The Frob-Coleco User's Manual

A True Connoisseur's Guide

January 1984

FROBCO
6Ø3 Mission Street
Santa Cruz, CA 95Ø6Ø
(4Ø8) 429-1551

The Miracle of Creation
Can Be YoursTM

COPYRIGHT NOTICE

Copyright (c), 1984 by FROBCO. All Rights Reserved Worldwide. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the express written permission of FROBCO, 603 Mission St., Santa Cruz, California 95060 U.S.A.

TRADEMARKS

The names FROB and FROBCO are trademarks of FROBCO.

AppleSoft and Apple are registered trademarks of Apple Computers, Inc.

Coleco is a registered trademark and Adam and ColecoVision are trademarks of Coleco Industries, Inc.

CP/M is a registered trademark of Digital Research Corporation.

Microsoft is a registered trademark and Softcard is a trademark of Microsoft Corporation.

TI is a trademark of Texas Instruments, Inc.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

FROBCO makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. FROBCO software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. In no event will FROBCO be liable for direct, indirect, incidential, or consequential damages resulting from any defect in the software, or use of this product in any way, even if FROBCO has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

SPECIAL NOTE TO FROB USERS:

Please fill-out the two pages of the "LICENSE AGREEMENT" and return promptly to the FROBCO address presented below. A copy of the agreement will be returned to the Licensee's address.

LICENSE AGREEMENT

FROBCO
603 Mission Street
Santa Cruz, California 95060
Telephone: 408-429-1551

FROB-COLECO DEVELOPMENT SOFTWARE

LICENSEE'S
NAME ____
Address ____
City ____
State ____________________ Country ____________________
Zip Code __________________
Telephone Number ______________________________________

PROGRAM PRODUCT:

1. Title of Software Disk ________________________________

Release Version Number _______ Disk Serial Number ______________

2. Title of Software Disk ________________________________

Release Version Number _______ Disk Serial Number ______________

FROB-COLECO Hardware Serial Number __________________________

LICENSE

FROBCO grants and Licensee accepts perpetual nonexclusive license to use the FROB-COLECO Development Software programs listed above, together with the modifications and additions as furnished by FROBCO, subject to the "Standard Terms and Conditions for End User Software License Agreement" attached hereto and incorporated herein by reference.

Licensee may not assign, sublicense, sell, transfer or otherwise make available to any third party Licensed Programs, or Derivative Works, without the prior written approval of FROBCO.

The Software is furnished to the Licensee for use only by Licensee, and only on the following single CPU:

____

Manufacturer

______________________   ______________________

Model   Serial Number

SIGNED BY FROBCO
CORPORATE OFFICER

By: ______________________

Title: ______________________

Date: ______________________

LICENSEE SIGNATURE

By: ______________________

Title: ______________________

Date: ______________________

TABLE OF CONTENTS

Chapter 1. INTRODUCING THE FROB

1.1 Legend of the Frob

Mythically, the Frob is an unusually intelligent creature, with some very special magical gifts. For example, the Frob has the ability to assume the identity of other creatures, duplicating their very nature to the smallest detail. Like a chameleon, the Frob can instantly change form, becoming an amoeba one instant, an extraterrestrial the next.

This, of course, brings up the question of the true identity of the Frob. If you could be anyone at any time, who would you be? The Frob often wondered about this and would ask all the creatures it' duplicated what they would be if they could be anything. S/he* got every answer imaginable and tried them all.

But the Frob found most fantasies got to be pretty boring after awhile, and kept changing hir* form. Not stopping with biological life forms and fantasies, s/he tried being fire, clouds, electricity, sounds, quartz crystals, galaxies, space and time, learning all kinds of things about everything.

And in the course of multi-dimensional events, s/he evolved a special love of humans, as they are creatures that often wondered about things too. The Frob learned much from the humans s/he became, assimilating many new things and experiences.

The Frob most enjoyed communicating with human children, truly curious creatures who easily accepted hir magical abilities. Through the children s/he learned about the nature of real dreams, and was soon able to communicate with adult humans during their sleeping hours.

It was in this way that s/he first contacted a human who was extremely knowledgeable of computers during his waking hours. S/he was very curious about this new human enterprise, learning all about computers and artificial intelligence from his sleeping biocomputer.

To the Frob's amazement and never-ending curiosity, s/he realized that both the computers and their creators, the biocomputers, were beginning to develop the ability to become and do many different things, just like hir.

The only problem was that, as curious as the humans were, they did not quite grok the possibilities and freedoms of being many many things. So s/he decided to help out a little by giving ideas to computer-oriented people in their dreams. Ideas that stimulated more interesting, multi-dimensional uses for the new human tool. Uses that would stimulate an increased sense of curiosity about what might really be going on in this universe.

* s/he and hir are the preferred pronouns of the Frob.

You see, despite all the knowledge and experience gathered by the Frob, s/he did not have the slightest clue of why all this was happening. S/he shared water with creatures who believed that you'd learn a lot by not asking why things existed. S/he traveled faster than light, going back in time to explore the beginnings of the physical universe. To get a bigger perspective, s/he expanded hir physical body so that every atom was a star, light years apart from the other. S/he even became mortal, to see what that was like.

But none of these things satisfied hir natural curiosity. And humans were the only other ones, so far, who shared this curious natural wonder with hir, exploring the same things, only from a more basic, less flexible form.

In a brilliant insight that came from hir new ability to dream, the Frob saw that collaboration of Silicon Intelligence with Biological Intelligence would give the curious humans more freedom to explore possibilities. All s/he had to do was translate this insight into a form understood by hir new human friends. For more than anything else, s/he wants company with others who are as versatile and curious as s/he is.

The following manual is the first part of hir translation, as filtered through the minds of humans.

1.2 Mission of the Frob

The Frob's mission begins with giving curious entrepreneurs and programmers, both rich and poor, new and used, an opportunity to create educational, business and gaming software cartridges for both specialized applications and the mass consumer marketplace.

The Frob-Coleco system is a professional software development environment for the ColecoVision. The ColecoVision, known for its ability to generate arcade quality, colorful, animated images and sophisticated sound effects, provides an inexpensive environment for creating dynamic software. With the Frob-Coleco system, you will be able to program your own ColecoVision cartridges, also compatible with the Adam Computer.

Until the Frob, creating arcade-quality machine cartridges required an investment of tens of thousands of dollars. Unlike the independent goldminers and adventurous oilmen of the past, individuals and small development companies can now have many of the same advantages as the big companies.

The Frob-Coleco system, in combination with a commercially available 2764 PROM programmer (not supplied with the Frob-Coleco system), allows a single individual to create software products along with ColecoVision and Adam cartridge prototypes.

With the advent of electronic distribution of software, there is a continued market for Coleco software. Within a couple weeks of the time it takes to think of and design a software product, merchandise can be delivered to the customer. There is no longer any need for maintaining an inventory of your software products, as hardware protected cartridge copies can be made on the spot, as orders are received. With low overhead (basically programmable cases and instructions), a very competitive price and a quick return for your software can be achieved.

The Frob-Coleco system is a combination of hardware, software, and user support. The Frob hardware includes the Frob Interface Unit, the Apple card, cable and Frob-Coleco EPROM adapter. The software includes an Apple Development package and the Lazer Monitor system. User support includes the documentation and technical guidance available by telephone from Frobco.

"May the Frob be With You"

1.3 Installing the Frob

The complete Frob-Coleco system requires a 64K Apple Computer with disk drive and a ColecoVision to function.

The components you receive from FROBCO are:

1) Frob-Apple Card - a PC board that plugs into slot #2 of the Apple.

2) Frob Interface Unit - a PC board in a case that plugs into the Colecovision Expansion slot.

3) Frob-Coleco EPROM Cartridge Adapter - a small PC board that plugs into the cartridge slot of the Colecovision.

4) A cable that connects the Frob-Apple Card with the Frob Interface Unit.

5) An Applesoft Basic disk entitled "Apple Development System - Coleco Loader."

6) A CP/M disk entitled "Lazer Z-8Ø Monitor."

7) The Frob-Coleco User's Manual.

The components you need besides the Frob are:

1) A 64K Apple II, II+ or IIe microcomputer.

2) An Apple disk drive with controller card.

3) An Apple monitor.

4) A ColecoVision video arcade system.

5) A color TV for the ColecoVision display.

6) A Microsoft Z-8Ø Softcard (or hardware equivalent) to run LZRMON program only.

To get the Frob-Coleco System working:

1) Get the Apple working normally (read Apple Reference Manual).

2) Get the ColecoVision working normally (see ColecoVision Instruction Booklet).

3) Plug the Frob-Apple card into Slot #2 of the Apple computer, with the cable inserted in the card so that cable comes out the back of the Apple.

4) Plug the Frob Interface Unit into the expansion bus of the ColecoVision console.

5) When used, the Frob-Coleco EPROM cartridge adapter plugs into the ColecoVision game cartridge slot. It has four sockets which will hold four 2764 (8K) EPROMS. This adapter and "burned" EPROMS (a maximum of 32K) is your demo cartridge.

Chapter 2. THE DEVELOPMENT SYSTEM - A FROB'S EYE VIEW

2.1 THE APPLE DEVELOPMENT SYSTEM

2.1.1 Introduction

With the Apple Development software, you create your own or modify ColecoVision programs on the Apple, and then test them on either the ColecoVision or the Frob Interface Unit. The Coleco Loader program, discussed below, will allow you to transfer data to and from the ColecoVision, Apple and the Frob Interface Unit.

2.1.2 Required Equipment

Besides the package from Frobco, the Frob-Coleco development system requires a 64K Apple II, II+ or IIe computer, with DOS 3.3, Applesoft Basic, and a ColecoVision arcade unit. A 2764 PROM programmer machine is also necessary, if you wish to duplicate your program onto blank Frob-Coleco cartridge adapters.

2.1.3 Menu Functions

After the Frob-Coleco development system is hooked up, insert the Apple Development System - Coleco Loader Disk into the Apple Drive 1 (one) and boot it up. You will see a menu:

```
PLEASE CHOOSE FUNCTION BY NUMBER:

1 — LOAD FROB MEMORY FROM APPLE DISK

2 -- MOVE FROB MEMORY TO APPLE DISK

3 — MOVE FROM CARTRIDGE TO APPLE DISK

4 -- MOVE FROM CARTRIDGE TO FROB MEMORY

5 — DISPLAY FROB CONTROL PARAMETERS

6 -- SET FROB CONTROL PARAMETERS

7 — RESET AND RUN PROG IN FROB MEMORY

8 -- RESET AND RUN PROG IN COLECO ROM

9 — EXIT
```

Let's take a look at each of these menu selections:

1 — LOAD FROB MEMORY FROM APPLE DISK

This option loads a binary program from the Apple disk to the Frob-Coleco development system. After choosing this option, you will be asked which slot the Frob-Apple Interface Card is plugged into:

INTERFACE SLOT NUMBER? (1-7)

Any slot will work, though for other Frob software (i.e. the Lazer Z-8Ø Monitor), the Apple card <u>must</u> be in slot 2.

Type in the slot number you have plugged the board into (usually slot 2) and press the Return key.

The next message you see is:

FILES MUST BE 8K BYTES OR LESS

HOW MANY 8K BLOCKS? (1-4)

The cartridge address space in the ColecoVision runs from Ø8ØØØH to ØFFFFH. This means that there is room for a 32K program, or four 8K blocks of memory. Let's say you have a binary program named 'FOOBAR' that is 24K bytes long. To store this program you need 3 of the 4 available 8K memory blocks. These would be arranged on your disk as:

FOOBAR.B4
FOOBAR.B5
FOOBAR.B6

So, in the case of our 'FOOBAR' example, the answer to the last question is '3'.

The next question is:

INPUT THE BASE FILE NAME

Here you type in the file name without the extension (.B?). In our 'FOOBAR' example you would type in 'FOOBAR', followed by a Return key.

The next question is:

INPUT THE STARTING BLOCK NUMBER (4-7)

In the case of FOOBAR you would type in '4' since FOOBAR.B4 is the first 8K file to be loaded.

The COLECO LOADER then loads in the file and simultaneously prints out the file name and the checksum on your CRT (the checksum is used for debugging purposes). In our example, you will see on the screen before you:

```
LOADING FOOBAR.B4   CHECKSUM=ØDØ8FE
LOADING FOOBAR.B5   CHECKSUM=Ø9B555
LOADING FOOBAR.B6   CHECKSUM=Ø55555
```

After loading the file, the COLECO LOADER program redisplays the option menu and waits for your next command.

The next option on the menu is:

2 — MOVE FROB MEMORY TO APPLE DISK

This command transfers the binary program residing in the Frob-Coleco development system onto the Apple disk. Type in a '2', followed by a Return key to select this option. You will then be asked a series of questions:

```
BASE FILENAME —
```

Type in the name of your program, without an extension. Like 'FOOBAR', without the '.B4' extension, for example.

Next you are asked to type in the starting block number.

```
INPUT STARTING BLOCK NUMBER (4-7)
```

As with option '1', the block number corresponds to an 8K block of ColecoVision development system memory. These are interpreted and represent the following locations in ColecoVision memory:

```
4 = Ø8ØØØH-Ø9FFFH
5 = ØAØØØH-ØBFFFH
6 = ØCØØØØ-ØDFFFH
7 = ØEØØØØ-ØFFFFH
```

If you want to save the whole binary program from the beginning, type in a '4'.

Eight blocks of 8K memory equal 64K. The Frob user program memory resides in the upper 32K of the ColecoVision, or blocks 4,5,6, and 7 of memory. Hence the starting blocks begin at 4 (Blocks start at zero in the ColecoVision and continue up to seven). Blocks Ø-3 contains the Coleco Monitor, 1K of scratchpad RAM, and unused addresses. See Appendix A, section A.1 regarding further information on the ColecoVision memory map.

After selecting the starting block number, type in the number of contiguous 8K blocks to save:

```
INPUT THE NUMBER OF 8K BLOCKS (1-4)
```

If all of memory is to be saved, then type in 4.

The COLECO LOADER then saves the file on the Apple disk. You will see messages to this effect on your CRT:

```
SAVING FILE FOOBAR.B4 CHECKSUM=Ø1FEØØØ

SAVING FILE FOOBAR.B5 CHECKSUM=Ø1FEØØØ

SAVING FILE FOOBAR.B6 CHECKSUM=Ø1FEØØØ
```

3 — MOVE FROM CARTRIDGE TO APPLE DISK

This option allows you to read any 8K block of memory from the ColecoVision and save it on the Apple disk. You select this option by typing a '3', followed by a Return key. You are then asked which slot the Apple card of development system is plugged into:

```
INTERFACE SLOT NUMBER? (1-7)
```

Type in the slot number the Frob-Apple Interface card is plugged into, followed by a Return key. You will then be asked for a file name:

```
BASE FILENAME --
```

Choose and type in a file name (i.e. FOOBAR). Now you will be asked where (the location in Colecovison memory) to save from:

```
INPUT STARTING BLOCK NUMBER (Ø-7)
```

You have the ability to save 8K blocks of memory from anywhere in the ColecoVision address space. Block Ø is the ColecoVision monitor area (See Appendix A for a description of the ColecoVision monitor). The cartridge area resides in blocks 4-7. If you wish to read the programs on the cartridge, type in a 4 in response to the above INPUT command.

The next question asked is the number of contiguous blocks to save:

```
INPUT THE NUMBER OF 8K BLOCKS (1-8)
```

For the whole cartridge space, type in 4.

The COLECO LOADER now saves the file on disk, displaying on the CRT:

```
SAVING FOOBAR.4 CHECKSUM=4444444
SAVING FOOBAR.5 CHECKSUM=5555555
SAVING FOOBAR.6 CHECKUSM=6666666
SAVING FOOBAR.7 CHECKSUM=7777777
```

After saving the file, the program redisplays the main menu.

4 — MOVE FROM CARTRIDGE TO FROB MEMORY

This option copies the programs stored in ROM on the cartridge into the 32K of RAM memory within the Frob Interface Unit. You select this option by typing a '4', followed by a Return key. The next thing you will see are the block checksums:

```
THE BLOCK Ø CHECKSUM IS ØØØØØØØ
THE BLOCK 1 CHECKSUM IS 1111111
THE BLOCK 2 CHECKSUM IS 2222222
THE BLOCK 3 CHECKSUM IS 3333333
```

After performing this operation, the COLECO LOADER redisplays the menu.

5 — DISPLAY FROB CONTROL PARAMETERS

This option prints out the status of software parameters selected by the user. Selected choices may include the memory space selected, the storage space selected, and write protection information. This option can be viewed by typing a '5', followed by a Return key. The following is an example of a display:

```
MEMORY SPACE IS SELECTED
COLECO CARTRIDGE IS SELECTED
INTERFACE IS WRITE PROTECTED
I/O SPACE IS SELECTED
```

After displaying the status, the main menu is redisplayed. The LEDs on the front panel of the Frob Interface Unit provide the same information continuously. See Chapter 3, section 3.1.2 for further details on the LED indicators.

6 — SET FROB CONTROL PARAMETERS

This option allows you to configure the hardware on the Frob-Coleco development system, selected by typing a '6', followed by a Return key. The first question asked is:

```
MEMORY SPACE OR I/O? (M/I)
```

This option allows one to choose whether access will be to the Colecovison or its I/O space.

```
            (*)
HOLD COLECO     BUS? (Y/N)
```

If the bus is held, processing stops, preventing the ColecoVision from running its program. As we use static RAMs, memory stays intact. Control is turned over to the RAM of the Frob Interface Unit.

(*) All references to Coleco are ColecoVision

RESET COLECO? (Y/N)

Resetting the ColecoVision makes the ColecoVision act as if it were just powered up. It will go through its entire intialization process, assuming that the bus is not held. Note, however, the reset must be turned off to start the program. Options 7 and 8 below do this function automatically by turning on the reset to reinitialize the ColecoVision and then turning off the reset to start the program in either the Frob Memory or the Coleco ROM.

SELECT CARTRIDGE? (Y/N)

Either the cartridge or the RAM area in the Frob-Coleco Interface Unit can be active, as they both map into the same Coleco address space. 'Y' will select the program in the cartridge. Answering 'N' will select the program in the Frob Interface Unit RAM space.

WRITE PROTECT INTERFACE? (Y/N)

This option allows you to prevent writes into the RAM area on the Frob-Coleco Interface Unit. This is useful while debugging programs, as it will prevent your code from being 'clobbered' or erased by a wild, out-of-control program.

At this late juncture, the COLECO LOADER summarizes your selections. They may be as follows:

```
I/O SPACE IS SELECTED [or MEMORY SPACE IS SELECTED]
BUS HOLD IN PROCESS
RESET HOLD IN PROCESS
COLECO CARTRIDGE IS SELECTED
INTERFACE IS WRITE PROTECTED
```

To safely run a software program stored in the RAM of the Frob Interface Unit, the following combination of choices is appropriate:

```
I/O SPACE IS SELECTED (or MEMORY SPACE IS SELECTED)
INTERFACE IS WRITE PROTECTED (optional)
```

7 — RESET AND RUN PROG IN FROB MEMORY

By typing a '7' followed by the Return key, you will automatically initialize the ColecoVision and select the program in the Frob Interface Unit. This is a fast way to perform this action.

8 — RESET AND RUN PROG IN COLECO ROM

Typing an '8' followed by a Return key, will reset the ColecoVision and select the program in the ColecoVision ROM cartridge and run the program in ROM.

9 — EXIT

This option, selected by typing a '9', followed by a Return key, stops the Coleco Loader program, bouncing you back to Applesoft.

2.2 THE LAZER Z-8Ø MONITOR SOFTWARE

2.2.1 Introduction

Similiar in some respects to the Coleco Loader software, the LZRMON program allows you to execute, examine, and debug Z-8Ø machine language programs directly on the ColecoVision system. Of the two pieces of software, the LZRMON is the more sophisticated, offering more system status information and greater debugging capabilities. It also requires more time to learn and more hardware.

Use of the software requires an Apple II, II+, or IIe with 64K RAM and a MicroSoft Z-8Ø Softcard (or hardware equivalent) running the CP/M operating system. The MicroSoft ALDS assembly language development package is another extremely useful tool worth having.

The Apple Card connecting to the Frob Interface Unit must be placed in slot #2.

The CP/M disk included with your Frob unit includes two files: LZRMON.COM and DBUG.FRB. The LZRMON program is the actual Z-8Ø monitor program; the DBUG.FRB file is a debugger program that loads into the ColecoVision memory space.

2.2.2 Execution Procedure

To execute the Lazer Monitor program, type "LZRMON" (or "B:LZRMON" if your Frob disk is in drive B:) followed by the Return key.

This command loads and runs the monitor program. You will then be presented with a short menu offering the following options:

2.2.3 Primary Menu Commands

Insert the Lazer Z-8Ø Monitor Disk into the Apple Drive 1 (one) and boot it up. Type: LZRMON and you will see a menu:

```
ENTER COMMAND:

C)  COPY CARTRIDGE TO RAM
L)  LOAD FILE INTO RAM
M)  ENTER MONITOR MODE
R)  RESET COLECOVISION
S)  SAVE RAM TO FILE

Q)  QUIT

COMMAND:              (Awaiting your response.)
```

C COPYing the data from the cartridge slot into the Frob RAM. This command copies the entire 32K present in the cartridge space. The Copy command is executed by pressing "C".

L LOADing a binary file from disk into the Frob memory. This command is executed by pressing "L".

Keep in mind that the LZRMON program only allows a file to be loaded onto a 256-byte page boundary. After you press "L" you will be asked which page you wish the file to be loaded onto. The desired starting page number must be entered in HEX for loading your Z-8Ø machine language program and be ORG'ed (originated)in $81 or above. This allows room for Lazer's debugger program, DBUG.FRB, to reside in page $8Ø.

M The MONITOR command halts the ColecoVision and places you into the LZRMON monitor mode. This is accomplished by pressing "M". See the following section for further details.

Try NOT to enter the LZRMON monitor mode when sound is being generated by the ColecoVision, as the system bus is held by the Frob in the current ColecoVision processing steps. The steady state of sound can be quite annoying when in the monitor mode. This phenomenon also happens with video information.

R RESETting the ColecoVision is accomplishing by pressing the "R" command. This begins execution of the Z-8Ø program stored in the Frob RAM memory. If you want to execute a cartridge program you must first copy the cartridge into the Frob RAM space with the "C" command.

S SAVING a file to disk from the Frob program memory is done by executing the "S" command (i.e., S ØØ.FF). You will be asked to enter starting and ending page numbers. These values may be in the range $ØØ . . . $FF, allowing you to save the ColecoVision monitor ROM or RAM values to disk.

Q QUITting the LZRMON program is accomplished by pressing the "Q" command. You are returned to the CP/M operating system.

2.2.4 The Monitor Mode

The Lazer monitor was designed to resemble the Apple's native 65Ø2 monitor program. Anyone familiar with the Apple's monitor will feel right at home with the LZRMON monitor program.

The monitor program provides the means for easily examining and modifying memory (or I/O ports), disassembling Z-8Ø instructions, moving memory around (including moving data between main memory and VRAM), resetting the ColecoVision, setting breakpoints, and executing Z-8Ø programs.

When the main menu asks for a command reponse, type "M" to enter the monitor mode. In the monitor mode you are greeted with the prompt message "MEMORY:". It is the first of three standard prompts: "MEMORY:", "VRAM:", and "PORTS:".

The "MEMORY:" prompt tells you that all monitor commands following the prompt operate on the Z-8Ø 64K memory space.

The lower 32K corresponds to the ColecoVision monitor ROM, RAM, and other unassigned locations; the upper 32K of the memory space corresponds to the 32K of RAM found in the Frob unit.

Although the TI 9928 VDP (Video Display Processor) in the ColecoVision addresses 16K of VRAM (Video RAM), it is NOT directly addressable by the Z-8Ø microprocessor chip. However, special provisions have been made in the LZRMON program to allow you to view the VRAM as though it were directly addressable.

V You can place the monitor program into VRAM mode by typing "V" after the monitor prompt. In this mode all memory accesses will reference the 16K of VRAM on the TI 9928 VDP instead of the Z-8Ø 64K address space. As there are only 16K of VRAM (instead of the 32K of RAM), all addresses beyond $3FFF are truncated to 14-bits, and put you back into the range $ØØØØ . . . $3FFF. While in the VRAM mode the monitor prompt is "VRAM:". For example, if you type in $FFFF, VRAM Location address would be $3FFF.

As the Z-8Ø can address 256 I/O ports (in addition to the 64K of RAM and 16K of VRAM), provisions have been made to allow you to directly access I/O ports from the monitor program.

I Pressing "I" after the monitor prompt places the Lazer monitor program into the "PORTS" mode. You will be greeted with a "PORTS:" prompt and all memory references will access the Z-8Ø I/O ports instead of memory.

NOTE: Because of the operation of the MicroSoft Z-8Ø Software, some timing-dependent peripherals may not operate properly when referenced by the LZRMON monitor program in "PORTS" mode. For example, the VRAM accesses had to be handled by calling 65Ø2 routines from the Z-8Ø code in the Lazer Monitor program. Though most peripherals available from the ColecoVision unit should work just fine, it's good to be aware of this little peculiarity. Port address may be corrected in a future release of this software if users report any difficult problems with the code.

As the Z-80 can only address 256 different I/O ports, all addresses while in the "PORTS:" mode are truncated to eight bits.

N The "MEMORY:" prompt will appear by default anytime you enter the LZRMON monitor mode from the main menu. If you switch over to VRAM or I/O ports using the 'V' OR 'I' command, you can return to the memory mode by pressing "N" (for Normal memory) after the "VRAM:" or "PORT:" prompt message.

X This command will eXit you from wherever you may be within the monitor mode to the main LZRMON menu.

2.2.5 Displaying and Modifying Memory

The Lazer monitor program can display and modify the contents of any Z-80 memory location, VRAM memory location, or I/O port. As outlined in the last section, the type of memory access is specified by the "N", "V", and "I" commands. The monitor prompt specifies the type of memory access that will take place.

To display a single memory location, type the address of the location (in hex) you wish to display, followed by a return. The Lazer monitor program will respond by printing the data stored at that location.

You can print the next eight additional, sequential memory locations by pressing return. Each time you press return the next eight memory locations will be displayed. The output of the Lazer monitor program will display on either a forty- or eighty-column screen.

To display a range of memory locations, type the first and last memory locations you want displayed, separated by a period. For example, the command "8000.807F" displays the contents of memory locations $8000 . . . $807F. The data is displayed eight bytes per line (it fits within 40-columns), with a maximum of about $9F values on the screen at any one time.

You can send this data to the printer (located in Apple slot #1) by typing control-P at the beginning of the line (which turns the printer on). You can then print as much data as you want, the printer capturing it all.

More information on displaying memory locations is available in the Apple reference manual.

To modify sequential memory locations, use the ":" operator. The syntax for the memory change command is:

```
<<address>>: <<data>>
```

where <<address>> is a valid Z-8Ø memory, port, or VRAM address and <<data>> is a collection of one or more single byte hexadecimal values, separated by spaces. If two or more data values follow the colon then these values are stored in sequential memory locations starting at address <<address>>.

Examples:

```
8Ø34:22 55 76 34 9Ø 2C
FFØØ:Ø Ø Ø Ø Ø 5C 36
18ØØ:1 2 3 A CC
```

2.2.6 Moving Memory Around

The Lazer monitor program provides a mechanism for moving around blocks of memory in the Z-8Ø memory and VRAM spaces. The syntax for the move command is:

```
<<destination>> < <<start>>.<<end>>M
```

where <<destination>> is the destination address where the data is to be moved to, <<start>> is the address of the source block, and <<end>> is the address of the last byte to be moved. For example, the command:

```
FFØØ<8ØØØ.8ØFFM
```

moves the block of memory $8ØØØ . . . $8ØFF to locations $FFØØ . . . $FFFF.

The block move command does not consider situations where the two blocks overlap. If you are moving data from a lower memory address to a higher memory address and the blocks overlap certain sections of the data will be duplicated. This allows you to fill memory with a specified value. For more information on this technique consult the Apple reference guide.

The standard block move command always moves data around in the current default memory space (MEMORY or VRAM only).

Sometimes it is useful to move data from the VRAM to main memory or from main memory to VRAM. This is accomplished with a simple extension of the Lazer monitor move command. By immediately following the "M" with a "V" you can transfer data between the MEMORY and VRAM spaces.

If the current memory mode is "MEMORY:" and the monitor command is of the form:

```
<<dest>> < <<start>>.<<end>>MV
```

then data is copied from Z-8Ø memory locations

<<start>>.<<end>> into VRAM locations <<dest>> and up. If the current memory mode is "VRAM:" when this command is executed, data is copied from VRAM into the Z-8Ø memory space.

WARNING: Moving memory to <<dest>> can erase present memory in that location.

2.2.7 Debugging Functions

The DBUG.FRB Program and the "G" Command

G The Lazer monitor "G" command ("G" for GO) initiates communication with the special debugging program for the ColecoVision System.

For the "G" command to operate, you must first load the DBUG.FRB program into the ColecoVision's memory space. Although the program data is in the ColecoVision memory space, the program physically resides in the Frob 32K Interface Unit. Using the "L" command from the main LZRMON menu, load DBUG.FRB in at location $8ØØØ (page $8Ø).

As this program is roughly 512 bytes long, it is advisable to ORG your programs at location $84ØØ or higher, leaving plenty of room for the debugging monitor program. If you plan to use all 32K of RAM for your program, however, the debugging monitor program can be displaced or replaced with a smaller program.

There are three allowable syntaxes for the G command. They are:

	Example
<<run address>>G	FØØØG
<<bpt1>>.<<run address>>G	FØ1Ø.FØØØG
and	
<<bpt2>> < <<bpt1>>.<<run address>>G	FØ3Ø5<5FØ1Ø.FØØØG

The first form simply runs the program at the specified address. The second form allows you to set a single breakpoint. The third form of the G command allows you to set two breakpoints.

2.2.8 Breakpoints

Breakpoints are locations in memory that stop a program from running. They help the programmer trace the flow of events occurring within a program and are particularly useful when a bug is being tracked. With well-placed breakpoints, you can see where things begin to go awry.

Breakpoints are extremely useful when running a new ColecoVision program in the Frob Interface Unit memory. Also useful for testing and experimentation, they work by storing a RST 56 instruction at a specified memory address. While the program is operating, the DBUG.FRB program will recognize and then intercept RST 56 calls, display memory registers and program parameters, and then return system control to the Lazer monitor. This will happen whenever an RST 56 instruction is stumbled upon.

As a programmer, you do not physically have to put a RST 56 instruction in your program for a breakpoint. LZRMON will automatically do that for you when you ask for a breakpoint. **However, the DBUG program does not remove the RST 56 instruction after encountering it.**

This is very useful for testing purposes, but not something you want to happen in your final program.

It is important that you remove the RST 56 instructions from your program after testing is completed.

2.2.9 Monitor Communications

Once control is transferred from the system monitor, LZRMON, to the primary menu, you still have three forms of communication with the monitor:

1) You can call your routine from the monitor with a CALL instruction.

2) You can terminate program execution with a simple RET instruction. It is preferred that you terminate your program using the DBUG.FRB program and the breakpoint processor.

3) You can also print HEX or ASCII data on the Apple's video display by sending a special sequence of commands to the Frob I/O port.

This last form of communication is made possible by the IN 7CH and OUT 7DH instructions (both Z-80 instructions). Port 7C is known as the Frob status register. If bit seven is high, you can then transfer data to the Apple through port 7DH. If bit seven of port 7CH is low, then you must wait until it is high. It must be high to transfer data.

To print an ASCII character on the Apple screen, output a one (the Value, not the character) to port 7DH, watch for the byte to be 'taken' (bit 7 of port 7CH to go high), and then send the ASCII character you want printed through port 7DH.

If you prefer to print a byte as two hex digits, output a two (the Value, not the character) to port 7DH, watch for the byte to be 'taken,' and then send the hex byte you want printed through port 7DH.

Normally the Lazer monitor sits in a tight loop looking for output commands or a command byte specifying the end of execution. You can get the Apple to break this loop at any time by pressing the Return key on the Apple's keyboard.

If your Z-80/ColecoVision program gets stuck in an infinite loop, you can get out of it by pressing the Return key (returning you to the Lazer monitor). Then hit the Return key again to reset the ColecoVision. If the program terminates from a breakpoint execution, all of the Z-80 registers will be displayed. Like the control panel of an aircraft, this feature allows you to check the current states of the machine at a given point.

2.3.0 Lazer Mini-disassembler

Built into the Lazer monitor is a mini-disassembler that allows you to view code sequences in the Z-80 memory space. The syntax for this command is identical to the Apple's:

```
<<address>> L
```

This command lists out 20 Z-80 instructions to the screen using standard Zilog mnemonics.

2.3.1 The Lazer Monitor List Commands

Monitor Command Summary

<<adrs>>	Displays the contents of memory location <<adrs>>
<<adrsl>>.<<adrs2>>	Displays the contents of the memory locations in the range <<adrsl>> . . . <<adrs2>>.
<<adrs>> : <<data>>	Sets sequential memory locations to the values specified by <<data>>.
<<adrs>>L	Disassembles and lists 20 instructions at the specified address.
<<adrs>>G	Executes the Z-80 subroutine at the specified address.

<<a1>>.<<a2>>G	Sets a breakpoint at address <<a1>> and begins execution at address <<a2>>.
<<a1>> < <<a2>>.<<a3>>G	Sets two breakpoints at addresses <<a1>> and <<a2>> then executes Z-8Ø code beginning at address <<a3>>.
C	Copys data from cartridge slot to Frob RAM.
<<dest>> < <<a1>>.<<a2>>M	Moves data from address <<a1>> through <<a2>> to <<dest>>. <<dest>> is always in the same memory space as <<a1>> and <<a2>>.
<<dest>> < <<a1>>.<<a2>>MV	Moves the data from the currently selected memory space (memory or VRAM) to the other memory space.
G	Initiates DEBUG.FRB program.
I	Selects I/O port addressing.
L	Loads binary file from disk to Frob memory.
M	Halts ColecoVision, entering MONITOR mode.
N	Selects normal memory addressing.
R	Resets the ColecoVision.
Q	Quits LZRMON program, returning to CP/M.
S	Saves Frob program memory to disk.
V	Selects VRAM addressing.
X	Exits to the main LZRMON menu.

Chapter 3. ANATOMY OF A FROB

3.1 THE FROB INTERFACE UNIT

The Frob Interface Unit uses the Apple memory addresses associated with the Apple I/O card slot. The 16 location device space is used for control, and the 256 I/O space (usually used for ROM) is a one page window into the address space of the ColecoVision. To the Apple, the ColecoVision memory looks like 128 pages (256 bytes/page) of memory. The Interface has 32K bytes of static RAM that operate in the ColecoVision address space from 8ØØØH to FFFFH, the same space used by the cartridge. A control bit in the interface allows you to select cartridge or interface memory.

3.1.1 Page Register

The 256 byte window in the Apple address space can be mapped to any page in the ColecoVision address space by sending the 8-bit page number to the Page Register. This value will be used as the upper 8 bits on any access to the ColecoVision.

EXAMPLE:

Suppose we want to stop a cartridge running on the ColecoVision to examine the contents of one of the ColecoVision RAM locations. Assuming the Frob-Apple interface card is in slot 2, and the RAM location we want to see is at 6Ø55H in the ColecoVision, the steps are as follows:

1. Write ØAH to location CØACH. This stops the ColecoVision by setting BUSREQ. It also sets ROMSEL, but that must have already been set for the ColecoVision to have been running a cartridge.

2. Write 6ØH to location CØADH. This sets the one page window to the address range 6ØØØH to 6ØFFH in the ColecoVision. (This could just as well have been done before the step above.)

3. Read location C255H. This is now mapped to 6Ø55H in the ColecoVision. We could also write to it if we want to change its contents.

4. Write Ø8H to location CØACH. This gives the bus back to the ColecoVision processor and allows it to continue with its program.

3.1.2 Bidirectional Port

The Interface has a bidirectional I/O port with handshaking between the ColecoVision and the Apple. The port is in the ColecoVision I/O space on its side and so does not use any of the cartridge memory locations.

The bidirectional port permits a program in the ColecoVision to communicate with a program that is running in the Apple. This port consists of a data and a status register.

DATA REGISTER:

The data register of the bidirectional port is a simple holding location for data passed between the two machines. It is analogous to an air lock between a space station and a supply ship. People and supplies must pass through this bidirectional air lock, leaving the atmosphere in both vehicles intact. Operators on both vechicles check to see if the pressure in the airlock is the same (compatible) with their ship's atmospheric pressure before the door between the ship and the air lock is opened.

STATUS REGISTER:

The status register is similiar to the internal status register of a UART. Similiar to the pressure indicating devices in the air lock of the space station, Bit 7 (Most Significant Bit) of this register indicates whether or not the data register is ready to accept data. If bit 7 is true (high=one), then the data register is empty and is ready to accept data. If Bit 7 is false (low=zero), then the data register has data in it and is not ready to accept data. If bit 6 is true (high), data is waiting in the data port from the other side (either the Apple or the ColecoVision). It is important that these status bits are checked when data transfers are made, as it is possible to write over the data in the data register, even when the status bits indicate that the data register is full and not ready to accept data.

3.1.3 LED Indicators

The Frob Interface Unit uses LED indicators for showing you the status of the unit. These status signals can be manually changed using the Coleco Loader, command number 6. The LEDs may be interpreted as follows:

LED	INTERPRETATION
1- RST	Reset in progress.
2- MEM	Memory space is selected.
3- HOLD	ColecoVision bus is being held.
4- I/O	I/O space is selected.

5- WP (Write Protect) — This LED is illuminated when Bit 4 of the Frob Control Register is set. This prevents writes to the Frob memory from the Apple or ColecoVision.

6- CART (Cartrige Select) — This LED is illuminated when Bit 3 of the Frob Control Register is set. This indicates that the ColecoVision ROM is accessible at locations 8ØØØ - FFFF.

POWER — ColecoVision power switch is on.

Sections 3.1.4 to 3.1.6 will be of great use to programmers who wish to write their own operating system or modify the existing software for the Frob-Coleco hardware. Both the Apple Development program and the LZRMON software were based on the following hardware data.

Reading these sections will allow one to become more familiar with the internal architecture of the Frob Development System.

3.1.4 Device I/O Locations

The Apple side map looks like this:

DEVICE I/O LOCATIONS Ø-F (For slot 2, CØAØ to CØAF)

CØAØ to CØAB		--	not used
CØAC	Write	--	Control Register, write only
CØAD	Write	--	Page Register, write only
CØAE	Read	--	Status Register
CØAE	Write	--	Interrupt ColecoVision
CØAF	Read	--	Data from ColecoVision to Apple
CØAF	Write	--	Data from Apple to ColecoVision

3.1.5 Control Register Bits

Bit Ø — IORQ — If IORQ is set, then an access from the Apple to the ColecoVision will be to the I/O space of the ColecoVision. If IORQ is zero, an access will be to the memory space.

Bit 1 — BUSREQ — If BUSREQ is set, the interface will assert BUSREQ on the ColecoVision. This will hold the processor and allow the Apple to get at the ColecoVision memory or I/O space.

Bit 2 — RESET If RESET is set, the interface will assert the RESET line on the ColecoVision. This will hold the processor and allow bus access as in BUSREQ, but when reset is cleared, the ColecoVision will vector through its reset vector.

Bit 3 — ROMSEL When set, this bit selects the ColecoVision cartridge to be in the ColecoVision address space from 8ØØØH to FFFFH. If ROMSEL is not set, this space is filled by the RAM on the interface.

Bit 4 -- WPROT If this bit is set, the RAM on the interface is write protected.

Bits 5-7 -- not used

3.1.6 I/O Ports

The ports between the Apple and ColecoVision have two "data status" bits. If you read the Status Register on either side, Bit 7 will be high if it is ready to send data, and Bit 6 will be high if the port has data for you from the other side.

The Status Register is at location 7CH in the ColecoVision I/O space, with the Data Register in the next location at 7DH. The Apple can get at both sides of this port structure because it can access both the ColecoVision memory and I/O space. We have found this useful only for testing the ports.

When the Apple writes to device location ØEH (i.e. CØAEH for slot 2), the interrupt line of the ColecoVision Z8Ø will be pulled low. Though of no obvious use, this observation is a way of obtaining the attention of the Z8Ø without sending it through the reset structure.

3.2 Frob-Apple Interface Card

The Frob-Apple Interface Card is the PC board that plugs into slot #2 of your Apple computer. Once in place, it gives your Apple access to the Frob Interface Unit and the ColecoVision, with the ability to store your programs on your Apple disk storage system. Install it and forget it.

3.3 Frob-Coleco Cartridge Adapter

The Frob-Coleco EPROM cartridge adapter plugs into the ColecoVision cartridge slot. It has four sockets which will hold four 2764 (8K) EPROMS. This adapter and "burned" EPROMS (a maximum of 32K) is your demo cartridge. It will plug into any ColecoVision system cartridge slot for demonstration purposes.

Appendix A. Inside the ColecoVision

A.1 Memory Map for the ColecoVision

Address Range:	Function:
ØH-1FFFH	ColecoVision monitor ROM
8ØØØH-FFFFH	Cartrige memory range

The ColecoVision has 4ØØ hex (1Ø24 dec) bytes of read/write memory. This memory image is duplicated in eight different address ranges. For example, a write to location 6ØØØH accesses the same memory location as a write to location 64ØØH.

This read/write memory resides in the following ranges:

6ØØØ-63FF
64ØØ-67FF
68ØØ-6BFF
6CØØ-6FFF
7ØØØ-73FF (Coleco uses this address range in their monitor.)
74ØØ-77FF
78ØØ-7BFF
7CØØ-7FFF

A.2 I/O Map

Port Address	Name	Read Function	Write Function
Ø8ØH	KEY	----	Select keypads
ØCØH	JSTK	----	Select joysticks
ØBEH	VRAM	VRAM data	VRAM data
ØBFH	VDPR	----	Video registers
ØBFH	VSTAT	Video status	----
ØFCH	CTROL1	Controller 1 data	----
ØFFH	CTROL2	Controller 2 data	----
ØFFH	AUDIO	----	Send data to sound

Each game controller has both a keypad and a joystick. The hardware at the interface maintains a select flip/flop that chooses which data (keypad vs. joystick) is returned when the processor reads the CTROL port.

A.3 The ColecoVision Monitor Disassembly Introduction

The ColecoVision Monitor will become more understandable if you are familiar with the Z-8Ø instruction set, the TMS9928A (Texas Instruments) video display processor, the SN76489AN (Texas Instruments) sound generator and the construction of monitors in general.

The Monitor is an 8K program residing in locations Ø to 1FFFH. Typical of Z-8Ø machines, the processor executes the monitor at location ØØØØH after powerup. At this location, the stack pointer is set and program initialization takes place.

This program provides the software developer with many ready-made routines to expedite the software writing process. These fall into four general classes: sound drivers, video drivers, game controller drivers and data structure manipulation programs.

Froboo provides a commented disassembly of the Monitor in order to allow programmers to use these routines in their programs. We have worked through the driver routines and many of the data structure routines. The advanced structure manipulator routines have been left to the programmer. If you wish to use these routines we suggest you disassemble some Coleco games and look at another programmer's use of the ColecoVision Monitor.

Many routines have dual entry points (two dispatch addresses in the main jump table). One entry takes parameters directly in the processor registers, whereas, the other gets the same parameters from list pointers and descriptor tables. In order to use the routines of the Monitor, it is necessary to become skilled with their calling sequences, returned values, and side effects. We recommend that the programmer study the disassembly and construct test calling routines.

When using the monitor routines, one should always remember that the Monitor probably has bugs. Over time Coleco has probably constructed a bug list to keep their programmers out of trouble. Froboo does not have this information, and makes no claim that any routine works the way you think it does. So be very careful.

A.3.1 Relevant Cartridge Locations

The Monitor examines several locations at the beginning of the Coleco cartridge and bases its actions on the values stored there. Here are just a few:

Location	Meaning
08000H 08001H	- If 8000 contains a 55 and 8001 an AA, then immediately jumps indirect through location 800A without performing any initialization. If 8000 contains an AA and 8001 a 55, then the monitor puts up an introduction before starting the cartridge.
0800AH	- Contains the starting address of the program in the cartridge.
0800C	- Jump vector for RST 1 instruction
0800F	- Jump vector for RST 2 instruction
08012	- Jump vector for RST 3 instruction
08015	- Jump vector for RST 4 instruction
08018	- Jump vector for RST 5 instruction
0801B	- Jump vector for RST 6 instruction
0801E	- Jump vector for RST 7 instruction
08021	- Jump vector for the NMI service routine (Non-maskable Interrupt).

A.3.2 Significant Memory Locations

The Monitor uses some memory locations in the Ø73ØØ page:

Location	Meaning
Ø7Ø2ØH	- Pointer to the base of an array of sound control tables.
Ø7Ø22H	- Pointer to a sound program for the noise part of the sound chip.
Ø7Ø24H	- Pointer to a sound program for tone generator 1.
Ø7Ø26H	- Pointer to a sound program for tone generator 2.
Ø7Ø28H	- Pointer to a sound program for tone generator 3.
Ø7Ø2AH	- Shadow of the last noise control code sent.
Ø73B9H	- Stack pointer (grows down).
Ø73BAH	- Start of an area used to store parameters pass to routines called through PCOPY.
Ø73C3H	- Shadow of VDP register #Ø.
Ø73C4H	- Shadow of VDP register #1.
Ø73C7H	- Flag used by the RAM copy routine.
Ø73C8H	- Pseudo-random number seed.
Ø73EBH	- Counter for roller controller #1.
Ø73ECH	- Counter for roller controller #2.
Ø73EEH	- Shadow of controller #1 joystick.
Ø73EFH	- Shadow of controller #2 joystick.
Ø73FØH	- Shadow of controller #1 keypad.
Ø73F1H	- Shadow of controller #2 keypad.
Ø73F2H	- Table of words containing the actual base addresses of the various VDP tables. This table is indexed by 2*VDP register code.
Ø73FEH	- Temporary 2 byte store location.

A.3.3 Initialization

Initialization starts at location ØØØØ, the stack pointer set to 73B9. The program then jumps to ØØ6E, continuing initialization.

The Monitor first checks cartridge locations Ø8ØØØ-Ø8ØØ1 to see if the cartridge should be executed immediately. If so, it jumps indirect through location Ø8ØØA. If not it calls dispatch #27H, initializing the sound generator. The program then stores a ØØ33 to location Ø73C8 to seed the pseudo-random number routine.

A routine is then called at location 11Ø5, selecting the keypads and setting some variable locations to zero. These variable locations are then specified by the contents of Ø8ØØ8-Ø8ØØ9 in the cartridge. After clearing two more memory locations, the monitor jumps to the routine at 1319.

The routine at 1319 continues the initialization process, calling yet another routine at location 18B4, clearing the entire VRAM memory. And then a routine at location 18E9 is called, initializing the VDP registers.

There are two primary routines called by the routine at 18E9. The first is at 1FD9 and is mainly used to program and shadow VDP registers 1 and 2 and to program (but not shadow) register 7. The second routine is at 1FB8 and is called to initialize VDP registers 2,3,4,5 and 6.

These 5 registers all take a base table address as part of their programming. The routine at 1FB8 shadows the base address in a table indexed by a register number at location Ø73F2. It then translates the actual memory location into a code in a format acceptable to the appropriate VDP register.

Next, a routine at 1927 is called, copying the font table at 158B into the VRAM.

All of the necessary tables are then loaded into the VRAM area using the routine at 1FBE.

The monitor then checks to see if cartridge location Ø8ØØØH has an 'AA' in it and location Ø8ØØ1H has a '55' in it. If so, it does an indirect jump through location Ø8ØØAH to start the cartridge program. If not, the monitor puts up a message telling the user to turn off the machine before inserting the cartridge or system adapter. If this is left alone, it times out, turns off the screen, and loops forever.

VRAM base addresses are initialized as follows:

Name Table	18ØØH
Color Table	2ØØØH
Pattern Generator	ØØØØH
Sprite Attribute	1BØØH
Sprite Pattern	38ØØH

A.3.4 The Jump Table

The jump table is located at the end of the Monitor, starting at location 1F61H and continuing to the end. This series of jump instructions will remain fixed.

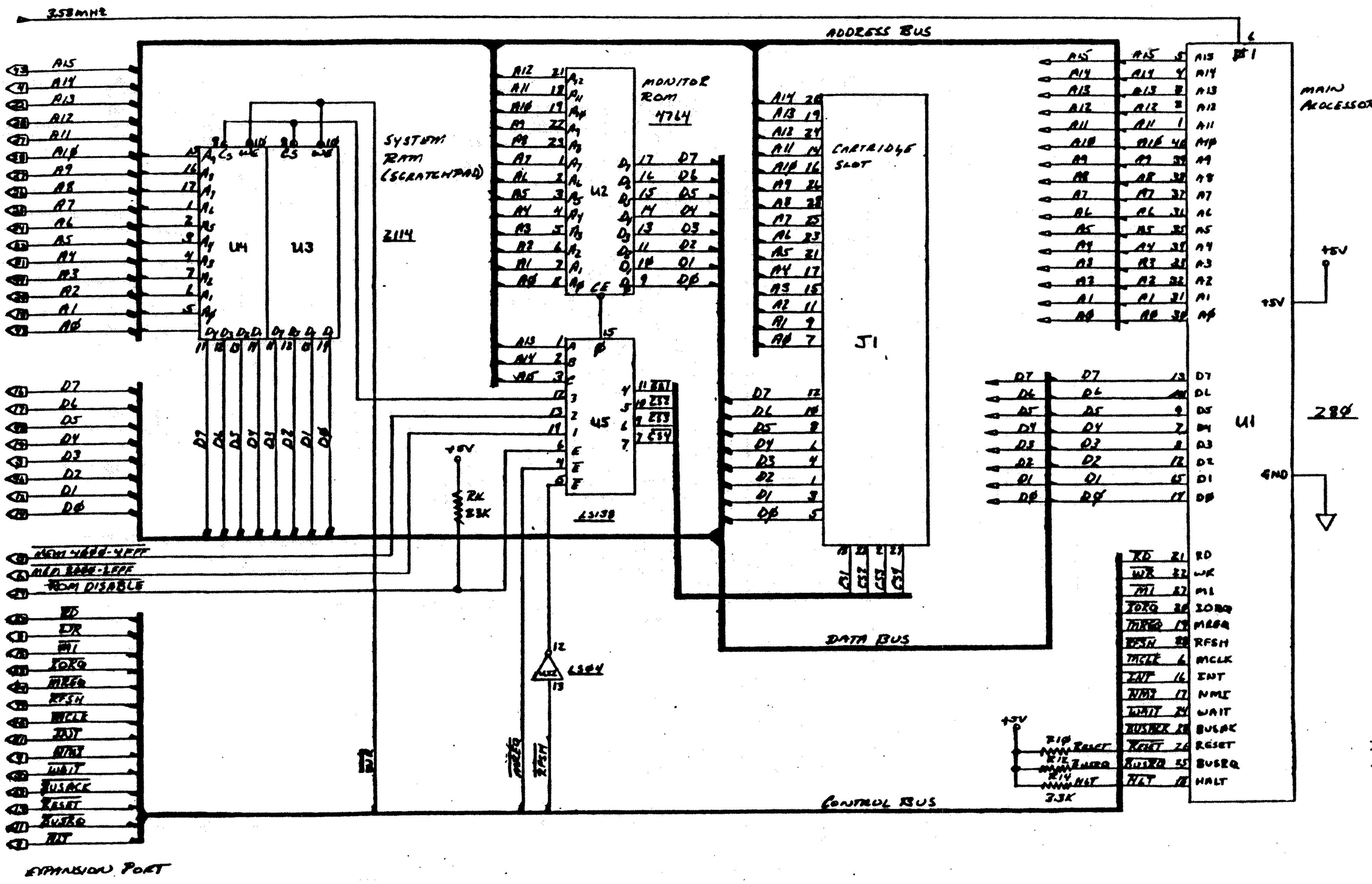
So, if you wish your cartridge programs to use Monitor routines, you should call them via the jump table entry points. Sometimes the monitor uses this jump table. Other times the monitor bypasses the table when using the jump routines.

As mentioned above, there are dual entry points for some routines. Careful study of the parameter passing routine at ØØØ98 should be made to understand how to call routines with indirect parameter passing.

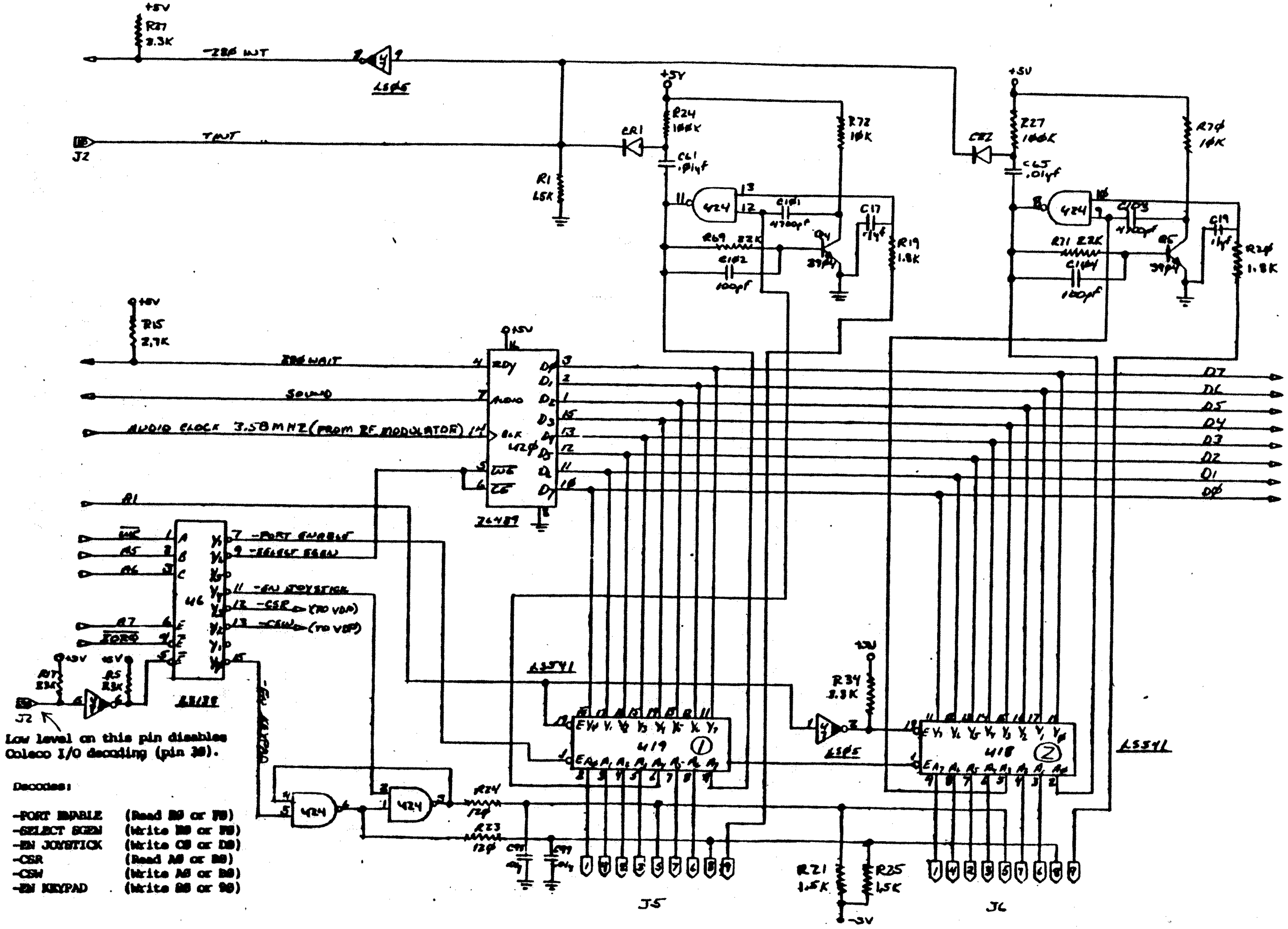
For further details about the function of the Monitor routines, refer directly to the commented disassembly of the Monitor contained in Appendix D.

Appendix A.4

COLECOVISION SCHEMATICS



A.4.1 PROCESSOR KERNEL

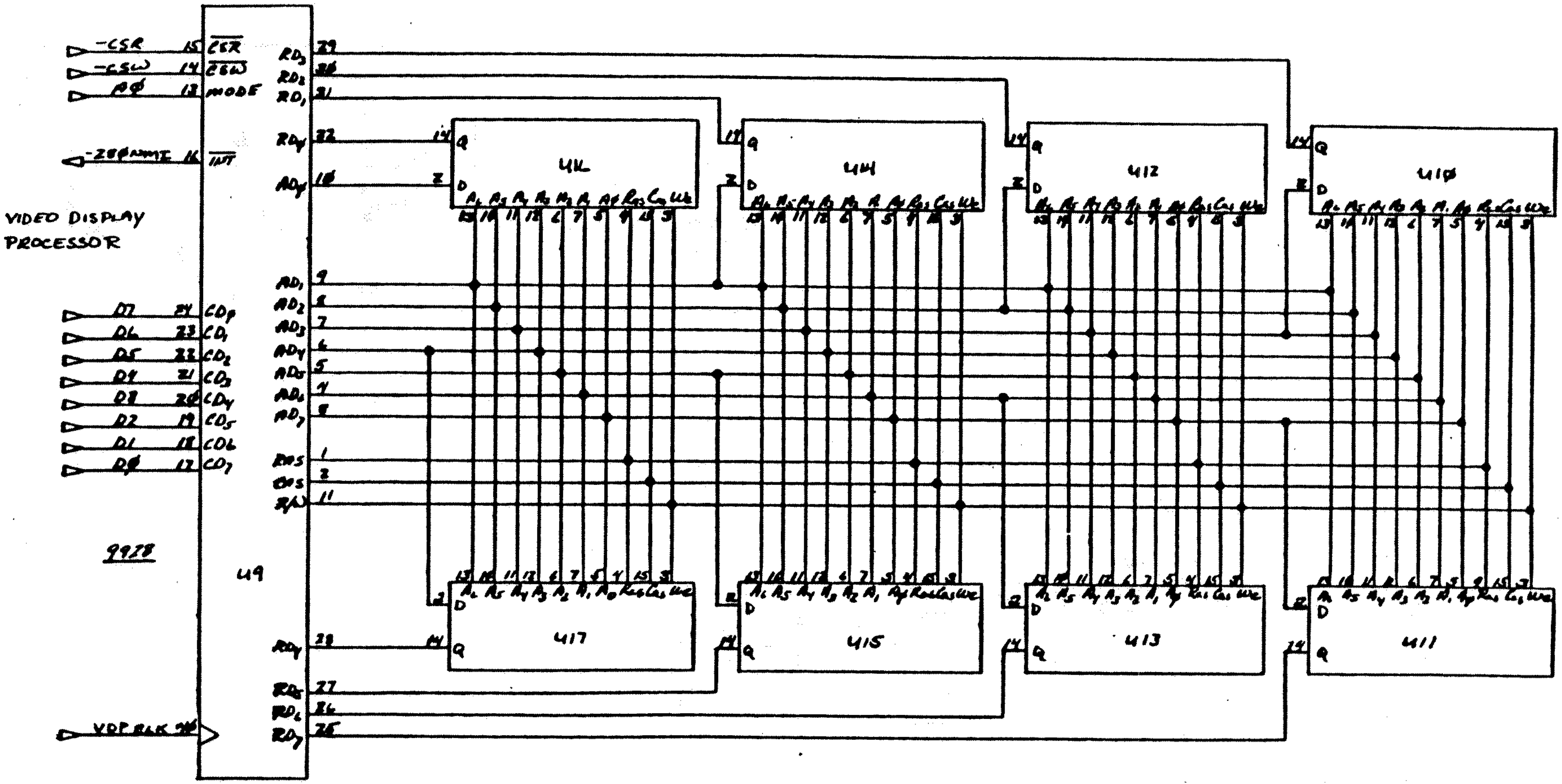


Low level on this pin disables Coleco I/O decoding (pin 38).

Decodes:

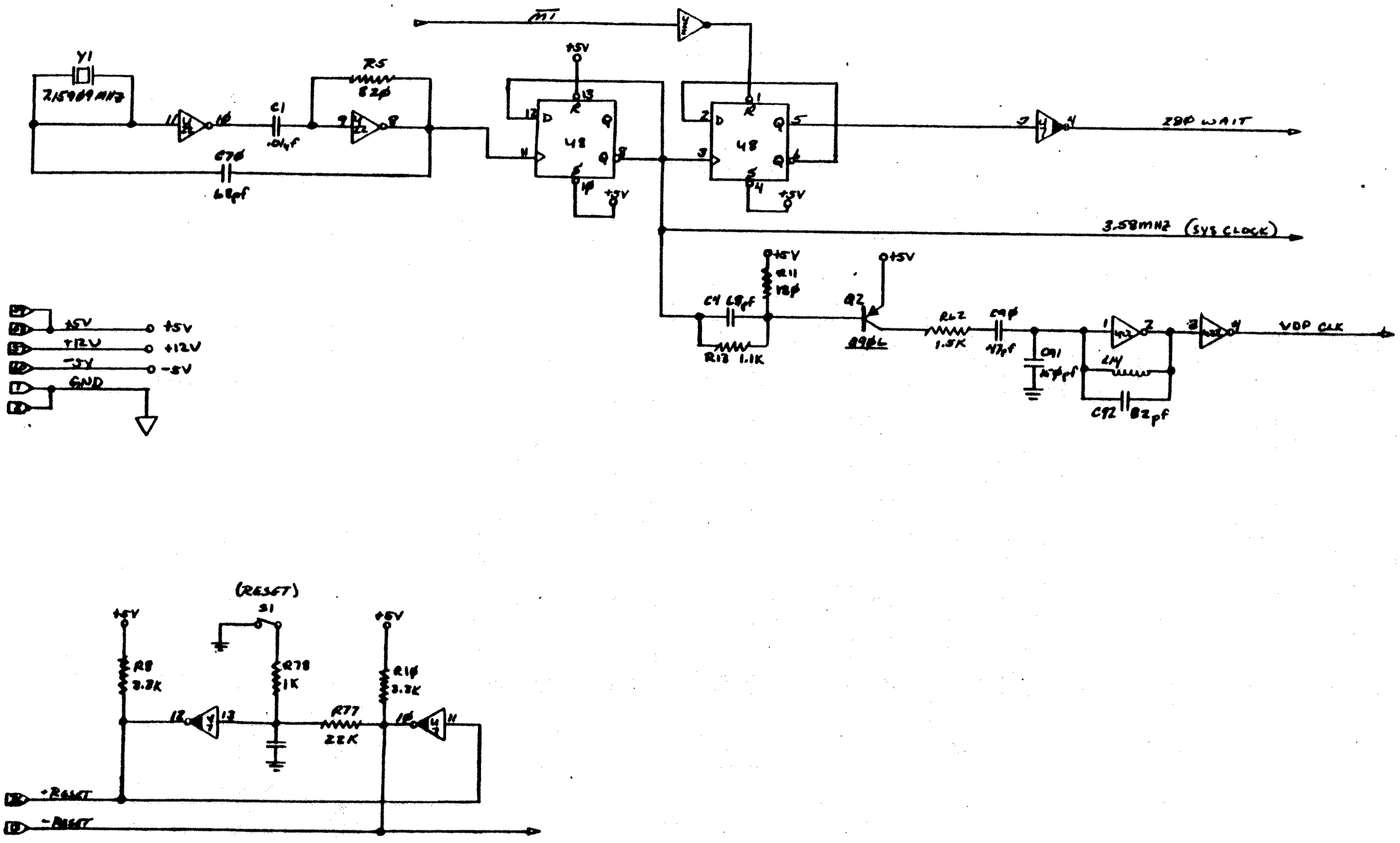
-PORT ENABLE	(Read E0 or F0)
-SELECT SGEN	(Write E0 or F0)
-EN JOYSTICK	(Write C0 or D0)
-CSR	(Read A0 or B0)
-CSW	(Write A0 or B0)
-EN KEYPAD	(Write 80 or 90)

A.4.2 I/O SECTION

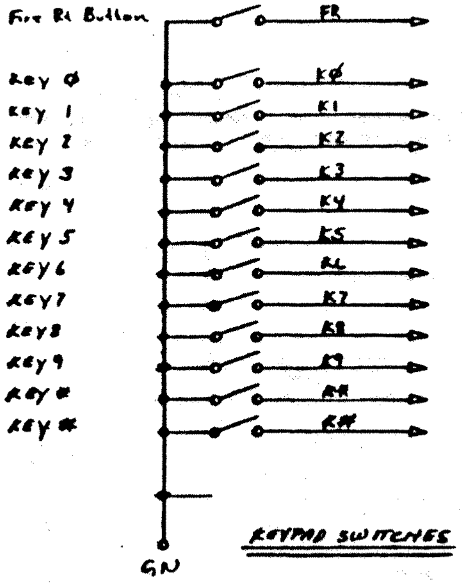


U10–U17 ARE 4116 COMPATIBLE
16K×1 DYNAMIC RAMS

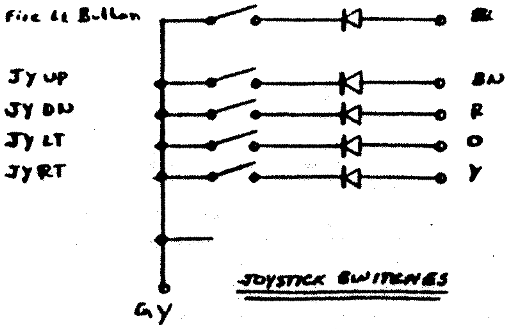
A.4.3 VIDEO PROCESSOR AND RAM



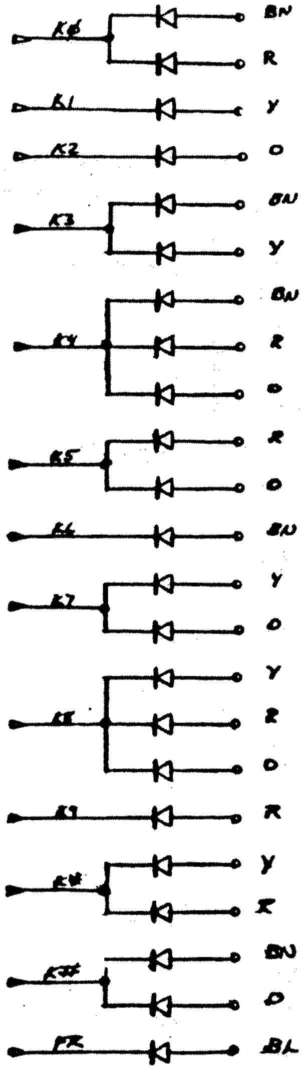
A.4.4 SYSTEM TIMING, POWER AND RESET SECTIONS



KEYPAD SWITCHES

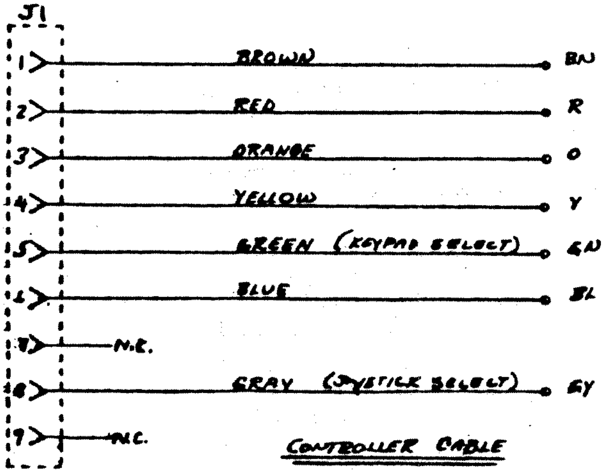


JOYSTICK SWITCHES



KEY PRESSED OR JOYSTICK POSITION	HEX CODE WITH FIRE BUTTON	HEX CODE WITHOUT FIRE BUTTON		1'S COMPLEMENT WITH FIRE BUTTON	1'S COMPLEMENT WITHOUT FIRE BUTTON
Ø	3A	7A		C5	85
1	3D	7D		C2	82
2	37	77		C8	88
3	3C	7C		C3	83
4	32	72		CD	8D
5	33	73		CC	8C
6	3E	7E		C1	81
7	35	75		CA	8A
8	31	71		CE	8E
9	3B	7B		C4	84
*	39	79		C6	86
#	36	76		C9	89
UP	3E	7E		C1	81
DOWN	3B	7B		C4	84
LEFT	37	77		C8	88
RIGHT	3D	7D		C2	82
UP+LEFT	36	76		C9	89
UP+RIGHT	3C	7C		C3	83
DOWN+LEFT	33	73		CC	8C
DOWN+RIGHT	39	79		C6	86

KEY PAD AND JOYSTICK CODES



CONTROLLER CABLE

Note:

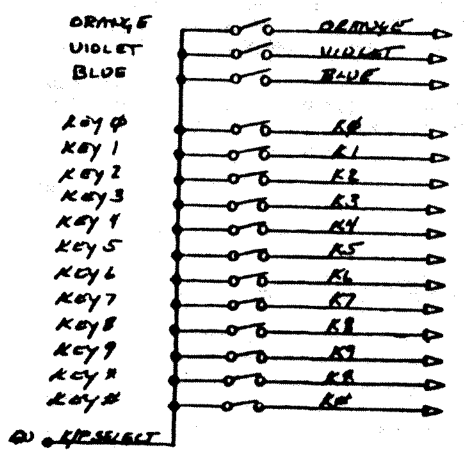
"GN" is the keypad select line. This line will go low and select the keypad switches if a write is made to any I/O location in the range $80 through $9F. It will remain low until the joystick is selected.

"GY" is the joystick select line. If goes low and selects the joystick switches when a write is made to any I/O location in the range $C0 through $DF. The joystick will remain selected until the keypad is selected.

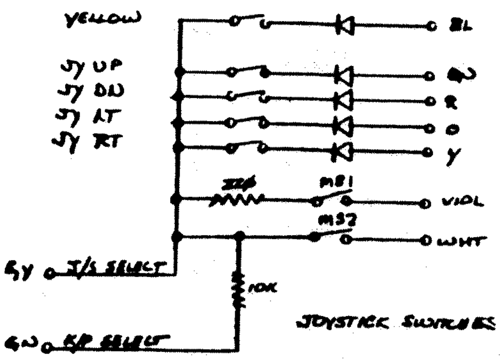
J1 is a 9-pin "D" connector that plugs into the ColecoVision.

A.4.5 STANDARD CONTROLLER

A.4.6 SUPER ACTION CONTROLLER

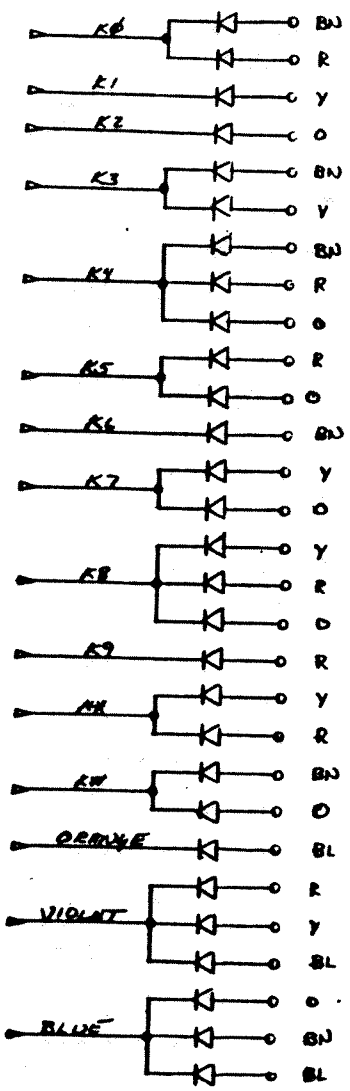


KEYPAD SWITCHES



Note:

Switches MS1 and MS2 are magnetically actuated by two magnets in the controller wheel. Each switch is actuated twice by one revolution of the wheel.



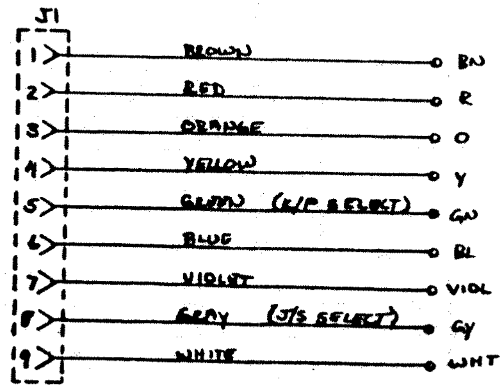
BUTTON PRESSED	HEX CODE	1's COMPLEMENT
VIOLET	39	C6
BLUE	36	C9
SEE NOTE	SEE NOTE	SEE NOTE

Note:

All other key and joystick codes are the same as standard controller codes.

The orange button is the same functionally as the fire left button on the standard controller.

The yellow button is the same as the fire right button.



CONTROLLER CABLE

J1 is a 9-pin "D" connector that plugs into the ColecoVision console.

A.4.7 Expansion Port Pin Number and Signal Names

1. GROUND
2. GROUND
3. DATA 3
4. ADDRESS 14
5. -MEM 4000-4FFF
6. -MEM 2000-2FFF
7. -HLT
8. -WR
9. -NMI
10. +INT
11. -BUSRQ
12. DATA 1
13. -RESET
14. DATA 0
15. -M1
16. DATA 7
17. DATA 6
18. ADDRESS 1
19. DATA 4
20. ADDRESS 2
21. ADDRESS 4
22. ADDRESS 13
23. ADDRESS 5
24. ADDRESS 6
25. ADDRESS 7
26. ADDRESS 8
27. ADDRESS 9
28. ADDRESS 10
29. -ROM DISABLE
30. -DIS COLECO DECODING
31. (FROM RF MODULATOR)
32. (FROM RF MODULATOR)
33. (FROM RF MODULATOR)
34. -RESET SWITCH
35. AUDIO CLOCK (3.58MHZ)
36. +RESET
37. ADDRESS 11
38. ADDRESS 12
39. -RESET/SYNC
40. 3.58 MHZ
41. N.C.
42. N.C.
43. ADDRESS 15
44. ADDRESS 3
45. -MCLK
46. DATA 2
47. ADDRESS 0
48. DATA 5
49. -RFSH
50. -WAIT
51. -INT
52. -BUSACK
53. -RD
54. -MREQ
55. -IORQ
56. SOUND
57. +12 VOLTS
58. +5 VOLTS
59. +5 VOLTS
60. -5 VOLTS

Appendix B. The TI 9928 Video Display Processor

B.1 Introduction

The ColecoVision's graphics capabilities are impressive, effective usage of the Texas Instrument 9928 Video Display Processor chip (VDP). This device generates all the necessary video, control, and synchronization signals used in creating a color graphics image on a home TV set. It also controls the storage, retrieval, and refresh of display data located in its own dynamic screen refresh memory.

NOTE: For a detailed technical programming description of the 9928, refer to documentation created for the chip by Texas Instruments.

The VDP communicates with the ColecoVision Z-8Ø microprocessor via an 8-bit bidirectional data bus. Three control lines, decoded from the Z-8Ø address and enable lines, determine interpretation of the bus. Through this bus, the Z-8Ø can write to VRAM, read from VRAM, write to VDP registers, and read the VDP status--important procedures to learn when programming the ColecoVision.

The VDP has its own 16K of dynamic RAM, separate and distinct from the 64K of main system RAM. The contents of VRAM define many of the parameters of the TV image. The VDP also handles its own refresh for its 16K of memory.

The video output of the VDP goes to an external video modulator. The output of the modulator goes to the TV set.

B.2 Sprites

One of the most important concepts in computer graphics, a sprite is a specific pattern positioned on the screen by a set of horizontal and vertical coordinates. Sprites allow relative ease of programming for creating three dimensional moving graphic effects. They come in two sizes: 8 X 8 pixels and 16 X 16 pixels. A pixel is the smallest point on the TV screen that can be independently controlled. The sprites can be further magnified by 2 to 16 X 16 and 32 X 32 pixels respectively.

Sprites exist on square two dimensional planes. Imagine 32 transparent, rectangular windows. Now put one in front of the other, forming a giant sandwich of windows. Each window (or more appropriately, 'plane') is the home of one sprite. Looking from the front end, you would see 32 sprites, all at home on each of their planes. Naturally, sprites in the front would partially block off the view of the sprites in the rear. This 'blocking off' effect is the result of a priority system assigned to the sprites.

Objects on planes closest to the viewer have higher priority. When the scenes on two different planes occupy the same spot on the screen, the scene on the higher priority plane will be seen by the viewer. For one of the lower priority planes to be viewed, all planes in front of that plane must be transparent at that point.

Remember, only one sprite is allowed per sprite plane and all area outside of the sprite itself is transparent. The sprite plane with the highest priority is called Sprite Ø and the lowest priority is Sprite 31.

There are two more planes behind Sprite 31. First, there is the pattern plane, used for non-sprite textual and graphics data. In an animation scene, for example, the pattern plane would contain parts of the picture that do not move, like the mountains or sky in the background.

Behind the pattern plane is the backdrop or background plane. It is larger than the other planes so that it can form a border around them and is always a solid color.

The last plane in the hierarchy is called the External VDP plane. The TMS9928A allows the input of an external video source, either from a camera, video tape recorder, computer CRT, or another VDP. This plane is not available on the ColecoVision.

The sprites, pattern plane, and background plane may also be thought of as independent video sources. Unlike single plane video display generators, the three sources can be combined to create a single image on the ColecoVision screen.

Moving objects in the sprite system does not require repainting the entire display screen. By simply changing 2 bytes in the Sprite Attribute Table (located in VRAM), each of the sprites can be made to move smoothly across the screen. Remember also, the pixel definitions will remain with the highest priority plane at any spot. This hidden view capability is provided in hardware and requires no complex software algorithms to implement.

A normal sprite is defined by an 8 X 8 pixel pattern stored in VRAM (8 bytes). **A sprite can only be a single color.** Wherever a 1 is stored in that pattern, the sprite will be colored and where it is a Ø, it will be transparent.

All the things that define a sprite, such as its color, coordinate position in its plane, and name are stored in a special table called the Sprite Attribute Table. Each set of attributes is 4 bytes. Since there are 32 sprites available for display, the table is 128 bytes long. Finding the attributes of a particular sprite is simply a matter of taking the sprite number times 4 plus the base address of the table.

The first two bytes of each entry of the Sprite Attribute Table set the X-Y positional coordinates of the sprite on the screen (referenced from the upper left corner). The first byte indicates the vertical distance of the sprite from the top of the screen, in pixels. The second byte describes the horizontal displacement of the sprite from the left edge of the screen.

The third byte is the pattern address of the specific sprite. This number specifies what the sprite should look like and is a pointer to the Sprite Generator Table, where more detailed information about the sprite is stored.

The fourth byte defines the sprite color (Refer to the TI 9928 data manual, page 2-17, for the color hex assignments).

B.3 Size and Magnification

Size and magnification are used to define the resolution and relative area covered by the sprites. Once defined, they apply to all the sprites. The size also determines the resolution of a sprite. A size of zero (Ø) says that the sprite will be 8 X 8 pixels and a size of one (1) says it will be 16 X 16 pixels. The smaller the sprite, the more resolution on the final display.

Magnification sets the relative area covered by the sprites. For a magnification setting of zero (Ø), the sprites are displayed as either 8 X 8 or 16 X 16, depending on the size. However, a magnification setting of one (1) doubles the area covered by doubling the area of each dot defined. An 8 X 8 sprite would be displayed as 16 X 16, but with half the resolution, and a 16 X 16 would be displayed as 32 X 32, again with half the resolution. This feature allows large areas of the screen to be covered by large sprites, but with reduced resolution. The low resolution image is a lot more 'blockish,' and so less appealing to the eye.

B.4 Example

Let's create a three dimensional image from sprites. Consider a scene where a flying saucer is zooming over cars slowed in typical rush hour traffic. In the background is the city of San Francisco, the Transamerica pyramid building silhouetted by a beautiful sunset, as the sun sinks into the Pacific Ocean.

To create this image, most of the background is "painted" on the pattern plane (similar to any conventional two-dimensional display).

Starting from the point closest to you (the observer), there is a hitchiker with hir thumb out, hoping to catch a ride from the rush hour people. Naturally, when the cars move past hir, they would appear to pass behind hir, yet stay in front of the city. This effect is readily achieved using the sprite system.

Being in the foreground, the hitchhiker is displayed as sprites Ø and 1 (in almost all cases, because of the limitions in sprite size and color, it is often necessary to use multiple sprites to define a single image. These sprites are then moved sychronously across the screen as one image). Sections of the freeway cars would be drawn on sprites 2 through 12. Clouds in the sky are drawn from 13 to 17. The setting sun, from 17 to 2Ø. Sprites 21 through 31 are left transparent.

The flying saucer may be created from any combination of sprites, breaking all the rules of foreground and background. One object can be created from several sprites. I leave this up to your imagination..Perhaps our hitch-hiker will get a ride, 'desprite' it all. Perhaps s/he will discover new forms of huemur on other planes....

Appendix C. The TI SN76489AN Sound Generator

C.1 Introduction

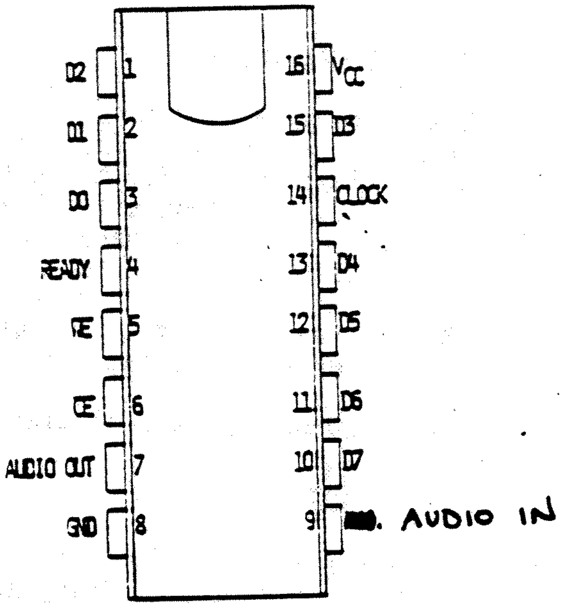
The sound system uses the very versatile SN76489AN digital sound generation chip by Texas Instruments. Is is easy to program and can create many complex, unusual sounds, including bizarre alien voices, comparable to today's professional arcade game chips.

The SN76489AN is a dedicated sound chip. There is no costly overhead on the Z-80 microprocessor, the chip doing all the sound generation work itself. It has eight internal registers used to control the three tone generators and the noise source. The wide range of frequencies possible is created from the input clock frequency on the chip which vibrates at 3.58 Mhz. Fun to play with, the output of this chip is sent directly into the modulator and then to the TV.

For a full technical description of the SN76489AN sound generator and examples of how to program the chip refer to the Texas Instruments documentation that follows.

C.2 TI SN76489AN Sound Generator Datasheet

NOVEMBER, 1981



FEATURES

- 3 Programmable tone generators
- Programmable white noise generator
- Programmable attenuation
- Simultaneous sounds
- TTL compatible
- Up to 4MHz clock input *

DESCRIPTION

The SN76489AN digital complex sound generator is an I²L/Bipolar IC designed to provide low cost tone/noise generation capability in microprocessor systems. The SN76489AN is a data bus based I/O peripheral.

RECOMMENDED OPERATING CONDITIONS

PARAMETER	MIN	TYP	MAX	UNITS
Supply Voltage, V_{CC}	4.5	5.0	5.5	V
High Level Output Voltage, V_{OH} (pin 4)			5.5	V
Low Level Output Current, I_{OL} (pin 4)			2	mA
Operating Free-Air Temperature, T_A	0		70	°C

* Part SN76489AN is identical to the SN76494N except that the maximum clock input frequency is 4MHz (500kHz for the SN76494N). A "divide-by-eight" stage is included in the input circuitry and 32 clock pulses are required to load the data, compared to 4 pulses for the SN76494N.

OPERATION

1. TONE GENERATORS

Each tone generator consists of a frequency synthesis section and an attenuation section. The frequency synthesis section requires 10 bits of information (F0-F9) to define half the period of the desired frequency (n). F0 is the most significant bit and F9 is the least significant bit. This information is loaded into a 10 stage tone counter, which is decremented at a N/16 rate where N is the input clock frequency. When the tone counter decrements to zero, a borrow signal is produced. This borrow signal toggles the frequency flip-flop and also reloads the tone counter. Thus, the period of the desired frequency is twice the value of the period register.

The frequency can be calculated by the following:

$$f = \frac{N}{32n}$$

where N = ref clock in Hz
n = 10 bit binary number

The output of the frequency flip-flop feeds into a four stage attenuator. The attenuator values, along with their bit position in the data word, are shown in Table 1. Multiple attenuation control bits may be true simultaneously. Thus, the maximum attenuation is 28 db.

Table 1 ATTENUATION CONTROL

BIT POSITION				
A0	A1	A2	A3	WEIGHT
0	0	0	1	2 db
0	0	1	0	4 db
0	1	0	0	8 db
1	0	0	0	16 db
1	1	1	1	OFF

2. NOISE GENERATOR

The Noise Generator consists of a noise source and an attenuator. The noise source is a shift register with an exclusive OR feedback network. The feedback network has provisions to protect the shift register from being locked in the zero state.

TABLE 2 NOISE FEEDBACK CONTROL

FB	CONFIGURATION
0	"Periodic" Noise
1	"White" Noise

Whenever the noise control register is changed, the shift register is cleared. The shift register will shift at one of four rates as determined by the two NF bits. The fixed shift rates are derived from the input clock.

TABLE 3 NOISE GENERATOR FREQUENCY CONTROL

BITS		
NFO	NFI	SHIFT RATE
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Tone generator #3 output

The output of the noise source is connected to a programmable attenuator as shown in Figure 4 (see page 11).

3. OUTPUT BUFFER/AMPLIFIER

The output buffer is a conventional operational amplifier summing circuit. It sums the three tone generator outputs and the noise generator output. The output buffer will generate up to 10mA.

To prevent oscillations in the output buffer, the output (pin 7) should be decoupled. This is done by putting 10 ohms in series with 0.1µF from pin 7 to ground (see figure 2).

4. CPU to SN76489AN INTERFACE

The microprocessor interfaces with the SN76489AN by means of the 8 data lines and 3 control lines ($\overline{WE}$, $\overline{CE}$ and READY). Each tone generator requires 10 bits of information to select the frequency and 4 bits of information to select the attenuation. A frequency update requires a double byte transfer, while an attenuator update requires a single byte transfer.

If no other control registers on the chip are accessed, a tone generator may be rapidly updated by initially sending both bytes of frequency and register data, followed by just the second byte fo data for succeeding values. The register address is latched on the chip, so the data will continue going into the same register. This allows the 6 most significant bits to be quickly modified for frequency sweeps.

5. CONTROL REGISTERS

The SN76489AN has 8 internal registers which are used to control the 3 tone generators and the noise source. During all data transfers to the SN76489AN, the first byte contains a three bit field which determines the destination control register. The register address codes are shown in Table 4.

TABLE 4 REGISTER ADDRESS FIELD

R0	R1	R2	DESTINATION CONTROL REGISTER
0	0	0	Tone 1 Frequency
0	0	1	Tone 1 Attenuation
0	1	0	Tone 2 Frequency
0	1	1	Tone 2 Attenuation
1	0	0	Tone 3 Frequency
1	0	1	Tone 3 Attenuation
1	1	0	Noise Control
1	1	1	Noise Attenuation

6. DATA FORMATS

The formats required to transfer data are shown below.

	REG ADDR			DATA			
1	R0	R1	R2	F6	F7	F8	F9

BIT 0 FIRST BYTE BIT 7

		DATA					
0	x	F0	F1	F2	F3	F4	F5

BIT 0 SECOND BYTE BIT 7

UPDATE NOISE SOURCE (SINGLE BYTE TRANSFER)

	REG ADDR					SHIFT	
1	R0	R1	R2	x	FB	NF0	NF1

BIT 0 BIT 7

UPDATE ATTENUATOR (SINGLE BYTE TRANSFER)

	REG ADDR			DATA			
1	R0	R1	R2	A0	A1	A2	A3

BIT 0 BIT 7

7. The microprocessor selects the SN76489AN by placing $\overline{\text{CE}}$ into the true state (low voltage). Unless $\overline{\text{CE}}$ is true, no data can occur. When $\overline{\text{CE}}$ is true, the $\overline{\text{WE}}$ signal strobes the contents of the data bus to the appropriate control register. The data bus contents must be valid at this time.

   The SN76489AN requires approximately 32 clock cycles to load the data into the control register. The open collector READY output is used to synchronize the microprocessor to this transfer and is pulled to the false state (low voltage) immediately following the leading edge of $\overline{\text{CE}}$. It is released to go to the true state (external pullup) when the data transfer is completed.

The data transfer timing is shown below.

DATA TRANSFER TIMING

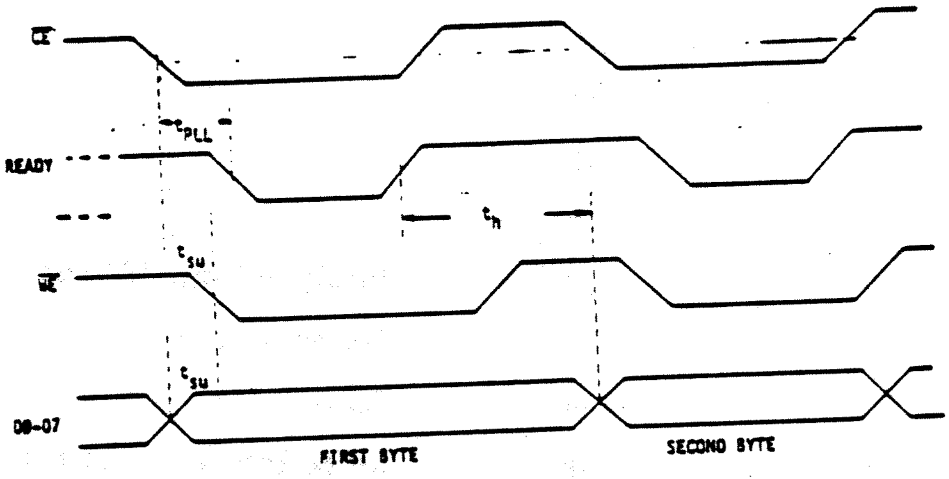


Figure 1.

TABLE 5 FUNCTION TABLE*

Inputs		Output
$\overline{\text{CE}}$	$\overline{\text{WE}}$	READY
L	L	L
L	H	L
H	L	H
H	H	H

* This table is valid when the device is:
(1) not being clocked, and
(2) is initialized by pulling $\overline{\text{WE}}$ and $\overline{\text{CE}}$ high.

8. PIN ASSIGNMENT

The table below defines the SN76489AN pin assignment and describes the function of each pin.

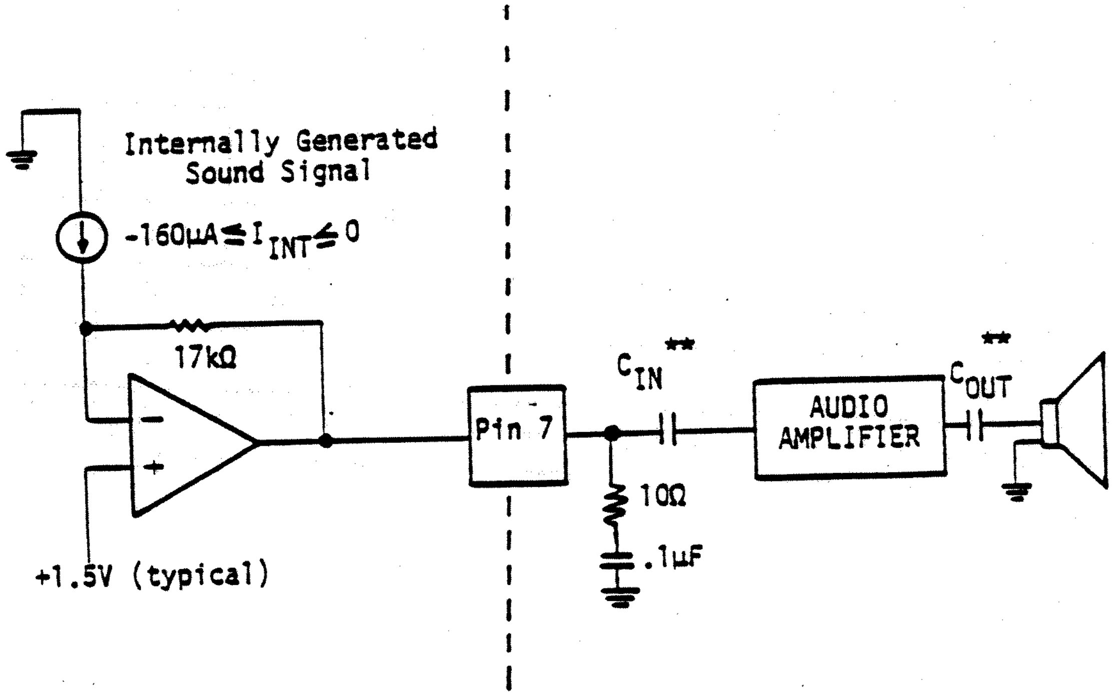
SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{\text{CE}}$	6	IN	Chip Enable - when active (low) data may be transferred from CPU to the SN76489AN.
D0(MSB)	3	IN	D0 through D7 - Input data bus through which the control data is input.
D1	2	IN	
D2	1	IN	
D3	15	IN	
D4	13	IN	
D5	12	IN	
D6	11	IN	
D7	10	IN	
VCC	16	IN	Supply Voltage (5V nom)
GND	8	OUT	Ground Reference
CLOCK	14	IN	Input Clock
$\overline{\text{WE}}$	5	IN	Write Enable - when active (low), $\overline{\text{WE}}$ indicates that data is available from the CPU to the SN76489AN.
READY	4	OUT	When active (high), READY indicates that the data has been read. When READY is low, the microprocessor shoulc enter a wait state until READY is high.
AUDIO IN ~~N.C.~~	9		~~No external connection should be made in this pi~~n.
AOUT	7	OUT	Audio Drive Out

Iectrical characteristics over recommended operating conditions (unless otherwise noted)

PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNITS
I_I Input Current	V_{IN} = GND to V_{CC}	$\overline{CE}$		-25	-175	µA
		DØ-D7, $\overline{WE}$, CLK		-10	-70	µA
V_{OL} Low Level Output Voltage	I_{OUT} = 2mA READY			.25	.4	V
I_{CC} Supply Current	Outputs Open			30	50	mA
C_I Input Capacitance					15	pF
I_{OH} High Level Output Current	READY				10	µA
V_{IH} High Level Input Voltage	DØ-D7, $\overline{WE}$, $\overline{CE}$, CLK		2			V
V_{IL} Low Level Input Voltage	BØ-D7, $\overline{WE}$, $\overline{CE}$, CLK				.8	V
2dB Attenuation			1	2	3	dB
4dB Attenuation			3	4	5	dB
8dB Attenuation			7	8	9	dB
16dB Attenuation			15	16	17	dB

AUDIO INPUT

EXTERNAL AUDIO OUTPUT INTERFACE



** These capacitance values are determined by the frequency response desired and the audio amplifier used.

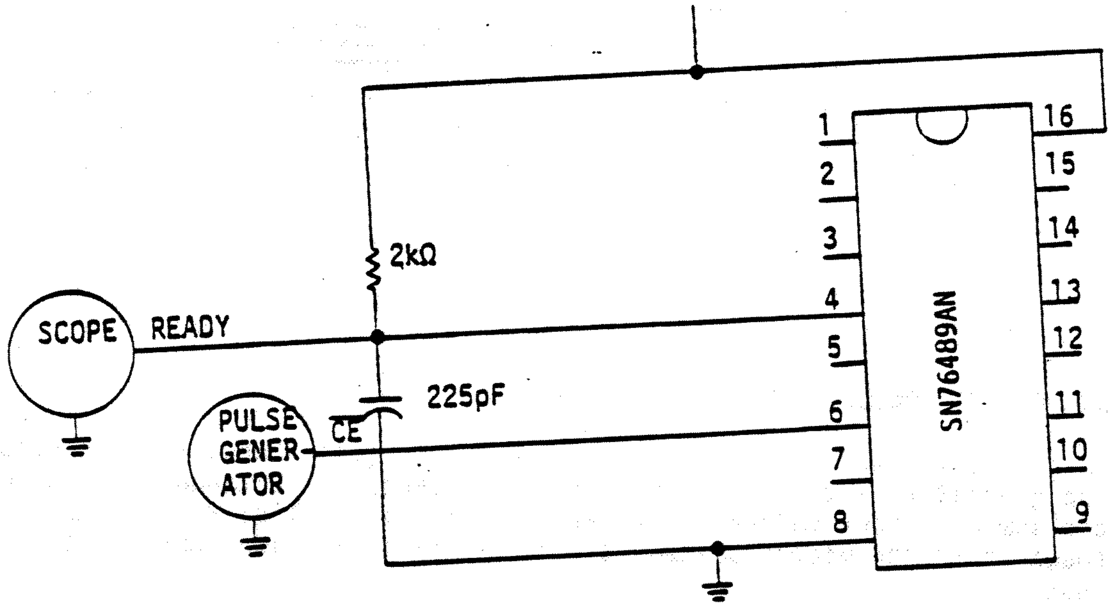
Figure 2.

ADVANCED CIRCUITS

Switching Characteristics, V_{CC} = 5V, T_A = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
* $\overline{CE}$ to READY t_{PLL}, 50% to 50%	C_L = 225pF R_L = 2K to V_{CC}		90	150	nS
f_{clock}, Input Clock Frequency	Clock Transition Time (10% to 90%) 10µS	DC	3.579	4	MHz
Setup Time, t_{su} (see Figure 1)	DATA W.R.T. $\overline{WE}$	0			nS
	$\overline{CE}$ W.R.T. $\overline{WE}$	0			nS
Hold Time, t_h (see Figure 1)	DATA W.R.T. READY	0			nS

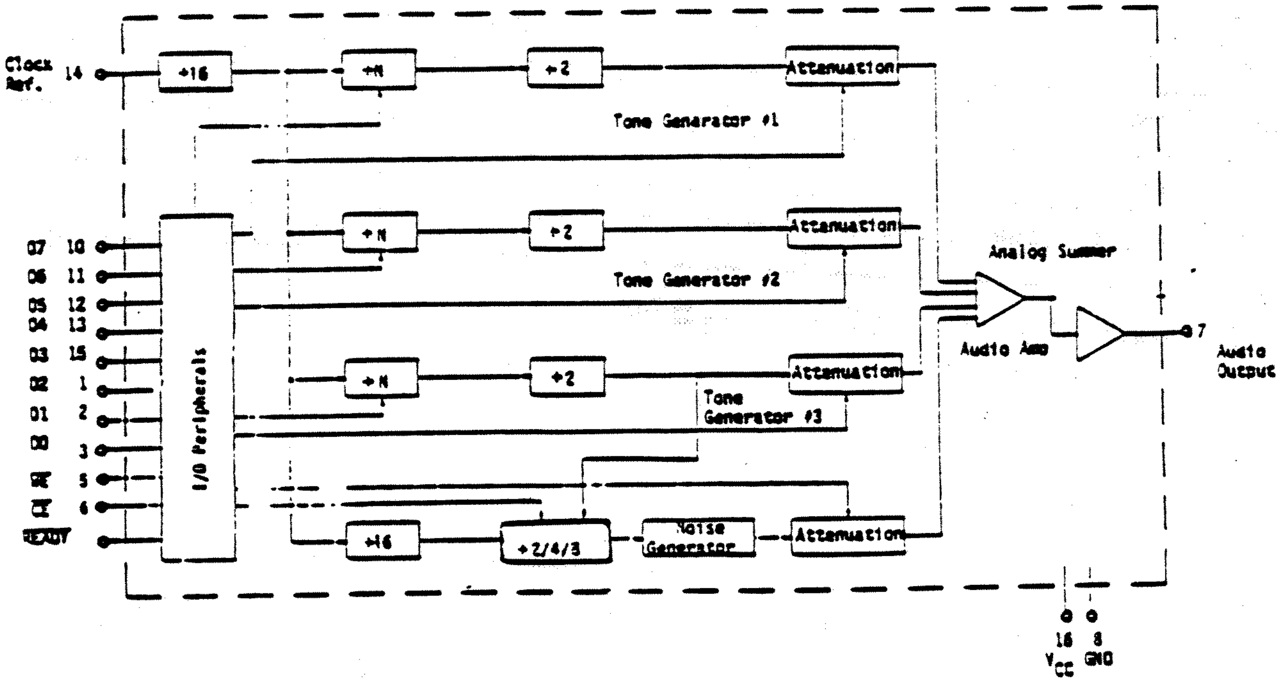
* $\overline{CE}$ Pulse: 0-3V, $t_{rise} \leq 7nS$, $t_{fall} \leq 7nS$



t_{PLL} TEST CIRCUIT

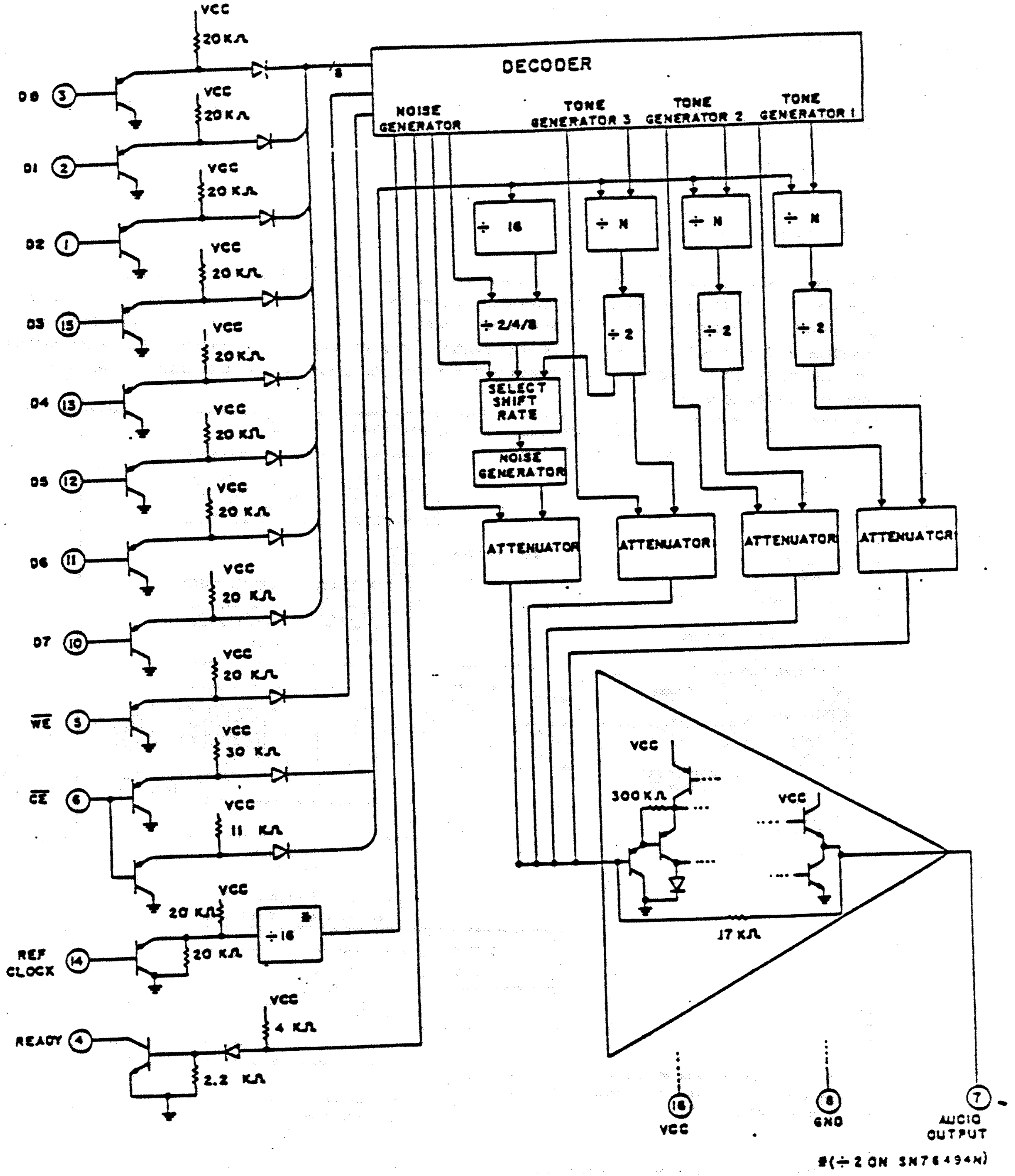
Figure 3.

BLOCK DIAGRAM



BLOCK DIAGRAM DESCRIPTION

This device consists of three programmable tone generators, a programmable noise generator, a clock scaler, individual generator attenuators and an audio summer output buffer. The SN76489AN has a parallel 8 bit interface through which the microprocessor transfers the data which controls the audio output.



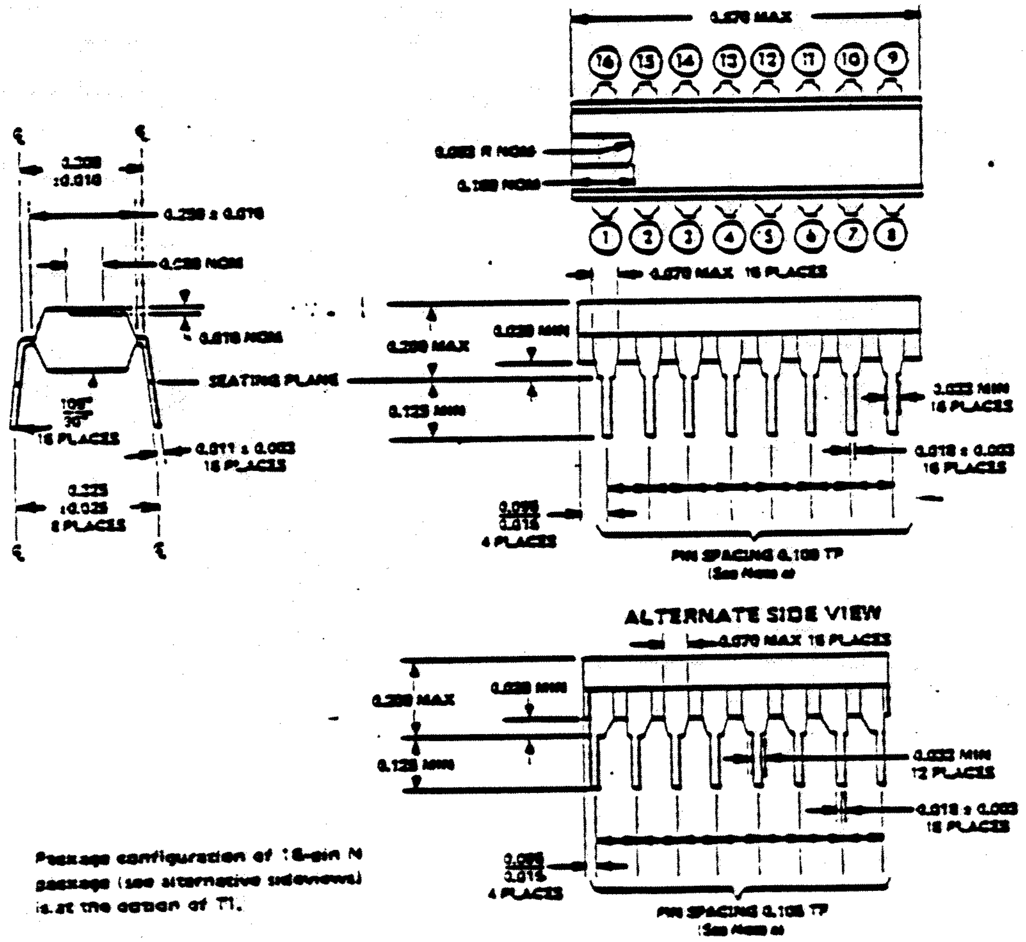
SN76489A / SN76494 SCHELOGIC

Figure 4.

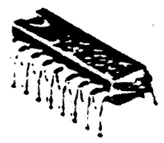
N package

This dual-in-line package consists of a circuit mounted on a 16-lead frame and encapsulated within an electrically nonconductive plastic compound. The compound will withstand soldering temperature with no deformation and circuit performance characteristics remain stable when operated in high-humidity conditions. The package is intended for insertion in mounting-hole rows on 0.300-inch centers. Once the leads are compressed and inserted, sufficient tension is provided to secure the package in the board during soldering. Leads require no additional cleaning or processing when used in soldered assembly.

16-PIN N PLASTIC DUAL-IN-LINE PACKAGE OUTLINE



Package configuration of 16-pin N package (see alternative sideviews) is at the option of TI.



NOTES: a. Each pin center line is located within 0.010 of its true longitudinal position.
b. All dimensions are in inches unless otherwise noted.

Texas Instruments

APPLICATIONS REPORT

by Ted Mahler Applications Engineer

A GUIDE TO USING THE TEXAS INSTRUMENTS SN76489A SOUND-GENERATOR

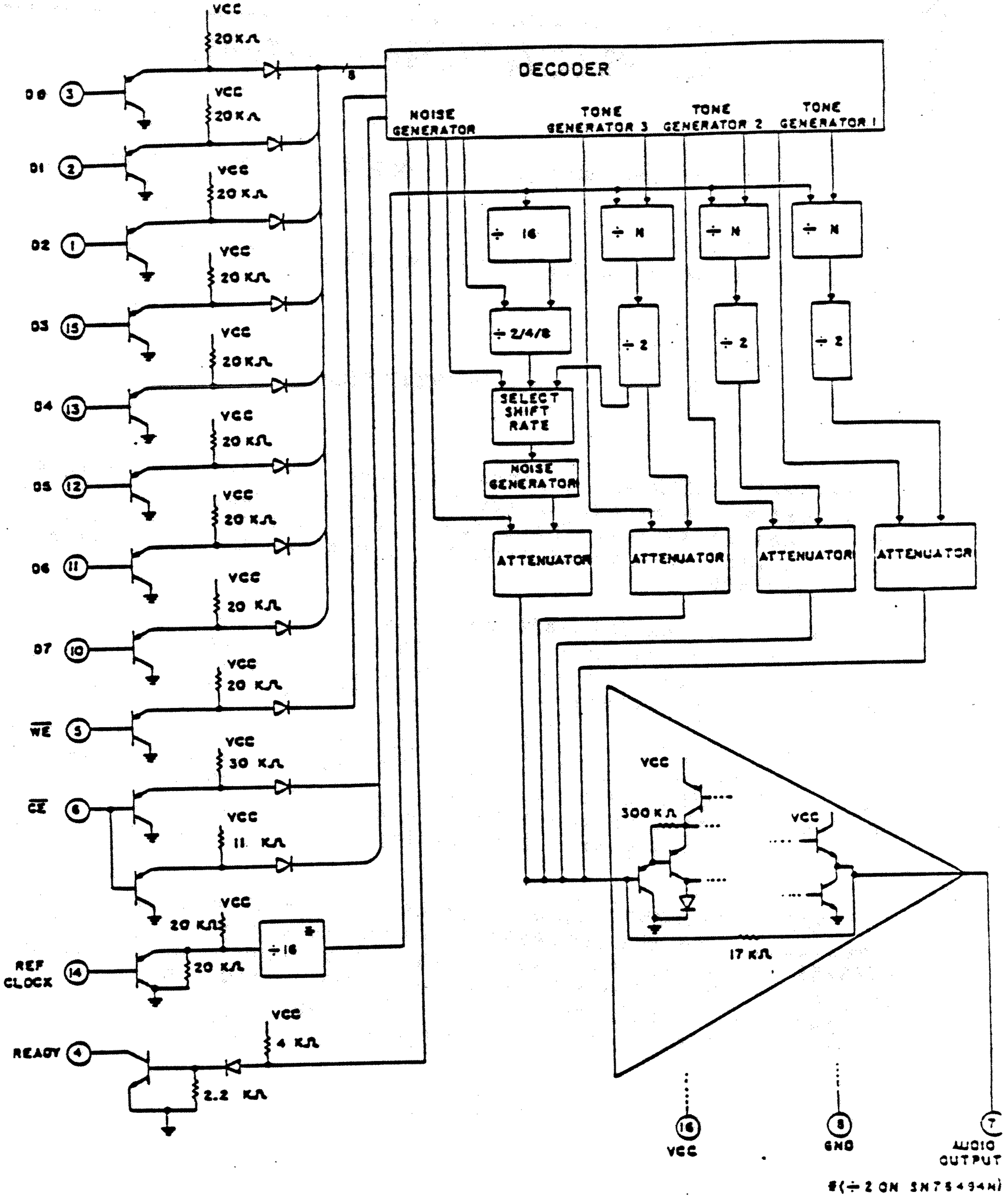
INTRODUCTION

The TEXAS INSTRUMENTS SN76489A sound generator is designed to provide a low cost means of adding sound generation capabilities to a microprocessor system. These sounds can include sound effects for video games, alarms for home or industrial use, or any application requiring audio feedback.

Internally, the SN76489A contains 3 programmable tone generators, each with its own programmable attenuator, and a noise source with its own attenuator. By using tone generators or combinations of tone generators and noise, an extremely wide variety of sounds can be easly created.

A complete description of the functions available in the SN76489A along with electrical characteristics is presented in the TEXAS INSTRUMENTS data sheet for this device. This application report contains examples of the methods needed to generate control bytes for the SN76489A, along with interfacing data and examples.

TEXAS INSTRUMENTS also manufactures the SN76489 which is pin compatable with the SN76489A with one exception. Audio out on the SN76489 is 100ma, which is capable of driving a small capacitively coupled speaker. The SN76489A has a 10ma output which will not drive a speaker but should be capacitively coupled into an amplifer. Software generation for both devices is identical.



SN76489A / SN76494 SCHELOGIC

The SCHELOGIC of the SN76489A illustrates both its logic arrangement and the actual components on each on the pins of this device. This information is presented to aid in understanding and interfacing the SN76489A.

EXAMPLE 1

I. The first example will demonstrate how to produce a 4167HZ tone with a clock of 2 MHZ.

A) First, with this equation find n. This number (n) will contain the frequency information for the SN76489A.

a) n = clock/(32(tone))
b) n = 2 MHZ/(32(4167HZ))
c) n ≃ 15

B) This number (n) is now converted to a 10 bit binary number.

Frequency bit#	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
	0	0	0	0	0	0	1	1	1	1

C) Now construct the 2 bytes neccessary to transfer this information to the SN76489A.

a) The format for for frequency generation is:

BYTE 1								BYTE 2							
*BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7	*BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
1	R0	R1	R2	F6	F7	F8	F9	0	X	F0	F1	F2	F3	F4	F5

* MOST SIGNIFICANT BIT (MSB)

where R0, R1, and R2, in this case, tell the chip which tone generator to use (table 1). For tone generator 1, R0 = 0, R1 = 0, and R2 = 0.

FREQUENCY REGISTER CODE			DEVICE
R0	R1	R2	
0	0	0	TONE 1
0	1	0	TONE 2
1	0	0	TONE 3

TABLE 1. FREQUENCY CONTROL CODES

b) Therefore the 2 bytes will be:

BYTE 1	1 0 0 0 1 1 1 1
BYTE 2	0 0 0 0 0 0 0 0

Notice that although the last byte is zero, it does need to be included so that the SN76489A internal registor does not contain information from previous frequency bytes.

EXAMPLE 2

This example will show how to program a 400HZ tone on tone generator 2 with a clock frequency of 4MHZ.

A) First find n;

a) n = clock/(32(tone))
b) n = 4Mz/(32(400HZ))
c) n = 312.5

Note that since n can only be an integer, you must change it to be either 312 or 313. This decesion depends on the particular sound you are trying to obtain. For this example, 312 will be used. By substituting this number back into the equation you ca find the actual frequency that will be produced.

a) n = clock/(32(tone))
b) tone = clock/(32(n))
c) tone = 4MHZ/(32(312))
d) tone = 400.6HZ

B) Now convert n to a 10 bit binary number.

Frequency bit#	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
	0	1	0	0	1	1	1	0	0	0

C) Using the frequency format;

* BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
1	R0	R1	R2	F6	F7	F8	F9

* BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0	X	F0	F1	F2	F3	F4	F5

* MSB

the two bytes needed will be:

BYTE 1 1 0 1 0 1 0 0 0

BYTE 2 0 0 0 1 0 0 1 1

EXAMPLE 4 ATTENUATION CONTROL

The third major function available in the SN76489A is attenuation. Each of the three tone generators and the noise source has its own attenuator. The procedure for controlling each of these is the same.

A) First select the device, within the SN76489A, which is to be attenuated. Each device and its associated code is :

ATTENUATION REGISTER CODE			DEVICE
R0	R1	R2	
0	0	1	TONE 1
0	1	1	TONE 2
1	0	1	TONE 3
1	1	1	NOISE

B) Next select the desired level of attenuation as shown in this table.

ATTENUATION	ATTENUATION CODE				ATTENUATION	ATTENUATION CODE			
	A0	A1	A2	A3		A0	A1	A2	A3
0 db	0	0	0	0	16 db	1	0	0	0
2 db	0	0	0	1	18 db	1	0	0	1
4 db	0	0	1	0	20 db	1	0	1	0
6 db	0	0	1	1	22 db	1	0	1	1
8 db	0	1	0	0	24 db	1	1	0	0
10 db	0	1	0	1	26 db	1	1	0	1
12 db	0	1	1	0	28 db	1	1	1	0
14 db	0	1	1	1	OFF	1	1	1	1

C) Now construct the attenuation byte using this format:

* BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
1	R0	R1	R2	A0	A1	A2	A3

* MSB

EXAMPLE 3 NOISE

White noise can easly be produced by selecting two options and using them to construct a one byte control word for the SN76489A.

A) The first option is the noise feedback control (FB). A FB bit o 0 selects periodic noise which sounds like a low frequency tone White noise, which sounds like a hiss, is generated by a FB bit of 1.

B) The noise source in the SN76489A is a shift register with an exclusive OR feedback. The rate at which it shifts is dependent on the clock and two noise frequency (NF) bits which make up the second option. A description of the two bits is given in th following table:

BITS		
NF0	NF1	
0	0	HIGHER PITCH (LESS COARSE HISS)
0	1	
1	0	LOWER PITCH (MORE COARSE)

C) The options are now placed in the proper format for a noise control byte.

* BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
1	R0	R1	R2	X	FB	NF0	NF1

* MSB

Bits 1, 2, and 3 tell the SN76489A that this is a noise byte and should be as follows, R0 = 1, R1 = 1, and R2 = 0. Therefore a noise byte to select white noise with the less coarse hissing sound will be:

1 1 1 0 0 1 0 0

The rate at which the noise clock shifts can also be controlled by the frequency of tone generator 3. To transfer control to tone 3 set both the NF0 and NF1 bits to 1. Now vary the noise generator, either periodic or white, by changing the frequecy of tone generator 3.

The SN76489A is interfaced through 3 control lines and 8 data lines. For the following definitions a high logic level is a minimum of 2 volts and a low is a maximum of .8 volts.

CHIP ENABLE ($\overline{CE}$) When low, this pin enables the SN76489A and drop the READY line to a low state. This pin must remain low until the completion of the data transfer. In most applications the CE pin is connected to address decoding logic.

WRITE ENABLE ($\overline{WE}$) When active, low, this pin signals the SN76489A that data is available on the data bus. As with C this pin must remain low during the data transfe this pin must remain low during the data transfe

READY When the SN76489A is ready to accept data this p will be high. It will drop to a low level when C goes low. From the falling edge of WE, the READY line will remain low for 32 clock cycles.

DATA LINES (D0-D7) These pins accept data for the SN76489A. Since t SN76489A requires 32 clock cycles for a data transfer, data must be available on these pins for that period of time.

INTERFACING EXAMPLES

CIRCUIT 1. TMS990/100M COMPUTER TO SN76489A INTERFACE

This interface illustrates the use of the READY line to synchronize the SN76489A and the TMS990/100M. With this circuit the CPU will halt, allowing the slower SN76489A to accept the data without necessitating any sort of software timing loops. Note that not all processors will function using this type of interface. A processor must be able to complete the write cycle (place data on the data bus and enable its WRITE ENABLE line) after its READY line is disabled. If the processor's architect is not designed this way or if the processor is not capable of being halted, an interface such as circuit 2 should be used.

Addressing used in CIRCUIT 1 will select the SN76489A when writing to any address from E000 to EFFF. This 4K block of memory can be reduced by further decoding of th address lines if necessary.

Therefore to attenuate Tone generator 3 by 4db the byte would be:

1 1 0 1 0 0 1 0

An 18db attenuation of the noise source would be:

1 1 1 1 1 0 0 1

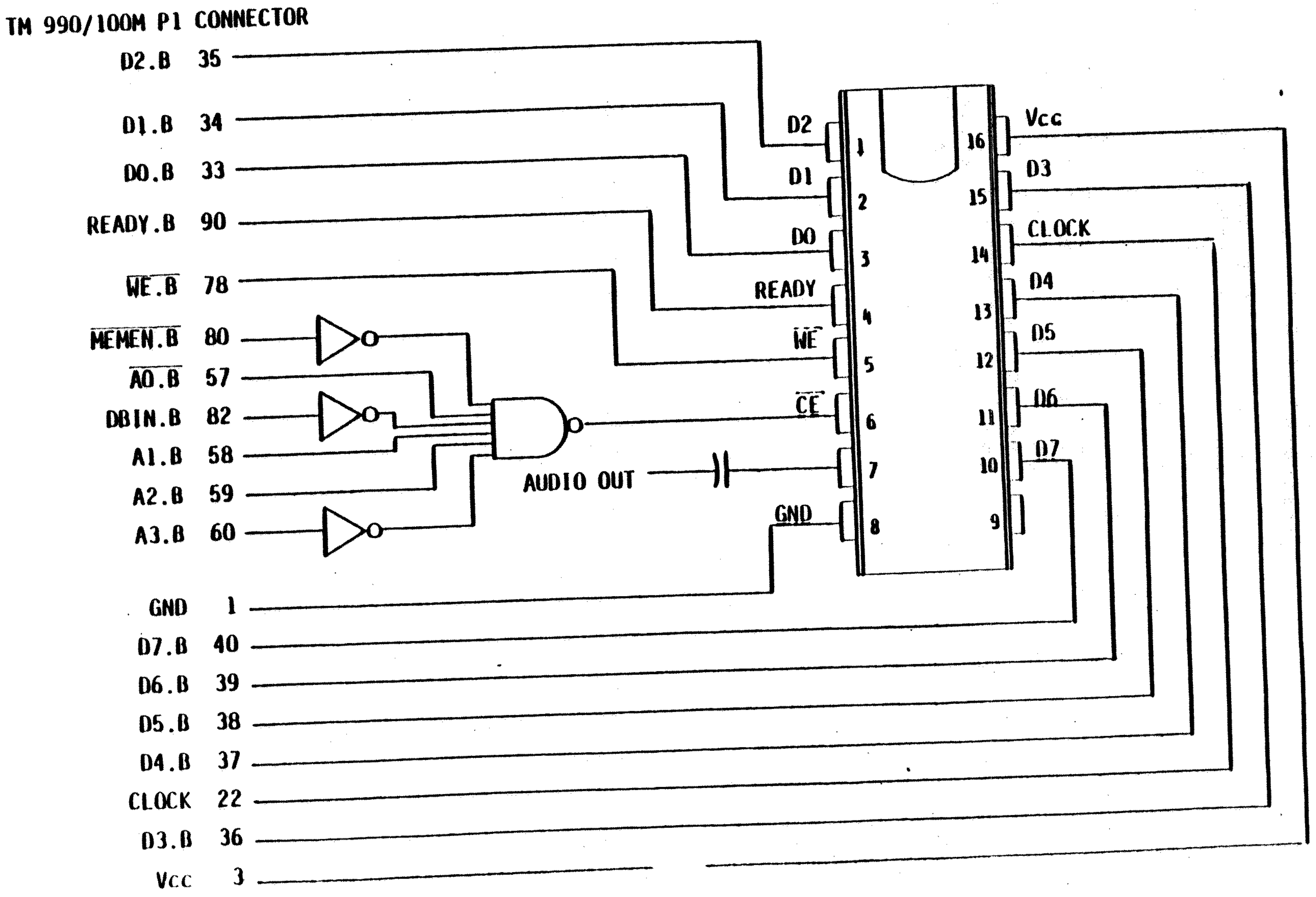
CIRCUIT 2. PARALLEL PORT INTERFACE

By using the READY line to control the duration of the WRITE ENABLE and CHIP ENABLE pulse, the SN76489A is assured that these pulses will be in their proper state for the proper amount of time. Two points must be considered when using this interface:

1. The data lines must maintain the data for the entire transfer period (32 clock cycles). This can be accomplished by latching the data lines.

2. Software using this interface must contain timing loops to avoid transfering a data byte to the SN76489A before it has had time to load the previo one. If a machine language program is being used, the number of machine cycles in the instructions can be counted and used to construct a 32 cycle lo If a higher level language is being used, generall timing loops for data transfers are not necessary.

TM 990/100M TO SN76489A INTERFACE



PROGRAMMING EXAMPLE

CLOCK CHIME

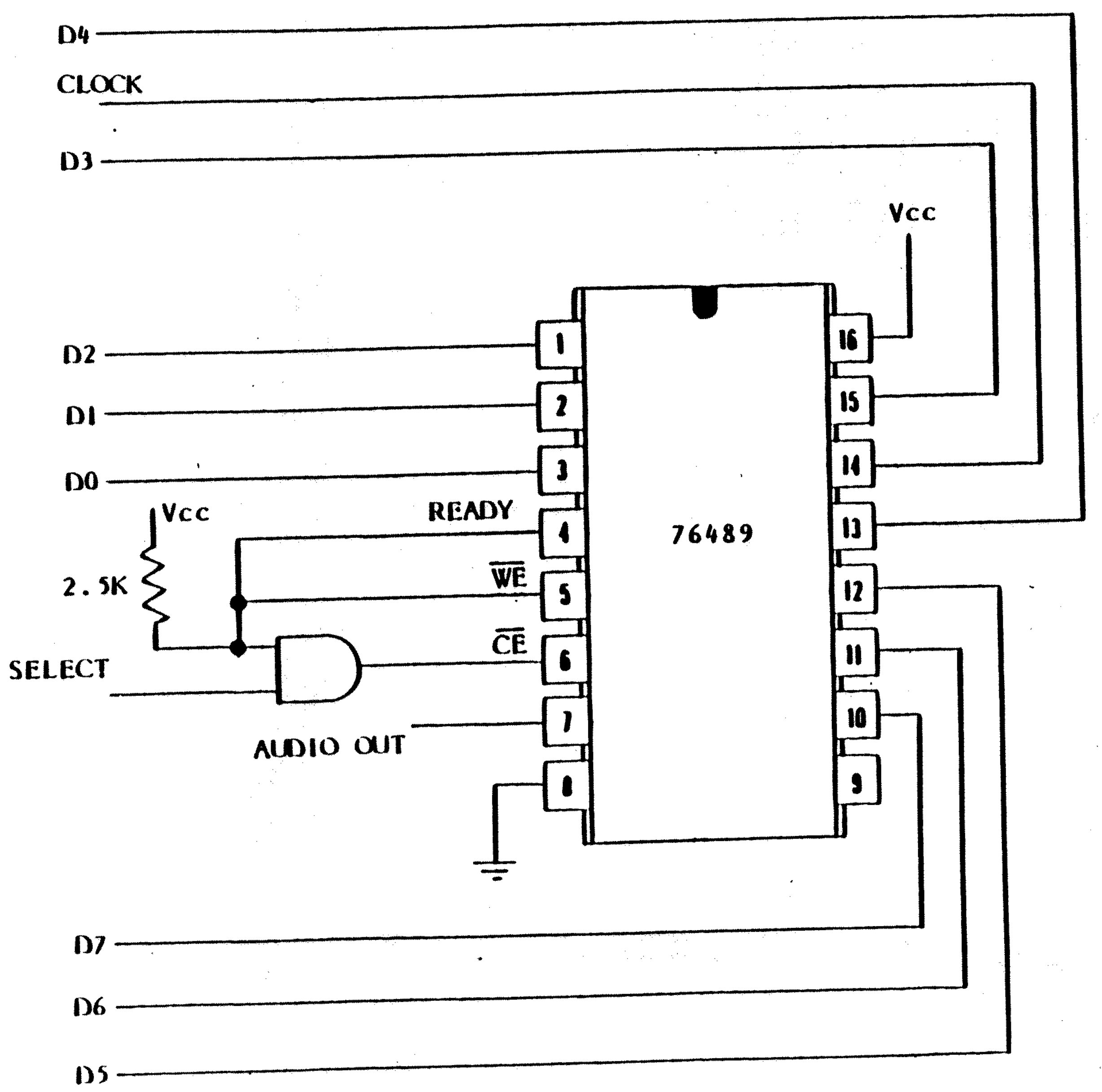
The clock chime sound can be generated by using two of the ton generators and their attenuators. Tone generator 1 is set to a frequenc of 679HZ at 0 dB down, while tone 2 is set to a frequency of 694 HZ at 24 dB down. While holding tone 2 steady, tone 1 attenuation in ramped u until completely off. A time delay is used to hold tone 1 for a perioc of time at each attenuation level. The two frequencies, being slightly different, beat to create the chime sound. The chime is repeated by ramping tone 1 attenuation again.

The following is a list of bytes which when presented to the SN76489A, with the suggested time intervals, will produce this sound. All frequencies were calculated using a 1MHZ clock.

HEX BYTE TO SN76489A	TIME INTERVAL FROM LAST BYTE	COMMENTS
9F	*	TURN OFF TONE 1
BF	*	TURN OFF TONE 2
DF	*	TURN OFF TONE 3
FF	*	TURN OFF NOISE
84		STE TONE 1 AT 679HZ
0A	*	
90		SET TONE 1 ATTENUATIC AT 0 dB DOWN.
A0	*	SET TONE 2 AT 694HZ
0A	*	
BB		SET TONE 2 ATTENUATIC AT 24 dB DOWN.
91	*	TONE 1 ATTENUATED 2dE
92	114 ms	" " " 4dE
93	"	" " " 6dE
94	"	" " " 8dE
95	"	" " " 10dE
96	"	" " " 12dE
97	"	" " " 14dE
.	.	
.	.	
.	.	
9F	"	TONE 1 OFF
BF	"	TONE 2 OFF

* This period of time does not affect the chime sound. It should be long enough to allow a data transfer from the controlling system to the SN76489A.

PARALLEL PORT INTERFACE



NOTES

1. The data lines must be latched so that the data remains on them for at least 32 clock cycles after the select line goes low.

2. The select pulse should be a negative going pulse a minimum of 150ns wide.

3. Audio out should be capacitively coupled to either an 8 OHM speaker (SN76489) or an audio amplifer (SN76489A).

4. Pin 9 must be left open.

Notice that a complete frequency sweep of tone three is done by incrementing the first byte of the two byte frequency word from C0 to CF while keeping the second byte at 00. After the last iteration of the first byte (CF) , the second byte is incremented by 1. Now the process repeats incrementing byte 1 from C0 to CF but this time holding byte 2 at 01. This process continues until byte 2 is at 0E. The noise attenuation can also be ramped down, during this sweep, from 0dB down to off (hex bytes F0 to FF). For the attenuation to reach maximum attenuation at the end of the sweep, it should be updated every other cycle of byte 1.

A jet sound can be produced using the same basic program. For this sound the frequency sweep of tone 3 is run from a low frequency to a high frequency then back to low frequency with the time interval between the frequency words increased.

EXPLOSIONS

Explosions are generated by setting the noise generator to white noise and ramping the noise attenuator from 0dB down (F0) to off (FF). Different time periods can be used between each attenuation update to simulate different types of explosions.

MISSILE

The sound of a missile or jet can be produced using the noise generator and tone generator number 3. Generation of the sound is done by clocking the noise generator from tone generator 3 and then sweepi the frequency of tone 3 from a high frequency to a low frequency. Du the sweep, the noise generator is attenuated to give the effect of distance to the sound.

HEX BYTE TO SN76489A	TIME INTERVAL FROM LAST BYTE	COMMENTS
9F	*	TURN OFF TONE 1
BF	*	" " " 2
DF	*	" " " 3
FF	*	" " NOISE
E7	*	CONTROL NOISE BY TONE 3
F0	*	TURN ON NOISE
C0	*	TONE 3 FREQUENCY (HIGH)
00		
C1	18 ms	TONE 3 FREQUENCY UPDATE
00	145 us	
C2	18 ms	TONE 3 FREQUENCY UPDATE
00	145 us	
C3	18 ms	TONE 3 FREQUENCY UPDATE
00	145 us	
.		
.		
.		
CF	18 ms	TONE 3 FREQUENCY UPDA
00	145 us	
F1	18 ms	NOISE ATTENUATION UPDATE
C0	18 ms	TONE 3 FREQUENCY UPDATE
01	145 us	
C1	18 ms	TONE 3 FREQUENCY UPDATE
01	145 us	
C2	18 ms	TONE 3 FREQUENCY UPDATE
01	145 us	
.		
.		
.		
CF	18 ms	LAST TONE 3 UPDATE WORD
0E	145 us	
FF		HIGHEST ATTENUATION LEVEL

* This period of time does not affect the sound. It should be long enough to allow the data transfer from the controlling system to the SN76489A.

BASIC Language program listings

To aid in sound development a higher level language can be used to control the SN76489A. Listings given here were used on a system where t sound chip was interfaced using a parallel port interface. The SN76489A will then appear as a memory location which can be written to using a F command (on some systems this is a POKE command). These programs, of course, will not be compatable with every system, but they should prese the basic method used to generate some given sounds.

```
10 REM ** BELL OR CHIME **
20 REM SYSTEM CLOCK FREQUENCY OF 2 MHZ
30 N=59392 : REM LOCATION OF SN76489A
40 FILL N,159 : REM TURN OFF TONE 1
50 FILL N,191 : REM TURN OFF TONE 2
60 FILL N,223 : REM TURN OFF TONE 3
70 FILL N,255 : REM TURN OFF NOISE
80 INPUT A$ : REM INPUT ANYTHING
90 FILL N,140 : TONE 1 AT 679 HZ
100 FILL N,5
110 FILL N,170 : REM TONE 2 AT 694 HZ
120 FILL N,5
130 FOR B=0 TO 11 : REM B=NUMBER OF BELLS
140 FOR I=145 TO 159 : REM LOOP TO GENERATE ATTENUATION STEPS
150 FILL N,I : FILL N,(I+32) : REM OUTPUT ATTENUATION TO 76489
160 FOR D=0 TO 75 : NEXT D : REM DELAY LOOP
170 NEXT I
180 NEXT B
190 PRINT "END OF SOUND"
```

```
10 REM ** BASIC BIRD SOUND **
20 REM SYSTEM CLOCK FREQUENCY OF 2 MHZ
30 N=59392 : REM LOCATION OF SN76489A
40 FILL N,159 : REM TURN OFF TONE 1
50 FILL N,191 : REM TURN OFF TONE 2
60 FILL N,223 : REM TURN OFF TONE 3
70 FILL N,255 : REM TURN OFF NOISE
80 INPUT A$ : S=0 : REM  INPUT ANYTHING
90 Z=INT(10 *(RND(0)) : REM RANDOM CHIRP LENGTH
100 FILL N,144 : REM SET TONE ATTENUATION
110 FOR I=0 TO 15 : REM START CHIRP LOOP
120 FILL N,(128+I) : REM STEP FREQUENCY FROM 3906 HZ TO 2016 HZ
130 FILL N,1
140 FOR D=0 TO Z : NEXT D : REM DELAY BY RANDOM AMOUNT
150 NEXT I
160 S=S+Z : REM COUNT TO STOP PROGRAM
170 IF S)200 THEN 190 : REM BRANCH TO END
180 GOTO 90
190 FILL N,159 : REM TURN OFF TONE 1
```

```
10 REM ** MISSILE SOUND **
20 REM SYSTEM CLOCK FREQUENCY OF 2MHZ
30 N=59392 :  REM LOCATION OF SN76489A
40 FILL N,159 : REM TURN OFF TONE 1
50 FILL N,191 : REM TURN OFF TONE 2
60 FILL N,223 : REM TURN OFF TONE 3
70 FILL N,255 : REM TURN OFF NOISE
80 INPUT A$ : REM INPUT ANYTHING
90 FILL N,231 : REM CONTROL NOISE BY TONE 3
100 FILL N,240 : REM SET NOISE ATTENUATION
110 FOR B=0 TO 15 : REM LOOP TO GENERATE SECOND BYTE
120 FOR A=192 TO 207 : REM LOOP TO GENERATE FIRST BYTE
130 FILL N,A : FILL N,B : REM OUTPUT BYTE TO 76489
140 NEXT A
150 FILL N,(240+B)  : REM ATTENUATE NOISE
160 NEXT B
170 GOTO 40
```

```
10 REM ** BOMB DROP AND EXPLOSION **
20 REM SYSTEM CLOCK FREQUENCY OF 2MHZ
30 N=59392 : REM LOCATION OF SN76489A
40 FILL N,159 : REM TURN OFF TONE 1
50 FILL N,191 : REM TURN OFF TONE 2
60 FILL N,223 : REM TURN OFF TONE 3
70 FILL N,255 : REM TURN OFF NOISE
80 INPUT A$ : REM INPUT ANYTHING
90 FILL N,144 : REM ATTENUATE TONE 1 0 DB
100 FOR J=5 TO 17 : REM LOOP TO GENERATE BYTE 2 FOR SWEEP
110 FOR I=128 TO 143 : REM LOOP TO GENERATE BYTE 1 FOR SWEEP
120 FILL N,I : FILL N,J : REM OUTPUT BYTE TO SN76489A
130 FOR D=0 TO 10 : NEXT D : REM DELAY LOOP
140 NEXT I
150 NEXT J
160 FILL N,159 : REM TURN OFF TONE 1
170 FILL N,228 : REM SET NOISE TO HIGH PITCH WHITE NOISE
180 FOR I=240 TO 255 : REM LOOP TO GENERATE ATTENUATION BYTES
190 FILL N,I : REM OUTPUT BYTE TO SN76489A
200 FOR D=0 TO 75 : NEXT D : REM DELAY LOOP
210 NEXT I
```

PLEASE NOTE:

TEXAS INSTRUMENTS cannot assume responsibility for any circuits or systems shown, or represent that they are free from patent infringement.

E.2 REFERENCES

Apple Computer Inc. Apple Reference Manual, Order Number A2LØØØ1A (Ø3Ø-ØØØ4-Ø1).

Barden, William, Jr. The Z-8Ø Microcomputer Handbook, 1978.

Bryant, Dorothy. The Kin of Ata. A fantasy novel about a society who believes in and uses the dreams occurring during their sleep in the daily life of the community.

Nichols, Joseph C. and Others. Z-8Ø Microprocessor Programming and Interfacing - Book One, 1979.

Microsoft Corp. Microsoft SoftCard System for Apple II - Installation and Operation Manual, 1982. Catalog No. 23Ø4, Part No. 23FØ4B, Document No. 88Ø5-223-Ø1.

Texas Instruments Inc. TMS9918A/TMS9928A/TMS9929A Video Display Processors Data Manual, November 1982. Document No. MPØ1ØA.

Texas Instruments Inc. Advanced Circuits - SN76489AN Sound Generator Data Sheet, November, 1981.

Texas Instruments Inc. (Mahler, Ted). Applications Report - A Guide To Using The Texas Instruments SN76489A Sound Generator.

E.5 USER REFERENCE CARD

The Apple Development System Menu

1 — LOAD FROB MEMORY FROM APPLE DISK

2 — MOVE FROB MEMORY TO APPLE DISK

3 — MOVE FROM CARTRIDGE TO APPLE DISK

4 — MOVE FROM CARTRIDGE TO FROB MEMORY

5 — DISPLAY FROB CONTROL PARAMETERS

6 — SET FROB CONTROL PARAMETERS

7 — RESET AND RUN PROG IN FROB MEMORY

8 — RESET AND RUN PROG IN COLECO ROM

9 — EXIT

Frob Interface Unit LED Indicators

LED	Interpretation
1- RST	Reset in progress.
2- MEM	Memory space is selected.
3- HOLD	ColecoVision bus is being held.
4- I/O	I/O space is selected.
5- WP (Write Protect)	This LED is illuminated when Bit 4 of the Frob Control Register is set. This prevents writes to the Frob memory from the Apple or ColecoVision.
6- CART (Cartrige Select)	This LED is illuminated when Bit 3 of the Frob Control Register is set. This indicates that the ColecoVision ROM is accessible at locations 8000 - FFFF.
Power	ColecoVision power switch is on.

The Lazer Monitor List Commands

<<adrs>>	Displays the contents of memory location <<adrs>>.
<<adrs1>>.<<adrs2>>	Displays the contents of the memory locations in the range <<adrs1>> . . . <<adrs2>>.

Lazer Monitor List Commands (continued)

<<adrs>> : <<data>>	Sets sequential memory locations to the values specified by <<data>>.
<<adrs>>L	Disassembles and lists 20 instructions at the specified address.
<<adrs>>G	Executes the Z-80 subroutine at the specified address.
<<a1>>.<<a2>>G	Sets a breakpoint at address <<a1>> and begins execution at address <<a2>>.
<<a1>> < <<a2>>.<<a3>>G	Sets two breakpoints at addresses <<a1>> and <<a2>> then executes Z-80 code beginning at address <<a3>>.
C	Copys data from cartridge slot to Frob RAM.
<<dest>> < <<a1>>.<<a2>>M	Moves data from address <<a1>> through <<a2>> to <<dest>>. <<dest>> is always in the same memory space as <<a1>> and <<a2>>.
<<dest>> < <<a1>>.<<a2>>MV	Moves the data from the currently selected memory space (memory or VRAM) to the other memory space.
G	Initiates DEBUG.FRB program.
I	Selects I/O port addressing.
L	Loads binary file from disk to Frob memory.
M	Halts Coleco, entering MONITOR mode.
N	Selects normal memory addressing.
R	Resets the ColecoVision.
Q	Quits LZRMON program, returning to CP/M.
S	Saves Frob program memory to disk.
V	Selects VRAM addressing.
X	Exits to the main LZRMON menu.