroOutput**

## Syntax

**WORD CtGetEnabDisabMicroOutput()**

When using the microphone input and the normal loudspeaker outputs of the audio card, audio feedback could result. In normal mode, this bit is set to 0. When set to 1, the microphone signal is cut from the output of the card and only sent to the telephone output, eliminating possible causes of feedback.

## Parameters

None

## Return Value

0 = Microphone output enabled, 1 = Microphone output disabled

## Comments

See **CtEnabDisabMicroOutput()**

# **CtEnabDisabInternPcSpeak**

## Syntax

**WORD      CtEnabDisabInternPcSpeak(WORD value)**

Enables/Disables redirection of the PC internal speaker output to the Gold mixer.output

## Parameters

**WORD      value**

0 = Disconnect internal PC speaker,

1 = Connect internal PC speaker

## Return Value

1 if ok.

## Comments

This can enable the PC internal speaker signal to be mixed with the audio signals of a Gold card (directly, without any mixer volume control).

# **CtGetEnabDisabInternPcSpeaker**

## Syntax

**WORD      CtGetEnabDisabInternPcSpeaker()**

Returns the state of redirection of the PC speaker.

## Parameters

None

## Return Value

0 = Internal PC speaker not redirected.

1 = Internal PC speaker redirected

## Comments

None

## CtSelectInterruptLineNbr

### Syntax

**WORD**      **CtSelectInterruptLineNbr(WORD value)**

Selects the interrupt request line used by the audio portion of the Gold hardware.

### Parameters

**WORD**      **value**

0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7

4 = IRQ10, 5 = IRQ11, 6 = IRQ12, 7 = IRQ15

### Return Value

1 if ok.

### Comments

The interrupt line is used by OPL3, MMA and telephone hardware. Valid interrupt lines on an XT are IRQ3, IRQ4, IRQ5 and IRQ7. Valid interrupt lines on an AT are IRQ3, IRQ4, IRQ5, IRQ7, IRQ10, IRQ11, IRQ12 and IRQ15.

# **CtGetInterruptLineNbr**

## Syntax

**WORD      CtGetInterruptLineNbr()**

Returns a number indicating the interrupt line used by the audio portion of the Gold hardware..

## Parameters

None

## Return Value

0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7

4 = IRQ10, 5 = IRQ11, 6 = IRQ12, 7 = IRQ15

## Comments

None

# **CtSelectDMA0ChannelSampChan**

**CtSelectDMA1ChannelSampChan**

## Syntax

**WORD**      **CtSelectDMA0ChannelSampChan(WORD value)**  
**WORD**      **CtSelectDMA1ChannelSampChan(WORD value)**

Allocates DMA channel for the specified MMA sampling channel.

## Parameters

**WORD**      **value**  
0 = DMA 0  
1 = DMA 1  
2 = DMA 2  
3 = DMA 3

## Return Value

1 if ok.

## Comments

Only DMA channels 1, 2 and 3 are available on model Gold 1000. All listed DMA channels are available on the Gold 2000 and 2000MC.

# CtGetDMA0ChannelSampChan

CtGetDMA1ChannelSampChan

## Syntax

```
WORD      CtGetDMA0ChannelSampChan()  
WORD      CtGetDMA1ChannelSampChan()
```

Returns a number indicating the DMA channel used by the specified sampling channel.

## Parameters

None

## Return Value

The sampling channel used.

- 0 = DMA 0
- 1 = DMA 1
- 2 = DMA 2
- 3 = DMA 3

## Comments

None

# CtEnabDisabDMA0SampChan

CtEnabDisabDMA1SampChan

## Syntax

**WORD**      CtEnabDisabDMA0SampChan(WORD value)  
**WORD**      CtEnabDisabDMA1SampChan(WORD value)

Disables or enables use of DMA channel for sampling channel.

## Parameters

**WORD**      value

0 = disable, 1 = enable

## Return Value

1 if ok.

## Comments

None

# **CtGetEnabDisabDMA0SampChan**

**CtGetEnabDisabDMA1SampChan**

## Syntax

<b>WORD</b>	<b>CtGetEnabDisabDMA0SampChan()</b>
<b>WORD</b>	<b>CtGetEnabDisabDMA1SampChan()</b>

Tells if the DMA channel is disabled or enabled for the specified sampling channel.

## Parameters

None

## Return Value

0 = disabled, 1 = enabled

## Comments

None

# CtSetRelocationAddress

## Syntax

**WORD CtSetRelocationAddress(value)**

Set s the base ports address for MMA, OPL3 and control chip.

## Parameters

**WORD value**

New I/O address, divided by 8.

Range is from 0 to 127

## Return Value

1 if ok.

## Comments

None

# **CtGetRelocationAddress**

## Syntax

**WORD      CtGetRelocationAddress()**

Returns the base port addresses for MMA, OPL3 and control chip.

## Parameters

None

## Return Value

New base I/O address, divided by 8.

Range is from 0 to 127

## Comments

None

# CtSetMixerLevelForFMLeft

CtSetMixerLevelForFMRight  
CtSetMixerLevelForLeftSamplePb  
CtSetMixerLevelForRightSamplePb  
CtSetMixerLevelForAuxLeft  
CtSetMixerLevelForAuxRight  
CtSetMixerLevelForMicrophone  
CtSetMixerLevelForTelephone

## Syntax

<b>WORD</b>	<b>CtSetMixerLevelForFMLeft(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForFMRight(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForLeftSamplePb(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForRightSamplePb(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForAuxLeft(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForAuxRight(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForMicrophone(WORD value)</b>
<b>WORD</b>	<b>CtSetMixerLevelForTelephone(WORD value)</b>

Sets the volume for the specified device

## Parameters

**WORD**      **value**

Volume level from 128 to 255 whereis 128 is the minimum, 255 the maximum.

## Return Value

1 if ok.

## Comments

Writing a value less than 128 will result in a signal with negative polarity and should be avoided because the resulting signal may cancel out another signal of opposite polarity.

# CtGetMixerLevelForFMLeft

CtGetMixerLevelForFMRight  
CtGetMixerLevelForLeftSamplePb  
CtGetMixerLevelForRightSamplePb  
CtGetMixerLevelForAuxLeft  
CtGetMixerLevelForAuxRight  
CtGetMixerLevelForMicrophone  
CtGetMixerLevelForTelephone

## Syntax

<b>WORD</b>	<b>CtGetMixerLevelForFMLeft()</b>
<b>WORD</b>	<b>CtGetMixerLevelForFMRight()</b>
<b>WORD</b>	<b>CtGetMixerLevelForLeftSamplePb()</b>
<b>WORD</b>	<b>CtGetMixerLevelForRightSamplePb()</b>
<b>WORD</b>	<b>CtGetMixerLevelForAuxLeft()</b>
<b>WORD</b>	<b>CtGetMixerLevelForAuxRight()</b>
<b>WORD</b>	<b>CtGetMixerLevelForMicrophone()</b>
<b>WORD</b>	<b>CtGetMixerLevelForTelephone()</b>

Returns the volume of the specified device.

## Parameters

None

## Return Value

Volume level from 128 to 255 where 128 is the minimum, 255 the maximum.

## Comments

None

# CtSetOutputVolumeLeft

CtSetOutputVolumeRight

## Syntax

**WORD**      **CtSetOutputVolumeLeft(WORD value)**  
**WORD**      **CtSetOutputVolumeRight(WORD value)**

Sets the final output volume

## Parameters

**WORD**      **value**

Volume level from 0 to 255

## Return Value

1 if ok.

## Comments

There are actually 64 final volume levels. The driver divides the specified value by 4.

# **CtGetOutputVolumeLeft**

**CtGetOutputVolumeRight**

## Syntax

<b>WORD</b>	<b>CtGetOutputVolumeLeft()</b>
<b>WORD</b>	<b>CtGetOutputVolumeRight()</b>

Returns the the final output volume

## Parameters

None

## Return Value

Final output volumefrom 0 to 255

## Comments

There are actually 64 final volume levels. The driver multiplies the specified value by 4 in the return value.the return value may not correspond exactly to the value specified with CTSetOutputVolumeXXX().

# **CtSetOutputBassLevel**

**CtSetOutputTrebleLevel**

## Syntax

**WORD**      **CtSetOutputBassLevel(WORD value)**  
**WORD**      **CtSetOutputTrebleLevel(WORD value)**

Sets the output bass and treble level.

## Parameters

**WORD**      **value**

Range from -128 to 127.

## Return Value

1 if ok.

## Comments

Negative values decreases treble or bass, positive numbers, increase treble or bass. 0 does not alter sound.

# CtGetOutputBassLevel

CtGetOutputTrebleLevel

## Syntax

```
WORD      CtGetOutputBassLevel()  
WORD      CtGetOutputTrebleLevel()
```

Returns the bass or treble level setting.

## Parameters

None

## Return Value

Bass or treble setting, from -127 to 127

## Comments

Since only 4 bits are actually used in the control Chip, the result obtained can differ with the value written using the CtSetOutputBassLevel() and CtSetOutputTrebleLevel function, due to rounding errors.

# CtEnabDisabOutputMuting

## Syntax

**WORD**      **CtEnabDisabOutputMuting(value)**

Disables or enables output muting.

## Parameters

**WORD**      value

0 = disable, 1 = enable

## Return Value

1 if ok.

## Comments

None

# CtGetEnabDisabOutputMuting

## Syntax

**WORD      CtGetEnabDisabOutputMuting()**

Returns a value indicating if output muting is disabled or enabled.

## Parameters

None

## Return Value

0: disabled, 1: enabled

## Comments

None

# CtSelectSCSIInterruptNumber

## Syntax

**WORD**      **CtSelectSCSIInterruptNumber(WORD value)**

Selects an interrupt request line for the SCSI hardware on the Goldcard.

## Parameters

**WORD**      **value**

0 = IRQ3  
1 = IRQ4  
2 = IRQ5  
3 = IRQ7  
4 = IRQ10  
5 = IRQ11  
6 = IRQ12  
7 = IRQ15

## Return Value

1 if ok.

## Comments

Valid interrupt lines on an XT are IRQ3, IRQ4, IRQ5 and, IRQ7.  
Valid interrupt lines on an AT are IRQ3, IRQ4, IRQ5, IRQ7,  
IRQ10, IRQ11, IRQ12 and IRQ15.

# CtGetSCSIInterruptNumber

## Syntax

**WORD      CtGetSCSIInterruptNumber()**

Returns a number indicating the interrupt request line used by the SCSI hardware on the Gold card.

## Parameters

None

## Return Value

Interrupt request line:

- 0 = IRQ3
- 1 = IRQ4
- 2 = IRQ5
- 3 = IRQ7
- 4 = IRQ10
- 5 = IRQ11
- 6 = IRQ12
- 7 = IRQ15

## Comments

None

# CtEnabDisabSCSIIInterrupt

## Syntax

**WORD**      **CtEnabDisabSCSIIInterrupt(value)**

Disables or enables interrupt from SCSII.

## Parameters

**WORD**      value

0 = disable, 1 = enable

## Return Value

1 if ok.

## Comments

None

# **CtEnabDisabSCSIDMA**

## Syntax

**WORD      CtEnabDisabSCSIDMA(value)**

Disables or enables DMA transfers on SCSI hardware.

## Parameters

**WORD      value**

0 = disable, 1 = enable

## Return Value

1 if ok.

## Comments

None

# **CtGetEnabDisabSCSIInterrupt**

## Syntax

**WORD      CtGetEnabDisabSCSIInterrupt()**

Returns 1 if interrupts are enabled on the SCSI hardware.

## Parameters

None

## Return Value

0: Interrupts are disabled  
1: Interrupts are enabled

## Comments

None

# **CtGetEnabDisabSCSIDMA**

## Syntax

**WORD      CtGetEnabDisabSCSIDMA()**

Returns 1 if DMA transfers are enabled on the SCSI hardware.

## Parameters

None

## Return Value

0: DMA is disabled  
1: DMA is enabled

## Comments

None

# CtSelectSCSIDMACHannel

## Syntax

**WORD**      **CtSelectSCSIDMACHannel(WORD value)**

Assigns a DMA channel to the SCSI hardware of the Gold Card.

## Parameters

**WORD**      **value**

0 = DMA 0

1 = DMA 1

2 = DMA 2

3 = DMA 3

## Return Value

1 if ok.

## Comments

Valid DMA channels are 0 - 3. Other channel numbers are reserved for future extensions.

## CtGetSCSIDMACHannel

### Syntax

**WORD      CtGetSCSIDMACHannel()**

Returns the number of the DMA channel Assigned to the SCSI hardware of the Gold card.

### Parameters

None

### Return Value

0 = DMA 0  
1 = DMA 1  
2 = DMA 2  
3 = DMA 3

### Comments

None

# CtSetSCSIRelocationAddress

## Syntax

**WORD**      **CtSetSCSIRelocationAddress(value)**

Sets the base port address addresses for SCSI controller.

## Parameters

**WORD**      **value**

New base I/O address divided by 8.

Range from 0 to 127.

## Return Value

1 if ok.

## Comments

None

# **CtGetSCSIRelocationAddress**

## Syntax

**WORD      CtGetSCSIRelocationAddress()**

Returns the base port address for SCSI controller.

## Parameters

None

## Return Value

New base I/O address divided by 8.  
Range from 0 to 127.

## Comments

None

# CtSetHangUpPickUpTelephoneLine

## Syntax

**WORD**      **CtSetHangUpPickUpTelephoneLine(WORD value)**

Hangs up or picks up telephone.

## Parameters

**WORD**      **value**

  0 = Disconnect telephone line,  
  1 = Connect telephone line

## Return Value

1 if ok.

## Comments

None

# **CtGetHangUpPickUpTelephoneLine**

## Syntax

**WORD      CtGetHangUpPickUpTelephoneLine()**

Returns a value telling if the telephone line is on-hook or off-hook.

## Parameters

None

## Return Value

0: telephone line is on-hook (not connected)  
1: telephone line is off-hook (connected)

## Comments

None

# **CtSelectOutputSources**

## Syntax

**WORD      CtSelectOutputSources(value)**

Selects final output mixing redirection.

## Parameters

**WORD      value**

0 = left mixer channel to left output & right mixer channel to right output,

1 = left mixer channel to both left and right outputs,

2 = right mixer channel to both left and right outputs.

## Return Value

1 if ok.

## Comments

On the Adlib Gold cards, mixing and volume control is performed in two stages. First, all sources are sent to a stereo mixer. Then, the stereo output of the mixer is fed into the final volume control circuitry. The final left and right outputs can be mixed in the fashion described above.

# **CtGetOutputSources**

## Syntax

**WORD      CtGetOutputSources()**

Returns the final mixer redirection mode.

## Parameters

None

## Return Value

0 = left mixer channel to left output & right mixer channel to right output,  
1 = left mixer channel to both left and right outputs,  
2 = right mixer channel to both left and right outputs.

## Comments

None

# **CtSelectOutputMode**

## Syntax

**WORD      CtSelectOutputMode(value)**

Controls the effect applied to the final output .

## Parameters

**WORD      value**

0 = Forced mono,

1 = linear stereo,

2 = pseudo stereo,

3 = spatial stereo.

## Return value

1 if ok.

## Comments

Linear stereo is ordinary, with no effects added. The spatial and pseudo-stereo effects will be useful primarily when the original source is monophonic.

# **CtGetOutputMode**

## Syntax

**WORD      CtGetOutputMode()**

Returns the effect applied to the final output .

## Parameters

None

## Return value

0 = Forced mono,  
1 = linear stereo,  
2 = pseudo stereo,  
3 = spatial stereo.

## Comments

None

# **GetControlRegister**

## Syntax

**WORD      GetControlRegister(reg)**

Returns value stored on register 'reg' of Ad Lib Control Chip.

## Parameters

**int            reg**

Which register to read from.

## Return value

Returns the WORD at the register position.

## Comments

None

## CtGetBoardIdentificationCode

### Syntax

**WORD      CtGetBoardIdentificationCode()**

Returns the board identification code.

### Parameters

None

### Return value

Board identification code:

- 0 -            Gold 2000,
- 1 -            Gold 1000,
- 2 -            Gold 2000 MC.

### Comments

None

# **CtGetBoardOptions**

## Syntax

**WORD      CtGetBoardOptions()**

Returns a bit pattern indicating the options present on boardpresent

## Parameters

None

## Return value

Bit 0-3 (0 = not present, 1 = installed)

bit 0 - Telephone,  
bit 1 - Surround,  
bit 2 - SCSI,  
bit 3 - Currently unused

## Comments

None

# CtGetControllerStatus

## Syntax

**WORD CtGetControllerStatus()**

Returns the interrupt controller status.

## Parameters

None

## Return value

bit 0 -	equals 1 when an OPL3 interrupt is pending,
bit 1 -	equals 1 when an MMA interrupt is pending,
bit 2 -	equals 1 when an telephone interrupt is pending,
bit 3 -	equals 1 when a SCSI interrupt is pending,
bit 6 -	equals 1 when the Control Chip is currently, occupied writing a value to the Mixer Chip or the Volume Control Chip.
bit 7	Set to 1 when the Control Chip is busy writing its internal registers to the external EEPROM chip. This bit must be polled after activating the "Store configuration" sequence to make sure that the Control Chip is free to proceed with another operation.

## Comments

Bit 7 and Bit 6 are polled by all set functions, prior to writing to the registers,  
to make sure that the Control Chip is free to proceed with another operation.

# CtGetRingTelephoneStatus

## Syntax

**WORD      CtGetRingTelephoneStatus()**

Gets telephone status.

## Parameters

None

## Return value

bit 0:      "Ring signal" (0 = no ring, 1 = ring)

## Comments

None

# CtGetInterruptRoutine

## Syntax

**WORD      CtGetInterruptRoutine()**

This routine returns the corresponding interrupt number associated with the interrupt request line used by the audio section.

## Parameters

None

## Return value

Corresponding interrupt number

## Comments

Useful utility mostly used when setting interrupt vectors.

# **CtGetGoldCardPresence**

## Syntax

**WORD      CtGetGoldCardPresence()**

Checks for Gold card presence.

## Parameters

None

## Return value

1 if any Gold card is found. 0 if no Gold card is found.

## Comments

None

# CtGetDriverPresence

## Syntax

**WORD CtGetDriverPresence()**

Checks for Ad Lib Gold Control Driver.

## Parameters

None

## Return value

1 if the Gold Control driver is found. Returns 0 otherwise.

## Comments

None

# CtProgramSurroundPreset

## Syntax

**WORD**      **CtProgramSurroundPreset(ptrData)**

This routine will store a preset into the surround module.  
The preset is defined by a 32 bytes array passed as argument.

## Parameters

**BYTE**      **\*ptrData**

Pointer to the array of 32 bytes.

## Return value

0 if no error, otherwise 1, no surround module.

## Comments

The 1 bytes of the Surround Preset are a 1 to 1 image of the 32 registers of the Surround processor.



The Ad Lib Gold FM Synthesis Driver offers services to access features of the OPL3 FM Chip.

## Voice Allocation Structure

---

The OPL3 chip contains 36 operators which can be combined in various ways to create 1-, 2- or 4-operator voices. (You may wish to refer to the "FM Driver Voices" table on the next page for the purposes of this discussion.)

The 4-operator voices offer the richest sound. Up to six 4-operator voices can be used simultaneously. In the FM Driver, the 4-operator voices are numbered 0, 2, 4, 6, 8 and 10. By default, all six 4-operator voices are enabled. They may be selectively disabled, thus creating two 2-operator voices.

In the FM Driver, when 4-operator voice  $x$  is disabled, the two 2-operator voices are numbered  $x$  and  $x+1$ . For example, if 4-operator voice #2 was disabled, the resulting 2-operator voices will be numbered 2 and 3.

Use Set4OpMaskOPL3() to determine the grouping of the units in either 2 operator or 4 operator voices.

Six of the chip's operators can only be used as three 2-operator voices. These three voices are numbered 12, 13 and 14.

The configuration of the remaining 6 operators depends on whether the card is in melodic or percussive mode. In melodic mode, these 6 operators are configured as three 2-operator voices: driver voice numbers 15, 16 and 18. In percussive mode, the 6 operators are used to create one 2-operator voice (the bass drum) and four 1-operator voices (the remaining drum sounds). The percussive voices are driver voice numbers 15 through 19.

Use SetPercModeOPL3() to configure this section in the melodic or percussive mode.

## DOS FM Synthesis Driver

### Voice Allocation

4 operator voice number	2 operator voice number	Percussive voice number
0	0, 1	-
2	2,3	-
4	4,5	-
6	6,7	-
8	8,9	-
10	10,11	-
-	12	-
-	13	-
-	14	-
-	15	15 (BD)
-	16	16 (HH)
-	-	17 (SD)
-	18	18 (TOM)
-	-	19 (CYMB)

FM Driver Voices

### Function Directory

The following section is an alphabetically arranged definition of all the functions available in the FM Synthesis Driver.

## InitOPL3

### Syntax

```
void InitOPL3(address)
```

Initializes the FM Chip.

### Parameters

<b>WORD</b>	address
-------------	---------

Port address of the FM chip.

### Comments

After initialization, percussion voices are available and all 4 op-voices are enabled.

# LeftRightOPL3

## Syntax

```
void LeftRightOPL3(voiceNum, leftRight)
```

Modifies the stereo position of the voice.

## Parameters

**int** voiceNums

VoiceNumber between 0 and 19.

**int** leftRight

Position of the specified voice:

0: Center.

1: Left.

2: Right.

## LevelOPL3

### Syntax

```
void LevelOPL3(voiceNum, level)
```

Specify the individual volume for a voice.

### Parameters

int                    voiceNum

Voice number between 0 and 19

int                    level

Volume for the voice.

This is an integer number between 0 and 127.

Volume scaling is linear.

### Comments

The volume is scaled linearly by the driver software.

## **NoteOffOPL3**

### Syntax

```
void NoteOffOPL3(voiceNum)
```

Starts the decay of the timbre currently playing on the voice.

### Parameters

**int                    voiceNum**

VoiceNumber between 0 and 19.

## NoteOnOPL3

### Syntax

```
void NoteOnOPL3(voiceNum, note)
```

Starts playing a note on the specified voice.

### Parameters

int                    voiceNum

VoiceNumber between 0 and 19.

int                    note

MIDI value for the note played, in the range 12-107.

### Comments

If a note is already playing on the specified voice, the frequency of the voice will be modified. However, the attack for the timbre will not be heard. To reattack the timbre on the specified voice, a NoteOffOPL3 must be issued.

# PitchbendOPL3

## Syntax

```
void PitchBendOPL3(voiceNum, pitchBend)
```

Modifies the pitch bend scaling factor for the melodic voice.

## Parameters

**Int** voiceNum

Melodic voiceNumber between 0 and 15.

**WORD** pitchBend

Pitch bend scaling factor within the range set in SetGlobalOPL3().

The pitch bend scaling factor is a 14 bit unsigned value. 0 is the maximum negative pitch bend, 0x2000 is no bend and 0x3FFF is the maximum positive pitch bend.

## Comments

Percussive voices cannot be bent.

## PresetOPL3

### Syntax

```
void PresetOPL3(voiceNum, timbrePtr)
```

Assigns a patch to the specified voice.

### Parameters

**int** voiceNum

voiceNumber between 0 and 19

**struct TIMBRE \*timbrePtr**

pointer to a description (28 bytes) of the patch assigned to the voice.

### Comments

If a 4 operator description is sent to a 2-op voice, only the first two operators are considered.

Appendix A: FM Patch format further describes the structure pointed to by timbrePtr.

# **QuitOPL3**

## Syntax

```
void QuitOPL3()
```

Resets the FM chip in the compatible mode.

## Parameters

None.

## Comments

This should be called by all applications prior to leaving, in order to put the OPL3 chip back in the Ad Lib compatible mode.

## Set4OpMaskOPL3

### Syntax

```
void Set4OpMaskOPL3(mask)
```

Enables or disables 4-op voices.

### Parameters

<b>WORD</b>	mask
-------------	------

Bit mask of enabled 4-op voices (in bits 0-5).

Bits 0-5 of mask specify whether the corresponding voice is in 4-op mode (bit set to 1) or in 2-op mode (bit cleared to 0).

Bit 0 corresponds to voice 0 (0-1 in 2 op), bit 1 to voice 2 (2-3 in 2 op) etc.  
(See to table 1 in the Voice Allocation section of this document).

### Comments

There is a maximum of 6 4-op voices.

# SetGlobalOPL3

## Syntax

```
void SetGlobalOPL3 (noteSelectEnable, amplitudeModEnable,  
vibDepthEnable, pitchBendRange)
```

Modifies global operating parameters of the OPL3.

## Parameters

**BOOL** noteSelectEnable

For future use. Set to 0 for now.

**BOOL** amplitudeModEnable

When non-zero, enables amplitude modulation for all timbres that have an amplitude modulation defined.

**BOOL** vibDepthEnable

When non-zero, enables vibrato for all timbres that have a vibrato depth defined.

**int** pitchBendRange

Range of the pitch bend in semitones.

Integer between 0-12.

## **SetPercModeOPL3**

### Syntax

```
void SetPercModeOPL3(newState)
```

Sets the OPL3 in melodic or percussive mode.

### Parameters

<b>BOOL</b>	<b>newState</b>
-------------	-----------------

True for percussive mode, false for melodic mode.

### Comments

If newState is true, disables melodic voices 15-18 and enables percussive voices 15-19 instead.

If newState is false, melodic voices 15-18 are enabled in place of percussive voices 15-19.



The Ad Lib Wave Driver is a high level software interface to the sampling hardware of the Gold Card. Its interface is inspired by the Microsoft Multimedia Wave Driver specifications. But in order to support the target hardware and software more efficiently, some adaptations were necessary. The main differences are:

- The support of ADPCM as well as PCM formats.
- The support of a stereo sample format.
- The control of multiple transfer modes from memory to hardware (polling, interrupt, DMA). (This implies an extension of the WaveFormat structure to include the new parameters.)
- The use of a callback function as a message-passing mechanism between the application and the driver during waveform recording and playback.

Some syntactical differences were introduced in the naming of functions and structures, in order to respect the Ad Lib naming conventions already in use. Please note that this specification is a preliminary document and is incomplete. More functions will be added to this preliminary specification.

The Wave Driver will first be available as a linkable library of functions. It will also be made available to developers in the form of a memory-resident driver, interacting with applications via an interrupt-driven protocol.

---

### DOS Wave Driver Functions

---

The following section is an alphabetically arranged definition of all the functions available in the Wave Driver.

# **InitWaveDriver**

## Syntax

```
void InitWaveDriver()
```

Initializes the wave driver.  
It is to be called only once by the application.

## Parameters

None

## Return value

None

## QuitWaveDriver

### Syntax

**Word QuitWaveDriver ()**

This function resets the driver. **IMPORTANT:** This must be called before returning to the DOS.

### Parameters

**None**

### Return value

**None**

# WaveInAddBuffer

## Syntax

**Word** **WaveInAddBuffer** (**hWaveIn**, **lpWaveInHdr**, **wSize**)

Sends a buffer to a waveform input device. When the buffer is full, the application is notified.

## Parameters

**HWaveIn**            **hWaveIn**

Specifies a handle to the waveform device which is to receive the buffer.

**LpWaveHdr**        **lpWaveInHdr**

Specifies a far pointer to a **WaveHdr** structure that identifies the buffer.

**Word**              **wSize**

Specifies the size of the **WaveHdr** structure.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid

# WaveInClose

## Syntax

**Word WaveInClose(hWaveIn)**

Closes the specified waveform input device.

## Parameters

**HWaveIn hWaveIn**

Specifies a handle to the waveform input device to be closed.  
If the function is successful, the handle is no longer valid after this call.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid

**WERR\_STILLPLAYING**

There are still buffers in the queue

## Comments

If there are input buffers that have been sent with **WaveInAddBuffer**, and have not been used, the close operation will fail. Call in **WaveInReset** to mark all pending buffers as done.

# **WaveInGetNumDevs**

## Syntax

**Word WaveInGetNumDevs()**

Retrieves the number of waveform input devices present in the system.

## Parameters

None

## Returns value

Returns the number of waveform input devices in the system.

# WaveInOpen

## Syntax

```
Word  WaveInOpen (iphWaveIn, wDeviceID, lpFormat, dwCallBack,
                   dwCallBackData, dwFlags)
```

Opens the specified waveform input device for recording.

## Parameters

**HWaveIn**            far \*lpWaveIn

Specifies a pointer to a HWaveIn handle. This location is filled with a handle identifying the opened waveform input device. Use this handle to identify the device when calling other waveform input functions.

This parameter may be NULL if the WAVE\_FORMAT\_QUERY flag is specified for the dwFlags.

**Word**            wDeviceID

Identifies the waveform input device that is to be opened.

**LpWaveFormat** lpFormat

Specifies a far pointer to a WaveFormat data structure that identifies the desired format for recording the waveform data.

**int (far \* dwCallBack) (HWaveIn dev, LpWaveHdr block,**  
**DWord dwCallBackData)**

Specifies the address of a callback function. The callback function is called by the driver during recording to process messages related to the progress of the recording.

Specify NULL for this parameter if no callback is desired.

**DWord**            dwCallbackData

Specifies 32 bits of user defined data that is passed to the callback function.

**DWord** dwFlags

Specifies flags for opening the device.

**WAVE\_FORMAT\_QUERY**

If this flag is specified, the device driver will determine if it supports the given format, but will not actually open the device.

#### Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_ALLOCATED**

Specified resource is already allocated.

**WERR\_BADDEVICEID**

Specified device is out of range.

**WERR\_BADTRANSFERMODE**

Specified transfer mode is unsupported or unavailable.

**WERR\_STEREOBADCHANNEL**

Invalid channel for stereo output (stereo output is only possible on channel 0).

**WERR\_STERONEED2FREECHNL**

Could not allocate two consecutive channels for stereo output.

**WERR\_UNSUPPORTEDFORMAT**

Attempted to open with an unsupported wave format.  
(This error code not currently supported).

#### Comments

Use **WaveInGetNumDevs** to determine the number of input devices present in the system. The device ID specified by wDeviceID varies from 0 to one less than the specified number of devices present.

The application should make sure that the transfer mode specified in the lpFormat variable is supported by the hardware configuration. The wave driver does NOT validate a DMA or interrupt transfer. This can be done by calling the appropriate functions in the control chip driver.

## WaveInReset

### Syntax

**Word WaveInReset(hWaveIn)**

Stops input on a given waveform device and resets the current position to 0. All pending buffers are marked as done.

### Parameters

**HWaveIn hWaveIn**

Specifies a handle to the input device that is to be reset.

### Return value

Returns zero if the function is successful. Otherwise, it returns an error code.  
Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

# WaveInStart

## Syntax

**Word WaveInStart(hWaveIn)**

Starts input on a given waveform input device.

## Parameters

**HWaveIn hWaveIn**

Specifies a handle to the input device to be started.

## Return value

Returns zero if the function is successful. Otherwise, it returns an error code.  
Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## Comments

Buffers are returned to the client when full or when WaveInReset is called (the dwBytesRecorded field in the header will contain the actual length of the data). If there are no buffers available, the data is thrown away without notification to the client and input will continue.

Calling this function when input is already started will have no effect and 0 will be returned.

# WaveOutBreakLoop

## Syntax

**Word WaveOutReset(hWaveOut)**

Breaks a loop on a given waveform device and allows playback to continue with the next block in the driver list.

## Parameters

**HWaveOut hWaveOut**

Specifies a handle to the waveform output device to receive the command.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid

## Comments

Waveform looping is controlled by the dwLoops and dwFlags fields in the **WaveHdr** structures passed to the device with **WaveOutWrite**. Use the **WHDR\_BEGINLOOP** and **WHDR\_ENDLOOP** flags in the **WaveHdr** structure to specify the beginning and ending data blocks for looping. To loop on a single block, specify both flags for the same block. Use the dwLoops field in the **WaveHdr** structure for the first block in the loop to specify the number of loops.

Calling this function when nothing is playing or looping will have no effect and 0 will be returned.

# WaveOutClose

## Syntax

**Word** **WaveOutClose(hWaveOut)**

This function closes the specified waveform output device.

## Parameters

**HWaveOut** hWaveOut

Specifies a handle to the waveform output device to be closed. If the function is successful, the handle is no longer valid after the call.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

**WERR\_STILLPLAYING**

There are still buffers in the device queue.

## Comments

If the device is still playing a waveform, the close operation will fail. Use **WaveOutReset** to terminate playback before calling **WaveOutClose**.

# **WaveOutGetNumDevs**

## Syntax

**Word WaveOutGetNumDevs()**

Retrieves the number of waveform output devices present in the system.

## Parameters

None

## Returns value

Returns the number of waveform output devices in the system.

# WaveOutGetVolume

## Syntax

**Word** **WaveOutGetVolume(hWaveOut, lpdwVolume)**

This function queries the current volume setting of a waveform output device.

## Parameters

**HWaveOut** hWaveOut

Identifies the wave output device.

**LPDWord** lpdwVolume

Specifies a far pointer to a location that will be filled with the current volume setting.

The high-order word contains the left channel volume and the low-order word contains the right channel volume.

If a device does not support volume control on both left and right channels (if the device is opened in mono), only the right channel value is used.

A value of 0xFFFF specifies full volume and a value of 0x0000 is silence.

## Return Value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## Comments

Volume control is supported on the left and right channels only if the device was opened specifying 2 in the nChannel field of the **lpWaveFormat** structure of **WaveInOpen**.

# WaveOutOpen

## Syntax

```
Word  WaveOutOpen  (lphWaveOut, wDeviceId, lpFormat, dwCallBack,
                     dwCallBackData, dwFlags)
```

Opens a specified waveform output device for playback.

## Parameters

**HWaveOut** far \*lphWaveOut

Specifies a pointer to an HWAVEOUT handle. This location is filled with a handle identifying the opened waveform output device.

Use the handle to identify the device when calling other wave output functions. This parameter may be NULL if WAVE\_FORMAT\_QUERY is specified in dwFlags.

**Word** wDeviceID

Identifies the waveform output device that is to be opened.

**LpWaveFormat** lpFormat

Specifies a pointer to a WaveFormat structure that identifies the format of the waveform that will be sent to the output device.

The WaveFormat structure is also used to specify the "mode" by which the data will be sent to the hardware (WAVE\_TRANS\_POLLING, WAVE\_TRANS\_INTERRUPT, WAVE\_TRANS\_DMA).

**int (far \* dwCallBack) (HWaveOut dev, LpWaveHdr block,**  
**DWord dwCallBackData)**

Specifies the address of a callback function. The callback function is called by the driver during playback to process messages related to the progress of the playback.

Specify NULL for this parameter if no callback is desired.

**DWord** dwCallbackData

Specifies 32 bits of user defined data that is passed to the callback.

**DWORD dwFlags**

Specifies flags for opening the device.

**WAVE\_FORMAT\_QUERY**

If this flag is specified, the device driver will determine if it supports the given format, but will not actually open the device.

**Return value**

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_ALLOCATED**

Specified resource is already allocated.

**WERR\_BADDEVICEID**

Specified device is out of range.

**WERR\_BADTRANSFERMODE**

Specified transfer mode is unsupported or unavailable.

**WERR\_STEREOBADCHANNEL**

Invalid channel for stereo output (stereo output is only possible on channel 0).

**WERR\_STERONEED2FREECHNL**

Could not allocate two consecutive channels for stereo output.

**WERR\_UNSUPPORTEDFORMAT**

Attempted to open with an unsupported wave format.  
(This error code not currently supported).

**Comments**

Use **WaveOutGetNumDevs** to determine the number of output devices present in the system. The device ID specified by **wDeviceID** varies from 0 to one less than the specified number of devices present.

The application should make sure that the transfer mode specified in the **lpFormat** structure is supported by the hardware configuration. The wave driver does NOT validate a DMA or interrupt transfer. This can be made by calling the appropriate functions in the control chip driver. The wave driver uses information stored in the control chip to determine which interrupt and which DMA line it will use.

# WaveOutPause

## Syntax

**Word WaveOutPause(hWaveOut)**

Pauses playback on a specified waveform output device. The current playback position is saved. Use **WaveOutRestart** to resume playback from the current playback position.

## Parameters

**HWaveOut hWaveOut**

Specifies a handle to the waveform output device to be paused.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## Comments

Calling this function when output is already paused will have no effect and 0 will be returned.

# WaveOutReset

## Syntax

**Word WaveOutReset(hWaveOut)**

Stops playback on a given waveform output device and resets the current position to 0. All pending playback buffers are marked as done.

## Parameters

**HWaveOut hWaveOut**

Specifies a handle to the waveform output device that is to be reset.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## WaveOutRestart

### Syntax

**Word** **WaveOutRestart(hWaveOut)**

This function restarts a paused waveform output device.

### Parameters

**HWaveOut** hWaveOut

Specifies a handle to the waveform output device that is to be restarted.

### Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

### Comments

Calling this function when the output is not paused will have no effect and 0 will be returned.

# WaveOutSetLeftRight

## Syntax

**Word** WaveOutSetLeftRight(**hWaveOut**, **leftRight**)

Selects which sides the output will be directed to.

## Parameters

**HWaveOut**      **hWaveOut**

Specifies a handle to the waveform output device that is to be restarted.

**Word**      **leftRight**

Flags specifying the output direction:

**WAVE\_STEREO\_LEFT**  
    **WAVE\_STEREO\_CENTER**  
    **WAVE\_STEREO\_RIGHT**

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid

## Comments

This function is useful only when the channel is monophonic. Stereophonic channels are always output left and right.

# WaveOutSetVolume

## Syntax

**Word** **WaveOutSetVolume(hWaveOut, dwVolume)**

Sets the volume of a waveform output device.

## Parameters

**HWaveOut** hWaveOut

Identifies the wave output device.

**Dword** dwVolume

Specifies the volume setting.

The high-order word contains the left channel volume and the low-order word contains the right channel volume.

If a device does not support volume control on both left and right channels (if the device is opened in mono), only the right channel value is used.

A value of 0xFFFF specifies full volume and a value of 0x0000 is silence.

## Return value

Returns zero if the function was successful. Otherwise, it returns an error code. Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## Comments

Volume control is supported on the left and right channels only if the device was opened specifying 2 in the nChannel field of the **IpWaveFormat** structure specified in **WaveOutOpen**.

Note that this controls output volume only.

# WaveOutWrite

## Syntax

**Word** **WaveOutWrite(hWaveOut, lpWaveOutHdr, wSize)**

Sends a data block to the specified waveform output device.

## Parameters

**HWaveOut** hWaveOut

Specifies a handle to the waveform device that the data is to be sent to.

**LpWaveHdr** lpWaveOutHdr

Specifies a far pointer to a **WaveHdr** structure containing information about the data block.

**Word** wSize

Specifies the size of the **WaveHdr** structure.

## Return value

Returns 0 if the function was successful. Otherwise, it returns an error code.  
Possible error codes are:

**WERR\_INVALIDHANDLE**

Specified device handle is invalid.

## Comments

Unless playback is paused by **WaveoutPause**, playback begins when the first data block is sent to the device.

When writing to a device opened using the **WAVE\_TRANSF\_POLLING** mode, control will be returned to the application only when the buffer has been completely played. Using this transfer mode, wave output must be paused with **WaveOutPause** prior to calling **WaveOutWrite** if the application must write more than one buffer.

The Ad Lib Gold card offers to developers 5 multi-purpose timers. They are physically located on two different chips but their implementation are similar.

All timers have their own base clock (time resolution) and counter size (maximum period). The controls available for all timers are:

- Write access in their register of different count values (divider).
- Stop and start (decrementing the initial stored count until it reach zero and re-writing the original count, again and again).
- Enable/disable interrupts to occur on zero count crossing.
- Read the interrupt status (access on the zero count crossing).

Some differences exist and need to be noticed:

- The timer 2 from the MMA chip is the only timer whose current count can be read.
- Yamaha in its own documentation use the terms timer 1 and 2 for the timers physically located in the OPL3 chip and timers located in the MMA chip.
- A base counter (another timer) is used in the MMA chip as an input clock for the timers 1 and 2. Those last two timers are decremented each time the base counter reaches zero. This means that the software must initialized the base counter with an appropriate value then the timer 1 or 2.

Here is a table that illustrates the specifications of all timers:

	OPL3 chip		MMA chip			
	Tim. 1	Tim. 2	Tim. 0	B. C.	Tim. 1	Tim. 2
time resolution in $\mu$ sec	80	320	1.89	1.89	1.89	1.89
max period length in msec	20.4	81.6	123.83	7.738	116.07	507116
counter size in bits	8	8	16	12	4+12	16+12

Table 1: Hardware specifications of timers

Remember that the MMA timer 1 and 2 are combined with the MMA base counter and that their combined specifications gives for the timer 1 a size of 16 bits and for the timer 2 a size of 28 bits.

The timer's function can be access directly or by the TimerDrvService functions which is a dispatcher.

Each timer function is presented in the following pages.

# **LoadStartOPL3Timer1**

**LoadStartOPL3Timer2**  
**LoadStartMMATimer0**  
**LoadStartMMATimer1**  
**LoadStartMMATimer2**

## Syntax

```
WORD    LoadStartOPL3Timer1(void)
WORD    LoadStartOPL3Timer2(void)
WORD    LoadStartMMATimer0(void)
WORD    LoadStartMMATimer1(void)
WORD    LoadStartMMATimer2(void)
```

This will load the physical counter with the count associated and start the counter.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured when loading.

## Comments

None

# **StopOPL3Timer1**

StopOPL3Timer2

StopMMATimer0

StopMMATimer1

StopMMATimer2

## Syntax

<b>WORD</b>	<b>StopOPL3Timer1(void)</b>
<b>WORD</b>	<b>StopOPL3Timer2(void)</b>
<b>WORD</b>	<b>StopMMATimer0(void)</b>
<b>WORD</b>	<b>StopMMATimer1(void)</b>
<b>WORD</b>	<b>StopMMATimer2(void)</b>

Stop the associated timer.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured when stoping.

## Comments

None

# SetOPL3Timer1Counter

SetOPL3Timer2Counter  
 SetMMATimer0Counter  
 SetMMATimer1Counter  
 SetMMATimer2Counter  
 SetMMABaseCounterCounter

## Syntax

<b>WORD</b>	<b>SetOPL3Timer1Counter(BYTE count)</b>
<b>WORD</b>	<b>SetOPL3Timer2Counter(BYTE count)</b>
<b>WORD</b>	<b>SetMMATimer0Counter(WORD count)</b>
<b>WORD</b>	<b>SetMMATimer1Counter(BYTE count)</b>
<b>WORD</b>	<b>SetMMATimer2Counter(WORD count)</b>
<b>WORD</b>	<b>SetMMABaseCounterCounter(WORD count)</b>

Set the OPL3 and MMA timer with the count value. Base clock periods are the following:

OPL3Timer1:	79.9682 us
OPL3Timer2:	319.873 us
MMATimer0:	1.89 us
MMATimer1:	1.89 us
MMATimer2:	1.89 us
MMATimerBaseCounter:	1.89 us

See table xx for more information the capacity of each timer.

## Parameters

**BYTE count**

**WORD count**

The parameters count specified the number of cycle the timer is supposed to do. Depending of timer count is BYTE or WORD parameter.

## Return value

**TIMER\_NO\_ERROR**

If the function was successful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured when setting.

## Comments

It is important to check the table xx because each timer don't use all of the bits in the count parameters.

# SetOPL3Timer1Period

SetOPL3Timer2Period  
SetMMATimer0Period  
SetMMATimer1Period  
SetMMATimer2Period  
SetMMABaseCounterPeriod

## Syntax

<b>WORD</b>	<b>SetOPL3Timer1Period(DWORD IPeriod)</b>
<b>WORD</b>	<b>SetOPL3Timer2Period(DWORD IPeriod)</b>
<b>WORD</b>	<b>SetMMATimer0Period(DWORD IPeriod)</b>
<b>WORD</b>	<b>SetMMATimer1Period(DWORD IPeriod)</b>
<b>WORD</b>	<b>SetMMATimer2Period(DWORD IPeriod)</b>
<b>WORD</b>	<b>SetMMABaseCounterPeriod(DWORD IPeriod)</b>

This set of functions offer another way to set the count of a timer. The period of a cycle is passed instead of passing the divider. It becomes more easy for the programmer to think in terms of period rather than in terms of a divider to associate with the required period.

## Parameters

### **DWORD IPeriod**

Period in usec to be passed to the timer.

## Return value

### **TIMER\_NO\_ERROR**

If the function was sucessful.

### **TIMER\_FUNCTION\_ERROR**

If a problem occured when setting.

## Comments

Check the table xx to be sure to respect the maximum capacity of the timer.  
The period will be round to the precision of the timer.

# EnableOPL3Timer1

EnableOPL3Timer2  
EnableMMATimer0  
EnableMMATimer1  
EnableMMATimer2

## Syntax

<b>WORD</b>	<b>EnableOPL3Timer1(void)</b>
<b>WORD</b>	<b>EnableOPL3Timer2(void)</b>
<b>WORD</b>	<b>EnableMMATimer0(void)</b>
<b>WORD</b>	<b>EnableMMATimer1(void)</b>
<b>WORD</b>	<b>EnableMMATimer2(void)</b>

This will set the mask bit associated with the timer interrupt.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured when enabling.

## Comments

None

# DisableOPL3Timer1

DisableOPL3Timer2

DisableMMATimer0

DisableMMATimer1

DisableMMATimer2

## Syntax

```
WORD    DisableOPL3Timer1(void)
WORD    DisableOPL3Timer2(void)
WORD    DisableMMATimer0(void)
WORD    DisableMMATimer1(void)
WORD    DisableMMATimer2(void)
```

This will reset the mask bit associated with the timer interrupt.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured when disabling.

## Comments

None

# GetOPL3TimerIntStatus

GetMMATimerIntStatus

## Syntax

<b>WORD</b>	<b>GetOPL3TimerIntStatus(void)</b>
<b>WORD</b>	<b>GetMMATimerIntStatus(void)</b>

These functions will return the state of timer interrupt of the OPL3 and MMA.

## Parameters

None

## Return value

### OPL3

return 0 if no timer has interrupted.  
return 2 if timer 1 has interrupted.  
return 1 if timer 2 has interrupted.  
return 3 if timer 1 and 2 has interrupted.  
return 0 if no timer has interrupted.  
return 1 if timer 0 has interrupted.  
return 2 if timer 1 has interrupted.  
return 4 if timer 2 has interrupted.  
or any combination of 1,2 and 4 if multiple timer has interrupted.

## Comments

The MMA chip has a special behavior: it will reset the interrupt bit after a status register reading. Note that this routine is automatically called by the main interrupt handler from the Control Chip Driver. Using GetOPL3TimerIntStatus will not reset the OPL3 status register bits.

# AssignOPL3Timer1IntService

AssignOPL3Timer2IntService

AssignMMATimer0IntService

AssignMMATimer1IntService

AssignMMATimer2IntService

## Syntax

<b>WORD</b>	<b>AssignOPL3Timer1IntService(void (*function)(void))</b>
<b>WORD</b>	<b>AssignOPL3Timer2IntService(void (*function)(void))</b>
<b>WORD</b>	<b>AssignMMATimer0IntService(void (*function)(void))</b>
<b>WORD</b>	<b>AssignMMATimer1IntService(void (*function)(void))</b>
<b>WORD</b>	<b>AssignMMATimer2IntService(void (*function)(void))</b>

Use by applications to assign their callback function on a specific interrupt.

## Parameters

**void (\*function)(void)**

The parameter is the callback prototype.

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occurred with the assign procedure.

## Comments

The application user must specify a callback routine that will automatically be called when the interrupt occurs. This callback function must be very short to execute because this is a timer interrupt that may occurs at a very high rate. At initialisation the default service hooked on each timer interrupt is a local DoNothing function that must be replaced by the application user.

# RestoreOPL3Timer1IntService

RestoreOPL3Timer2IntService  
RestoreMMATimer0IntService  
RestoreMMATimer1IntService  
RestoreMMATimer2IntService

## Syntax

WORD	<b>RestoreOPL3Timer1IntService(void)</b>
WORD	<b>RestoreOPL3Timer2IntService(void)</b>
WORD	<b>RestoreMMATimer0IntService(void)</b>
WORD	<b>RestoreMMATimer1IntService(void)</b>
WORD	<b>RestoreMMATimer2IntService(void)</b>

Use by applications to remove their callback function from the interrupt process.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured with the restore procedure.

## Comments

None

# **ExecOPL3Timer1IntService**

**ExecOPL3Timer2IntService**

**ExecMMATimer0IntService**

**ExecMMATimer1IntService**

**ExecMMATimer2IntService**

## Syntax

```
void      ExecOPL3Timer1IntService(void)
void      ExecOPL3Timer2IntService(void)
void      ExecMMATimer0IntService(void)
void      ExecMMATimer1IntService(void)
void      ExecMMATimer2IntService(void)
```

Those routines will execute the function associated with each interrupt.

## Parameters

None

## Return value

None

## Comments

None

# ResetOPL3LastTimerInt

## Syntax

**WORD      ResetOPL3LastTimerInt(void)**

This will reset the IRQ signal generated by timers 1 and 2.

## Parameters

None

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured with the reset procedure.

## Comments

This function does not exist for the MMA because the MMA clear the status after each reading of the status register.

# **AllocateOPL3Timer1**

**AllocateOPL3Timer2**  
**AllocateMMATimer0**  
**AllocateMMATimer1**  
**AllocateMMATimer2**  
**AllocateMMABaseCounter**

## **Syntax**

```
WORD    AllocateOPL3Timer1(void)
WORD    AllocateOPL3Timer2(void)
WORD    AllocateMMATimer0(void)
WORD    AllocateMMATimer1(void)
WORD    AllocateMMATimer2(void)
WORD    AllocateMMABaseCounter(void)
```

This procedure will reserve and from then denied any external application access to this timer.

## **Parameters**

None

## **Return value**

1: if available  
0: if not available

## **Comments**

Any application who wants to use the service of any timers should ask the Timer Driver for its disponibility using an allocation routine. The application should free the timer after use.

# FreeOPL3Timer1

FreeOPL3Timer2

FreeMMATimer0

FreeMMATimer1

FreeMMATimer2

FreeMMABaseCounter

## Syntax

<b>WORD</b>	<b>FreeOPL3Timer1(void)</b>
<b>WORD</b>	<b>FreeOPL3Timer2(void)</b>
<b>WORD</b>	<b>FreeMMATimer0(void)</b>
<b>WORD</b>	<b>FreeMMATimer1(void)</b>
<b>WORD</b>	<b>FreeMMATimer2(void)</b>
<b>WORD</b>	<b>FreeMMABaseCounter(void)</b>

Free the the timer.

## Parameters

None

## Return value

1: if operation succed  
0: if operation not succed

## Comments

None

# **GetMMATimer2Content**

## Syntax

**WORD      GetMMATimer2Content(void)**

This routine returns the content of the MMA timer 2.

## Parameters

None

## Return value

16 bit content of MMA timer 2

## Comments

This is the only timer that can be read. These timers respect the specification of Windows Multi-Media.

# GetOPL3Timer1Caps

GetOPL3Timer2Caps  
GetMMATimer0Caps  
GetMMATimer1Caps  
GetMMATimer2Caps

## Syntax

<b>WORD</b>	<b>GetOPL3Timer1Caps</b> (DWORD far *IPeriodMin, DWORD far *IPeriodMax)
<b>WORD</b>	<b>GetOPL3Timer2Caps</b> (DWORD far *IPeriodMin, DWORD far *IPeriodMax)
<b>WORD</b>	<b>GetMMATimer0Caps</b> (DWORD far *IPeriodMin, DWORD far *IPeriodMax)
<b>WORD</b>	<b>GetMMATimer1Caps</b> (DWORD far *IPeriodMin, DWORD far *IPeriodMax)
<b>WORD</b>	<b>GetMMATimer2Caps</b> (DWORD far *IPeriodMin, DWORD far *IPeriodMax)

Used by external modules to query the driver on physical limits of each timer. It returns the minimum and maximum period covered by the timer in micro seconds.

## Parameters

**DWORD far \*IPeriodMin**  
**DWORD far \*IPeriodMax**

These two address will receive the minimum and the maximum period capacity respectively of the timer.

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured with the procedure.

## Comments

None

# **InitTimerDriver**

## Syntax

**WORD      InitTimerDriver(WORD base)**

This procedure initialize the Timer Driver structure with default values.  
This procedure should be used the first time the driver is called.

## Parameters

**WORD base**

Actual address of the Ad Lib control chip.

## Return value

**TIMER\_NO\_ERROR**

If the function was sucessful.

**TIMER\_FUNCTION\_ERROR**

If a problem occured with the procedure.

## Comments

None

# TimerDrvService

## Syntax

**WORD far TimerDrvService(WORD segm, WORD offs)**

Entry point for the AdLib timer dispatcher. The segment and offset of the argument structure are passed as argument.

## Parameters

**WORD segm**

**WORD offs**

These two parameters specify the segment and the offset of the following structure which is used to pass parameters to the TimerDrvService routine.

struct TimerArgum {

WORD	controlID;	which service to be used
WORD	timerDv;	on which timer
DWORD	param;	optionnal based on service used
DWORD	param2;	optionnal based on service used
void	(interrupt far *function)();	optionnal based on service used
}		

## Return value

Service result if any.

## Comments

See TimerDrv.h for all ID of services.



# Chapter 7 - Low-level Programming

<b>7.1 Mixer and Setup Features</b>	<b>1</b>
Register Access	1
Status Register	2
Register Map	2
Register Reference	4
Control/ID	4
Telephone Control	5
Sampling Gain	5
Final Output Volume	5
Bass	6
Treble	7
Output Mode	7
Mixing Volumes	8
Audio Selection	8
Register 12h	9
Audio IRQ/DMA Select - Channel 0	9
DMA Select - Channel 1	10
Audio Relocalisation	11
SCSI IRQ/DMA Select	12
SCSI Relocalization	13
Surround	13

<b>7.2 FM Synthesis</b>	<b>15</b>
Programming the YM3812	17
The Ad Lib Music Synthesizer Card	17
Operators	18
ALMSC Input / Output Map	19
Register Reference	21
Test Register/WSE	21
Timers	21
Status Register	22
CSM/Keyboard Split	23
AM/VIB/EG-TYP/KSR/Multiple	23
KSL/Total Level	25
ADSR	26
BLOCK/F-Number	26
Rhythm/AM Dep/VIB Dep	27
FeedBack/Connection	27
Wave Select	28
Programming the YMF262	29
Register Array 0	29
Register Array 1	33
4-Operator Voices	33

<b>7.3 Digital Input and Output (Digital Audio and MIDI)</b>	<b>37</b>
Register Reference	40
Status Register	40
Register 00H: Test Register	40
Registers 02H - 07H: Timer Counters	41
Register 08H: Timer Control	42
Stand-by Mode	42
Timer Interrupt Masks	42
Timer Controls	42
Register 09H: Playback and Recording Control	42
Reset PCM/ADPCM	42
Select Output Channel	42
Select Frequency	43
PCM/ADPCM Selection	43
Select Record/Playback	43
Start/Stop Record/Playback	43
Register 0AH: Output Volume Control	43
Register 0BH: PCM/ADPCM Data	44
Register 0CH: Sampling Format and Control	44
Interleaving	44
Set Data Format	44
Set FIFO Interrupt	45
FIFO Interrupt Mask	45
DMA Mode Specification	46
Register 0DH: MIDI and Interrupt Control	46
Mask Digital Overrun Error	46
Mask MIDI Overrun Error	46
Reset MIDI transmit circuit	46
Mask MIDI transmit FIFO interrupts	46
Reset MIDI Receive Circuit	46
Mask MIDI Receive FIFO Interrupts	46
Register 0EH: MIDI Data	46
MMA Programming Tips	47



## Register Access

The control chip registers are implemented as a set of phantom registers to the second bank of FM registers. Access to the the control chip is triggered by writing 0FFh to the address register of the second FM bank (38Ah). Thereafter, all reads/writes will access the control chip. Access to the second FM bank is returned by writing 0FEh to the same address register.

As with the FM and sampling chips, the control chip uses two port addresses. The first address, 38Ah, is the address register and writing a register number to this address selects a given data register. The second address, 38Bh, is the data address. Values written to this address are directed to the register number specified by the previous write to the address register. There are delays that must be respected when writing to certain registers. These delays are explained in detail in the *Status Register* section.

By default, the control chip is located at 38Ah and 38Bh. However, the chip may be relocated (as explained in the section *Audio Relocalization*). Regardless of where the chip is located, the data register port address is always one greater than the address register port address.

All data registers on the control chip are read/write. Reading a register will return its current value. The only exception to this are registers 0 and 1. All registers are explained below in detail.

The Gold cards contain permanent memory (EEPROM) in which the boot-up values for all registers are stored.

## Mixer and Setup Features

### Status Register

#### **Status Register**

Reading the address port (38Ah by default) when the control chip access has been triggered returns the following information:

D7	D6	D5	D4	D3	D2	D1	D0
RB	SB	X	X	SCSI	TEL	SMP	FM

The 4 least significant bits indicate interrupt status. Reading this register does not reset the interrupt status. A zeroed bit indicates which section of the board has generated an interrupt. FM indicates the FM section has generated an interrupt; SMP, the sampling section; TEL, the telephone section; SCSI, the SCSI section. SB set indicates that the card is busy writing to a register. RB set indicates that the card is busy writing its registers to memory.

A delay of approximately 450  $\mu$ sec is required after writing to any of registers 4 to 8. A delay of approximately 5  $\mu$ sec is required after writing to any of registers 9 through 16. As well, the chip must not be accessed while the chip is saving its registers to memory. In order to respect these delays, the SB and RB bits should be polled until they become zero. As a general rule, always poll the SB and RB bits before writing anything to the chip.

As well, the chip must not be accessed while it is restoring its registers from memory. This process takes a bit less than 2.5 milliseconds. As there is no status bit for this action, the timing must be done in software.

**IMPORTANT:** Before returning access to the FM chip (writing FEh to 38Ah), all delays must have expired. Results will be unpredictable otherwise.

#### **Register Map**

The diagram on the following page is a summary of the control chip registers. When writing to registers which contain undesignated bits, these bits must be set to zero. Locations where certain bits must be set are indicated by a "1" in the register map.

## Register Map, Control Chip

REG	D7	D6	D5	D4	D3	D2	D1	D0				
00							ST	RT				
01							RING	TC				
02	SAMPLING GAIN - LEFT											
03	SAMPLING GAIN - RIGHT											
04	1	1		FINAL OUTPUT VOLUME - LEFT								
05	1	1		FINAL OUTPUT VOLUME -RIGHT								
06	1	1	-1	1	BASS							
07	1	1	1	1	TREBLE							
08	1	1	MU	ST-MONO	SOURCE							
09	FM VOLUME - LEFT											
0A	FM VOLUME - RIGHT											
0B	SAMPLING VOLUME - LEFT											
0C	SAMPLING VOLUME - RIGHT											
0D	AUX VOLUME - LEFT											
0E	AUX VOLUME - RIGHT											
0F	MICROPHONE VOLUME											
10	TELEPHONE VOLUME											
11		SPKR		MFB	XMO	FLT0	FLT1					
12												
13	DENO	DMA SEL 0			AEN	INT SEL A						
14	DEN1	DMA SEL 1										
15		AUDIO RELOCATE										
16	DENS	DMA SEL S			SIEN	INT SEL S						
17		SCSI RELOCATE										
18	SURROUND											

## Mixer and Setup Features

### Register Reference

### Register Reference

#### Control/ID

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	ST	RT

Register #0: Write

Writing to the Control/ID byte with the ST bit set will cause all control chip registers, in their current state, to be written to memory. If RT is set, then all registers will be restored from memory. When the operation is finished, the control chip sets the appropriate bit back to zero. It is not necessary to manually clear the bit.

D7	D6	D5	D4	D3	D2	D1	D0
X	OP2	OP1	OP0				MODEL ID

Register #0: Read

Reading this register gives information on the model of the card and which options are present. The currently defined MODEL ID's are:

ID	Gold Model
0	2000
1	1000
2	2000MC

The OP0, OP1 and OP2 bits indicate which of the board options are present and are SET when the option is NOT present.

Bit	Option
OP0	Telephone
OP1	Surround
OP2	SCSI

**Telephone Control**

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	X	TC

Register #1: Write

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	RING	TC

Register #1: Read

Setting TC engages the telephone line; clearing the bit hangs up. Reading this register returns the state of the telephone ring signal: RING set indicates that the line is NOT ringing and TC returns the status of the telephone line (i.e. the previously written value of TC).

**Sampling Gain**

Registers 2 and 3 control the gain on sampling channels 0 (left) and 1 (right). 256 different gain values are possible, giving a range from approximately 0.04 to 10 times the input value. The exact gain is given by the equation:

$$\text{Gain} = (\text{RegisterValue} * 10) / 256$$

**Final Output Volume**

These registers control the overall output volume of the card. They replace the potentiometer found on the original Ad Lib card. Adjusting for left and right channels separately allows the balance to be varied.

The volume ranges from +6 dB to -64 dB in steps of 2 dB. An additional step gives -80 dB (off). **IMPORTANT:** Bits D6 and D7 must be set to 1.

## Mixer and Setup Features

### Register Reference

dB	D5-D0
6	3F
4	3E
:	:
-62	1D
-64	1C
-80	1B
:	:
-80	0

Registers #4 and #5

### Bass

The bass control has a range of +15dB to -12 dB in 3 dB steps. The bass is set using bits D0-D3. **IMPORTANT:** Bits D4 - D7 must be set to 1.

dB	D3-D0
15	F
:	:
15	B
12	A
:	:
0	6
:	:
-12	2
:	:
-12	0

Register #6

**Treble**

The treble control has a range of +12dB to -12 dB in 3 dB steps. The treble is set using bits D0-D3. **IMPORTANT:** Bits D4 - D7 must be set to 1.

<b>dB</b>	<b>D3-D0</b>
12	F
:	:
12	A
:	:
0	6
:	:
-12	2
:	:
-12	0

Register #7

**Output Mode**

D7	D6	D5	D4	D3	D2	D1	D0
1	1	MU	ST-MONO		SOURCE		

Register #8

This register controls the final output. This final output section takes as its input the output from the mixing section. SOURCE indicates which channels from the mixer are selected for final output. If only one input channel is selected, it is directed to both output channels. Stereo input results in stereo output.

<b>SOURCE</b>	<b>Channels</b>
6	Left and right
4	Right only
2	Left only

ST-MONO selects the type of effect applied to the final output:

ST-MONO	Effect
3	Spatial stereo
2	Pseudo stereo
1	Linear stereo
0	Forced mono

Linear stereo is ordinary, stereo output with no effects added. The spatial and pseudo stereo effects will be useful primarily when the original sources are monophonic. If the surround option is present, the output signal is modified after mixing and the attributes of this register are then applied.

Setting MU enables muting; clearing it disables muting.

**IMPORTANT:** Bits D6 and D7 must be set to 1.

## Mixing Volumes

Registers 9 through 10h are individual volume control registers and constitute the mixing section of the card. 128 different linear volume levels are possible, ranging from 128 (silent) to 255 (maximum gain). Note that writing values less than 128 will result in a signal with negative polarity and should be avoided because the resulting signal may cancel out another signal of opposite polarity.

## Audio Selection

D7	D6	D5	D4	D3	D2	D1	D0
X	X	SPKR	X	MFB	XMO	FLT0	FLT1

Register #11h

The Gold card uses antialiasing filters during sampling and playback to ensure maximum audio quality. Because these operations are mutually exclusive on a given channel, the same antialiasing filter is used for sampling and playback. When FLT0 is set, the filter for Channel 0 (left) is set for input (recording); clearing the bit sets the filter for output (playback). FLT1 operates similarly, but is applied to Channel 1 (right).

Normally, the Aux input on the card is sampled in stereo on both channels at the same time. This stereo input can be turned monophonic and sampled on Channel 0 by setting XMO. Clearing XMO returns Aux input to its normal state.

When the telephone option of the Gold card is present, microphone input is directed to both the loudspeaker output as well as the telephone when MFB is cleared. However, this could cause feedback to occur. When MFB is set, the microphone signal is not directed to the loudspeaker output, thus eliminating possible causes of feedback. Although this feature is intended for use with the telephone option, it is operational at all times so that setting MFB always removes the microphone from the final output.

The internal audio speaker from the PC can be mixed directly with the final audio signal of the Gold Card. When SPKR is cleared, the signal is disconnected; when set it is connected.

### Register 12h

Register 12h is unused and should be ignored or set to 0 otherwise.

### Audio IRQ/DMA Select - Channel 0

D7	D6	D5	D4	D3	D2	D1	D0
DENO	DMA SEL 0		AEN		INT SEL A		

Register #13h

Audio interrupts (FM, sampling and telephone) are enabled when AEN is set. The following values for INT SEL A select the corresponding interrupt line:

## Mixer and Setup Features

### Register Reference

INT SEL A	IRQ
0	3
1	4
2	5
3	7
4	10
5	11
6	12
7	15

Only IRQ 3, 4, 5, and 7 are available on model Gold 1000. All listed interrupts are available on the Gold 2000 and the Gold 2000MC.

DMA for sampling channel 0 is enabled when DEN0 is set. The following values for DMA SEL 0 select the corresponding DMA line:

DMA SEL 0	DMA Line
0	0
1	1
2	2
3	3

Only DMA 1, 2 and 3 are available on model Gold 1000. All listed DMA lines are available on the Gold 2000 and the Gold 2000MC.

#### DMA Select - Channel 1

D7	D6	D5	D4	D3	D2	D1	D0
DEN1		DMA SEL 1		X	X	X	X

Register #14

DMA for sampling channel 1 is enabled when DEN1 is set. The following values for DMA SEL 1 select the corresponding DMA line:

DMA SEL 1	DMA Line
0	0
1	1
2	2
3	3

Only DMA 1, 2 and 3 are available on the model Gold 1000. All listed DMA lines are available on the Gold 2000 and the Gold 2000MC.

### **Audio Relocalisation**

D7	D6	D5	D4	D3	D2	D1	D0
X	AUDIO RELOCATE						

Register #15h

This register indicates the port address for the audio section (FM, sampling, control chip). Writing here immediately relocates the audio section to the specified address. The AUDIO RELOCATE value is the port address divided by eight. This forces the address to be on an 8-byte boundary.

The audio section uses 8 port addresses. It is the first of these 8 addresses which is used in this register. Note that the control chip address is considered to be part of the audio section, so that the address of the control chip changes as soon as this register is modified.

The following is the default configuration for the audio section:

Address	Section
388h, 389h	FM Bank 0
38Ah, 38Bh	FM Bank 1, Control Chip
38Ch, 38Dh	Sampling Channel 0
38Eh, 38Fh	Sampling Channel 1

**SCSI IRQ/DMA Select**

D7	D6	D5	D4	D3	D2	D1	D0
DENS		DMA SEL S		SIEN		INT SEL S	

Register #16h

SCSI interrupts are enabled when SIEN is set. The following values for INT SEL S select the corresponding interrupt line:

INT SEL S	IRQ
0	3
1	4
2	5
3	7
4	10
5	11
6	12
7	15

Only IRQ 3, 4, 5, and 7 are available on the model Gold 1000. All listed interrupts are available on the Gold 2000 and the Gold 2000MC.

SCSI DMA is enabled when DENS is set. The following values for DMA SEL S select the corresponding DMA line:

DMA SEL S	DMA Line
0	0
1	1
2	2
3	3

Only DMA 1, 2 and 3 are available on model GOLD 1000. All listed DMA lines are available on the Gold 2000 and the Gold 2000MC.

## SCSI Relocalization

D7	D6	D5	D4	D3	D2	D1	D0
X	SCSI RELOCATE						

Register #17h

This register indicates the port address for the SCSI section. Writing here immediately relocates the SCSI section to the specified address. The SCSI RELOCATE value is the port address divided by eight. This forces the address to be on an 8-byte boundary. The SCSI section uses 8 port addresses. It is the first of these 8 addresses which is used in this register. The default configuration has the SCSI section at addresses 390h to 397h.

## Surround

D7	D6	D5	D4	D3	D2	D1	D0
SURROUND							

Register #18h

The surround sound option of the card is accessed via this register. It will be documented at a later date.



This chapter explains the features of the new FM synthesis chip, the YM262, on the Ad Lib Gold cards. This chip is similar to the YM3812, the chip on the original Ad Lib card, and contains a compatibility mode to emulate the YM3812. Because of this similarity, the first part of this section discusses the features of the YM3812. Those of you who are already familiar with this chip may wish to skip this section and proceed to *Programming the YM262*, which discusses the differences between the two chips.



# Programming the YM3812

*(NOTE: This section is reproduced from the original Ad Lib Synthesizer Card Programmer's Manual. It is necessary for understanding the functioning of the new FM chip, the YMF262. If you are already familiar with this material, you may wish to proceed to the following section which discusses the YMF262.)*

This section provides information about the Ad Lib Music Synthesizer Card for advanced programmers who wish to program it directly. There is information on the components of the card, a technical description of the operators, the input / output map and a register reference section.

## The Ad Lib Music Synthesizer Card

The card is equipped with a vibrato oscillator, an amplitude oscillator (tremolo), a noise generator which allows for the combination of a number of frequencies, two programmable timers, composite sine wave synthesis and 18 operators.

A white noise generator is used to create rhythm sounds. This white noise generator uses voices 7 and 8 (melodic voices), frequency information (Block, F-Number, Multi), and the proper phase output. Various rhythm sounds are produced by combining this output signal with white noise. The resulting signal is then sent to the operators. Experience has shown that the best ratio for the two frequencies is 3:1 (melodic voice 7 frequency = 3 times melodic voice 8 frequency). Finally, envelope information is multiplied with the wave table output. As the envelope is set for one operator which corresponds to a single rhythm instrument, the values which express that instrument's characteristics are set in the parameter registers in the same manner as for melody instruments.

Operators

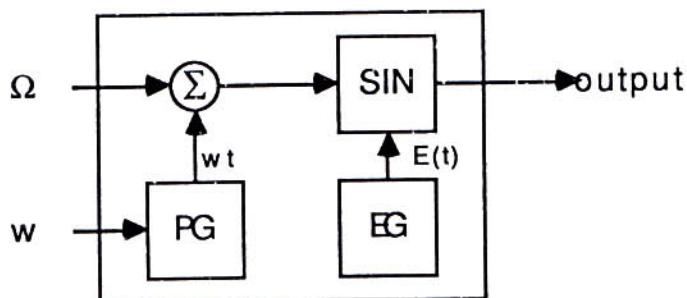
The ALMSC uses pure sine waves that interact together to produce the full harmonic spectrum for any voice. Each digital sine wave oscillator is combined with its own envelope generator to form an "operator".

An operator has 2 inputs and 1 output. One input is the pitch oscillator frequency and the other is for the modulation data. The frequency and modulation data (phases) are added together and converted to a sine wave signal. The phase generator (PG) converts the frequency ( $w$ ) into a phase by multiplying it by time ( $t$ ). An envelope generator (EG) produces a time variant amplitude signal (ADSR). The EG's output is then multiplied by the sine wave and output to the outside world.

The operator output can be expressed as a mathematical expression:

$$F(t) = E(t) \sin(wt + \Omega)$$

$E(t)$  is the output from the EG,  $w$  is the frequency,  $t$  is time and  $\Omega$  is the phase modulation.



The operators can be connected in three different ways: additive, frequency modulation and composite sine wave.

• **FM synthesis**

FM synthesis uses two operators in series. The first operator, the modulator, modulates the second operator via its modulation input. The name given to the second operator is the carrier. The modulator can feed back its output into its modulation data input;

$$F_m(t) = E_m(t) \sin(w_{mt} t + \beta F_m(t)) \quad \text{Modulator and feedback}$$
$$F_c(t) = E_c(t) \sin(w_{ct} t + F_m(t)) \quad \text{Carrier and Modulator}$$

- **Additive synthesis**

Additive synthesis connects two operators in parallel, adding both outputs together. This method of synthesis is not as interesting as FM synthesis, but it can generate good organ type sounds.

The simplified formula for the additive synthesis is:

$$F(t) = E_1(t) \sin(\omega t + \Omega_1) + E_2(t) \sin(\omega t + \Omega_2)$$

- **Composite sine wave synthesis**

Composite sine wave synthesis (CSW) may be used to generate speech or other related sounds by playing all voices simultaneously. When using this mode the card cannot generate any other sounds. This mode is not used because other methods have proved to provide better quality speech.

---

### ALMSC Input / Output Map

The ALMSC is located at address 388H in the i/o space. The card decodes two addresses: 388H and 389H. The first address is used for selecting the register address and the second is used for writing data to the selected register. There also exists the possibility of using three other addresses: 218H, 288H and 318H. The port address is currently hard-wired, but address jumpers may be added in the future so you may want to take into account the possibility of using different addresses when programming. Here is a register map of the ALMSC:

## Programming the Synthesizer

### ALMSC Input/Output Map

REG	D7	D6	D5	D4	D3	D2	D1	D0
01			WSE				TEST	
02					TIMER-1			
03					TIMER-2			
04	RST	T1	mask	T2				start/stop T2 T1
08	CSM	SEL						
20-35	AM	VIB	EG	KSR		MULTI		
40-55		KSL			TL			
60-75			AR			DR		
80-95			SL			RR		
A0-A8				F-NUMBER (L)				
B0-B8			KON		BLOCK		F-NUM (H)	
BD	DEP AM	DEP VIB	R	BD	SD	TOM	TC	HH
C0-C8					FB		C	
E0-F5							WS	

Because of the nature of the card, you must wait 3.3  $\mu$ sec after a register select write and 23  $\mu$ sec for a data write. Only the status register located at address 388H can be read.

For many parameters, there is one register per operator. However, there are holes in the address map so that the operator number cannot be used as an offset into the map. The operator offsets are as follows:

	Operator Address Offset									
Opr.	1	2	3	4	5	6	7	8	9	
Off. (hex)	00	01	02	03	04	05	08	09	0A	
Opr.	10	11	12	13	14	15	16	17	18	
Off. (hex)	0B	0C	0D	10	11	12	13	14	15	

For example, the KSL/TL registers are at 40H-55H. If we wish to access the register for operator 8, we must write to register 49H (NOT 48H).

## Register Reference

### Test Register/WSE

This register must be initialized to zero before taking any action. The wave select enable/disable bit (WSE) is D5. If set to 1, the value in the WS register will be used to select the wave form used to generate sound. If the WSE is set to 0, the value in the WS register will be ignored and the chip will use a sine wave. (The available waveforms are detailed later in this section).

### Timers

The timers are not wired on the card. However, the following information is included since the timers can be used to detect the presence of our card in the computer.

Timer-1 is an upward 8 bit counter with a resolution of 80  $\mu$ sec. If an overflow occurs, the status register flag FT1 is set, and the preset value (address = 02) is loaded into Timer-1. Timer-2 (address = 03) is an upward 8 bit counter just like Timer-1 except that the resolution is 320  $\mu$ sec.

$$T_{\text{overflow}}(\text{ms}) = (256-N) * K$$

N is the preset value and K is the timer constant equal to 0.08 for Timer-1 and 0.32 for Timer-2. Register address 04 controls the operation of both timers. ST1 and ST2 (start/stop T1 or T2) bits start or stop the timers. When the corresponding bit is 1 the counter is loaded and counting starts, but when 0 the counter is held.

The Mask bits are used to gate the status register timer flags. If a mask bit is 1 then the corresponding timer flag bit is kept low (0) and is active when the mask bit is cleared (0). The most significant bit (MSb) is called IRQ-RESET. It resets timer flags and IRQ flag in the status register to zero. All other bits in the control register are ignored when the IRQ-RESET bit is 1.

## **Status Register**

Reading at address 388H yields the following byte of information:

- D0 - D4 are unused.
- D5 Timer 2 flag: Set to 1 when the preset time in Timer 2 has elapsed. The flag remains until reset.
- D6 Same as D5, except for Timer 1.
- D7 IRQ flag: set if D5 or D6 are 1.

As mentioned earlier, the timer interrupts are not connected, but the timers can be used to detect the presence of the board as follows:

1. Reset T1 and T2: write 60H to register 4.
2. Reset the IRQ: write 80H to register 4 (this step must NOT be combined with Step #1).
3. Read status register: read at 388H. Save the result.
4. Set timer-1 to FFH: write FFH to register 2.
5. Unmask and start timer-1: write 21H to register 4.
6. Wait (in a delay loop) for at least 80  $\mu$ sec.
7. Read the status register and save the result.
8. Reset T1, T2 and IRQ as in steps #1 and #2.
9. Test the results of the two reads: the first should be 0, the second should be C0H. If either is incorrect, then an ALMSC board is not present. (NOTE: You should AND the result bytes with E0H as the unused bits are undefined.)

**CSM/Keyboard Split**

This register (address = 08) will determine if the card is to function in music mode (CSM = 0) or speech synthesis mode (CSM = 1) as well as the keyboard split point.

When using composite sine wave speech synthesis mode all voices should be in the KEY-OFF state. The bit NOTE-SEL (D6) is used to control the split point of the keyboard. When 0, the keyboard split is the second bit from the MSb (bit 8) of the F-Number. The MSb of the F-number is used when

NOTE-SEL = 1. This is illustrated in the following table:

NOTE-SEL = 0

BLOCK/OCT	0	1	2	3	4	5	6	7								
FNUM(MSb)	X	X	X	X	X	X	X	X								
FNUM(8)	0	1	0	1	0	1	0	1	0	1	0	1				
Split Num.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

NOTE-SEL = 1

BLOCK/OCT	0	1	2	3	4	5	6	7								
FNUM(MSb)	0	1	0	1	0	1	0	1								
FNUM(8)	X	X	X	X	X	X	X	X								
Split Num.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

X = Ignored

**AM/VIB/EG-TYP/KSR/Multiple**

This group of registers (addresses 20H to 35H), one per operator, controls the frequency conversion factor and modulating wave frequencies corresponding to the frequency components of music.

The MULTI 4-bit field determines the multiplication factor applied to the input pitch frequency in the PG section. That is, an operator's frequency will automatically be multiplied according to the value in this field. The multiplication factors are given in the following table:

## Programming the Synthesizer

### Register Reference

MULTI	Factor
0	1/2
1	1
2	2
3	3
4	4
5	5
6	6
7	7

MULTI	Factor
8	8
9	9
10	10
11	10
12	12
13	12
14	15
15	15

The operator output can then be expressed, with "d" as the multiplication factor, as follows:

$$F(t) = E_c(t) \sin(\partial_c w_{ct} + E_m \sin(\partial_m w_{mt}))$$

The KSR bit (position = D4) changes the rates for the envelope generator (EG). This parameter makes it possible to gradually shorten envelope length (increase EG rates) as higher notes on the keyboard are played. This is particularly useful for simulating the sound of stringed instruments such as piano and guitar, in which the envelope of the higher notes is noticeably shorter than the lower notes. The actual rate is then equal to the ADSR value plus an offset:

$$\text{Actual rate} = 4 * \text{Rate} + \text{KSR offset}$$

The KSR offset is specified in the following table:

Rate	KSR=0	KSR=1
0	0	0
1	0	1
2	0	2
3	0	3
4	1	4
5	1	5
6	1	6
7	1	7

Rate	KSR=0	KSR=1
8	2	8
9	2	9
10	2	10
11	2	11
12	3	12
13	3	13
14	3	14
15	3	15

The EG-Type activates the sustaining part of the envelope when the EG-Type is set (1). Once set, an operator's frequency will be held at its sustain level until a KEY-OFF is done.

The VIB parameter toggles the frequency vibrato (1 = on, 0 = off). The frequency of the vibrato is 6.4 Hz and the depth is determined by the DEP VIB bit in register 0BDH.

The AM parameter is similar to the VIB parameter except that it is an amplitude vibrato (tremolo) of frequency 3.7Hz. The amplitude vibrato depth is determined by the DEP AM bit in register 0BDH.

### KSL/Total Level

These registers (addresses 40H to 55H, 1 per operator) control the attenuation of the operator's output signal. The KSL parameter produces a gradual decrease in note output level towards higher pitch notes. Many acoustic instruments exhibit this gradual decrease in output level. The KSL is expressed on 2 bits (value 0 through 3). The corresponding attenuation is given below:

D7	D6	Attenuation
0	0	0
1	0	1.5dB/oct
0	1	3.0dB/oct
1	1	6.0dB/oct

The Total Level (TL) attenuates the operator's output. In FM synthesis mode, varying the output level of an operator functioning as a carrier results in a change in the volume of that operator's voice. Attenuating the output from a modulator will change the frequency spectrum produced by the carrier. In additive synthesis, varying the output level of any operator varies the volume of its corresponding voice. The TL value has a range of 0 through 63 (6 bits). To convert this value into an output level, apply the following formula:

$$\text{Output level} = (63 - \text{TL}) * 0.75\text{dB}$$

## **ADSR**

These values change the shape of the envelope for the specified operator by changing the rates or the levels. The attack (AR) and the decay (DR) rates are at addresses 60H to 75H (1 per operator). The Sustain Level (SL) and Release Rate (RR) are located at addresses 80H to 95H. All of these values are 4 bits in length (range 0 to 15). Refer to the diagram on page 11 for more information.

The attack rate (AR) determines the rising time for the sound. The higher the value in this register, the faster the attack.

The decay rate (DR) determines the diminishing time for the sound. The higher the value in the DR register, the shorter the decay.

The sustain level (SL) is the point at which the sound ceases to decay and changes to a sound having a constant level. The sustain level is expressed as a fraction of the maximum level. When all bits are set, the maximum level is reached. Note that the EG-Type bit must be set for this to have an effect.

The release rate (RR) determines the rate at which the sound disappears after a Key-Off. The higher the value in the RR register, the shorter the release time.

## **BLOCK/F-Number**

These parameters determine the pitch of the note played. The Block parameter determines the octave while the F-Number (10 bits) further specifies the frequency. The following formula is used to determine the value of F-Number and Block:

$$F\text{-Num} = F_{\text{mus}} * 2^{(20-b)} / 49.716 \text{kHz}$$

In this formula,  $F_{\text{mus}}$  is the desired frequency (Hz) and "b" is the block value (0 to 7). Refer to Appendix C for a table of note frequencies.

The D5 bit in the register that contains the BLOCK information is called KEY-ON (KON) and determines if the specified voice (0 to 8) is enable (1) or disable (0). The lower bits of F-Number are at location A0H through A8H (1 per voice) and the 2 MSb are at positions D0 and D1 of addresses B0H to B8H.

REG	D7	D6	D5	D4	D3	D2	D1	D0
A0H-								F-Number
A8H	2 <sup>7</sup>	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
B0H-			KEY		Block			F-Number
B8H		ON		2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>		2 <sup>9</sup> 2 <sup>8</sup>

### Rhythm/AM Dep/VIB Dep

This register allows for control over AM and VIB depth, selection of rhythm mode and ON/OFF control for various rhythm instruments. Bit D5 (R) is used to change the mode from melodic (0) to percussive (1). When in percussive mode, bits D0 through D4 are the KEY-ON/KEY-OFF controls for the rhythm instruments listed below. The KEY-ON bit in registers B6H, B7H and B8H must always be 0 when in percussive mode.

D0	Hi-Hat
D1	Cymbal
D2	Tom-Tom
D3	Snare Drum
D4	Bass Drum

The AM Depth is 4.8dB when D7 is 1 and 1dB when 0. The VIB Depth is 14 cents when D6 is 1, and 7 cents when zero. (A "cent" is 1/100th of a semi-tone.)

### FeedBack/Connection

These two parameters influence the way the operators are connected together and the  $\beta$  factor in the feedback loop of the modulator. These parameters are assigned 1 per voice at locations C0H through C8H. The Connection bit (C) determines if the voice will be functioning in Additive synthesis mode (C = 1) or in Frequency modulation mode (C = 0). The other parameter, Feedback (FB), gives the modulation factor,  $\beta$ , for the feedback loop:

	0	1	2	3	4	5	6	7
$\beta$	0	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	$\pi$	$2\pi$	$4\pi$

### Wave Select

- The WS parameter enables the card to generate other kinds of wave shapes. This is done by changing the sine function of the specified operator. (Note that the WSE bit must be set in order to use this feature.) The addresses of this feature are E0H to F5H. The following figure gives the corresponding wave forms:

D1	D0	Wave Form
0	0	 Sine
0	1	 Half-sine
1	0	 Abs-sine
1	1	 Pulse-sine

# Programming the YMF262

This section explains the differences between the Ad Lib Gold Sound Adapter and the original Ad Lib Music Synthesizer Card as regards FM synthesis. A previous knowledge of the original Ad Lib card is assumed. If you are unfamiliar with the original card, you should first read the following section: "Programming the Synthesizer", which is reproduced from the original Programmer's Manual .

You can see from the register map on the following page that the new FM section is quite similar to the original FM chip but with extra features added. Register Array 0 is accessed by writing to addresses x and x+1 (388H and 389H by default). Register Array 1 is accessed by writing to addresses x+2 and x+3 (38AH and 38BH by default). This scheme allows for complete compatibility with older software which recognizes only the original Ad Lib card.

All registers are cleared at reset. The TEST registers at 01 should be cleared or not accessed at all. Bits in the register map which are not designated should be left in their cleared state.

## Register Array 0

Register Array 0 emulates the original chip and will be used as such by software written for the original card. However, there are several changes to be noted.

The Wave Select Enable bit (WSE, D5 at 01) no longer exists. Wave Select is now "on" permanently. Writing 1 to D5 at 01 has no effect so that compatibility is thereby maintained.

The CSM bit (D7 at 08) found on the original chip is no longer present. Although this bit was documented on the original chip, it was non-functional. Compatibility is, therefore, not an issue.

The timers are now functional. How to program them is explained in the *Timers* section of *Programming the Synthesizer*.

New FM Features

Register Map

Register Map, FM Array 0

REG	D7	D6	D5	D4	D3	D2	D1	D0			
01	TEST										
02	TIMER-1										
03	TIMER-2										
04	RST	T1	mask	T2				start/stop T2   T1			
05											
08		SEL									
20-35	AM	VIB	EG	KSR	MULTI						
40-55	KSL		TL								
60-75	AR				DR						
80-95	SL				RR						
A0-A8	F-NUMBER (L)										
B0-B8			KON	BLOCK			F-NUM (H)				
BD	DEP AM	DEP VIB	R	BD	SD	TOM	TC	HH			
C0-C8			SRL	STR	FB			C			
E0-F5						WS					

## Register Map, FM Array 1

REG	D7	D6	D5	D4	D3	D2	D1	D0
01								TEST
02								
03								
04								CONNECTION SELECT
05								NEW
08								
20-35	AM	VIB	EG	KSR				MULTI
40-55	KSL							TL
60-75	AR							DR
80-95	SL							RR
A0-A8								F-NUMBER (L)
B0-B8		KON		BLOCK				F-NUM (H)
BD								
C0-C8		SRL	STR		FB			C
E0-F5								WS

## New FM Features

### 4-Operator Voices

Each voice now has two bits which control stereo output: STL and STR (D5/D4 at C0-C8). Setting STL enables output to the left channel. Setting STR enables output to the right channel. Clearing both bits will result in no output for a given voice. However, for these bits to have effect, the NEW bit (explained in the next section) must be set. If NEW is not set (its default state), then the STL and STR bits are ignored and sound is output to both channels. This maintains compatibility with older software which ignores the existence of the stereo bits.

The stereo bits affect pairs of operators, which creates a particularity in percussive mode. The stereo bits in C7 simultaneously affect the Hi-Hat and Snare Drum; C8 affects the Tom-Tom and Cymbal similarly. The Bass Drum (C6) uses two operators and functions the same as a melodic voice.

The Wave Select has been expanded to 3 bits, thus allowing for a total of 8 different waveforms. The waveforms are shown below.

D2 - D0	Waveform
0	Sine
1	Half-sine
2	Abs-sine
3	Pulse-sine
4	Square
5	Triangle
6	Square
7	Exponential Decay

## Register Array 1

Register Array 1 is similar to Register Array 0 with some omissions and additions. The timer registers are unused or are used for other purposes. Register Array 1 does not offer percussive voices, so the bits relating to percussive mode are not present.

The SEL, DEP AM and DEP VIB bits are globally affective and so are found only in the first register array. Setting any one of these three bits will affect both register arrays.

The NEW bit (D0 at 05) enables the new features of the new chip. If this bit is zero, then writes to any other register in Register Array 1 will be blocked. When NEW is zero, Register Array 0 functions as if it were the original chip: the stereo bits will be ignored and the high bit of the wave select will be ignored.

**IMPORTANT:** All software should enable the NEW bit during its initialization sequence. However, it should clear the NEW bit when exiting. This is so that if an older piece of software is subsequently run, the card will be in the mode which emulates the original card.

The CONNECTION SELECT bits control the 4-operator voice, as explained in detail in the next section.

## 4-Operator Voices

A significant new feature of the FM section of the Ad Lib Gold card is the presence of 4-operator voices, which are capable of creating a large variety of rich timbres. To enable a 4-operator voice, you must set the appropriate bit in the CONNECTION SELECT register. The following table shows which bit corresponds to which 4-operator voice and the pair of 2-operator voices which correspond to the 4-operator voice.

Connection Select (05H, Register Array 1):

	D5	D4	D3	D2	D1	D0
4-op voice	6	5	4	3	2	1
2-op voices	3,6	2,5	1,4	3,6	2,5	1,4
	Array 1				Array 0	

## New FM Features

### 4-Operator Voices

With 2-operator voices, the connection bit at C0-C8 specifies one of two possible methods for connecting the operators. With 4-operator voices, there are 4 methods of connecting the operators. This is done by using both connection bits of the pair of 2-operator voices involved. The following table shows the relationship between the 4-operator voice and its connection bits. The diagram on the next page illustrates the connection methods.

Connection bit (C) addresses for 4-operator voices:

4-op voice	1	2	3	4	5	6
C addresses	C0,C3	C1,C4	C2,C5	C0,C3	C1,C4	C2,C5
Array 0				Array 1		

Note that even if all six 4-operator voices are used, there are still three 2-operator voices available on Register Array 1 and three 2-operator or five percussive voices available on Register Array 0. The CONNECTION SELECT register allows you to selectively use 4-operator voices so that you can mix 2 and 4-operator voices as you wish.

The following table is a combination of the preceding two tables. You may find it useful for reference purposes.

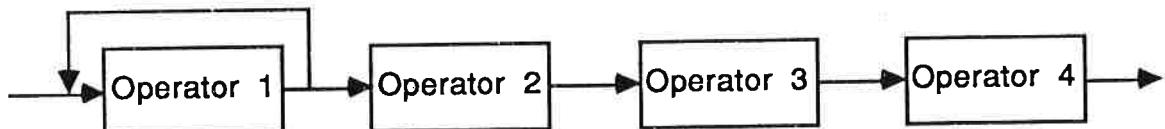
Connect Sel	D5	D4	D3	D2	D1	D0
4-op voice	6	5	4	3	2	1
2-op voices	3,6	2,5	1,4	3,6	2,5	1,4
C addresses	C2,C5	C1,C4	C0,C3	C2,C5	C1,C4	C0,C3
Array 1				Array 0		

Feedback in a 4-operator voice is applied to the first operator only, as indicated by the loop around Operator 1 in the diagram on the following page. The feedback value is determined by the value written in the register for the first register pair ( $Cx$ ). The value in the second register pair ( $Cx+3$ ) is ignored.

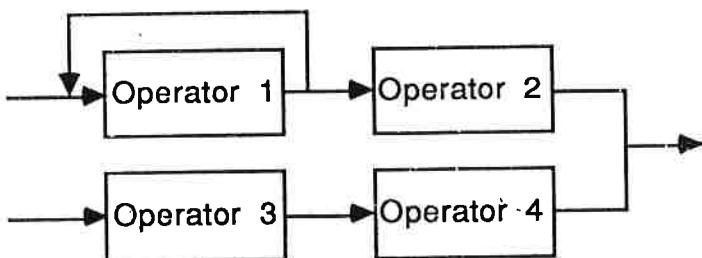
Similarly, the F-NUMBER, KON, and BLOCK parameters for a 4-operator voice are determined by the values written in the registers for the first register pairs ( $Ax$  and  $Bx$ ). The values in the second register pairs ( $Ax+3$  and  $Bx+3$ ) are ignored.

Note that the state of the STL and STR bits for a 4-operator voice must be the same for both register pairs ( $Cx$  and  $Cx+3$ ) or else the output of all four operators will be disabled. For example, if STL at C0 is 1 and STL at C3 is 0, then this 4-operator voice will not be output to the left channel.

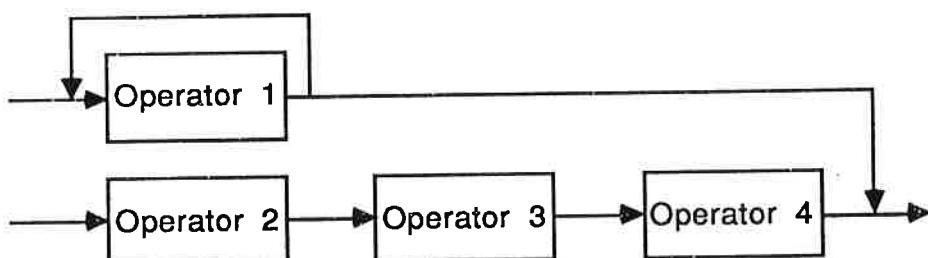
$Cx=0, Cx+3=0$



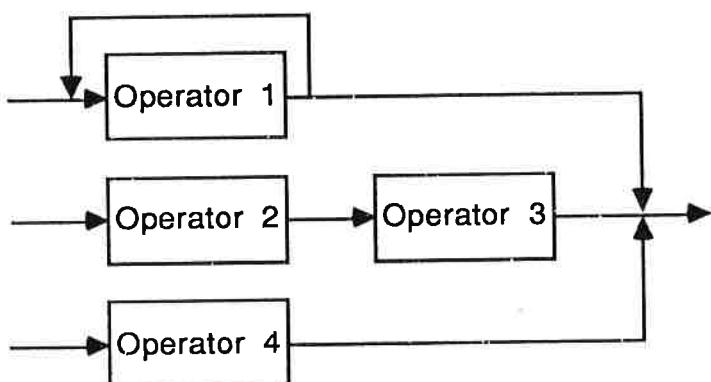
$Cx=0, Cx+3=1$



$Cx=1, Cx+3=0$



$Cx=1, Cx+3=1$



### Connection Methods



## 7.3

# Digital Input and Output (Digital Audio and MIDI)

---

The digital I/O functions are handled by the YMZ263 chip, also known as the MMA. The MMA handles the following functions:

- 2 channels of digital audio input and output
- MIDI input and output
- Three high-speed timers

The digital I/O functions are accessed via three addresses. The first address is located four bytes past the address of FM Array 0 (38CH by default).

Accessing a MMA register is done in two steps:

- 1) write the index of the register to be accessed to the "register select" port, located at 38CH
- 2) write or read the desired value for the selected register, either in the channel 0 port, located at 38DH or in the Channel 1 port located at 38FH

A 470 nanosecond delay is necessary between read/write at any address of the MMA

## Digital Input and Output

### Register Map

REG		D7	D6	D5	D4	D3	D2	D1	D0				
01	-	TEST											
02	W	TIMER-0 (L)											
03	W	TIMER-0 (H)											
04	W	BASE COUNTER (L)											
05	W	TIMER 1				BASE COUNTER (H)							
06	RW	TIMER 2 (L)											
07	RW	TIMER 2 (H)											
08	W	SBY	T2M	T1M	T0M	STB	ST2	ST1	ST0				
09	W	RST	R	L	FREQ		PCM	P/R	GO				
0A	W	VOLUME CONTROL											
0B	RW	PCM DATA											
0C	W	ILV	DATA FMT			FIFO INT			MSK				
0D	W			MSK POV	MSK MOV	MDI TRS	MSK RST	MDI RCV	MSK RRQ				
0E	RW	MIDI DATA											

Register Map, Channel 0

Digital Input and Output

Register Map

REG		D7	D6	D5	D4	D3	D2	D1	D0
01	-								
02	W								
03	W								
04	W								
05	W								
06	RW								
07	RW								
08	W								
09	W	RST	R	L	FREQ	PCM	P/R	GO	
0A	W	VOLUME CONTROL							
0B	RW	PCM DATA							
0C	W		DATA FMT		FIFO INT	MSK	ENB		
0D	W								
0E	RW								

Register Map, Channel 1

## **Register Reference**

### **Status Register**

Reading the port at address 38CH returns the following information:

D7	D6	D5	D4	D3	D2	D1	D0
OV	T2	T1	T0	TRQ	RRQ	FIF1	FIFO

Status Byte

OV becomes 1 when a MIDI receive overrun error or a PCM/ADPCM record or playback overrun error occurs.

T0, T1 and T2 become 1 when the specified time elapses in the corresponding timer.

TRQ becomes 1 when the MIDI transmit FIFO buffer is empty.

RRQ becomes 1 when the MIDI receive FIFO buffer has data in it.

FIFO and FIF1 become 1 when the PCM/ADPCM FIFO reaches the status that was specified in FIFO INT. FIFO corresponds to channel 0; FIF1 to channel 1.

### **Register 00H: Test Register**

Register #1, Channel 0 is used for testing the LSI. It should not be accessed.

## Registers 02H - 07H: Timer Counters

Timer 0 (Registers #1 and 2, Channel 0) is a 16-bit programmable down counter with 1.88964 usec resolution. This constant will be referred to as **clockFreq**. In the following examples. The interrupt is triggered when the counter value reaches 0. The time **t0**, in usec, until IRQ is generated may be calculated as follows:

$$t0 = \text{TIMER0(H)} * (256 * \text{baseFreq}) + \text{TIMER0(L)} * \text{baseFreq}$$

The BASE COUNTER (Register #4 and 5, Channel 0) is a 12-bit counter that supplies the period for each tick of TIMER1 and TIMER2. The base counter has a resolution of 1.89 usec. The period **bc**, in usec, may be calculated as follows:

$$bc = \text{BASE COUNTER(H)} * (256 * \text{baseFreq}) + \text{BASE COUNTER(L)} * \text{baseFreq}$$

Timer 1 (Register #5, Channel 0) is a 4-bit programmable down counter that is controlled by the base counter clock. The 4-bit value is placed in the high nibble of the register. The interrupt is triggered when the counter value reaches 0. The time **t1**, in usec, until IRQ is generated may be calculated as follows:

$$t1 = \text{TIMER1} * bc$$

Timer 2 (Register #6 and 7, Channel 0) is a 16-bit programmable down counter that is controlled by the base counter clock. The interrupt is triggered when the counter value reaches 0. The time **t2**, in usec, until IRQ is generated may be calculated as follows:

$$t2 = (\text{TIMER2(H)} * 256 + \text{TIMER2(L)}) * bc$$

TIMER2 may be read to determine the count value. When TIMER2(L) is read the 16-bit count value is latched and the latched value of TIMER2(L) is output. Subsequently, when TIMER2(H) is read, the latched value of TIMER2(H) is output. (Latching a value means taking a "snapshot" of that value at a given moment.) TIMER2(L) must be read first as it is this read which triggers the latching mechanism.

## Digital Input and Output

### Select Frequency

## Register 08H: Timer Control

D7	D6	D5	D4	D3	D2	D1	D0
SBY	T2M	T1M	T0M	STB	ST2	ST1	ST0

Register #8: Channel 0

### Stand-by Mode

Setting SBY to 1 reduces the internal clock frequency in order to minimize power consumption. This must be set to 0 when doing any I/O operations.

### Timer Interrupt Masks

Setting T0M, T1M or T2M disables the interrupt generated by the corresponding timer. Hence, the bit must be cleared if you wish to use the interrupt timer.

### Timer Controls

ST0, ST1, ST2 and STB (base counter) control the start and stop of each timer. Setting a bit loads the reload value and starts counting down. Clearing the bit stops the timer.

## Register 09H: Playback and Recording Control

D7	D6	D5	D4	D3	D2	D1	D0
RST	R	L	FREQ	PCM	P/R	GO	

Register #9: Channels 0 & 1

### Reset PCM/ADPCM

RST bit is used to reset PCM and ADPCM playback for the channel. Resetting a channel clears the FIFO buffers and resets the FIFO flags. In order for reset to operate properly, all other bits should be 0. The sequence for a channel reset should then be: 1) write 80H to register 9 2) write the desired values to register 9.

### Select Output Channel

Setting L or R enables output from the left or right channel respectively. Clearing the bit disables output.

### Select Frequency

FREQ selects the PCM/ADPCM frequency as indicated below:

FREQ	Sampling Frequency (KHz.)	
	PCM Mode	ADPCM Mode
0	44.1	22.05
1	22.05	11.025
2	11.025	7.35
3	7.35	5.5125

### PCM/ADPCM Selection

Setting PCM selects PCM mode (data is not compressed). Clearing PCM selects ADPCM mode (data is compressed to 4-bits).

### Select Record/Playback

Clear P/R to record; set it to playback.

### Start/Stop Record/Playback

In playback, the FIFO buffers should never be empty when the GO bit is set. To start playback, the proper procedure is: 1) write data into the FIFO buffer for the channel. The FIFO should be filled to a level exceeding the FIFO interrupt level (see register 0CH description) 2) Set the GO bit to start playback.

## Register 0AH: Output Volume Control

VOLUME CONTROL (Register #0Ah, both channels) sets the output attenuation value. A value of 0 is the minimum output volume, a value of FF is the maximum output volume.

## Register 0BH: PCM/ADPCM Data

Register #0Bh (both channels) is used for writing data into the FIFO buffer and reading data from the FIFO buffer. . Each channel has its own buffer. Data written into this register is transferred into the FIFO buffer, and data transferred from the FIFO buffer is written into this register. In PCM mode, 12-bit data is accessed in one or two passes. The data format for this access follows the specification of the FORMAT register. In ADPCM mode, each access inputs or outputs two 4-bit data. The high 4 bits and the low 4 bits are each ADPCM data. The high data is followed immediately by the low data.

## Register 0CH: Sampling Format and Control

D7	D6	D5	D4	D3	D2	D1	D0
ILV	DATA FORMAT		FIFO INT		MSK	ENB	

Register #0Ch: Channels 0 & 1

### Interleaving

Setting ILV (Channel 0 *only*) to 1 will cause the chip to do interleaving. Data will be alternately input/output from each channel. Channel 0 initiates the transfer. ENB must be 1 for both channels, otherwise the data transfer is not performed. Both channels operate in the same mode so that the P/R,FREQ and GO bits will be controlled by the values set for channel 0.

### Set Data Format

There are 3 possible data formats for sampling input and output. The format is selected by writing 0, 1 or 2 to the DATA FORMAT register. "3" is an invalid format... This is ignored in ADPCM mode.

Format 0 is an 1-byte format which contains the 8 most significant bits of the sample.

Format 1 is a 2-byte format. The first byte contains the 8 least significant bits. The lower nibble of the second byte contains the 4 most significant bits of the sample. The MSB of the sample is repeated in all bits of the upper nibble.

Format 2 is a 2-byte format as well. The upper nibble of the first byte contains the 4 LSBs of the sample. The lower nibble is zero. The second byte contains the 8 MSB's.

FORMAT	PCM Data Byte 1	PCM Data Byte 2
0	MSB b10 b9 b8 b7 b6 b5 b4	There is no 2nd byte
1	b7 b6 b5 b4 b3 b2 b1 b0	MSB MSB MSB MSB MSB b10 b9 b8
2	b3 b2 b1 b0 0 0 0 0	MSB b10 b9 b8 b7 b6 b5 b4

PCM Data Formats

### Set FIFO Interrupt

The FIFO INT register is used to specify when an interrupt will be generated while the 128-byte FIFO buffer is being filled or emptied. The following table documents the possible interrupt points.

FIFO INT	Interrupt Generation Point (bytes)
0	112
1	96
2	80
3	64
4	48
5	32
6	16
7	Prohibited

### FIFO Interrupt Mask

Setting MSK disables the FIFO interrupt.

**DMA Mode Specification**

Set ENB to enable the DMA mode. Clear ENB when not using DMA to transfer data.

**Register 0DH: MIDI and Interrupt Control**

D7	D6	D5	D4	D3	D2	D1	D0
		MSK POV	MSK MOV	MDI TRS RST	MSK TRQ	MDI RCV RST	MSK RRQ

Register #0Dh: Channel 0

**Mask Digital Overrun Error**

Set POV to disable interrupt signals generated by overrun errors during PCM/ADPCM recording and playback.

**Mask MIDI Overrun Error**

Set MOV to disable interrupt signals generated by overrun errors during MIDI reception or transmission.

**Reset MIDI transmit circuit**

Set MDI TRS RST to 1 to reset the MIDI transmit circuit and clear the MIDI transmit FIFO buffer. Zero MDI TRS RST to terminate the reset status.

**Mask MIDI transmit FIFO Interrupts**

Set MSK TRQ to disable interrupt signals generated by the MIDI transmit FIFO. When interrupts are enabled, an interrupt is generated when the MIDI transmit FIFO buffer is emptied.

**Reset MIDI Receive Circuit**

Set MDI RCV RST to 1 to reset the MIDI receive circuit and clear the MIDI receive FIFO buffer. Zero MDI RCV RST to terminate the reset status.

**Mask MIDI Receive FIFO Interrupts**

Set MSK RRQ to disable interrupt signals generated by the MIDI receive FIFO buffer. When interrupts are enabled, an interrupt is generated on reception of a MIDI byte.

**Register 0EH: MIDI Data**

This register is used for writing data into the MIDI FIFO buffer and reading data from the MIDI FIFO buffer. Data written in this register is transferred to the transmit FIFO buffer and data transferred from the receive FIFO buffer can be read from this register.

## MMA Programming Tips

- Reset a MMA channel after each sample (using the RST bit in register 9), after stopping the sample playback. This makes sure that the FIFO buffer for the channel is emptied.
- In playback mode, when processing a FIFO interrupt, a situation occurs where your application is filling in the FIFO while the playback mechanism is emptying the FIFO at the same time. In some cases this can cause "false triggers" of the FIFO interrupt. In order to avoid this, a simple trick is to temporarily lower the FIFO level, while your application fills in the FIFO, and restore the original level before leaving the interrupt procedure.
- A similar situation can occur in recording mode.
- To avoid the same situation during playback and recording using DMA transfers, you can double-check if the interrupt is valid by reading the DMA controller's counters or status register. they should indicate that data transfer is over.
- The MMA FIFO buffers should never be left to empty themselves during playback (this is when GO bit is set) This implies that the FIFO buffers should be filled to a level exceeding the FIFO interrupt level before the GO bit is set.

Special care should be taken during high-speed transfers (44.1K, 12 bit stereo samples, for example) on slower computers.

- All masks (mask T2, T1, T0, FIFO, POV, MOV, TRQ and RRQ) have no effect whatsoever on the status register. They are only used to disable the hardware interrupt.
- Respect the 470ns delay between writes to the MMA registers.



---

*YAMAHA*

# Gold Sound Standard

March 17, 1992

---



# Contents

<b>Introduction .....</b>	<b>1</b>
<b>Overview .....</b>	<b>2</b>
<b>GSS Implementation.....</b>	<b>3</b>
MMA: Digital Audio, MIDI, and Game Port .....	4
OPL3: FM Synthesis .....	6
Mixer and Set-up Section .....	9
Software Issues .....	11
<b>Mixer and Set-up Function Implementation .....</b>	<b>12</b>
Access Method .....	12
Status Register .....	13
Index Register Map .....	14
Register Reference .....	15
<b>Conclusion .....</b>	<b>28</b>



## Introduction

The rapid evolution of multimedia has necessitated an audio standard to be defined. Although Windows alleviates some of the need for compatibility, it is important that an audio standard is established for DOS, game, and "edutainment" applications. The implementation of an audio multimedia standard lessens the concerns of both hardware and software developers.

This document describes recommended procedures and practices for implementing multimedia audio hardware using the "Magic" chip set from YAMAHA. By conforming to the Gold Sound Standard, hardware manufacturers can be assured that software written for Gold Sound Standard compatible cards will run on their product.

The Gold Sound Standard is a hardware implementation specification, as well as the requirements of hardware compatibility at the register level for the mixer and set up functions. This ensures that software, which writes directly to the hardware, will run on any Gold Sound Standard implementation. This also means that any Gold Sound Standard driver kit for DOS or Windows will be capable of driving any Gold Sound Standard hardware. The Gold Sound Standard provides a safe development path for both software and hardware designers.



## Overview

The Gold Sound Standard (GSS) is composed of the YAMAHA "Magic" chip set and a form of mixer and set up circuitry. In the case of the YAMAHA "Magic" chip set, this document will summarize its functions. A more detailed reference for the individual registers of the "Magic" chip set may be found in the YAMAHA reference manual for the particular chip.

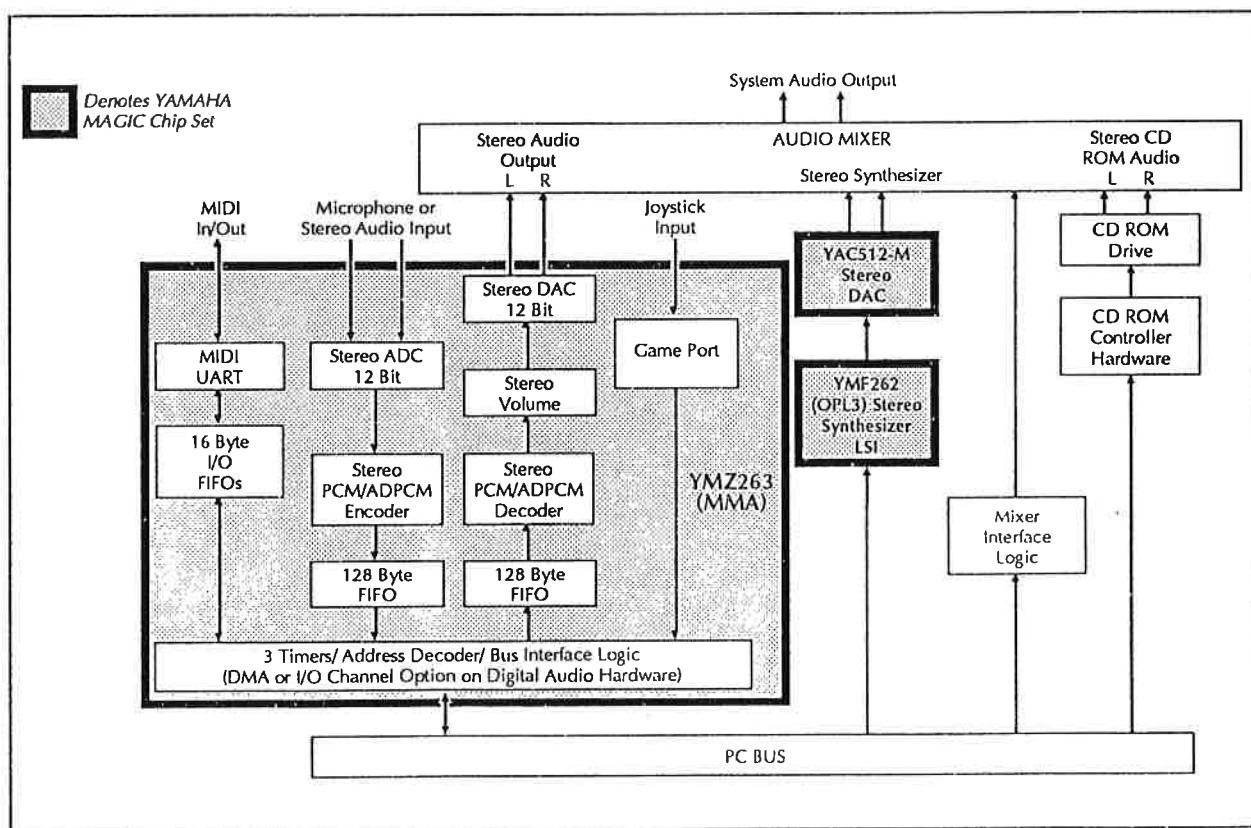
Only the minimum requirements are defined in this document. The individual hardware designer may implement additional features. The GSS provides the necessary functionality to be Level 1 MPC compatible.

The "Magic" chip set is designed as a highly integrated solution to developing a Level-1 MPC compatible audio subsystem. The following sections illustrate design concerns when using the "Magic" chip set. They also provide the I/O register map requirements to be Gold Sound Standard compatible.



## GSS Implementation

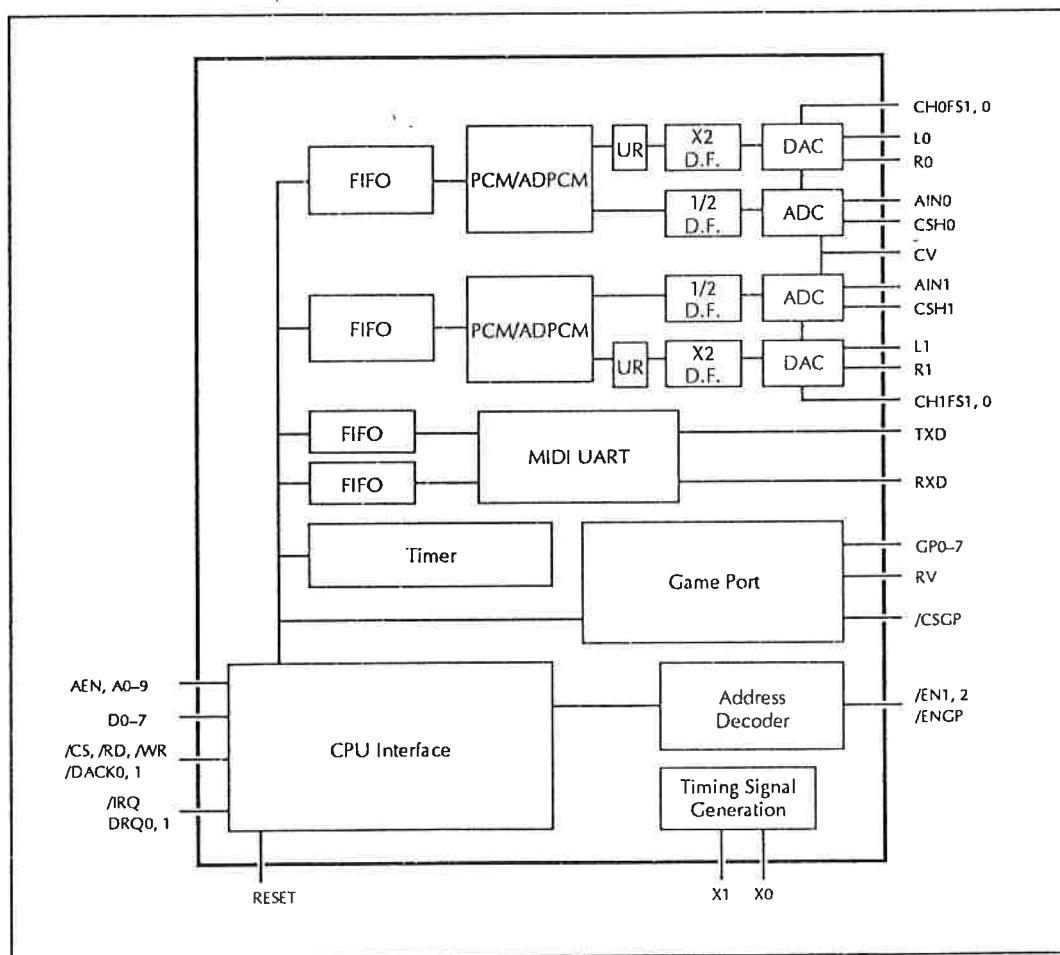
The basic features of GSS compatible hardware are a MIDI port, microphone input, stereo input/output, joystick input, and a mixer to produce the stereo audio output. Optionally, a SCSI interface may be implemented for use with CD-ROM drives. A hardware design conforming to the GSS will allow software that directly accesses the hardware to run on all GSS compatible cards. The design concerns are minimized by implementing the "Magic" chip set, which includes the YMZ263 (MMA) Multimedia Audio LSI, the YMF262 (OPL3) Advanced Algorithm Synthesizer LSI, and the YAC512-M Stereo Serial DAC. The following figure illustrates the hardware components of a typical Level 1 multimedia PC using the "Magic" chip set.





## MMA: Digital Audio, MIDI, and Game Port

The MMA integrates a stereo digital audio, game port, and MIDI interface into one LSI. The MMA also contains internal bus decode logic, two DMA channels, and two FIFOs. The internal block diagram illustrates the various portions of control circuitry.





The CPU interface is directly connectable to the address, data, and I/O control lines of the PC bus. The interrupt line is connected to the PC bus and may either be asserted when there is data in the input FIFOs, when the output FIFOs are able to receive more data, or if a timer interrupt is generated.

The DMA channels may be programmed to provide two methods of operation, allowing simultaneous record and playback. The first method uses a separate DMA channel for each channel. The second method is to interleave channel information using one DMA.

A PAL device may be used to decode jumper block settings, allowing DMA channel and IRQ level selection.

The direct ISA bus interface of the MMA contains an address decoder for built in fixed addresses. The I/O address the MMA will respond to is determined by the state of the /EN1, /EN2, and /ENGP signals. The recommended default I/O address for the MMA is from 38CH to 38FH (Channel 0: 38CH-38DH, Channel 1: 38EH-38FH). The MMA uses two port addressing for each channel. The first address of a channel is the address register, which is used to access the desired internal register. The second register is the data register. The data written to this register will be sent to the register specified by the index written to the address register.

The two inputs to the wave audio section of the MMA are a low impedance stereo microphone or a high impedance stereo audio. These inputs are A/D converted at twice the selected sampling frequency and decimated before being fed into the PCM/ADPCM encoder. External capacitors are attached to pins CSH1 and CSH2, which stabilize the analog signals during sample hold operations. A reference center voltage for the A/D converter is supplied through the CV pin of the MMA.

The buffered PCM/ADPCM decoder output is amplified and over sampled at twice the selected frequency (except for 44.1 kHz PCM mode) before passing through the DAC. The DAC output signals are fed through a series of operational amplifiers ending with a low pass reconstruction filter before final output.

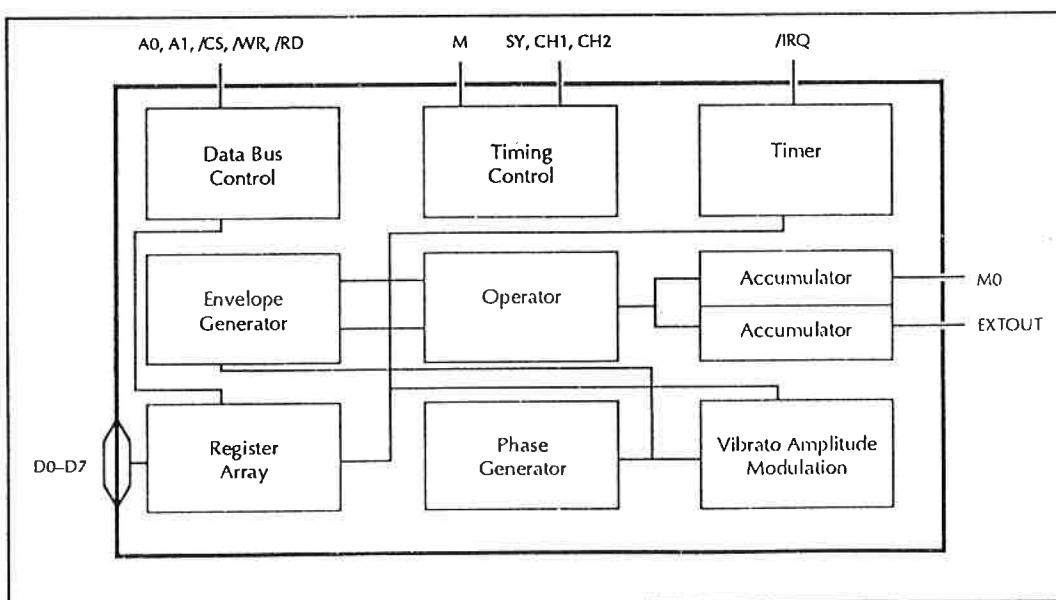


The MMA contains a MIDI subsystem with three timers, an asynchronous UART, and two 16 byte FIFOs for sending and receiving MIDI data. The MIDI output channel (TXD) is inverted and sent through an external 5 mA current loop to the external MIDI output connector. The external MIDI input connector's RXD signal is optically isolated to avoid ground loops. An optional MIDI thru port may be added by transferring the MIDI input signal through two inverters out an additional MIDI output port.

The game port interface of the MMA uses internal voltage comparison circuitry to isolate changes in the signals from the game port connector.

## OPL3: FM Synthesis

The FM synthesis is produced by the OPL3 and uses the YAC512-M stereo DAC for output. The OPL3 is backward compatible with the YM3812 (OPL2) used in most popular PC audio cards, yet offers much higher quality synthesized sound. The OPL3 was designed to be directly controlled by software. The following diagram illustrates the internal functions of the OPL3.

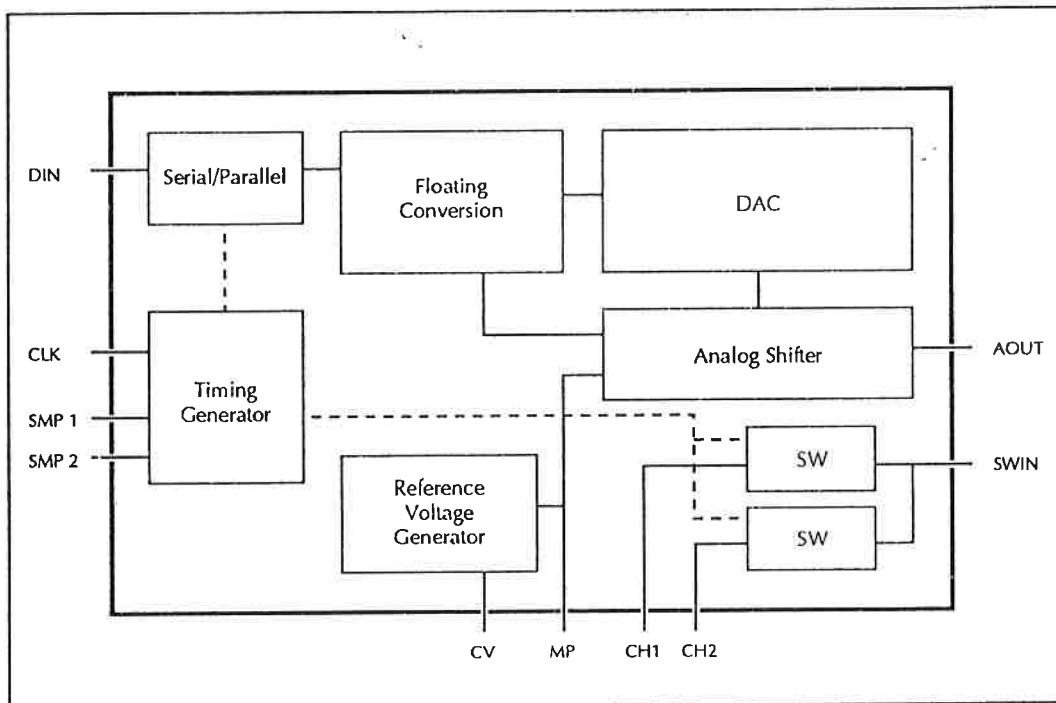




The OPL3 requires a 14.3127 MHz oscillator, which is taken directly from the PC bus. The address, data, and I/O control lines are also taken directly from the PC bus.

The recommended default base address of the OPL3 is 388H. The OPL3 also uses two port addressing for each channel and the internal register access is identical to the procedure used with the MMA. Channel 0 will be accessed at 388H and Channel 1 at 38AH.

The internal block diagram of the YAC512-M is shown below.

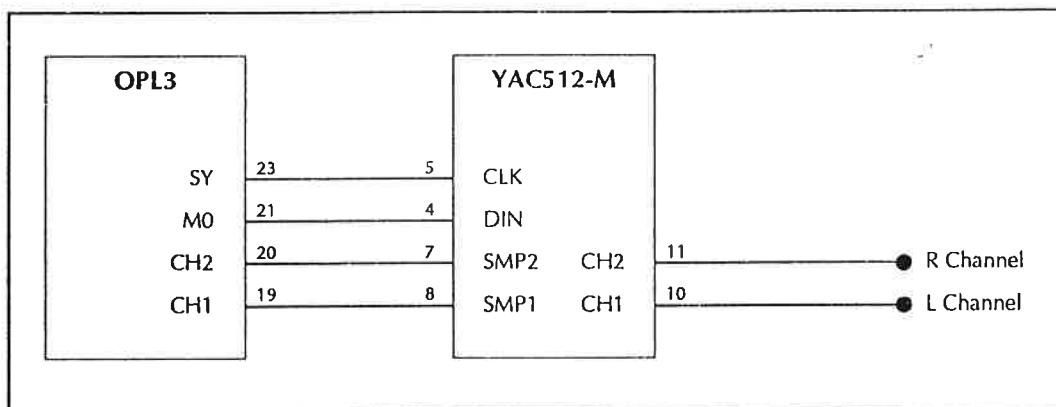


The serial data output and sample/hold channel signals are directly connected to the YAC512-M. The serial data between the OPL3 and the YAC512-M is synchronized using the OPL3's SY clock signal. This data is then latched when the channel sampling lines of the OPL3 fall.



The data is converted to floating point with the mantissa being processed by the DAC and the exponent by the analog shifter. This process produces effectively 16 bit resolution . The data is then converted into a D/A voltage which is sent out the AOUT terminal. This signal passes through a buffer operational amplifier for sample holding and is used for channel 1/2 common input.

The YAC512-M converts the digital serial stream into an analog signal. The YAC512-M requires external capacitors for stabilizing the analog output. The stereo output of the YAC512-M is ready for mixing with the MMA's stereo audio output. The block diagram below illustrates the simple connections between the OPL3 and the YAC512-M.





## Mixer and Set-up Section

The mixer section is where the hardware designer may differentiate their audio implementation. The Gold Sound Standard implementation allows software to control the individual volumes of the mixer inputs.

The GSS allows for relocation of the MMA and OPL3. Software should not assume an absolute address. This document will use the default base address of the OPL3 (388H) and the MMA (38CH). The mixer section may be accessed by outputting an invalid index address to the second channel address register of the OPL3.

This access method uses the second channel of the OPL3 for the mixer registers. When the mixer registers are enabled, an index into the internal mixer registers is written to 38AH, or OPL3 base address +2, with the desired register data written to 38BH, or OPL3 base address +3.

In order to avoid the problem of hardware reentrance when accessing the mixer registers the software must push the flags and disable interrupts. Then by writing a value of FFH to port 38AH, or OPL3 base address +2, the hidden mixer registers will appear on top of the second channel of the OPL3.



The next output to port 38AH or OPL3 base address +2 will be an index into the internal registers. The specified internal register is ready for read/write operations. Multiple reads or writes may be made without continually resetting the address register. Once all accesses are complete, the internal register access is disabled by writing a FEH to port 38AH, or OPL3 base address +2. This access method is illustrated by the listing below.

```
PUSHF           ; Push the CPU status flags.  
CLI             ; Disable Interrupts  
OUT  OPL3_base +2, FFH   ; Switch to mixer register access  
OUT  OPL3_base +2, 02H   ; Select register 2, Left Smpl Gain  
OUT  OPL3_base +3, 34H   ; Set gain level to 34H  
    . . .  
OUT  OPL3_base +2, FEH   ; Close access to internal mixer  
                          ; registers.  
  
POPF           ; Restore interrupt status.
```

This approach minimizes the address space occupied by the GSS implementation and reduces the risk of hardware conflicts with other resources.

The implementation allows access to the address and data I/O registers of the first channel of the OPL3, while accessing the internal mixing registers. The address decoding being handled by the mixer section should have no influence on the operation of the second channel of the OPL3.

As is the case with all other aspects of the GSS hardware, special care must be taken for the possibility of reentrance when separate applications access the mixer section and the OPL3 chip simultaneously. This is discussed in the section entitled "Software Issues" below.



## Software Issues

In the GSS architecture, hardware reentrance may be possible. Applications may be using the MMA to record and play back samples while the timer or MIDI functions are being used by other applications. Multiple operations are required to access the OPL3 and the MMA. There is a possibility of an interrupt occurring during these accesses, which would cause problems.

In a single-task system (such as DOS), this reentrance may be handled by disabling interrupts during accesses to the GSS hardware.

This reentrance may be handled through the use of an asynchronous queue manager, which would handle mixer register requests when appropriate.

The GSS hardware implementation will allow any standard PCM wave data files to be played, provided it is sampled at the supported frequencies of the MMA.

A method of verifying GSS audio hardware is to write a signature value to the PCM data registers of Channel 1 and 2 of the MMA. If the value read back matches the value written, the audio hardware is GSS compatible.



## Mixer and Set-up Function Implementation

The internal mixing register will have read/write capability with the exception of the supported features register. The supported features register will be read only.

### Access Method

The mixer and set-up registers, when enabled, use two port addressing like the MMA and OPL3. The first address being an index into the register map with the next address for data.

A delay of 450 microseconds is recommended after writing to the output volume registers (04H – 08H).

A delay of 5 microseconds is recommended after writing to the mixing volume registers and channel/IRQ registers (09H – 15H).

In order to ensure these delays, the mixer access status register should be polled.

While writing the mixer registers to memory the mixer must not be accessed. This is achieved by polling the mixer register write status bit, checking for a "0".



## Status Register

The index address register of the mixer is used as a status register. Reading the status register returns information as defined in the following figure.

Status Register								
D7	D6	D5	D4	D3	D2	D1	D0	Function
							X	Indicates FM Interrupt
						X		Indicates Sampler Interrupt
		X	X	X	X			Reserved
	X							Mixer Register Access Status
X								Mixer Register Write Status

**D0.** This bit provides interrupt information for the FM portion. When bit D0 is set to 0, it indicates the FM portion of the hardware has generated an interrupt.

**D1.** This bit provides interrupt information for the sampling portion. When bit D1 is set to 0, it indicates that the sampling portion of the hardware has generated the interrupt.

**D2-D4** These bits are reserved.

**D6.** This bit indicates the mixer register write status. If bit D6 is set the card is currently writing to a mixer register.

**D7.** This bit indicates the card is currently writing its registers to memory. This is useful for notebook and portable devices utilizing power saving features.



## Index Register Map

The following figure defines the mixer index and data register map. The index value is written to the address register of the mixer to access the desired register. The data is then written to the data register of the mixer.

Mixer and Set-up Index Register Map		
Index	Read/Write	Description
00	R	ID/Feature Register
02	R/W	Left Channel Sampling Gain
03	R/W	Right Channel Sampling Gain
04	R/W	Left Channel Output Volume
05	R/W	Right Channel Output Volume
06	R/W	Bass Output
07	R/W	Treble Output
08	R/W	Output Mode
09	R/W	Left Channel FM Volume
0A	R/W	Right Channel FM Volume
0B	R/W	Left Sampling Volume
0C	R/W	Right Sampling Volume
0D	R/W	Left Auxiliary Volume
0E	R/W	Right Auxiliary Volume
0F	R/W	Microphone Volume
11	R/W	Audio Selection
13	R/W	Audio IRQ/DMA Select—Channel 0
14	R/W	DMA Select Channel 1
15	R/W	Audio Relocation



The register map is a complete set of registers required to be compatible with the GSS. These are the minimum features required for mixer implementation on a GSS compatible audio card.

## Register Reference

### *0H: ID/Feature Register*

This read-only register provides information on supported features of the board. The bit functions are identified in the following figure.

0H: ID/Feature Register								
D7	D6	D5	D4	D3	D2	D1	D0	Function
			X	X	X	X	X	Reserved
		X						Surround Sound Option
	X							SCSI Option
X								Reserved = 1

### *02H: Left Channel Sampling Gain*

The left channel sampling gain is controlled by values written to this register. There are 256 possible values. The amount of gain and step is dependent on the mixer implementation. The recommended gain implementation is computed by the following equation:

$$\text{Gain} = (\text{Register Value} * 10) / 256$$



The bit functions of this register are identified in the following figure.

02H: Left Channel Sampling Gain								
D7	D6	D5	D4	D3	D2	D1	D0	Function
X	X	X	X	X	X	X	X	Left Channel Output Gain

### 03H: Right Channel Sampling Gain

The right channel sampling gain is controlled by values written to this register. There are 256 possible values. The amount of gain and step is dependent on the mixer implementation. The recommended gain implementation is computed by the following equation:

$$\text{Gain} = (\text{Register Value} * 10) / 256$$

The bit functions of this register are identified in the following figure.

03H: Right Channel Sampling Gain								
D7	D6	D5	D4	D3	D2	D1	D0	Function
X	X	X	X	X	X	X	X	Right Channel Output Gain



**04H: Left Channel Output Volume**

This read/write register controls the overall left channel output volume. The bit functions of this register are identified in the following figure.

04H: Left Channel Output Volume								
D7	D6	D5	D4	D3	D2	D1	D0	Function
		X	X	X	X	X	X	Left Channel Output Volume
X	X							Reserved = 1

The left channel volume may range from 0 to 64 dB. The recommended volume range is from +6 dB to -64 dB, in two dB steps. The decibel values are listed in the following figure.

Decibels	D5–D0
6	3F
-62	1D
-80	0



**05H: Right Channel Output Volume**

This read/write register controls the overall right channel output volume. The bit functions of this register are identified in the following figure.

05H: Right Channel Output Volume								
D7	D6	D5	D4	D3	D2	D1	D0	Function
		X	X	X	X	X	X	Right Channel Output Volume
X	X							Reserved = 1

The right channel volume may range from 0 to 64 dB. The recommended volume range is from +6 dB to -64 dB, in two dB steps. The decibel values are listed in the following figure.

Decibels	D5-D0
6	3F
-62	1D
-80	0



**06H: Bass Output**

This read/write register controls the bass output with a range of values from 0 to 16. The bit functions of this register are identified in the following figure.

06H: Bass Output								
D7	D6	D5	D4	D3	D2	D1	D0	Function
				X	X	X	X	Bass Output
X	X	X	X					Reserved = 1

The recommended decibel range is from +15 dB to -12 dB, in 3 dB steps. The decibel values are listed in the following figure.

Decibels	D3-D0
15	F
15	B
0	6
-12	2
-12	0



**07H: Treble Output**

This read/write register controls the treble output with a range of values from 0 to 16. The bit functions of this register are identified in the following figure.

07H: Treble Output								
D7	D6	D5	D4	D3	D2	D1	D0	Function
				X	X	X	X	Treble Output
X	X	X	X					Reserved = 1

The recommended range is from +12 dB to -12 dB, in 3 dB steps. The decibel values are listed in the following figure.

Decibels	D3-D0
12	F
12	A
0	6
-12	2
-12	0



**08H: Output Mode**

This read/write register controls the final output. The final output uses the input and output of the mixer. The bit functions of this register are identified in the following figure and defined below.

08H: Output Mode								
D7	D6	D5	D4	D3	D2	D1	D0	Function
					X	X	X	Source of Final Output
			X	X				Type of Effect
		X						Mute
X	X							Reserved = 1

**D2–D0.** These bits determine the channels to be selected for the final output. If only one output channel is selected, it will be directed out to both channels. The following figure defines the signal configurations.

D2	D1	D0	Channels
1	1	0	Left and Right
1	0	0	Right Only
0	1	0	Left Only

**D4–D3.** These bits determine the output effect. The following figure defines the signal configurations.



D4	D3	Type of Effect
1	1	Spatial Stereo
1	0	Pseudo Stereo
0	1	Linear Stereo
0	0	Forced Stereo

D5. This bit enables or disables mute.

D7-D6. These bits are reserved and set to 1.

#### 09H – 0FH

Registers 09H through 0FH are the individual mixing controls, and comprise the mixer section of the audio card. The following figure provides descriptions for these registers.

09H — 0FH	
Register	Description
09H	Left Channel FM Volume
0AH	Right Channel FM Volume
0BH	Left Sampling Volume
0CH	Right Sampling Volume
0DH	Left Auxiliary Volume
0EH	Right Auxiliary Volume
0FH	Microphone Volume



There are 128 possible linear volume levels, ranging from silent (80H) up to a maximum gain (0FFH). If values less than 80H are written to this register, a negative voltage signal (negative polarity) would result. This may cancel out another signal and should be avoided. The following figure specifies the volume range.

Value	Volume Range
FFH	Maximum Volume
80H	Minimum Volume
00H	Negative Maximum Volume

#### 11H: Audio Selection

This read/write register controls the antialiasing filters (input/output) and the auxiliary input. The same antialiasing filter is used for sampling and playback. The bit functions of this register are identified in the following figure and defined below.

11H: Audio Selection								
D7	D6	D5	D4	D3	D2	D1	D0	Function
							X	Right Channel Filter
						X		Left Channel Filter
					X			Auxiliary Input Control
		X						Internal Speaker Mixer
X	X		X	X				Reserved



**D0.** This bit is used to set the filter for Right Channel. When this bit is set to 1, the filter is set for recording. When set to 0, the filter is set for playback.

**D1.** This bit is used to set the filter for Left Channel. When this bit is set to 1, the filter is set for recording. When set to 0, the filter is set for playback.

**D2.** This bit controls the auxiliary input. When this bit is set to 1 forces the stereo input to monophonic to be sampled on Left Channel. When set to 0, the auxiliary input to stereo is restored.

**D3–D4.** These bits are reserved.

**D5.** This bit enables or disables the internal speaker of the PC to be mixed with the final audio output.

**D7–D6.** These bits are available.

#### *13H: Audio IRQ/DMA Select—Left Channel*

This read/write register controls the interrupt and DMA functionality of the FM and sampling features. The bit functions of this register are identified in the following figure and defined below.

13H: Audio IRQ/DMA Select –Channel 0								
D7	D6	D5	D4	D3	D2	D1	D0	Function
					X	X	X	Select Interrupt
				X				Enables Audio Interrupt
	X	X	X					Selects DMA
X								Enables Left Channel DMA



**D2–D0.** These bits control the IRQ selection, as defined in the following figure.

Interrupt Select	IRQ
0	3
1	4
2	5
3	7
4	10
5	11
6	12
7	15

**D3.** When this bit is set to 1, audio interrupts are enabled.

**D6–D4.** These bits select the DMA line for Left Channel, as defined in the following figure.

DMA Select	DMA Line
0	0
1	1
2	2
3	3

**D7.** When this bit is set to 1, the DMA for Left Channel is enabled.



***14H: DMA Select Right Channel***

The DMA select Right Channel register controls the DMA for Right Channel and is identical to the Left Channel register except for the IRQ information. The bit functions of this register are identified in the following figure and defined below.

14H: DMA Select Channel 1								
D7	D6	D5	D4	D3	D2	D1	D0	Function
			X	X	X	X	X	Reserved
	X	X						Selects DMA
X								Enables Right Channel DMA

**D4–D0.** These bits are reserved.

**D6–D5.** These bits select the DMA line for Right Channel, as shown in the following figure.

DMA Select	DMA Line
0	0
1	1
2	2
3	3

**D7.** When this bit is set to 1, the DMA for Right Channel is enabled.



***15H: Audio Relocation***

The audio relocation register provides the flexibility of relocating the I/O map of the FM banks and the sampling channels. The value written to this register is the port address divided by eight, which will force the location to be on an even byte boundary. The OPL3 and MMA use eight I/O ports, the desired base address divided by eight is the value written to this register. The value written will immediately relocate the audio functions. The bit functions of this register are identified below.

15H: Audio Relocation								
D7	D6	D5	D4	D3	D2	D1	D0	Function
	X	X	X	X	X	X	X	Audio Relocation Address
X								Reserved

The following figure provides the recommended default addresses.

Address	Section
388H, 389H	FM Bank 0
38AH, 38BH	FM Bank 1
38CH, 38DH	Sampling Left Channel
38EH, 38FH	Sampling Right Channel



## Conclusion

The Gold Sound Standard completely defines the requirements of hardware compatibility on the register level. This low level of compatibility provides the software developer, who writes directly to the hardware, with a large variety of implementations based on a common audio platform. The Gold Sound Standard is a minimum implementation standard, offering a safe migration path for today's hardware and software designs.

For more information on the "Magic" chip set and implementing the Gold Sound Standard, call your local YAMAHA representative or contact YAMAHA at:

981 Ridder Park Drive  
San Jose, CA 95131

(408) 437-3133

FAX (408) 437-8791



# **Appendix B: Surround Sound**

---

## **Introduction**

This section briefly describes how to program the Yamaha YM7128 Surround Processor (SP2) on the Gold Card.

A first section describes the method used to access the SP2 chip through the Control Chip on the Ad Lib Gold card.

The second part is a hardware description of the SP2 chip.

Sample source code is also available in the Developer Toolkit disk, in the <SURROUND> directory. This sample source code demonstrates the procedure used to download a surround preset to the SP2.

## **Communicating with the SP2**

Register 18H of the Control Chip is used to interface with the SP2 Surround Processor.

Communication with the SP2 is done using a bit-serial protocol.

Modifying a register value to the SP2 involves sending a "register address - register value" pair to the SP2 using a special bit-serial protocol through register 18H.

### **SP2 bit-serial protocol**

Bit 0 of register 18H is the Data Bit (DATA).

Bit 1 of register 18H is the Clock Bit (CLK).

Bit 2 of register 18H is the Address Latch Bit (ADR).

Each bit of a message is sent to the SP2 by first sending a byte with the CLK bit low.

The message bit is sent in a byte with CLK low and the DATA bit containing the desired bit value.

By sending a third byte with CLK high, and DATA set to the correct value, the bit is "latched" into the SP2.

The ADR bit is used to differentiate the register-address- register value parts of the message.

When the bytes related to the register address part of the message are sent, ADR should be low. When all 8 bits of the address have been sent, a bit should be sent with ADR high, to latch the register address to the SP2.

ADR should then be high while the bytes related to the register value part of the message is sent.

Finally a last byte with ADR low should be sent, to latch the register value part of the message.

Sample code on how this procedure is accomplished is supplied on the Developer Toolkit diskette (in directory SURROUND).

While communicating with the SP2, we recommend that interrupts be disabled, in order to avoid access conflicts with background applications that could access the OPL3 chip or the Control Chip.

# **YAMAHA® LSI**

## **YM7128**

### **Surround Processor (SP2)**

#### **■ OUTLINE**

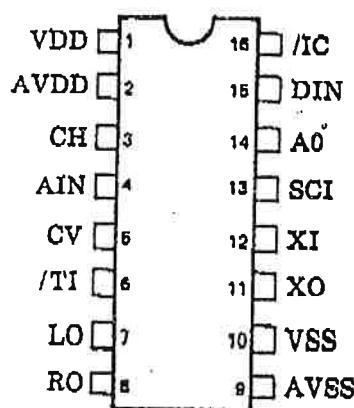
This is an LSI which has quality digital surround sound capabilities realized by Yamaha's digital audio technology. The LSI has built-in A/D and D/A converters which enable digital surround sound processing for analog input/output. Its eight digital delay lines may provide delay time of up to 100 msec. for each, and digital adding up of delay line signals for two-channel output assures a wide range of application.

#### **■ FEATURES**

- The built-in RAM realizes digital delay time of 100 msec.\* at the maximum.
- Feedback loop can be constructed for reverberation.
- Various surround effect can be obtained by controlling this LSI with serial data from microprocessor.
- Digital attenuator is built in for surround sound volume control.
- Sampling frequency is 23.6 kHz\*, and 14 bit floating A/D converter is built in.
- Two-times oversampling digital filter and 14 bit floating D/A converter are built in.
- 16 pin DIP package, silicone gate CMOS 5V power supply.

Note) When XI clock frequency is 7.16 MHz (304 fs is required for XI clock).

## ■ PIN CONFIGURATIONS



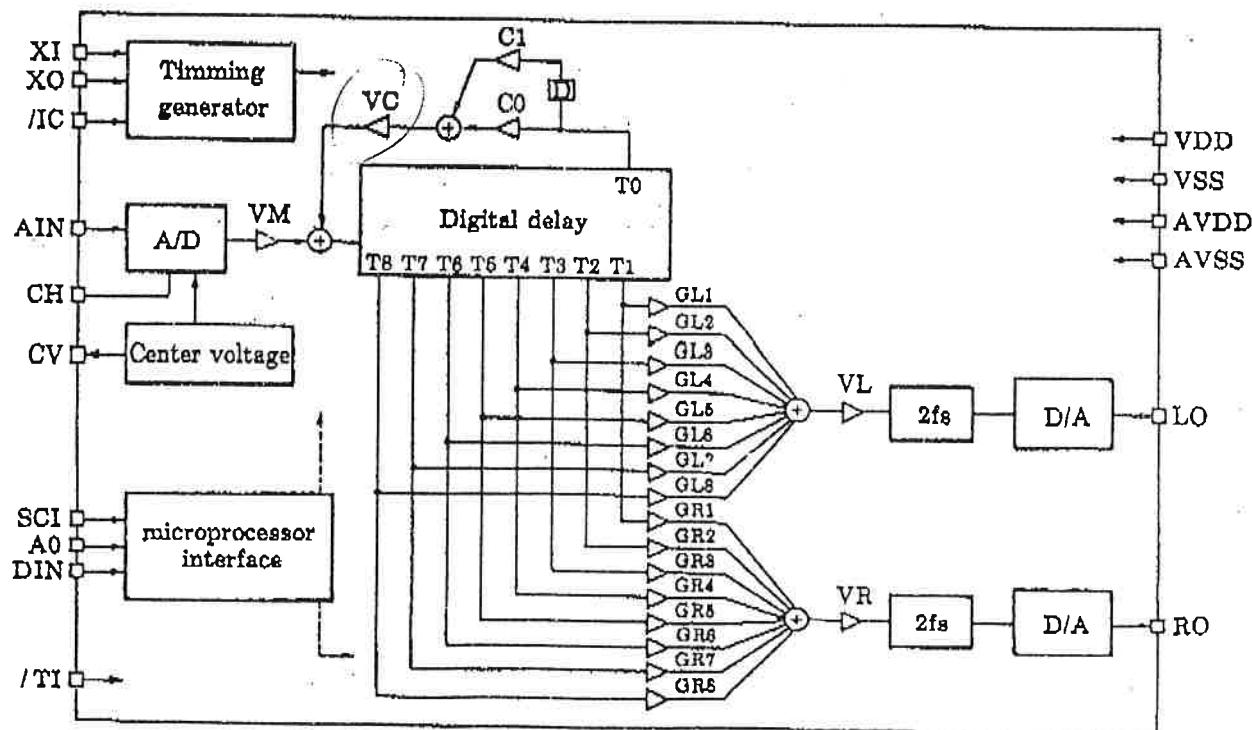
TOP VIEW

## ■ PIN DESCRIPTIONS

Pin No.	Name	I/O	Function
1	VDD	-	Digital +5V power supply
2	AVDD	-	Analog +5V power supply
3	CH	O	Sample/hold capacitor terminal
4	AIN	I	Analog signal input
5	CV	O	Center voltage of A/D
6	/TI	I+	Test terminal (without connection)
7	LO	O	L channel, analog out
8	RO	O	R channel, analog out
9	AVSS	--	Analog ground
10	VSS	-	Digital ground
11	XO	O	
12	XI	I	X'tal oscillator terminal (7.16 MHz typ.)
13	SCI	I	Bit clock for microprocessor interface
14	A0	I	Word clock for microprocessor interface
15	DIN	I	Serial data for microprocessor interface
16	/IC	I+	Initial clear terminal

+; pulled up

## ■ BLOCK DIAGRAM



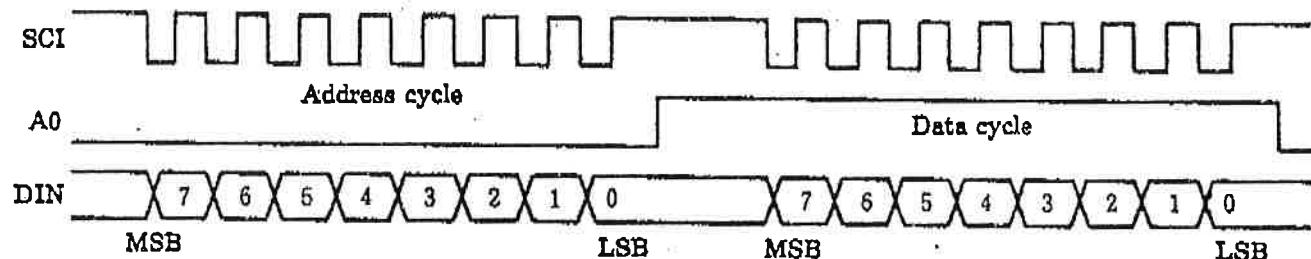
## ■ FUNCTION DESCRIPTION

As shown in the block diagram, analog signal input at AIN terminal are converted to 14 bit digital signal with the sampling frequency of 28.6 kHz using 14 bit floating type A/D converter, and then attenuated by the digital attenuator VM. Tap T0 output of digital delay passes through first order FIR type low pass filter and attenuated by VC. These signals are added before they are input to digital delay. Digital delay has nine output taps and tap positions can be switched by the registers T0 to T8. Outputs of eight taps from T1 to T8 are attenuated and added for each channel with the digital attenuator from GL1 to GL8 and GR1 to GR8 respectively, and attenuated by digital attenuator VL or VR to be input to two-times oversampling digital filter. Since this filter attenuates aliasing noise, it reduces the burden on external analog low pass filter. Digital input to D/A converter shall be with doubled value, which is 47.1 kHz sampling rate.

## ■ MICROPROCESSOR INTERFACE

Digital attenuation value, delay time and FIR type low pass filter coefficients are all set by writing data into registers.

With A0 = "L", 8 bit address data are sent synchronizing with SCI. At the rising edge of A0, register address is taken in. With A0 = "H", 8 bit data are sent synchronizing with SCI, then register data are changed at the falling edge of A0.



- At the time of initial clear, VM, VC, VL and VR registers are reset to 0. Other register values are not fixed.

## ■ REGISTER MAP

Address (HEX)	Data						Function		
	7	6	5	4	3	2	1	0	
00	x	x							GL1
01	x	x							GL2
02	x	x							GL3
03	x	x							GL4
04	x	x							GL5
05	x	x							GL6
06	x	x							GL7
07	x	x							GL8
08	x	x							GR1
09	x	x							GR2
0A	x	x							GR3
0B	x	x							GR4
0C	x	x							GR5
0D	x	x							GR6
0E	x	x							GR7
0F	x	x							GR8

Address (HEX)	Data						Function		
	7	6	5	4	3	2	1	0	
10	x	x							VM
11	x	x							VC
12	x	x							VL
13	x	x							VR
14	x	x							C0
15	x	x							C1
16	x	x	x						T0
17	x	x	x						T1
18	x	x	x						T2
19	x	x	x						T3
1A	x	x	x						T4
1B	x	x	x						T5
1C	x	x	x						T6
1D	x	x	x						T7
1E	x	x	x						T8

Note 1) x; Don't Care

Note 2) Don't write to the other address

## ■ REGISTER DATA DESCRIPTION

### (1) Attenuation value setting (GL1 to GL8, GR1 to GR8, VM, VC, VL, VR)

#### • Output polarity (bit 5)

When bit 5 = "1": Output signal is in phase with input signal.

When bit 5 = "0": Output signal is reversed phase with input signal.

#### • Attenuation value (bit 4-0)

Level (dB)	Data					
	4	3	2	1	0	(HEX)
0	1	1	1	1	1	1F
-2	1	1	1	1	0	1E
-4	1	1	1	0	1	1D
-6	1	1	1	0	0	1C
-8	1	1	0	1	1	1B
-10	1	1	0	1	0	1A
-12	1	1	0	0	1	19
-14	1	1	0	0	0	18
-16	1	0	1	1	1	17
-18	1	0	1	1	0	16
-20	1	0	1	0	1	15
-22	1	0	1	0	0	14
-24	1	0	0	1	1	13
-26	1	0	0	1	0	12
-28	1	0	0	0	1	11
-30	1	0	0	0	0	10

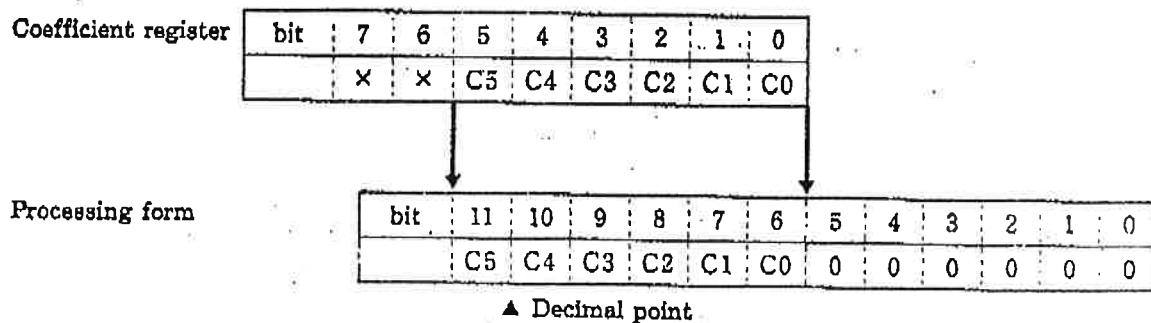
Level (dB)	Data					
	4	3	2	1	0	(HEX)
-82	0	1	1	1	1	0F
-84	0	1	1	1	0	0E
-86	0	1	1	0	1	0D
-88	0	1	1	0	0	0C
-40	0	1	0	1	1	0B
-42	0	1	0	1	0	0A
-44	0	1	0	0	1	09
-46	0	1	0	0	0	08
-48	0	0	1	1	1	07
-50	0	0	1	1	0	06
-52	0	0	1	0	1	05
-54	0	0	1	0	0	04
-56	0	0	0	1	1	03
-58	0	0	0	1	0	02
-60	0	0	0	0	1	01
-∞	0	0	0	0	0	00

## (2) Delay time setting (T0 to T8) (XI = 7.18 MHz)

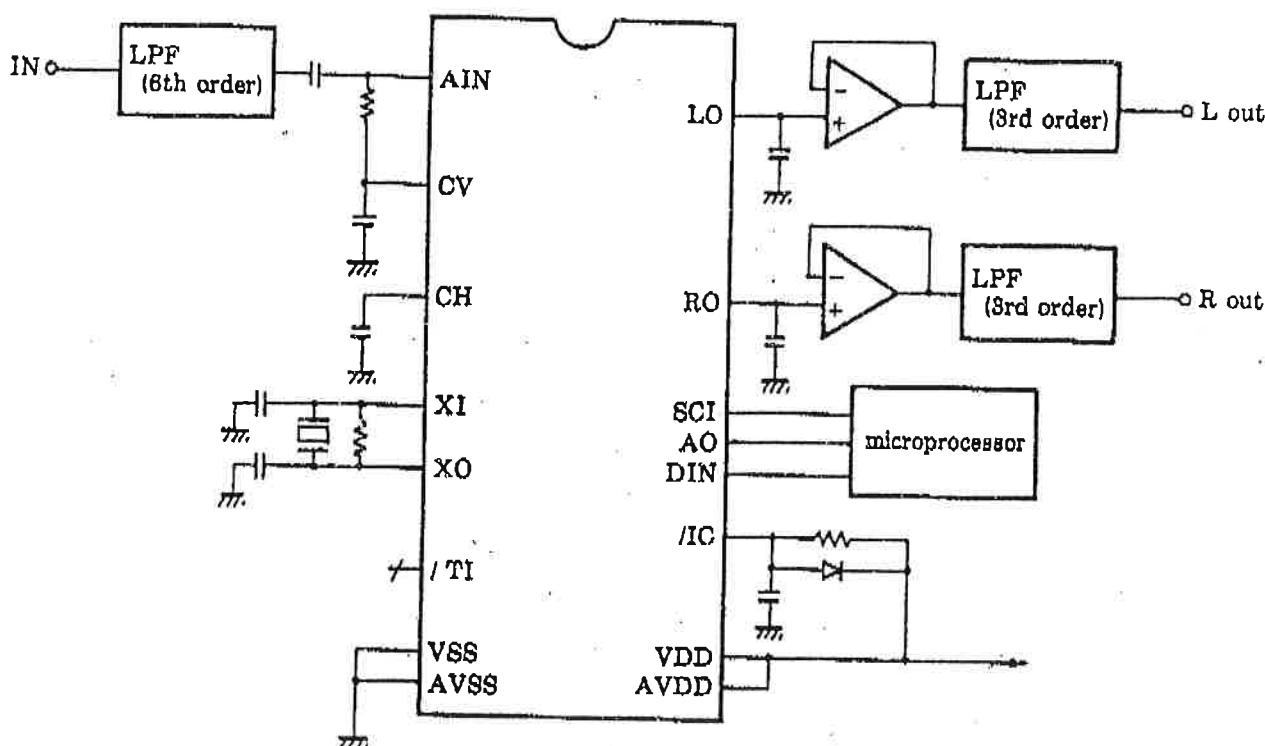
Delay time (ms)	Data					Delay time (ms)	Data					
	4	3	2	1	0		4	3	2	1	0	(HEX)
0.0	0	0	0	0	0	00	51.6	1	0	0	0	10
8.2	0	0	0	0	1	01	54.9	1	0	0	0	11
6.5	0	0	0	1	0	02	58.1	1	0	0	1	12
9.7	0	0	0	1	1	03	61.3	1	0	0	1	13
12.9	0	0	1	0	0	04	64.5	1	0	1	0	14
16.1	0	0	1	0	1	05	67.8	1	0	1	0	15
19.3	0	0	1	1	0	06	71.0	1	0	1	1	16
22.6	0	0	1	1	1	07	74.2	1	0	1	1	17
25.8	0	1	0	0	0	08	77.4	1	1	0	0	18
29.0	0	1	0	0	1	09	80.7	1	1	0	0	19
82.3	0	1	0	1	0	0A	83.9	1	1	0	1	1A
35.6	0	1	0	1	1	0B	87.1	1	1	0	1	1B
38.7	0	1	1	0	0	0C	90.4	1	1	1	0	1C
41.9	0	1	1	0	1	0D	93.6	1	1	1	0	1D
45.2	0	1	1	1	0	0E	96.8	1	1	1	1	1E
48.4	0	1	1	1	1	0F	100.0	1	1	1	1	1F

## (3) FIR Low Pass Filter coefficient setting (C0, C1).

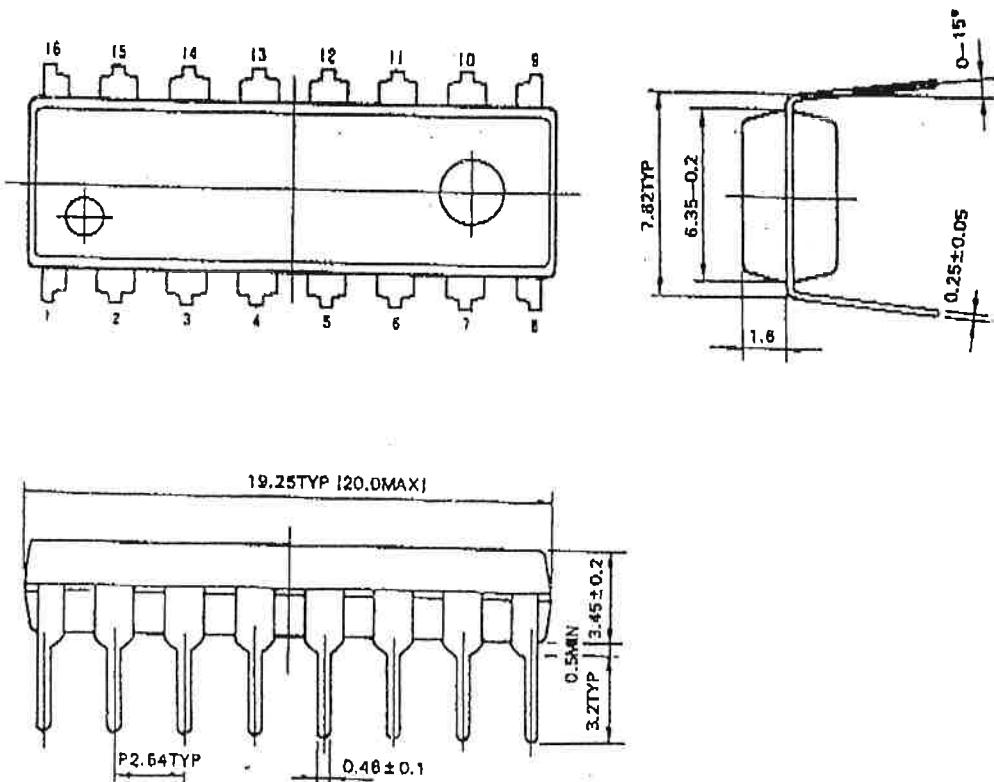
The lower 6 bits of coefficient register are used as the upper 6 bits of 12 bit 2's compliment data actually processed inside.



## ■ SYSTEM BLOCK DIAGRAM



## ■ EXTERNAL DIMENSIONS



## ■ ELECTRICAL CHARACTERISTICS

- Absolute maximum ratings

Parameter	Symbol	Rating	Unit
Supply voltage	VDD	-0.3 ~ +7.0	V
Operating temperature	Top	-20 ~ +85	°C
Storage temperature	Tstg	-50 ~ +125	°C

- Recommended operating conditions

Parameter	Symbol	Min.	Typ.	Max.	Unit
Supply voltage	VDD	4.75	5.0	5.25	V
Operating temperature	Top	0	25	70	°C

- DC characteristics (Conditions: Ta = 25°C, VDD = 5.0V)

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Supply current	IDD				50	mA
High-level input voltage (1)	VIH1		2.0			V
Low-level input voltage (1)	VIL1				0.8	V
High-level input voltage (2)	VIH2		4.0			V
Low-level input voltage (2)	VIL2				0.8	V
High-level output voltage	VOH	IOH = -0.4mA	4.0			V
Low-level output voltage	VOL	IOL = -0.2mA			0.4	V
Input leakage current	IIL	VI = 0 ~ 5V	-10		10	μA
Input capacitance	Ci			5.0	12.0	pF
Output capacitance	Co				10.0	pF

Note 1: Applicable to the input terminals except XI

Note 2: Applicable to XI terminal

- AC characteristics (Conditions: Ta = 25°C, VDD = 5.0V)

Parameter	Symbol	Min.	Typ.	Max.	Unit
XI Input frequency	fc	8.6	7.16	8.0	MHz
Duty		40	50	60	%
Rise time	TCR			50	ns
Fall time	TCF			50	ns
SCI Input frequency	fs			fc/8	MHz
On-off time	Ts	600			ns
Rise time	TSR			200	ns
Fall time	TSF			200	ns

• ANALOG characteristics (Conditions:  $T_a = 25^\circ\text{C}$ ,  $V_{DD} = 5.0\text{V}$ )

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Analog input voltage	V <sub>IA</sub>	AIN terminal			4.5	V <sub>p-p</sub>
Analog output voltage	V <sub>OA</sub>	LO, RO terminal			4.5	V <sub>p-p</sub>
DC offset voltage	C <sub>V</sub>			2.5		V
Total harmonic distortion	THD	output voltage 0dB		0.3	0.4	%
		-10dB		0.4	0.5	%
		-20dB		0.4	0.5	%
		-30dB		0.6	0.8	%
S/N	S/N	S=0dB	75	80		dB

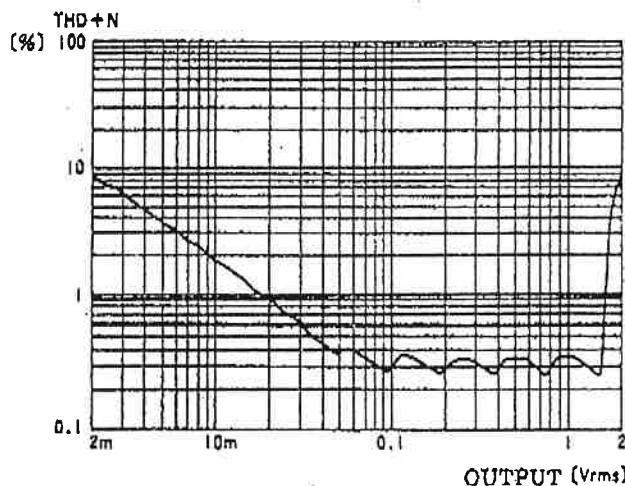
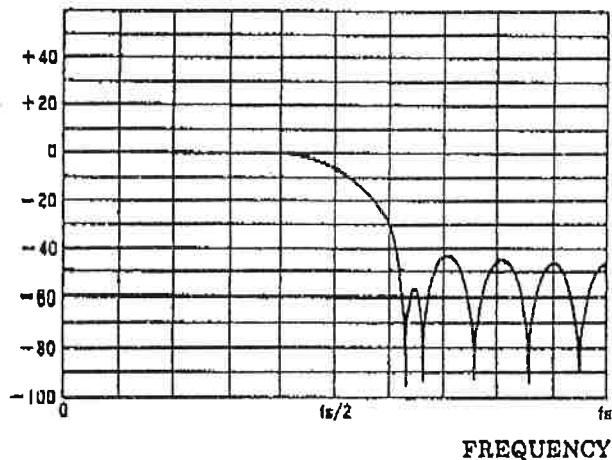
Note) 0dB = 1.5VRms

■ REFERENCE CHARACTERISTICS

2 times oversampling filter

Output vs THD + NOISE

(dB) OUTPUT



The specifications of this product are subject to improvement changes without prior notice.

— AGENCY —

— YAMAHA CORPORATION —

Address inquiries to:  
Semi-conductor Sales Department

■ Head Office 209, Matsunokijima, Toyooka-mura,  
Iwata-gun, Shizuoka-ken, 438-01  
Electronic Equipment business section  
Tel. 0539-62-4918 Fax. 0539-62-6054

■ Tokyo Office 8-4, Surugadai Kanda, Chiyoda-ku,  
Tokyo, 101  
Ryumeikan Bldg. 4F  
Tel. 03-255-4481 Fax. 03-255-4488

■ Osaka Office 3-12-9, Minami Senba, Chuo-ku,  
Osaka City, Osaka, 542  
Shinsaibashi Plaza Bldg. 4F  
Tel. 06-262-7980 Fax. 06-252-5815

■ U.S.A. YAMAHA Systems Technology,  
655 Ridder Park Drive San Jose, CA95131  
Tel. 408-437-3133 Fax. 408-437-8791

The following source code demonstrates how to program a Surround preset.

Function Write\_Srnd\_Reg writes the specified value to the YM7128 through the control chip register 18H. The sample code assumes that the control chip is located at address 38AH.

Function Write\_Surround sends the 32 bytes of a surround preset to the YM7128 using the bit-serial protocol. For each byte of the Surround preset, the register number (variable *addr*) is sent first , followed by the register value (variable *data*).

```
/*
   SURR.C
   Write a preset to the surround chip.
   Copyright 1992, Ad Lib Inc.
*/
unsigned control_io = 0x38a;      /* address of control chip section */

/* Write 'val' to the surround register in the control chip. */
static void __fastcall Write_Srnd_Reg (unsigned val)
{
    _asm {
        mov     dx, control_io
l10:
        in     al, dx
        test   al, 0c0h      ;status bits indicating chip is busy
        jnz    l10
        mov     ax, 18h        ;surround register number
        out    dx, al
        mov     ax, val
        inc    dx
        out    dx, al
    }
}
```

## Appendix B: Surround Sound

### Sample Source Code

```
/* NOTE: When writing a byte to the control chip, it is very important
   that the transfer not be interrupted. Therefore, interrupts are
   disabled while the preset is being sent. */
void Write_Surround (unsigned char *preset)
{
    unsigned addr, data, cmd;
    int i, k;

    _asm {
        pushf          ;preserve the current interrupt state
        push    dx
        cli             ;disable interrupts
        mov    dx, control_io
        mov    al, 0ffh      ;disable OPL3, enable control bank
        out    dx, al
    }

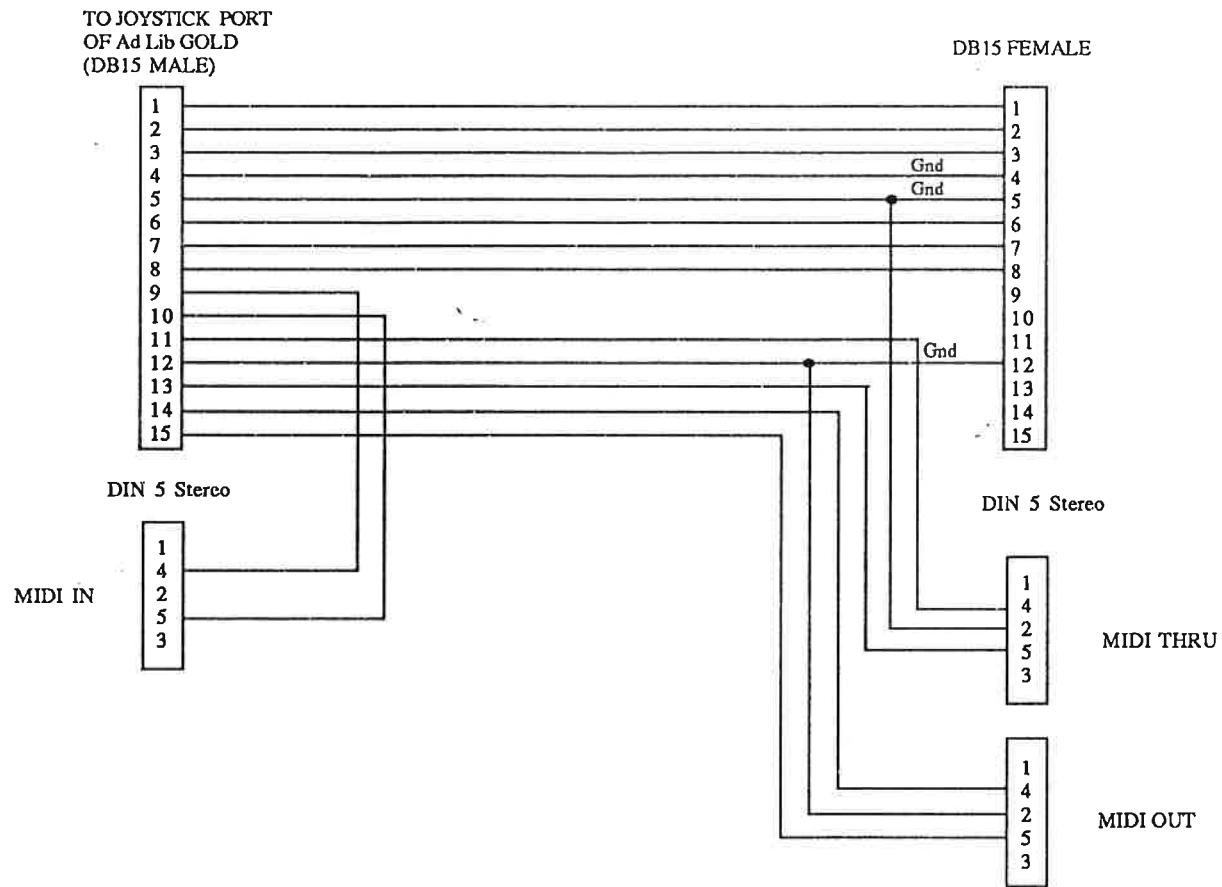
    /* Send the 31 array elements: */
    for (i = 0; i < 31; i++) {
        cmd = 0;           /* clock LOW, A0 LOW */
        addr = i;
        for (k = 7; k >= 0; k--) {
            cmd &= ~2;       /* clock LOW */
            Write_Srnd_Reg (cmd);
            cmd = (cmd & ~1) | ((addr >> k) & 1);
            Write_Srnd_Reg (cmd);
            cmd |= 2;         /* clock HIGH */
            Write_Srnd_Reg (cmd);
        }
        cmd |= 4;           /* Set A0 to 1 */
        Write_Srnd_Reg (cmd);

        data = preset [i];
        for (k = 7; k >= 0; k--) {
            cmd &= ~2;       /* clock LOW */
            Write_Srnd_Reg (cmd);
            cmd = (cmd & ~1) | ((data >> k) & 1);
            Write_Srnd_Reg (cmd);
            cmd |= 2;         /* clock HIGH */
            Write_Srnd_Reg (cmd);
        }
        cmd &= ~4;           /* Set A0 to 0 */
        Write_Srnd_Reg (cmd);
    }

    _asm {
        mov    dx, control_io
    l20:
        in     al, dx
        test   al, 0c0h      ;status bits indicating chip is busy
        jnz    l20
        mov    al, 0feh      ;enable OPL3, disable control bank
        out    dx, al
        popf
    }
}
```

# Appendix C: Pin Out for Joystick-MIDI Connector

PIN OUT FOR JOYSTICK-MIDI CONNECTOR  
OF THE Ad Lib GOLD CARD





## **Appendix D: List of Installed Files**

---

The Ad Lib Gold Developer Toolkit software included in the diskettes contains, when decompressed and installed, several files related to the utilization of the Gold card: drivers, application programs, music, sounds, and other various files. These files are:

- **README.TXT**  
This file is not compressed on the diskette. It contains information on the latest program updates, if there are any, and any other pertinent information.
- **CTRLDRV.EXE**  
This file is not compressed on the diskette. It contains the Ad Lib Gold Control chip driver. This low level driver is used by other programs, such as the Setup program, to implement: DMA channel & interrupt number select; sampling source select; sampling gain and input filter; microphone input gain; sampling output filtering and volume & tone control; mixing control; card localization setup and ID code reading; saving registers in non volatile memory.
- **SETUP.EXE**  
This file is not compressed on the diskette. It contains the Installation and Configuration program. This program enables you to install the drivers and all associated programs, and to configure your Ad Lib Gold card.

### **Drivers and TSRs**

Are located in the "DRIVERS" subdirectory.

- **FMDRV.EXE**  
This file contains the FM driver. This low level driver implements: preset change; note on; note off; pitch bend; volume and stereo positioning.
- **WAVEDRV.EXE**  
This file contains the Sampling driver. This low level driver implements: recording and playback of samples by DMA and interrupt.
- **TIMERDRV.EXE**  
This file contains the Timer driver. This low level driver implements: the five timers of the Yamaha Magic Chip Set.
- **MIDIDRV.EXE**  
This file contains the MIDI driver. This low level driver implements: MIDI In and Out serial port control.
- **RL2DRV.EXE**  
This file contains the ROL2 driver. This low level TSR driver implements: playback of the .RL2 music files and user control commands .

## Appendix D

### List of Installed Files

- **MIXER.EXE**  
This file contains the Mixer Panel TSR. This memory resident application allows for the control of the programmable volume and tone control, mixer settings, surround features, and setting of activation and volume keys.
- **PLAYRL2.EXE**  
This file contains the executable code of ROL2 Playback utility.
- **PLAYDIGI.EXE**  
This file contains the executable code of Digitized Sound Playback utility.

### Application Programs (Executables)

- **TESTGOLD.EXE**  
This file contains the Ad Lib Gold Test Program. This program enables you to verify that the Gold card is functioning properly in all of its different components.
- **JUKEG.EXE**  
This file contains the executable code of Juke Box Gold Music Playback Program.
- **ED.EXE**  
This file contains the executable code of Instrument Maker Gold.
- **SAMPL.EXE**  
This file contains the executable code of Sample Maker Program.
- **SURR.EXE**  
This file contains the executable code of Juke Box Gold with the Surround Sound Editor.

### Batch Files

- **TEST.BAT**  
This file contains the DOS command sequence which loads the necessary drivers and calls the Ad Lib Gold Test Program.
- **JUKEGOLD.BAT**  
This file contains the DOS command sequence which loads the necessary drivers and calls the Juke Box Gold Music Playback Program.
- **INSGOLD.BAT**  
This file contains the DOS command sequence which loads Instrument Maker Gold.
- **SURROUND.BAT**  
This file contains the DOS command sequence which loads the necessary drivers and calls the Juke Box Gold Music Playback Program with the Surround Sound Editor.

- **DRIVERS.BAT**  
This file contains the DOS command sequence which loads all Ad Lib Gold drivers.

#### Other Files

- **\*.RL2**  
The ".RL2" files contain the pieces of music that will be played with the Juke Box Gold.
- **\*.SMP**  
The ".SMP" files contain the PCM digitized sounds. TESTGLD1.SMP is the sound file that will be used by the Test Program.
- **SAMPLBNK.EQU**  
This file contains a translation table of digitized instrument sound names, which is used by the ROL2 Playback driver.
- **OPL3.BNK**  
This file contains the FM synthesized instrument sounds compatible with the OPL3 FM synthesis chip.
- **ED.RSR**  
This file contains the resources required by Instrument Maker Gold.
- **SAMPL.RSR**  
This file contains the resources required by Sample Maker Program.

#### Files Created by Programs

- **JUKEGOLD.DAT**  
This file is created the first time you make a selection of songs in the Juke Box Gold program, permitting not to lose your selection even after rebooting the computer.
- **TESTGLD1.SMP**  
This file is created by the Test Program when you test Sampling and Playback.
- **SAMPLES.BNK**  
This bank file is created by the Sample Maker Program the first time you save a digitized sound in the ADPCM format.