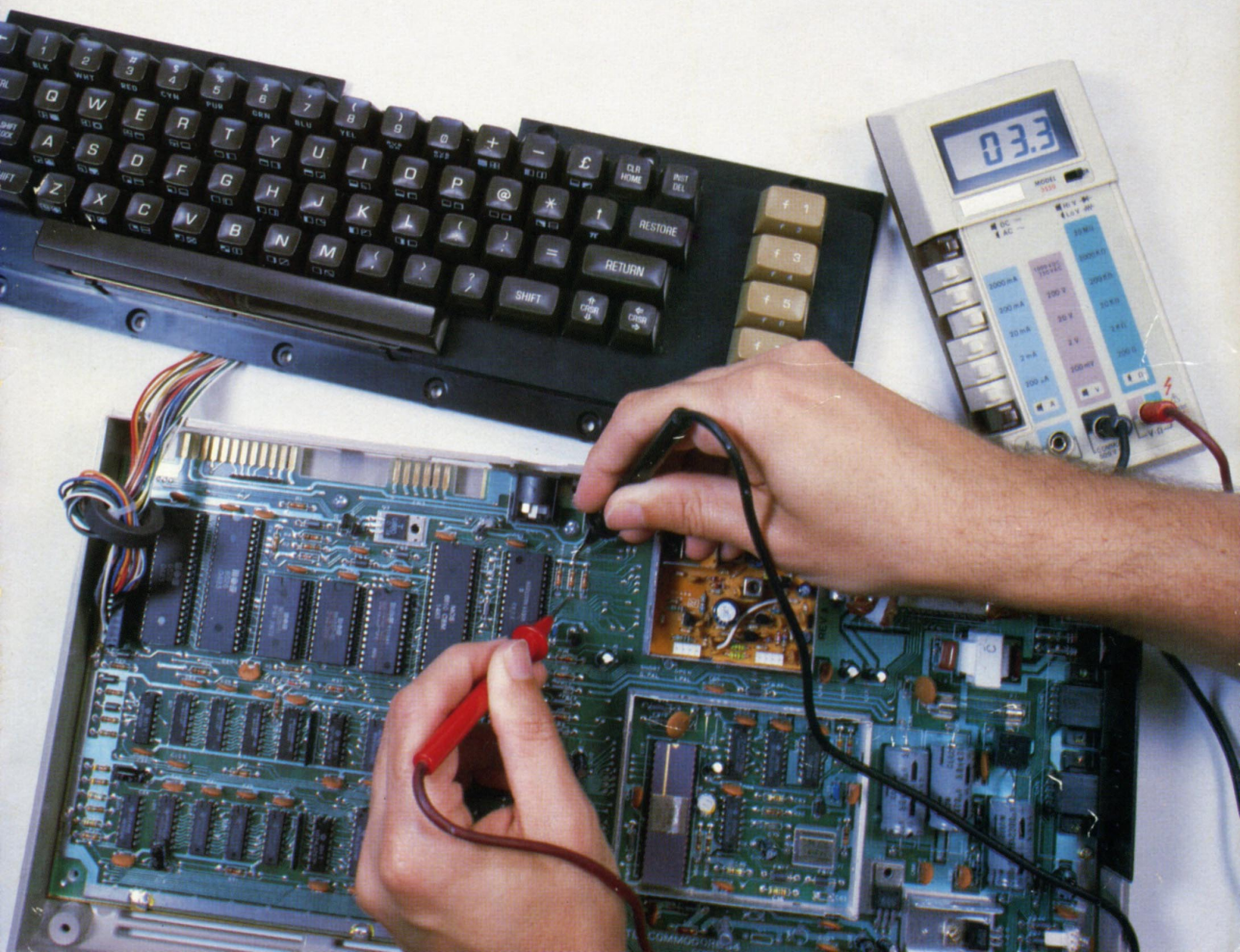


# TROUBLESHOOTING & REPAIRING YOUR COMMODORE 64™

ART MARGOLIS





TROUBLESHOOTING & REPAIRING  
YOUR  
COMMODORE 64™



TROUBLESHOOTING & REPAIRING  
YOUR  
COMMODORE 64<sup>TM</sup>

ART MARGOLIS



TAB BOOKS Inc.

Blue Ridge Summit, PA 17214

FIRST EDITION

FOURTH PRINTING

Printed in the United States of America

Reproduction or publication of the content in any manner, without express permission of the publisher, is prohibited. No liability is assumed with respect to the use of the information herein.

Copyright © 1985 by TAB BOOKS Inc.

Library of Congress Cataloging in Publication Data

Margolis, Art.

Troubleshooting and repairing your Commodore 64.

On t.p. the registered trademark symbol "TM" is superscript following "64" in the title.

Includes index.

1. Microcomputers—Maintenance and repair.

2. Commodore 64 (Computer) I. Title.

TK7887.M374 1985 621.3819'584 85-4640

ISBN 0-8306-0889-3

ISBN 0-8306-1889-9 (pbk.)

Questions regarding the content of this book should be addressed to:

Reader Inquiry Branch

Editorial Department

TAB BOOKS Inc.

P.O. Box 40

Blue Ridge Summit, PA 17214

# Contents

---

<b>Test Point Charts</b>	<b>ix</b>
<b>Introduction</b>	<b>x</b>
<b>1 Interpreting the Symptoms</b>	<b>1</b>
Common Symptoms 1	
Dead Computer—Garbage Display—Empty Display Block—No Color—No Video, Sound OK—No Sound—External Device Troubles	
Diagnostic Programming 5	
PEEK and POKE—Diagnostic Programs	
Inside the 64 8	
Complex Interface Adapter—Microprocessor—Video Interface Chip—Sound Interface Device	
Troubleshooting Charts 15	
<b>2 Disassembly</b>	<b>16</b>
Removing the Top 16	
Removing the Circuit Board 19	
Disassembly as a Cure 24	
Visual Inspection 26	
Ground Plane Short—Board Defects	
Cleaning 28	
Static Electricity 28	
What Is It?—Wrist Strap—Additional Problems	
<b>3 Chip Location Guide</b>	<b>31</b>
Drawing a Guide 31	
Chip Survey 34	

Microprocessor—Complex Interface Adapters—Video Interface Chip—Sound Interface Device—  
Random Access Memory—BASIC ROM—Kernal ROM—Character ROM—Color RAM—Other Board  
Landmarks

Sockets 39

Using the Location Guide 40

## **4 Chip Changing Techniques 42**

TTL, DTL, and RTL 42

Three-State TTL 46

MOS Chips 48

DIP Package 50

Socketed Chips 52

Chip Removal—Chip Replacement

Soldered-In Chips 55

Desoldering—Resoldering

## **5 LSI Chips 57**

6510 Microprocessor 57

6526 Complex Interface Adapter 61

6567 Video Interface Chip 65

6581 Sound Interface Device 69

## **6 Random Access Memory 73**

Static RAM 74

Dynamic RAM 77

Memory Refresh—Memory Layout—Operation—Timing—Refresh Timing

## **7 Read Only Memory 90**

Block Diagram 93

Character ROM 96

Kernal and BASIC ROM 97

BASIC ROM—Kernal ROM—Jump Tables

Programmable Logic Array 102

82S100 Pinout—PLA Inputs

Service Charts 107

## **8 Other Integrated Circuits 109**

7406A Hex Inverter 109

74LS08 Quad 2-Input AND Gate 111

74LS74 Dual D Flip-Flop 114

74LS193 Up/Down Counter 116

74LS139 2-4 Decoder 118

74LS257 Quad 2-Input Multiplexer 121

74LS258 Quad 2-Input Multiplexer 122

74LS373 Octal D-Type Latch 126

4066 Quad Bilateral Switch 130

556 Dual Timer 131

Other Chip-like Components 136

## **9 System Block Diagram 138**

Block Diagram 138

MPU and Dynamic RAM 140

MPU and ROM 143

MPU and CIA Interface 143

MPU and VIC II 147

MPU and SID 151

<b>10</b>	<b>Servicing Logic Gates</b>	<b>153</b>
	Decimal and Binary 153	
	Hexadecimal 156	
	PEEK and POKE 158	
	Gates 158	
	YES Gate—NOT Gate—AND Gate—OR Gate—XOR Gate—NOR and XNOR Gates	
	Gate Testing Technique 172	
<b>11</b>	<b>Servicing Digital Registers</b>	<b>174</b>
	Flip-Flops 174	
	74LS74 D Flip-Flop—74LS373 D Latch—74LS193 Up/Down Counter—556 Dual Timer	
	Computing Registers 183	
	Shifting—Clearing—Complementing—Increment, Decrement, and Jump—ANDing and ORing	
<b>12</b>	<b>6510 Microprocessor</b>	<b>192</b>
	Addressing 192	
	Stack 194	
	Arithmetic Logic Unit 195	
	Shifting—Logic Manipulations	
	Accumulator 200	
	Instruction Set 202	
	Instruction Byte—Fetch and Execute	
	Index Register 206	
	Flag Register 207	
	Interrupts 209	
	Vector Addresses 211	
	Other Pins 211	
	Testing 211	
<b>13</b>	<b>Memory Maps</b>	<b>214</b>
	6510 I/O Port 214	
	Default Map 217	
	I/O Signals 217	
	Memory Decks 218	
	Reading and Writing to the Decks 221	
	Other Possible Maps 222	
	PEEKing and POKEing the Map 223	
<b>14</b>	<b>Clock</b>	<b>224</b>
	Sine Wave to Square Wave 224	
	Working Frequency 225	
	6510 Timing Control 228	
	Other Timing Signals 230	
	Address Signals—Data Timing—Reading and Writing to Peripherals	
	Testing the Clock 233	
<b>15</b>	<b>Address, Data, and Control Buses</b>	<b>236</b>
	Address Bus 236	
	Address Assignments—Address Bus Connections	
	Data Bus 239	
	Control Lines 241	
	R/*W Line—*IRQ Line—Reset Line— $\phi 0$ and $\phi 2$ Signals	
	Testing the Bus Lines 242	
	Using the Tester—PEEK and POKE Tests—Control Line Tests	
<b>16</b>	<b>Complex Interface Adapters</b>	<b>253</b>
	Addressing and Controlling 253	



Internal Data Transfer	255	
Timing	255	
Write Timing—Read Timing		
I/O Ports	257	
*PC and *Flag Pins	259	
Timers	259	
Real Time Clock	259	
Interrupt Control Register	262	
Serial Data Register	265	
Operation	265	
CIA1—CIA2		
Testing	268	
<b>17 Video Interface Chip</b>		<b>269</b>
Operation	269	
Special Address Lines	271	
Data Bus Connections	271	
Modes	275	
Displaying Characters—Character Fetch—Character Color—Character Modes—Bit Map Modes		
Sprites	280	
Other VIC Features	283	
Raster Register—Interrupt Register		
Light Pen	285	
Video Output	286	
Testing	289	
<b>18 Sound Interface Device</b>		<b>290</b>
Pinout	290	
Special Inputs	291	
Operation	291	
Registers	295	
First Voice Registers—Voice Control Register—Envelope Generator Register—Other Voices—Filter Registers—Mode/Volume Register—Writing and Reading—Pot Registers—Osc 3/Random Register—Env 3 Register		
Timing	303	
Testing	303	
<b>19 Inputs and Outputs</b>		<b>305</b>
Control Ports	306	
Expansion Ports	306	
Audio-Video Ports	309	
Serial I/O Ports	309	
Cassette I/O Slot	311	
User Connector	312	
<b>20 Power Supply</b>		<b>313</b>
Trouble Locations	313	
Fuse Considerations	316	
Source Checking	316	
The Main Line	317	
<b>Appendix Master Schematic</b>		<b>319</b>
<b>Index</b>		<b>349</b>

### Test Point Charts

Figure #	Chip #	Chip Job
6-4	2114	Color RAM
6-14	4164	Dynamic RAM
7-3	2332	Character ROM
7-8	2364	Basic ROM
		Kernal ROM
7-12	82S100	PLA
8-1	7406	Hex Inverter
8-3	74LS08	Quad 2-Input AND
8-6	74LS74	Dual D Flip Flop
8-9	4044	Phase/Freq Detector
8-11	74LS193	Up/Down Counter
8-14	74LS139	Decoder
8-16a,b,c	74LS257	Multiplexers
8-17	74LS258	Multiplexer
8-21	74LS373	Octal Latch
8-25	4066	Quad Switch
8-27	556	Dual Timer
8-28	74LS629	Dual Voltage Osc
12-16	6510	Microprocessor
16-8	6526	CIA 1
16-9	6526	CIA 2
17-15	6567	VIC II
18-1	6581	SID

# Introduction

---

When I first turn on my Commodore 64, there is a pause between the time the switch is flicked and the blue on blue picture comes on with the READY sign and the blinking cursor. It's little more than a second, but the pause makes me nervous. The thought always flashes through my mind: come on, fellow, come on.

Unfortunately, with the millions of 64s in daily use, even though the 64s are quite reliable, they are still only assembled pieces of electronic gear. On any given day there are many of them that do not come on. If yours is one of the unlucky, what can you do? Obviously, the reason why your little favorite is not coming on will have to be discovered.

That is what this book is all about. Even though you might be a top programmer or an expert computer operator, the troubleshooting and repairing of your 64 can only be accomplished by approaching the machine from a different point of view. It will be easier to gain the technician's viewpoint if you are a programmer or operator, but the approach is a new one and requires a different dimension of thought.

The analogy has often been given that the programmer can be compared to the driver of a car. Just as the auto driver does not need to know how the car's engine burns gasoline, the programmer does not need to know how the computer's circuits are consuming electricity. When the auto breaks down, the driver takes it to a mechanic for the fix. In the same way, a programmer can take an ailing 64 to a computer tech for the repair. You could think of this book as the manual the tech refers to as he makes the necessary tests to get the machine back into operation.

However, this book is much more than that. Besides the needed specific information on voltages and parts, this is also a training course to give Commodore 64 owners the information that, added to programming skills, will give you the missing dimension that will enable you to gain mastery over your computer.

In order to be able to figure out and repair a circuit failure in your 64, you must first of all be able to picture in your mind how the circuit is working. There are also tried and true electronic service

pathways that have been developed over the years. By following these methods and using your puzzle solving abilities, you will be able to locate defective components or connections. Once the trouble is pinpointed, which is the most trying part of a repair job, then the rest of the chore is either replacing the part or repairing the connection.

The starting point of any repair is to carefully analyze the symptoms of trouble. The computer not coming on is only one symptom. There are a number of others. In Chapter 1, all the common symptoms are discussed. Each symptom points to a circuit area that could possibly contain the source of the trouble. In each circuit, there are primary suspects that should be examined first. However, in order to get to the circuits you must be able to take the 64 apart. It is not difficult, and Chapter 2 shows you how. It's all up to knowing where the screws are. Dismantling the machine must be performed correctly and carefully to avoid accidents. A wrong move could cause additional troubles that can really complicate things.

Chapter 3 contains the most used piece of service information—the Chip Location Guide. This is the Printboard layout of the 32 chips in the computer. It lets you relate all the information you learn to the location of any chip you might want to examine on the board. This Guide will be used on every repair job and will be constantly referred to during the course of the troubleshooting. It is indeed valuable.

Chapter 4 is the directive that provides the mechanical and electronic techniques that must be used if and when chips have to be replaced. The tools and thinking that goes into the ticklish job of changing an IC is reviewed. Changing a chip is not at all like changing a vacuum tube and similar but much more exacting than changing a transistor. These first four chapters, amazingly, will teach you how to repair quickly at least 50 percent, and possibly more, of possible breakdowns.

Chapters 5 through 8 introduce you with a bit more intimacy to the chips in your machine. Starting in Chapter 6 you'll find the first of the 21 Test Point charts that make it easy to quickly check out every chip in the computer. Each chart is the top

view of a chip showing the exact pinout. The name of each pin is given and, where practical, a sketch of the chip's insides. Then arrows point the direction of the signal flow. Attached to each arrow is the actual reading you should receive if you probe the test point with a logic probe or vom.

The readings were all made on my Commodore 64 with my logic probe and vom. The readings were taken when the computer was first turned on with the READY sign on and the cursor blinking. The idea is when you read your chips the state of each test point should match up with my charts. If the chip reads correctly, then that chip is deemed ok. Should one or more readings on a chip be incorrect, that is a symptom of trouble and bears further investigation.

Actually, what you are doing is checking the input and output status of the chips. The inputs and outputs are clearly shown by the arrows. If all signals are being input properly but not outputting correctly, chances are the chip's internal circuits are not passing the signals. That would indicate a bad chip. Should an input signal be incorrect, odds are the chip is ok but the circuit feeding the chip is not getting the signal to the pin. That circuit or bus line could have a short or an open condition. The 21 charts are your entree to getting repairs underway quickly and locating troubles fast.

You'll find the charts in Chapters 6, 7, 8, 12, 16, 17 and 18. A list of the Test Point charts is found immediately after the Table of Contents.

Chapters 5 through 20, besides being a servicing manual, are also, with the exception of Chapters 10 and 11, a technical reference manual written on a technician's level. There are discussions of each chip with considerable detail and even some timing diagrams for the 6510 MPU, RAM and the CIAs. You can really gain mastery over your machine by gradually absorbing the material. You'll find your programming skills will improve immeasurably as you gradually realize what is actually happening to the 1's and 0's as they flash around the digital circuits.

The book ends up with a Master Schematic of your machine. The schematic is needed during a difficult repair after the circuit area containing the

trouble has been located and the wiring details are needed to pinpoint the prime suspects. The schematic contains all the part numbers. These same part numbers are found printed clearly on the print board. For example, a U7 is found printed on the board next to the 6510 MPU. The schematic reads U7 as the part number for the MPU. All the capacitors, resistors, etc. are identified in the same way. With the Chip Location Guide, the Test Point charts, the theoretical discussions of the circuits, and the Master Schematic, you should be able to cover all the information needs you'll ever require for troubleshooting and repair of your computer.

Chapters 10 and 11 act as a short course in logic gates and digital registers with specific reference to the gates and registers in your computer. The machine's numbering methods are included too. The BASIC language in your 64 uses ordinary decimal as its number system. The 64 changes the decimal to binary and then after processing changes the binary back to decimal once again. With the PEEK and POKE commands, you can signal trace circuits that the MPU is able to address. The ability to convert from decimal to binary and then back should be one of the tools in your servicing repertoire.

From a money and convenience point of view, it really pays off to be able to at least make the easy repairs on your computer. The easy ones are at least half of the total. While your 64 is still in warranty, there is no problem. Should your machine give up the ghost during the first 90 days, all you have to do is take it back to the dealer. He will give you a new one. Commodore is very accommodating that way. However, once the warranty period is over, the complication begins. If the machine fails, you must send it back to the factory repair facility. This usually takes some weeks and it is returned with a healthy repair bill. If you are able to do the troubleshooting and repair yourself, the savings in time and money is quite worthwhile. If you happen to get one repair in the life of your machine from this book, it will more than pay for the price of the book.

I'd like to thank my wife Lea for running interference and fielding interruptions while I wrote the book. I would also like to thank my son-in-law Michael Gorzeck for taking the black and white photos. I hope your computing is uninterrupted, but should your 64 act up, perhaps this book will be directly responsible for getting it back on line again.



# 1. Interpreting the Symptoms

**I**N THE OLD DAYS OF COMPUTERS, WAY BACK as far as 1976, before micros hit the homes, users for the most part inputted and outputted through a teletype machine. A teletype cost about \$1000 and was touted as being able to provide almost every function of a computer system. These were considered the keyboard input, the printer output, and paper tape punching and reading. Notice that a TV display is not even mentioned.

There were, of course, TV displays around called CRTs. They were able to show numbers and letters easily. They could produce dot matrix patterns in good fashion. However, graphic pictures could only be produced with equipment that carried astronomical price tags. The CRTs were a peripheral accessory that was not used as much as the teletype. Things have changed.

The TV display is the most used output device for the computer. Teletype machines are almost unknown to the home computer or small business user. Alphanumeric characters, dot patterns, and complicated graphics are routine uses for the computer owner. The TV display is thought of as part

of the computer. You wouldn't think of using your Commodore 64 without the TV display.

From a troubleshooting and repair point of view, the TV display is a diagnostic tool. When the computer is okay, the TV picture shines in bright colors and the audio emanates correct sounds. If trouble strikes the computer, symptoms of the condition often appear dramatically on the TV screen or are heard from the TV speaker. By careful eyeball analysis, the troubleshooter is often able to pinpoint the seat of the trouble, or at least know where to start.

## COMMON SYMPTOMS

The following collection of general symptoms can help you determine which section of your computer has failed. Narrowing down the location of the problem is the first step in troubleshooting. Let's give these troubles some thought.

### Dead Computer

The most common problem that you'll en-

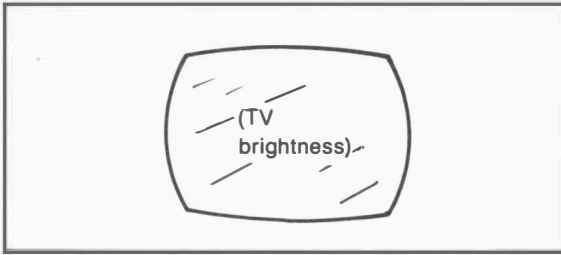


Fig. 1-1. When your TV brightness comes on ok but your 64 won't display anything, your computer is playing dead.

counter is a blank TV screen. You have your 64 connected to the TV ok, and the TV is showing light probably with snow (see Fig. 1-1). When you

flip the off-on switch, nothing happens. The LED pilot light could shine red or not show light at all.

Of course, you must be sure that the computer is plugged in properly at both the wall socket and that the power cable shown in Fig. 1-2 is plugged into the side of the case. If everything is plugged in, feel the casing of the ac adapter for your 64. Is it getting warmish? If it is, then it is probably ok. If it is not, it could be the trouble. Trying a new one will prove the point. If a new one works then it was bad. Should the new not work either, then the old one is probably fine.

Once you complete the above quick checks and the computer is still dead, then turn to Chapter 20

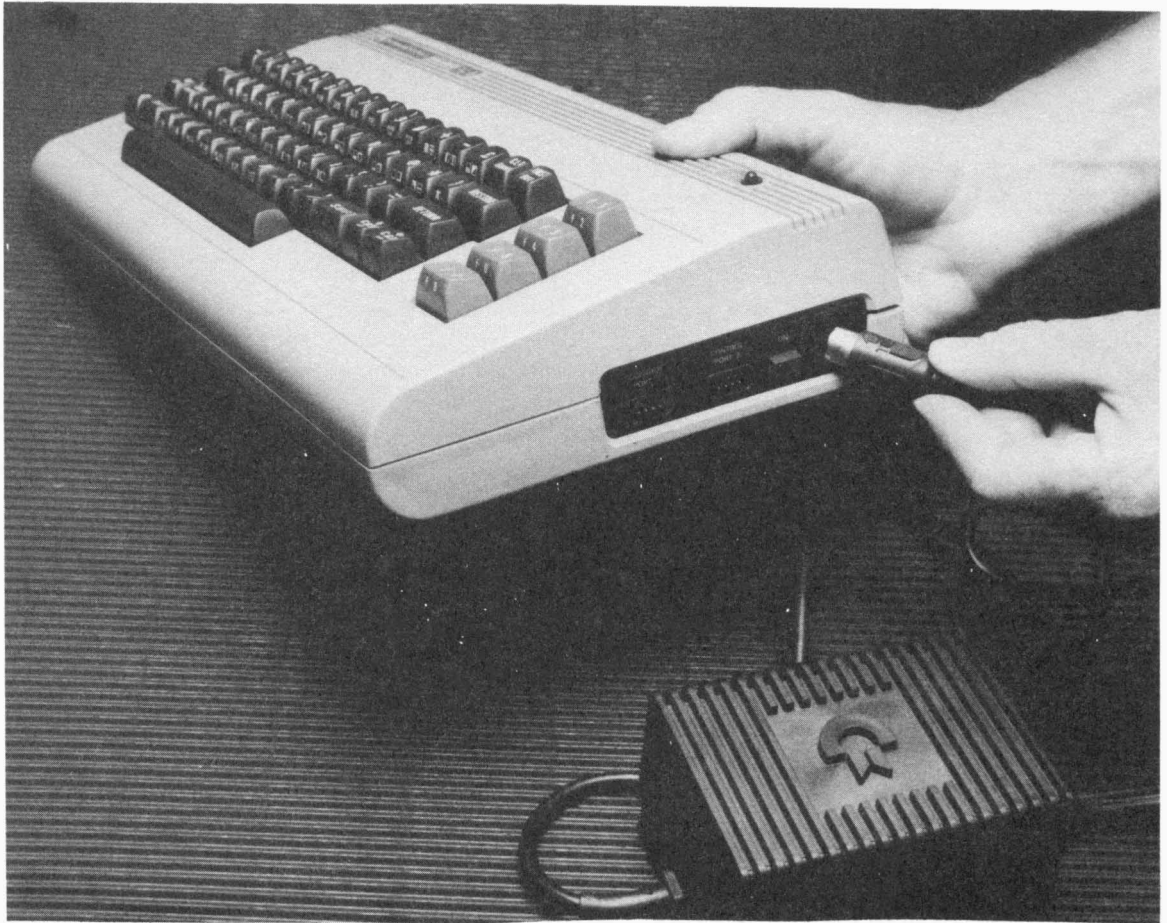


Fig. 1-2. The first obvious step to take when your 64 acts dead is to make sure it is plugged in at the wall socket and at the side of the case.

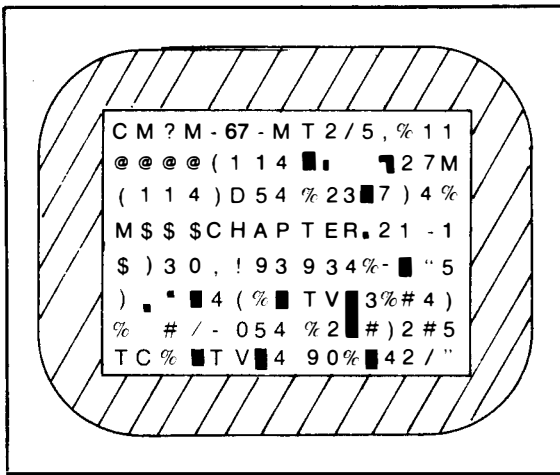


Fig. 1-3. When the TV fills with GARBAGE instead of the READY sign and the blinking cursor, there is some trouble in the digital circuits of the computer.

for the next troubleshooting steps. You'll need to take the 64 apart and make some ac and dc voltage reading with a vom.

## Garbage Display

The next most common symptom is called "garbage." The extreme case of garbage is shown on the screen in Fig. 1-3. When you turn on your 64, the TV display immediately fills up with

numbers, letters, symbols, white spaces and black spaces. It looks like a cartoon character's collection of swear words. Your 64 should have come on with the READY sign, but you are looking at garbage instead.

The TV display is normally able to show the four items (Fig. 1-4). The border should come on in light blue. Second, a dark blue display block should appear contained in the border. The third item in the picture consists of numbers, letters, and symbols. The lettering ends in the word READY. The characters that the 64 displays upon turn on are also in light blue. The display is also capable of changing the color of the characters. The colors do not appear automatically upon turn-on. You must type in the colors you want.

The reason why garbage appears on the screen, instead of the two-tone blue, sign-on message, is because the microprocessor is running wild. Normally, the processor performs its duties under the careful control of the 64's operating system, which is contained in the ROM chips. When the operating system for some reason loses control, the processor simply goes mad, spewing addresses, data and control signals without any rhyme or reason. The result is, the TV display fills up with meaningless characters, numbers, symbols, and spaces.

The garbage can be caused by failure in almost

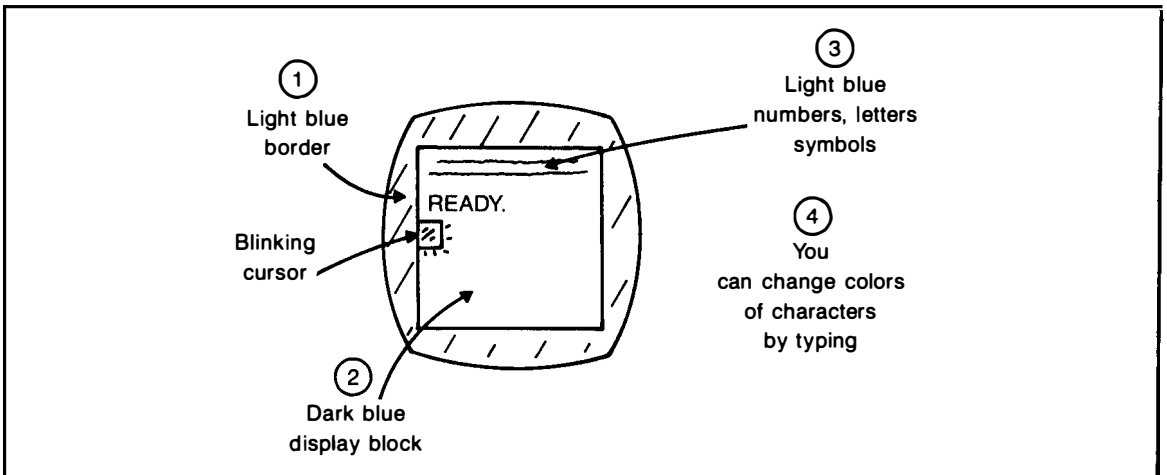


Fig. 1- and the ability to have the character colors changed easily.



any of the digital circuit components or chips. The approach to first locating the general circuit area of the trouble and then pinpointing the actual failed component or connection requires the following. You must learn how the digital circuits are processing data from a hardware point of view at a technician level of understanding. Then you can intelligently make voltage, logic probe, and scope readings to seek out defects. Chapters 5 through 16 contain the information that will help you discover what is causing a garbage condition.

### Empty Display Block

A related trouble to garbage happens when the border and display block come on ok, but none of characters are appearing. You can strike the keyboard to your heart's content, but nothing happens. The block remains empty. Refer to Fig. 1-5.

Here again the same circuits are suspect. The condition could possibly be located anywhere in the digital circuits. It is the troubleshooter's job to take test readings and from the results of the readings deduce what circuit area is in trouble. Understanding Chapter 5 through Chapter 16 should clear up the maze of chips and print board copper traces.

These chapters contain troubleshooting techniques and schematic drawings with the voltage readings and logic states that are normally present on the test nodes when the READY sign appears. If you run some tests and one or more of the test results do not match up with what is supposed to be at the connections, you have found a clue that could lead to the fault.

### No Color

The 64 is a color computer. It has circuits that produce colors that are placed into the TV display. When the computer is putting out a correct display of characters and symbols but will not show any colors, it indicates that the trouble is in the color circuits.

The color signal originates in the clock circuit. Chapter 14 covers the system clock. The color is output by the Video Interface Chip, or VIC. Chapter 17 reveals the VIC workings and its associated cir-

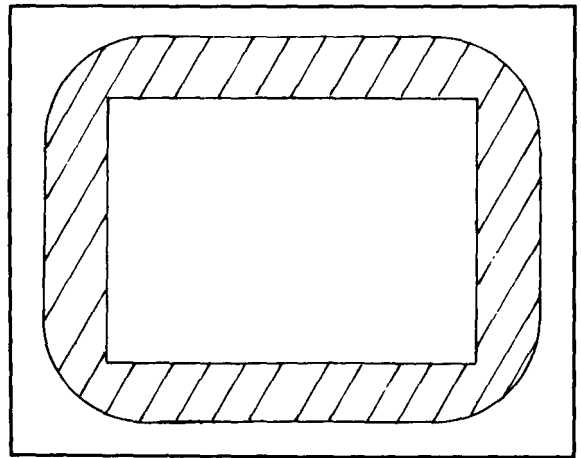


Fig. 1-5. A variation of garbage is an empty display block. The border and display block appear but no characters appear. Striking the keyboard has no apparent effect.

uits that could be involved in the "no color" symptom.

### No Video, Sound Ok

This symptom resembles the "dead set" power supply troubles except for one thing, the sound is ok. This means the digital circuits are working. The trouble in this case is in the video output circuits. The VIC and its output transistors are the probable cause of this malfunction. They are covered in Chapter 17.

### No Sound

When everything is working except the audio, the trouble is located in the area of the sound circuits. Chapter 18 covers the 64's sound circuit inside and around the Sound Interface Device, or SID.

### External Device Troubles

The 64 is able to receive inputs from the keyboard, the joysticks, the paddles, a cassette, the disk system, and many other devices. It is able to output to a printer, the cassette, the disk drive, and other external units. When these units start acting up the trouble could be either in the device itself

or the I/O circuitry that connects the device to the computer.

If you have device problems, the first step is to try another known good device. If the new device works, then the old one was defective. Should the symptoms remain with the new unit, then the problem is being caused by the computer. Chapter 19 covers all the interface connections on the back and side of the 64 that devices can plug into. Chapter 16 covers the I/O chips that the plugs connect to inside the 64. This type of trouble is usual-

ly confined to these circuits.

The keyboard is connected internally to the I/O chips in the 64. Figure 1-6 shows the keyboard wires disappearing into the 64's innards. When the keyboard does not work properly, you can test it as if it is part of the computer and not an external device. The keyboard connections are discussed in Chapter 16.

## DIAGNOSTIC PROGRAMMING

Odds are you have already spent a lot of time work-

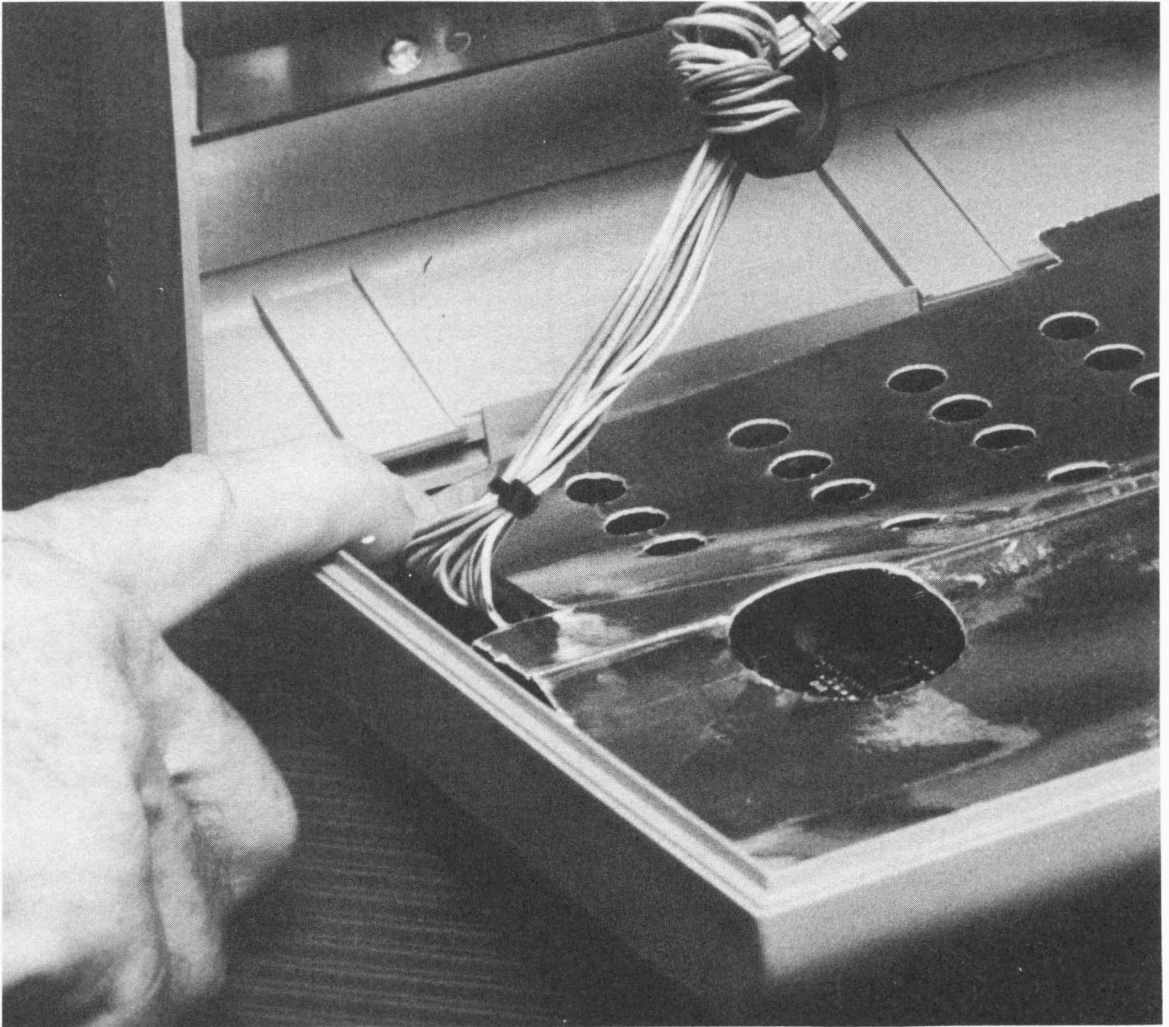


Fig. 1-6. The keyboard is connected inside the case at a special plug on the print board. The keyboard is disconnected only as a troubleshooting or repair measure.

ing on BASIC programs on the 64. BASIC programming can be a very important tool to isolate circuit sections where troubles might lie. I use BASIC programming as a signal injection technique. The function PEEK and the statement POKE are the mechanisms that perform the signal injections. They are powerful troubleshooting test instruments. You can write your own diagnostic programs or buy commercial software to perform tests.

## PEEK and POKE

Peek allows you to read the contents of any of the thousands of locations in the 64's memory map. POKE permits you to load a byte of data into all

of the memory map addresses except the read-only locations. This gives you tests whereby you can run data back and forth from the microprocessor to all locations of the memory map.

The PEEK and POKE tests work in the immediate mode or contained in a program. It should be obvious, however, that diagnostic software tests have their limitations. If the computer is dead or the processor is out, you cannot use the test functions or statements. They simply won't work. On the other hand, when the computer comes on normally and displays READY and the blinking cursor, you can try the software testing. Chances are good it will provide you with some valuable service

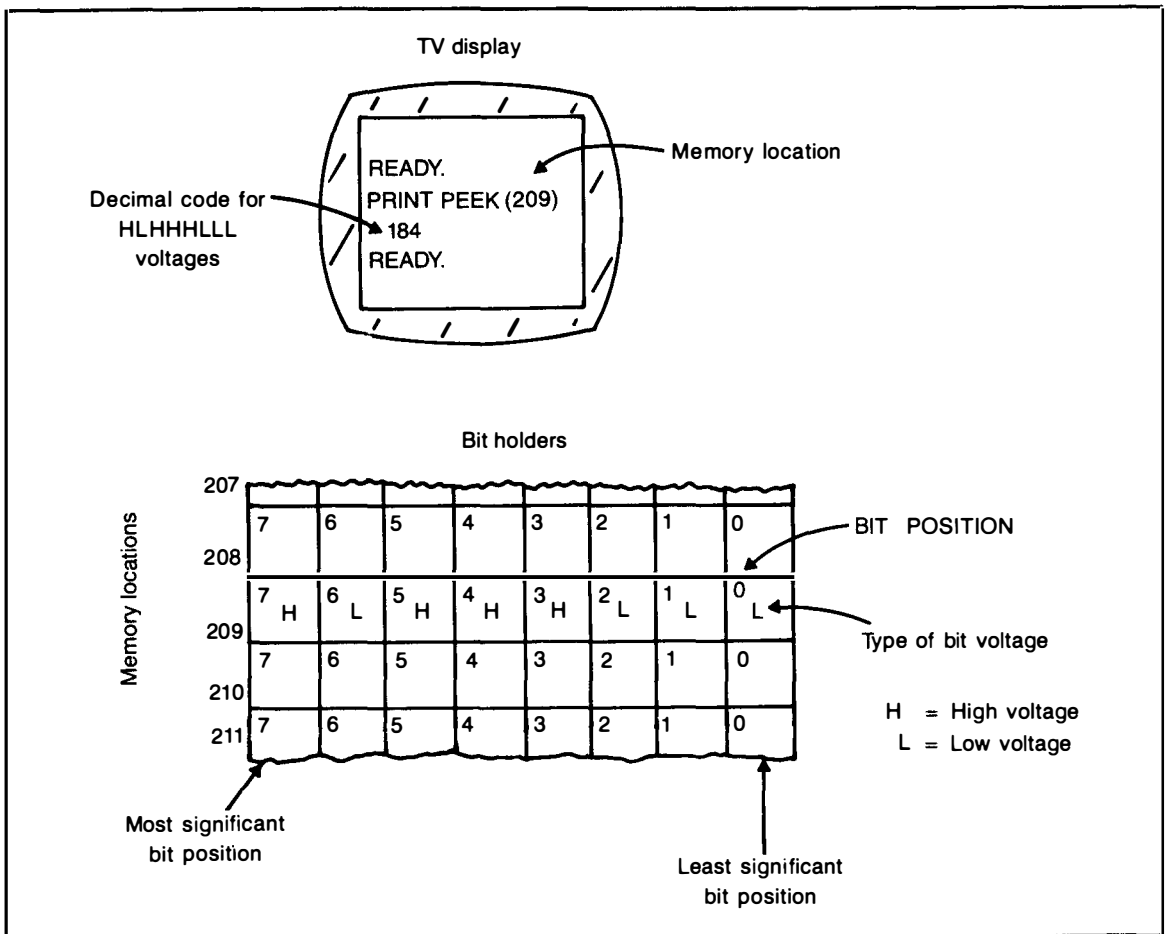


Fig. 1-7. When the technician sees the PEEK return decimal number 184, he knows that location 209 contains the voltages HLHHLLL.

information to guide you to the ultimate repair.

The PEEK and POKE used for troubleshooting requires that you have a clear idea of the relationship between decimal numbers and the set of bits they represent. When you PEEK a memory location, a decimal number is returned to you and is printed on the screen. The decimal number is the code for the set of eight bits contained in the register you have just read.

For instance, suppose you tell the 64 to PRINT PEEK (209). The 64 returns the decimal number 184. The 184 is the decimal code for the binary bit collection 10111000. Figure 10-7 illustrates this idea. The bits could also be described as HLHHLLL. The programmer usually thinks of the bits in terms of 1s and 0s while the technician finds the Hs and Ls more convenient. Anyway, the PEEK function has permitted you to read the contents of location 209. In certain cases, learning the bit contents is a piece of valuable service information.

At other times in a repair process, it might become necessary to write a set of bits to a location. For example, if you want to quick check the color RAM chip, you could write to the location that changes the color of the border. If you enter POKE 53280,8 the border of the TV display should change to orange. Try it. If it does, then the color RAM chip seems to be working ok. When the POKE produces no effect or the wrong effect, the chip could be in trouble. At any rate, the 53280 is the decimal address of the color RAM register. The 8 is a set of bits, 00001000 or LLLLHLLL. The color RAM uses the four lowest bits, HLLL, to change the border color.

PEEK and POKE can be used throughout the entire memory map, which contains 64K of dynamic RAM and additional memory locations for static RAM, ROM, and I/O. The servicer must view the locations as bit holders and the decimal numbers as code for the bits. There will be a lot more detail on this subject in Chapter 10 and Chapter 11.

## Diagnostic Programs

Besides being able to read or write to individual addresses with PEEK or POKE in the immediate

mode, you can write programs that perform a whole battery of tests and check out large portions of the memory map. One way is to place PEEK and POKE statements in loops. A POKE statement in a loop can make the 64 write to a large group of memory locations. A PEEK function in a loop could have the 64 read many memory locations and print the results in decimal on the screen. Another way that PEEK and POKE can be used is to test memory locations and the bus lines to the locations. You could first POKE some data to a location. Then you would PEEK the location to see if the data ever arrived. This could all be contained in one program. There are examples of this technique in Chapter 13 and Chapter 15.

In addition to writing your own diagnostic programs as you need them, there is software available that will perform some limited jobs. One piece I've seen in the software stores is a program called 64 DOCTOR. It is manufactured by Computer Software Associates, 50 Teed Dr., Randolph, MA 02368.

It is useful and will normally operate when the 64 comes on ok. The program comes on a disk and also on cassette tape. I only saw the disk version.

It takes a few minutes for the disk to load the program into the 64. Then a menu gives a number of options. There is one test that checks out the disk system and the 64's internal RAM. Next there are tests for the keyboard, a printer (if it is attached), a cassette (if it is attached), and joysticks. Another test provides patterns for the color TV or monitor. The tests are similar to the ones you would use to set up the colors and convergence of any color TV. The DOCTOR also gives the SID chip a music lesson.

The program is interesting and uses the 64's graphic capabilities to full advantage. You'll see sprites that look like a TV, printer, etc. running about the screen as well as all sorts of other patterns. The program was selling for about \$30 at the two places I visited.

While these large diagnostic programs can be interesting and occasionally useful during troubleshooting, they can be especially important before programming. If you are going to be

spending days on end programming a heavy project, it is a good idea to exercise the 64, everyday before work. If the machine exercises ok then it is safe to program on. It is very frustrating to spend many hours on a large number of program lines and then find them trapped in the machine.

## INSIDE THE 64

As you look over the 64 circuit board in Fig. 1-8, the main landmarks are the large chips. A closer look shows the data and address buses coursing

over the board. Figure 1-9 shows in a very basic manner how these various chips interact.

## Complex Interface Adapter

The input ports are on the extreme left side of the board in Fig. 1-10. There you'll find two 40 pin chips. They are both numbered 6526 and are called CIAs for Complex Interface Adapters. The leftmost one is connected to a plug that goes through a bundle of wires to the keyboard. It is the keyboard's port of entry. It also connects through the print

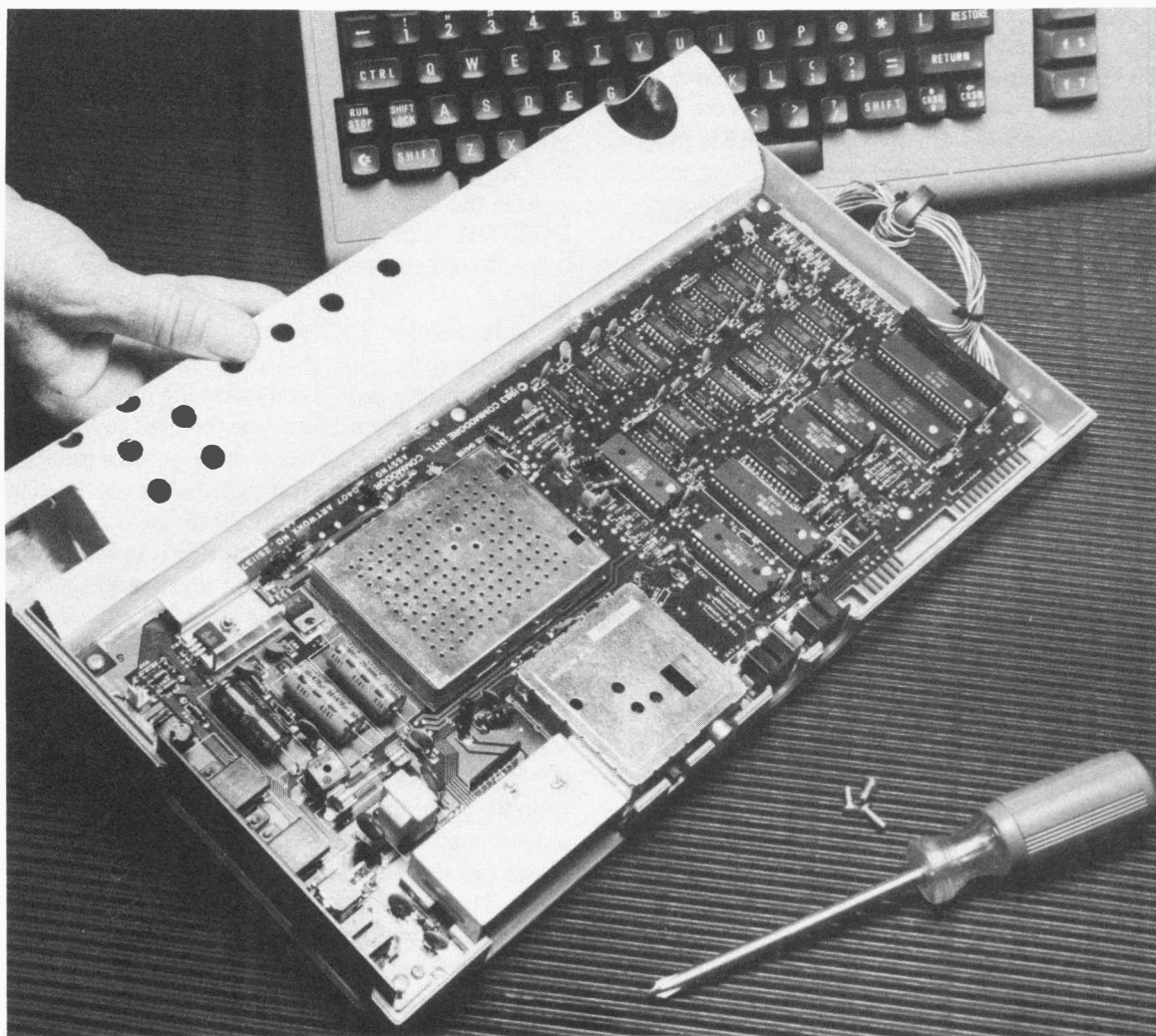


Fig. 1-8. At first glance the print board of your 64 can look like a jumbled mass of components.

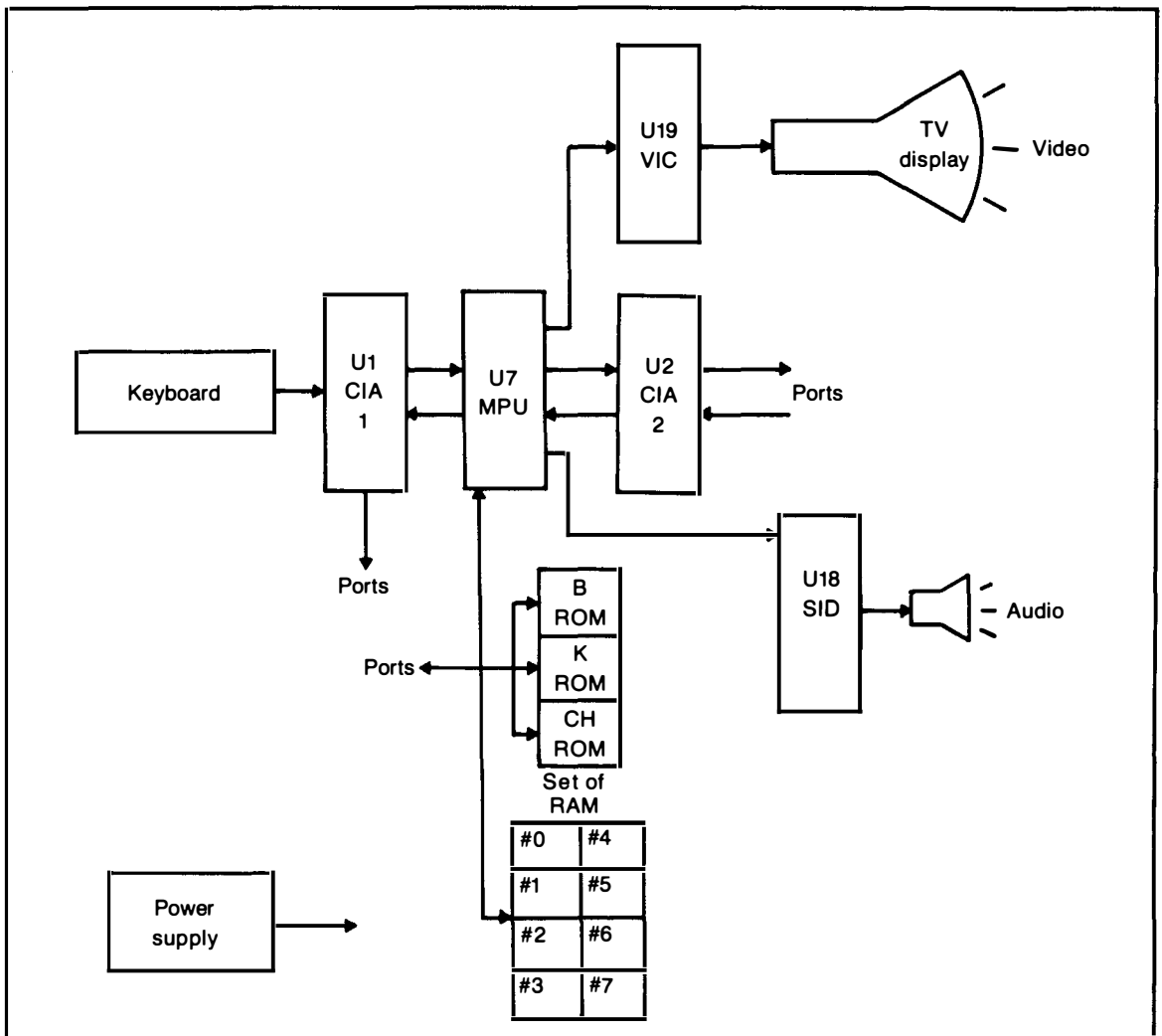


Fig. 1-9. This block diagram shows how the chips in the 64 interact.

board to the joystick input plugs. When a key is struck or a joystick is moved the signal is transferred to the CIA and CIA places the signal onto a group of eight parallel copper traces called the data bus. The data bus transports the signal to the microprocessor further down the board.

The other CIA is connected to the nearby user port and the serial port. The CIA performs like the keyboard CIA on the user and serial ports. It will receive the port inputs. In addition, it is able to output to these ports. The keyboard and joysticks, in

contrast, use their CIA as an input only device. This CIA also is able to put the signal out on the data bus. It receives data from the processor on the bus.

## Microprocessor

The processor is a 6510. It sits in the center of the board. It is the originator of the data bus. Besides connecting up to the CIAs via the data bus, it also hooks up to a number of other chips, as seen in Fig. 1-11. The data bus is sent to eight 4164 RAM chips, three ROMs, a 2114 RAM, the VIC,

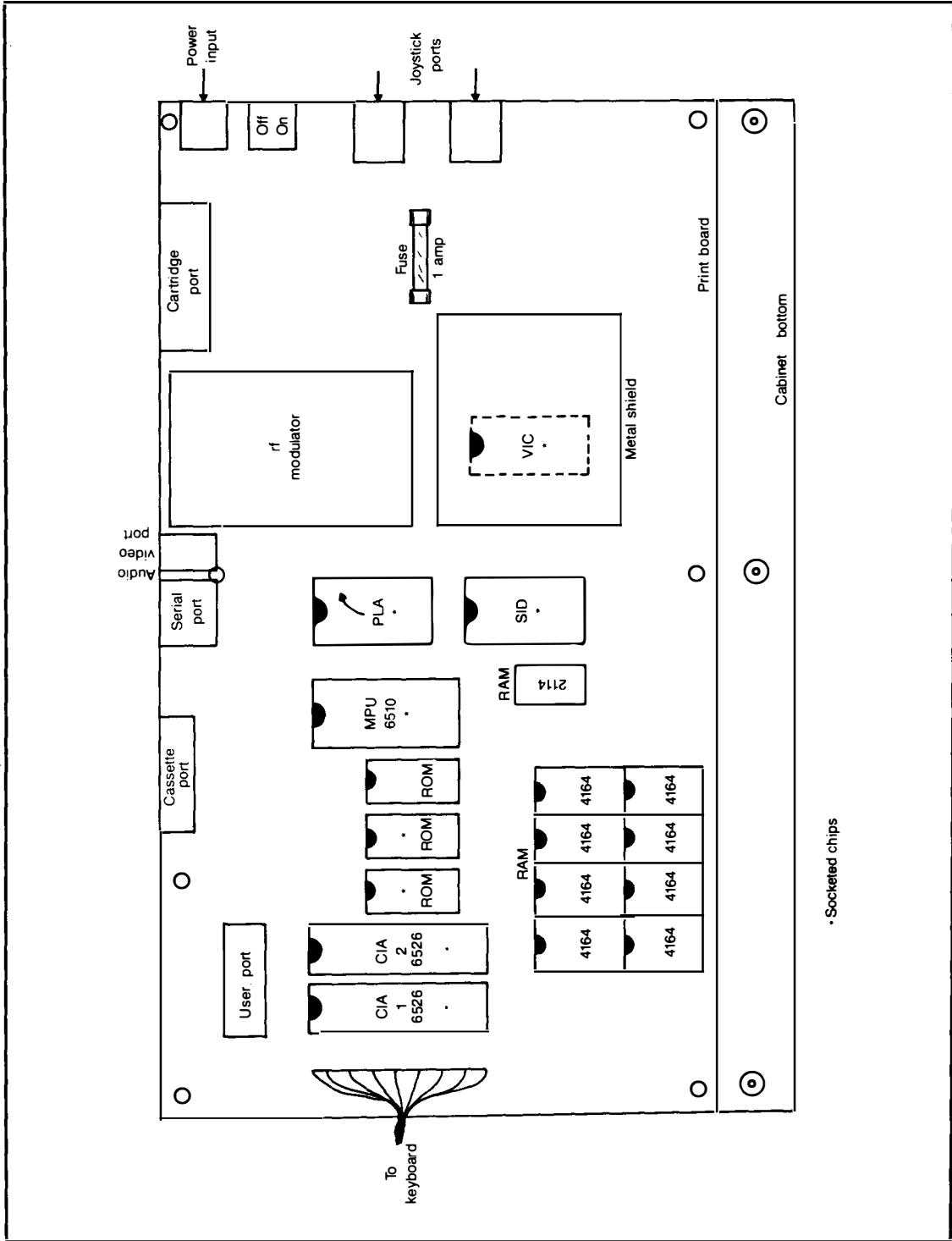


Fig. 1-10. A lot of troubleshooting confusion disappears as you become familiar with your 64's print board landmarks.

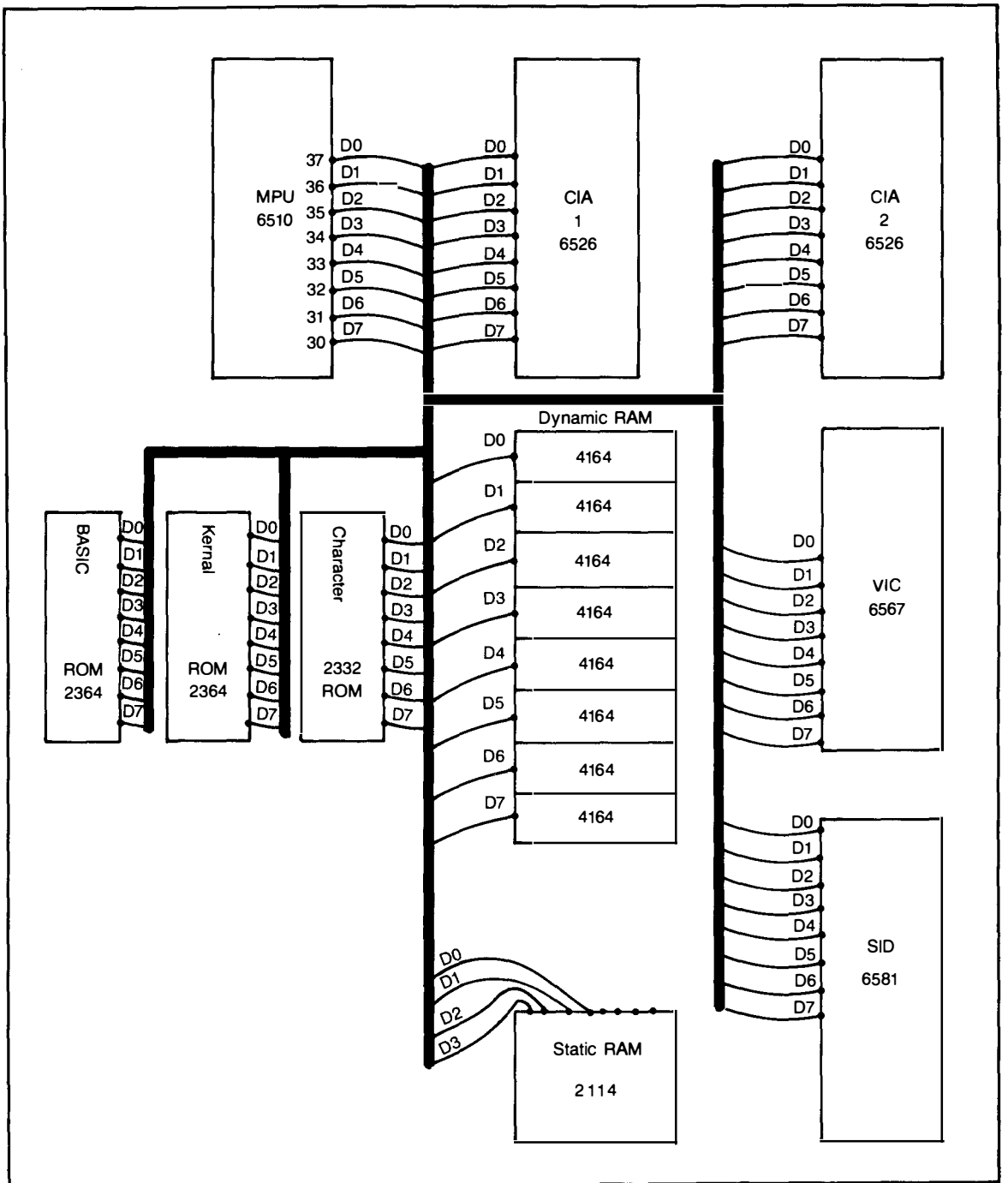


Fig. 1-11. The eight lines of the data bus originate in the MPU. They are named D0 through D7. The CIAs, ROMs, VIC, and SID are connected to all eight copper lines. The static RAM chip attaches to only the four lowest numbered lines. The eight dynamic RAM chips each connect to a different one of the eight lines. The lines are all bidirectional.



and the SID. All of the chips that the data bus is connected to have a decimal number that identifies specific locations. The numbers are called addresses.

The addresses range from 0 to 65535. Each address can be accessed individually. All of the addresses connect to the 6510 microprocessor. A listing of the addresses and what the address is connected to is called the Memory Map.

The processor can access the addresses through 16 copper traces called the address bus. It travels around the print board from chip to chip. The processor is able to place combinations of 16 highs and lows onto the address bus. There are 65536 possible combinations of highs and lows that can be placed on the address bus. Starting with address 0 and ending with address 65535 there are 65536 individual addresses. There are only a couple dozen or so chips, but the chips contain a lot of registers. Some of the chips contain thousands of registers. Each register has its own address.

When the 6510 dials an address, all of the highs and lows do not go directly to the address. Some of them travel first to substation type chips called multiplexers and decoders. There are two 74LS257 multiplexers to handle the RAM addresses. An 82S100 PLA chip conducts the ROM decoding. Two 74LS239 decoder chips do the CIA, VIC, SID, and other decoding. The address decoding is shown in Fig. 1-12.

Once addressed, the location can then be either read or written to over the data bus. The reading and writing takes place as the 6510 sends eight highs and lows to an address during a write or receives eight highs and lows from an address during a read. All of the locations, except those on the color RAM chip, consist of eight bit registers. The color RAM registers only have four bit holders.

An eight bit register can hold a binary coded number from 0 to 255. There are 256 possible combinations of highs and lows in eight bits. As you no doubt know, eight bits make one byte. A four bit register can hold a number from 0 to 15. There are 16 possible combinations of highs and lows in four bits. Four bits make one nybble. All of the

registers in the memory map are byte size except for the color RAM which holds nybbles.

When you POKE data to an address, such as POKE 1099,65, the first number 1099 is a RAM address. The second number after the comma is the code for the byte LHLLLLLH. The byte is installed in address 1099. As it turns out, 1099 is a location in video RAM. If you POKE any bytes into video RAM, the VIC reads the byte as code for a symbol. The VIC then fetches that symbol from a ROM chip and displays it on the screen. If you POKE that number into video RAM you'll see a symbol replace the last asterisk in the top sign-on line. Try it. This development will be covered as you proceed in the book. If you are curious, the location could be probed with PRINT PEEK(1099). Your 64 should print 65.

The processor traffics with the chips in these ways. It sends and receives highs and lows over the data bus with the 4164 RAM chips. It receives but does not send to the ROM chips. It sends and receives data to and from the VIC and the SID.

## Video Interface Chip

The VIC plays an unusual role. It is not only a resident of the memory map and is addressed by the 6510 in normal fashion, but it also can take over and act as the addresser. It is able to address more than 16000 of the total addresses. If any of the chips have addresses in the VIC's 16K range, the VIC can access the chip and read bytes of data out of it. This important VIC ability is covered in detail in Chapter 17.

The VIC does not need an I/O port chip. It handles its own outputs. It does not have a provision for any video inputs. The VIC is the manufacturer of the composite color TV signal that is drawn on the face of the TV.

The 64 comes equipped with an rf modulator that makes the VIC output into a channel 3 or 4 TV compatible signal. The modulator is contained in a metal case and is located at the right rear of the print board. To the right of the modulator are power supply components including the power input plug, the off-on switch and the power fuse.

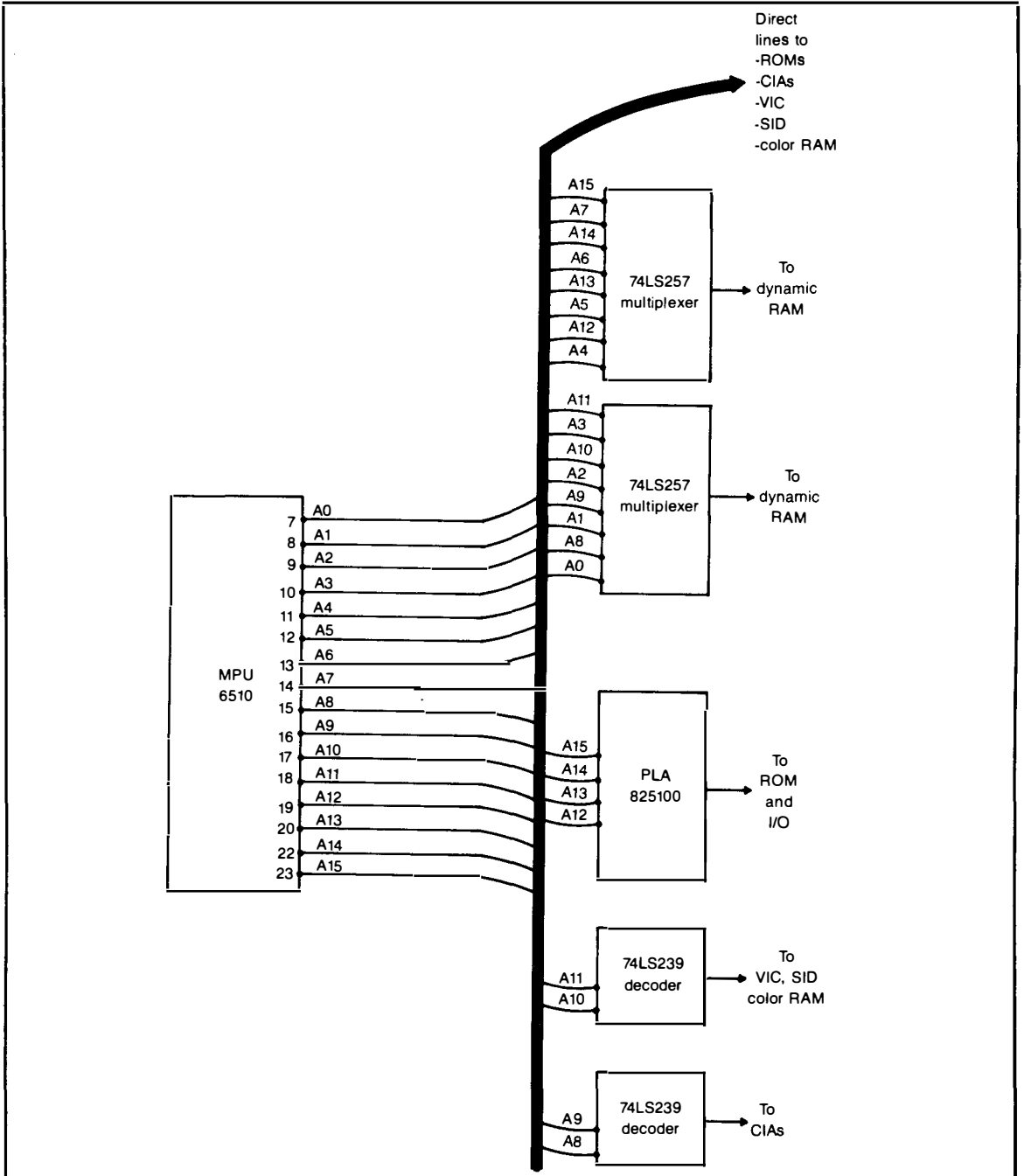


Fig. 1-12. The 16 lines of the address bus originate in the MPU too. They are named A0 through A15. The A15-A8 address lines are used quite a lot to select the chip that the MPU wants to contact. The A7-A0 address lines can be used to choose among the thousands of locations on the selected chips. The address lines go from the 6510 to the memory address locations.

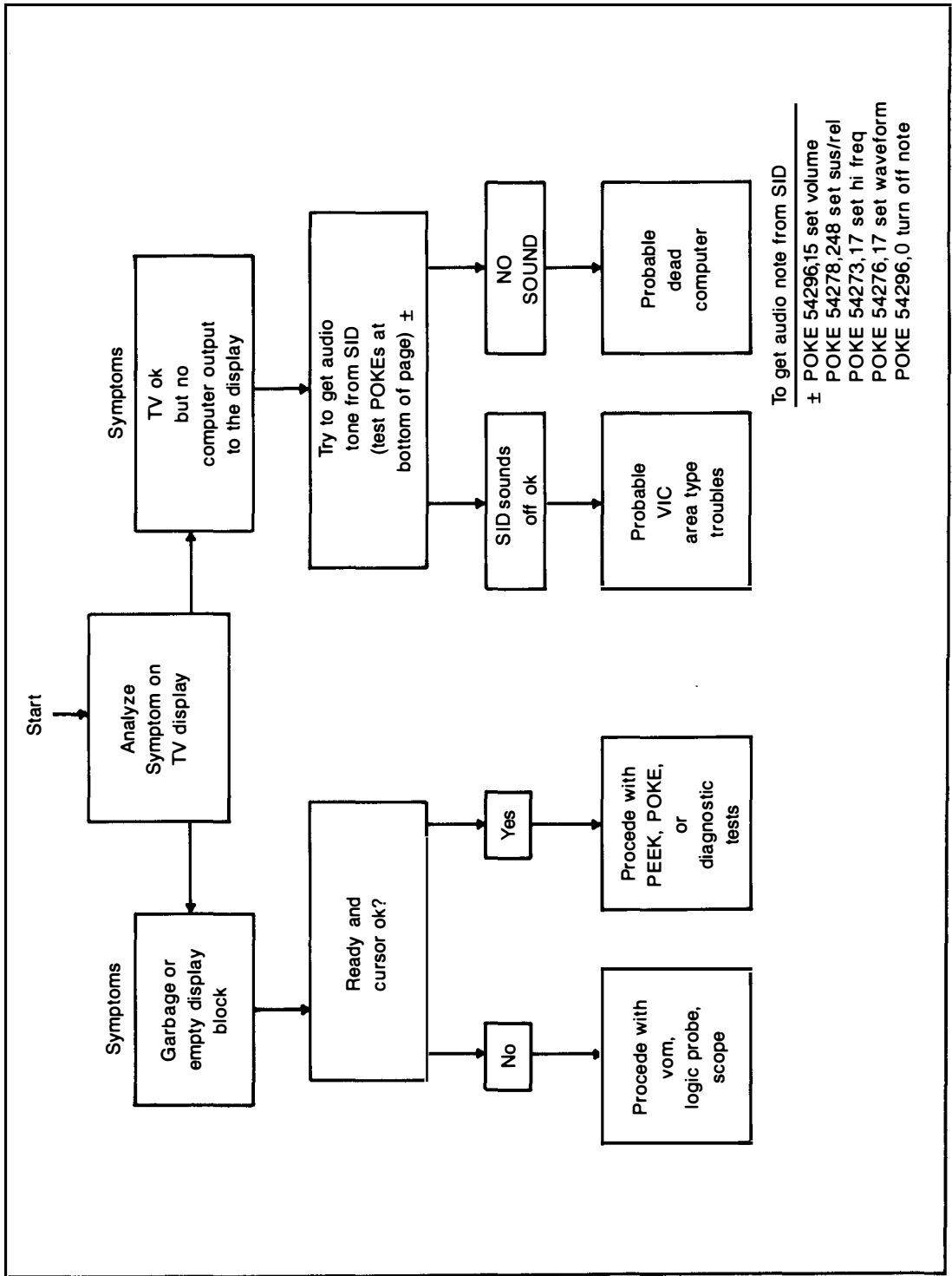


Fig. 1-13. When trouble strikes, it is often covered in this service procedure. The idea is to logically follow the steps till one of the four approaches can be confidently begun.

Symptom	Possible Defect	Try
Disk won't work Printer stops Keyboard won't work	CIA2 CIA1/CIA2 CIA1 or Keyboard	Replacing chip Replacing chip Replace chip or repair or replace keyboard
Modem not operating Audio-Video port loses sound	CIA1/CIA2 SID	Replacing chips Replacing chip
Audio-Video Port loses video	VIC	Replacing chip
Cartridge port not operating properly	PLA/Kernal	Replacing chips
Control port inoperative	CIA1	Replace chip

**Table 1-1. Peripheral Troubleshooting Chart.**

## Sound Interface Device

The SID is the composer of the audio signal that the 64 puts out. The SID is also able to receive an audio input and process it along with sounds of its own. Like the VIC, it handles its own I/O without the aid of an I/O port chip.

## TROUBLESHOOTING CHARTS

When a trouble occurs with your 64 and you realize it is being caused by the computer and not one of the external devices, refer to Fig. 1-13. If a device is at fault refer to Table 1-1. The first step is to analyze the symptom on the TV display. If the display is just a snowy TV picture, try a regular TV channel. Should the TV program be normal, move to the next block on the right side of Fig. 1-13.

Now the computer could really be dead and need power supply repairing, or the VIC could have died. To determine which section is at fault, try to produce a sound out of the SID. If the SID is quiet, then the power supply is indicated. Should the SID sound off though, the computer is ok except for the VIC and it's surrounding circuits. Turn to Chapter

18 for a SID test program of POKEs. If the sound test fails, then the power supply is suspect. Turn to Chapter 20 and follow the step-by-step instructions.

If the TV channel is also snowy, your TV is bad and you need TV service, which is not covered in this book. The same goes for any other recognizable TV problem. The computer is not the problem in these cases.

When the screen fills with garbage or comes on with an empty display block (the left side of Fig. 1-13), all of the digital circuit components are suspect. This includes the processor, RAM, ROM, CIAs, and decoders. The VIC and SID are probably not involved in the trouble.

When these symptoms occur, the READY sign and cursor might or might not be displayed. If they are not displayed, then diagnostic software cannot be used. The repair will begin by making voltage, logic probe, and scope tests. When the READY sign and cursor do appear, chances are you can make use of the various diagnostic PEEK and POKE tests or longer programs, as demonstrated throughout the book.

A black and white photograph of a Commodore 64 computer keyboard lying on a light-colored surface. To the right of the keyboard, a screwdriver with a dark handle and a metal shaft is visible. The keyboard is a standard QWERTY layout with dark keys and a lighter-colored numeric keypad on the right side. The title '2. Disassembly' is overlaid in large, bold, black letters across the middle of the keyboard.

## 2. Disassembly

**W**HEN A COMMODORE 64 GIVES UP THE ghost and you make the decision to repair it, most of the time you have to take it apart. The first time you do the disassembly you are sure to be hesitant. You know from long experience with all types of machinery that you could possibly cause some additional troubles just by taking it apart. You don't want to start a repair job by causing more trouble.

Fortunately, the 64 is assembled in a sensible manner, which makes the disassembly relatively easy, although extreme care and slow moves are the order of the day. The first steps are common sense. Arrange a large enough place on your work bench. Gather your tools together. Be sure to have good bright lighting and place a rubber mat on the bench. Disconnect all the attachments to the 64. Place it on the soft mat. Figure 2-1 illustrates your progress to this point.

It is a good idea to have the bench area as clean as possible. Dirt and filings that manage to enter the computer could cause troubles. It is not a good idea to work on the 64 in a low humidity environ-

ment. Should it be cold outside but nice and warm inside and static electric sparks are popping as you walk on the carpet, be careful about working on the computer. Chances are good you could blow out chips with the static activity. There will be more about the static electricity danger in Chapter 4.

### REMOVING THE TOP

Place the 64 upside down on the soft pad. Position the 64 with the rounded front facing you. In the bottom of the case, there are three recessed screws along the rounded front. Take a medium sized Phillips head screwdriver and unscrew them as shown in Fig. 2-2.

Once the screws are out, turn the 64 back to its normal position. You'll find that the top will now swing up at the front. As Fig. 2-3 shows, the rear end is sort of hinged. I say "sort of" because the hinging is just the plastic top and bottom cut to hold on to each other. When you swing the top all the way up, the hinging will disconnect and the top piece of plastic disconnects from the bottom. However, you must be careful. There are two miniature



Fig. 2-1. The 64's case can be opened with a Phillips head screwdriver. The print board is removed with the aid of a soldering iron and longnose pliers.

wiring systems that connect the top casing to the bottom.

The first wires are three leads that connect the pilot light to the print board. You can remove it gingerly, as shown in Fig. 2-4, making sure you do not bend any connections or cause any damage to either end. The second set of wires are from the

keyboard to the keyboard port. The keyboard port is a 20 pin male connector. The cable from the keyboard terminates in a 20 pin female plug as shown in Fig. 2-5. They can be pulled apart, but before you do, find out if it is necessary to disconnect them for purposes of repair.

In the 64, the keyboard is mounted firmly on-

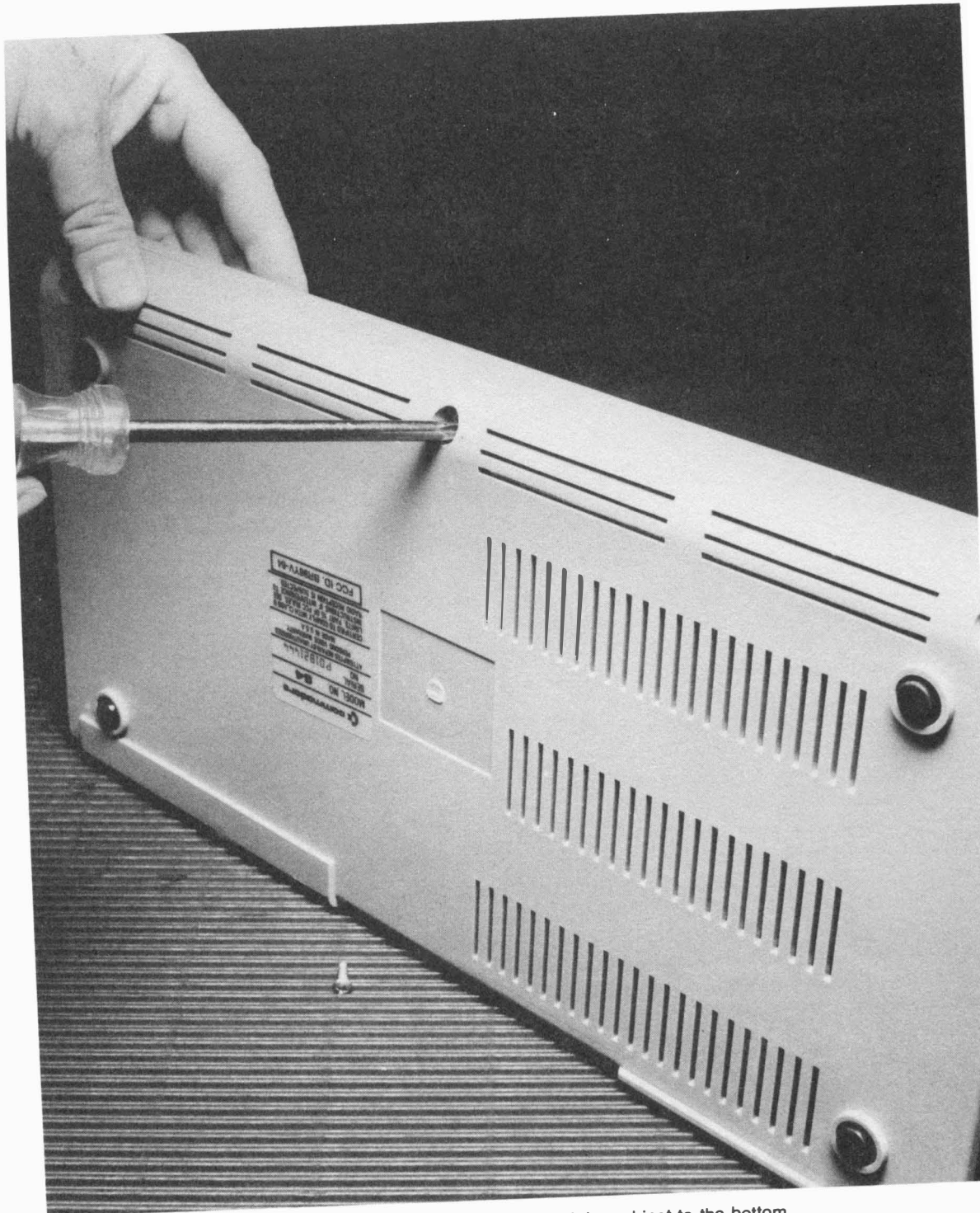


Fig. 2-2. There are three Phillips head screws holding the top of the cabinet to the bottom.

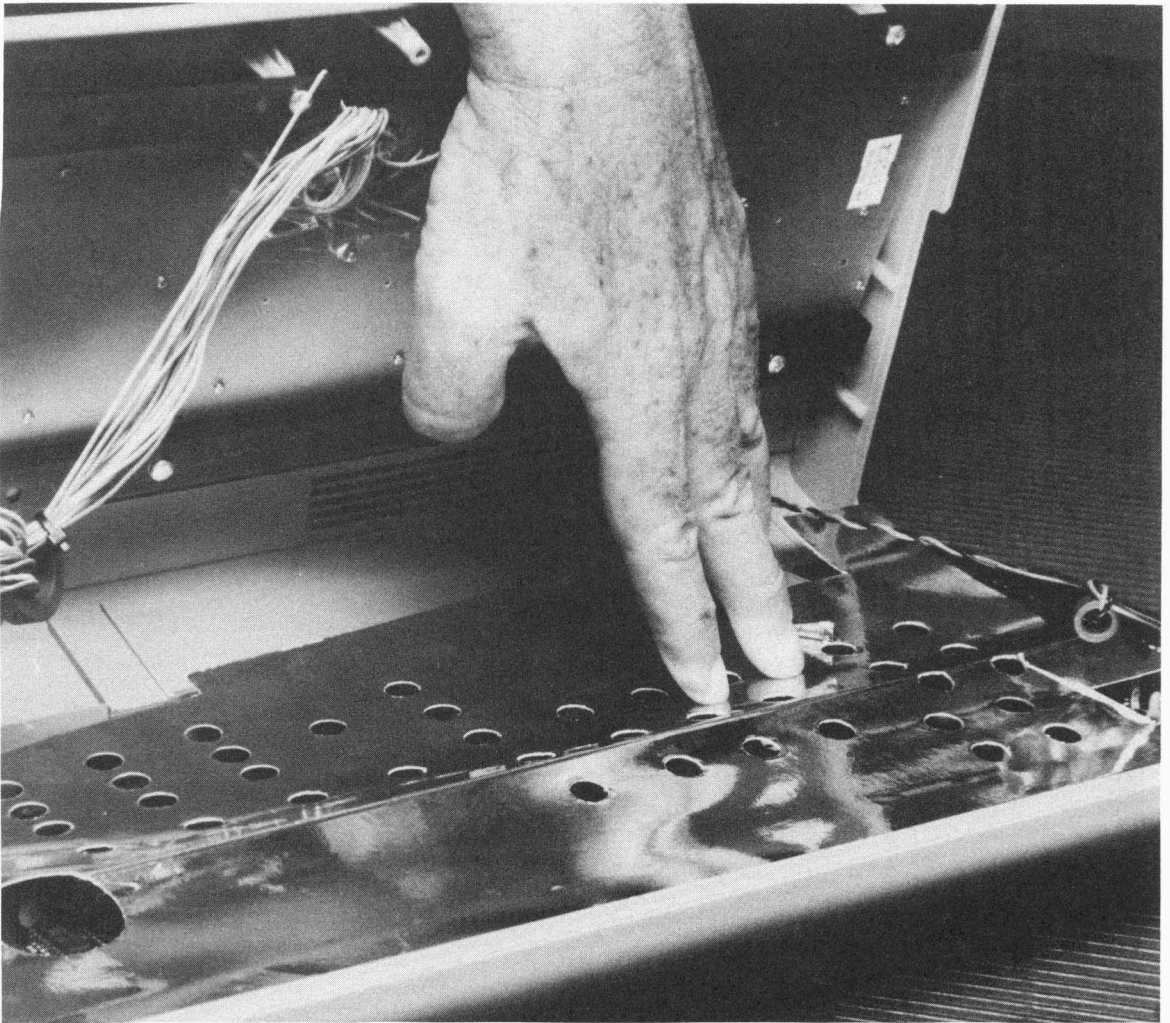


Fig. 2-3. Plastic type hinging permits the top to be swung away from the bottom.

to the plastic case top. If you do not have to remove the keyboard from the top or unplug the 20 pin port connections, don't do it. The more you disconnect, the more you have to reconnect and the more exposure you'll have to possibly causing additional troubles. The good repairman only takes apart what is absolutely necessary. You can place the still connected keyboard to one side. Disconnecting the pilot light will give you plenty of room.

Once the top is to one side you'll be facing a piece of cardboard covered with silver foil that is hiding the print board. The covering is taped down

to a shield on the print board. The tape is covered with a coppery cover and the tape is soldered to the shield in a couple of places. Figure 2-6 shows how to remove the print board cover. Desolder the tape with as little heat as possible. Then bend the cardboard back as shown in Fig. 2-7. That does it. You are into the print board area.

### **REMOVING THE PRINTED CIRCUIT BOARD**

Separating the top of the cabinet from the bottom is easy. Three screws, a pilot bulb connection, one



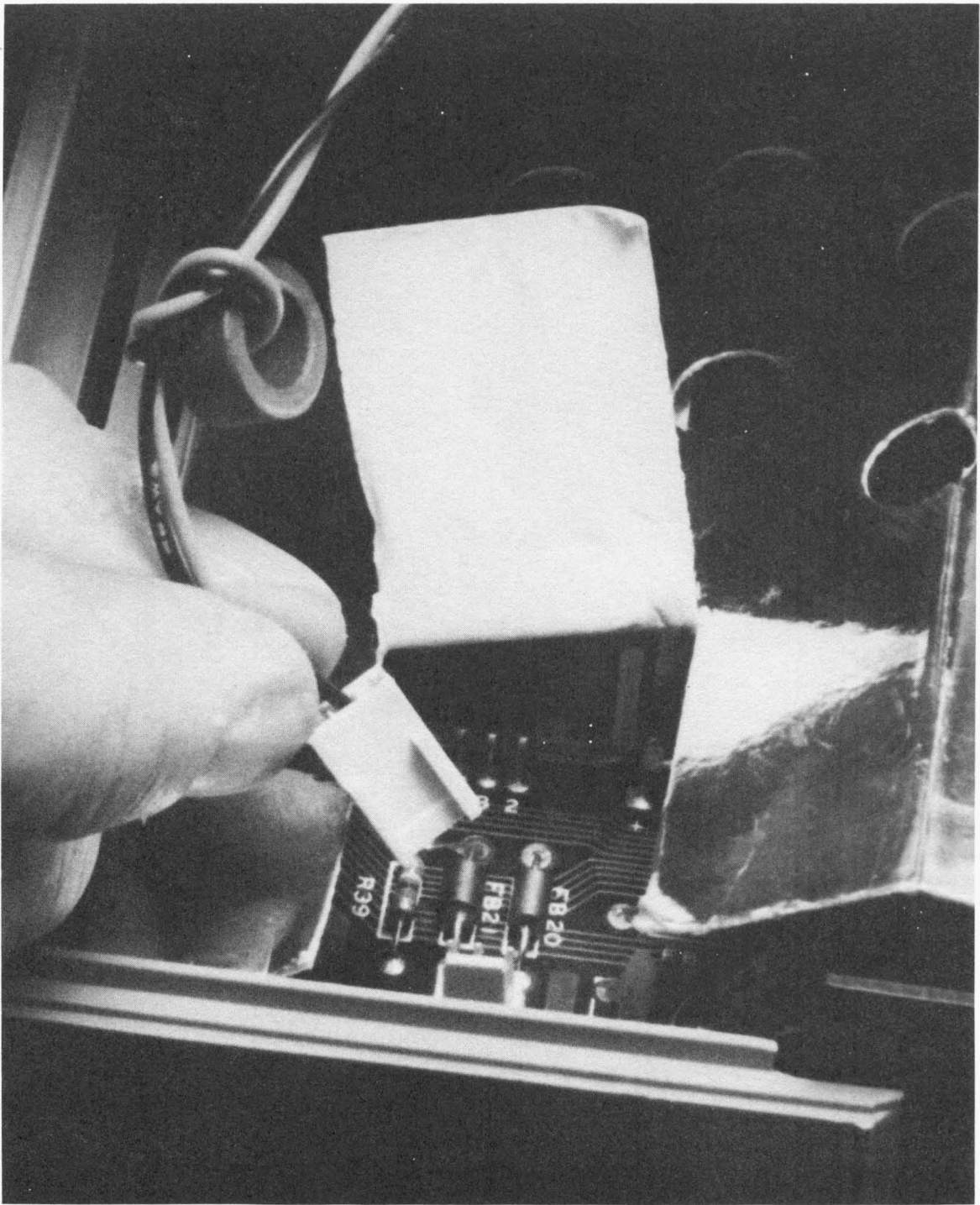


Fig. 2-4. The LED Pilot bulb is connected to the board with a small pressure connector.

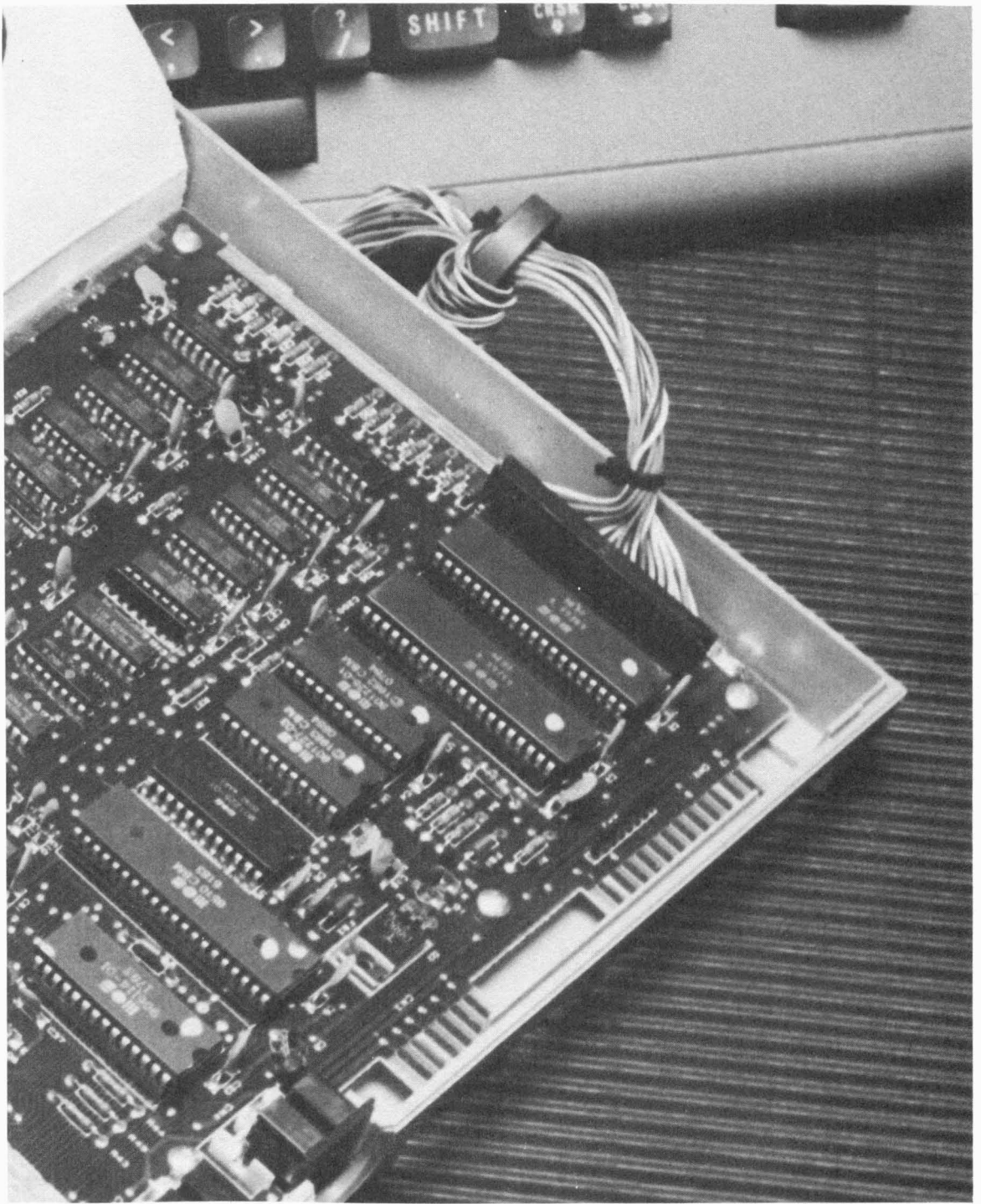


Fig. 2-5. The keyboard is plugged into a socket at the end of the print board.

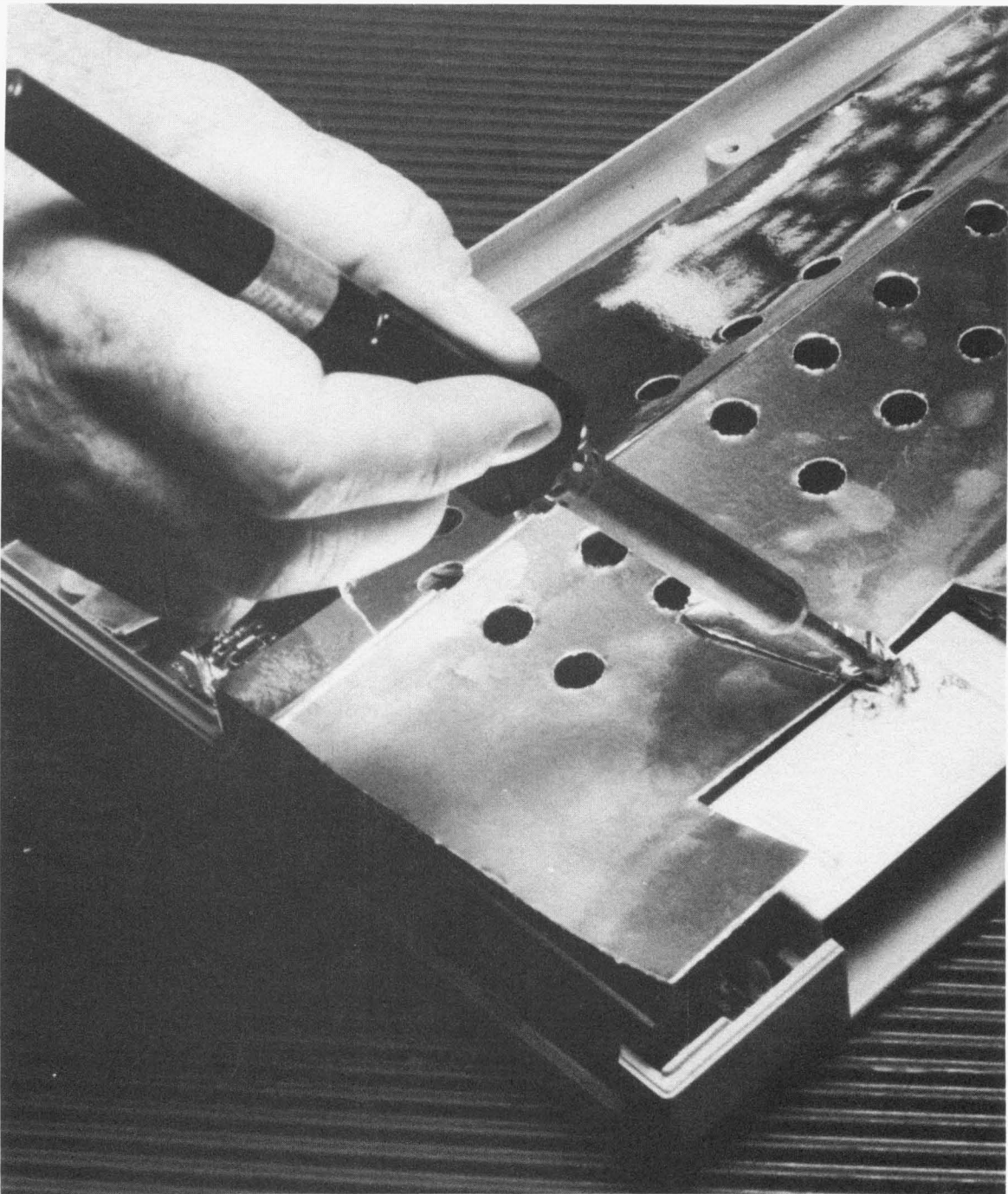


Fig. 2-6. The metallized cardboard covering on the board is soldered to the cover of the rf modulator. It is removed with a low-wattage iron.

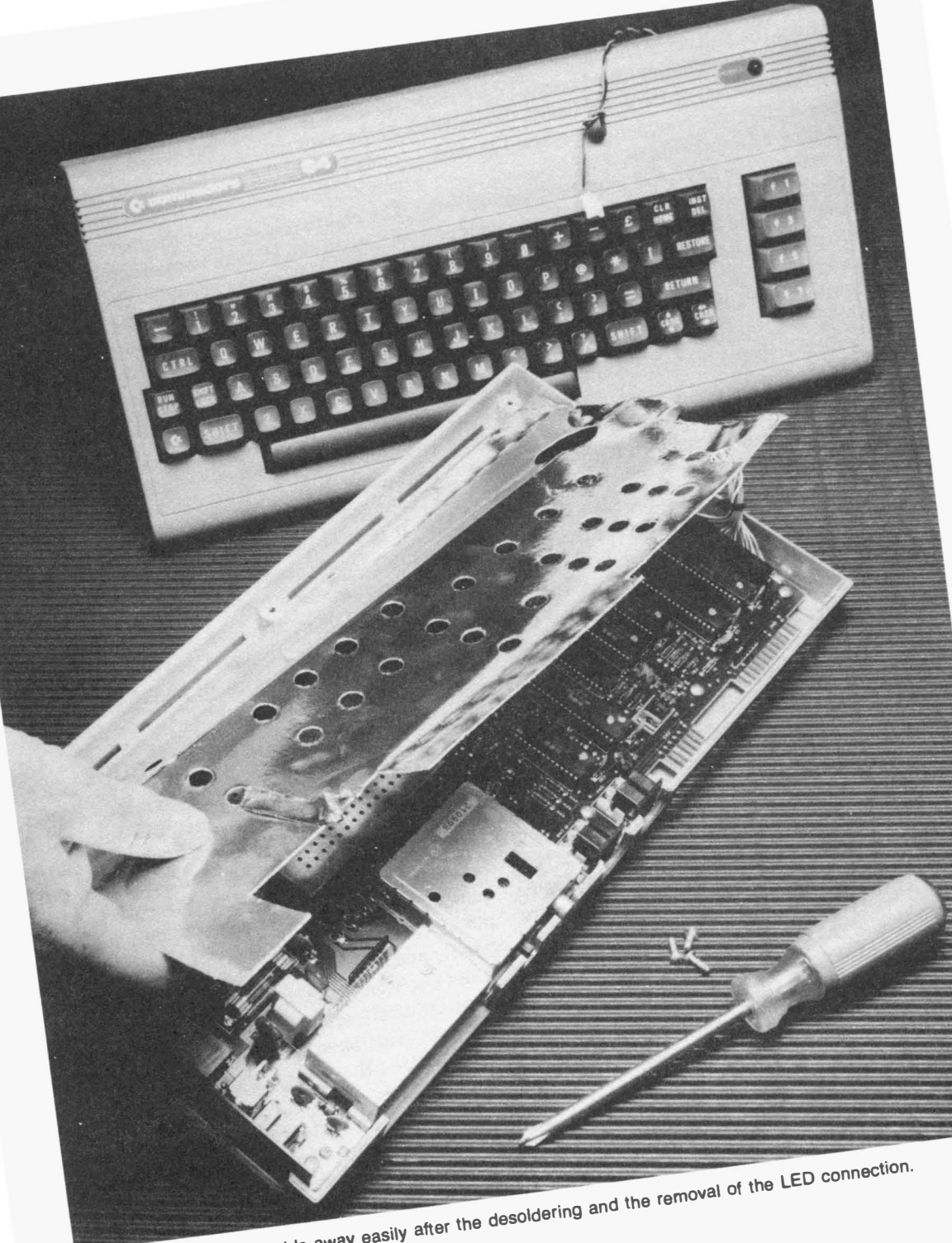


Fig. 2-7. The cardboard folds away easily after the desoldering and the removal of the LED connection.

solder joint, and the job is done. Removing the print board from the bottom of the cabinet is somewhat more difficult. Fortunately, most repair jobs do not require the removal of the print board. When the job does dictate the complete separation though, here is how it is to be done.

First of all, the keyboard must be disconnected at its plug near the end CIA. The top can then be separated from the bottom. Place the keyboard top in a safe place. It should be clear of any undue heat and away from any liquids. If it is going to be disassembled for any length of time, box it or do something to keep it free of dust.

Figure 2-8 shows the print board screwed and soldered onto the cabinet bottom. There are seven Phillips head screws holding the board tight. Remove all of the screws shown without touching any nearby chips or ports. Keep the screws with the three you took out to open the case. These seven screws are a little smaller than the three for the case.

Once you remove the seven screws, the board can come free. If you grasp the board by the rf modulator, you can lift it right out of the cabinet. You'll see seven little plastic mold holes where the screws had been attached to the cabinet floor.

On the bottom of the board, you'll find a shiny metal cover. This is the ground plane. A closer look shows a tan sheet of insulation between the board and the ground plane. It keeps the board bottom connections from shorting to the ground plane. Notice the ground plane has a number of tabs that are folded over and connected to the print board grounding strip. There are ten tabs, and they are soldered to the grounding strip.

The rest of the disassembly job is desoldering. If you must gain access to the bottom of the print board you will have to desolder the ten ground plane tabs from the grounding strip. That will undo the sandwich of print board, insulator and grounding plane.

The desoldering is a tedious job and you must take care not to apply too much heat or pressure. The only reason that you would take the sandwich apart is to replace chips that are soldered onto the print board.

There are 32 chips on the board of various sizes and shapes. Of the 32, eight of the largest are plugged into chip sockets. There are 24 smaller chips soldered into the board. If you must replace one of them, the sandwich must come apart. If you count the visible chips, there are seven in sockets. The eighth one is under the metal shield. To open the metal shield, take a small screwdriver and turn it in the notch on the left side. The top of the shield is hinged and the notch is where it opens.

There is one more socketed chip under the shield along with four smaller chips that are soldered in place. The socketed chip is the VIC. When you open the shield, note that the VIC has a heat sink that touches the shield. The sink is covered with a white silicon paste to conduct the heat away from the VIC. Should you replace the VIC be sure to reapply the silicon paste.

It is vital that, as you take apart the 64, you perform the disassembly in a slow and deliberate manner. Carelessness will result in "induced" troubles. That is, troubles caused by you. If you disassemble correctly, then you'll be able to get it back together again without undue difficulty.

## **DISASSEMBLY AS A CURE**

The step-by-step procedure of disassembly often provides a fix all by itself. Sometimes during print board manufacturing, a form of booby trap can be accidentally installed on the board. After the print board is assembled, it is typically soldered with automatic machinery. During the soldering process, the machines generate hot gases that can expand rapidly and shoot liquid solder into the air. As the solder cools and falls, it hardens. The solder then forms little balls and drizzles down onto the board. The little balls have flux on them and often will stick where they hit. They are tiny little troublemakers.

Of course, the factory is very aware of the solder drizzle, and the board goes through an extensive cleaning procedure after soldering to eliminate this potential source of trouble. However, no matter how hard they try an occasional sliver of solder will stick in a place where it is not seen but does

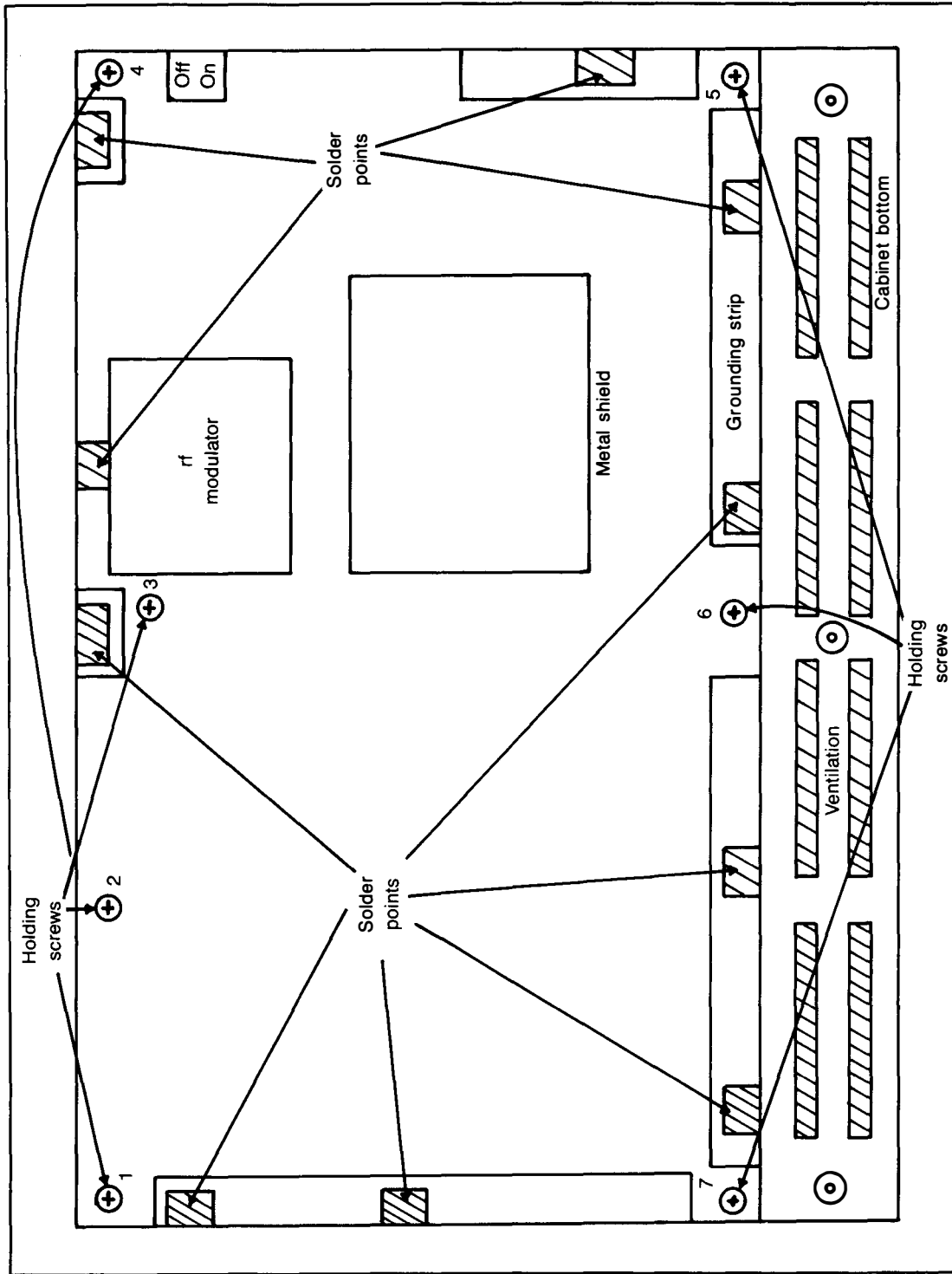


Fig. 2-8. When an unsocketed chip must be replaced, the print board must be removed. This is accomplished by taking out these 7 Phillips head holding screws and desoldering the 10 folded over tabs.

not cause any trouble at that time. It sneaks through the extensive board inspection and is shipped out with the finished computer.

Somewhere down the line, as the computer is taken home and put into service, a solder ball gets loose and starts rolling around the print board. It settles into a spot such as between two pins on the CPU. The computer stops operating in a logical manner. A screenful of garbage appears. The computer needs service.

If this happens to you and you have had the computer more than the limited warranty time of 90 days and you decide to check it out yourself, the first thing you do is take it apart. As you remove the casing and turn the board over on its side, a tiny solder ball could plink out. Then when you attach power and turn it on, the garbage is gone! The computer is now working fine. Yes, you have completed a fix. The removal of the solder ball short has cured the trouble. The technique was simply taking the computer apart.

## VISUAL INSPECTION

While the solder ball type of short does happen on occasion, most of the time you are not so lucky. There are, however, a few other service moves that are easy to make that do often produce good results. First of all there is the visual inspection.

The first step of a visual inspection is to take the computer apart and expose the print board. This includes removing the metal shield over the VIC after you have inspected the open areas of the board and found nothing out of the ordinary.

To inspect the board, a bright light and a good magnifying glass are very helpful. You are mainly looking for short circuits, open circuits, and burnt or mangled components.

## Ground Plane Short

One common short happens when a chip lead pushes into and punctures the insulation between the print board and the ground plane the board is

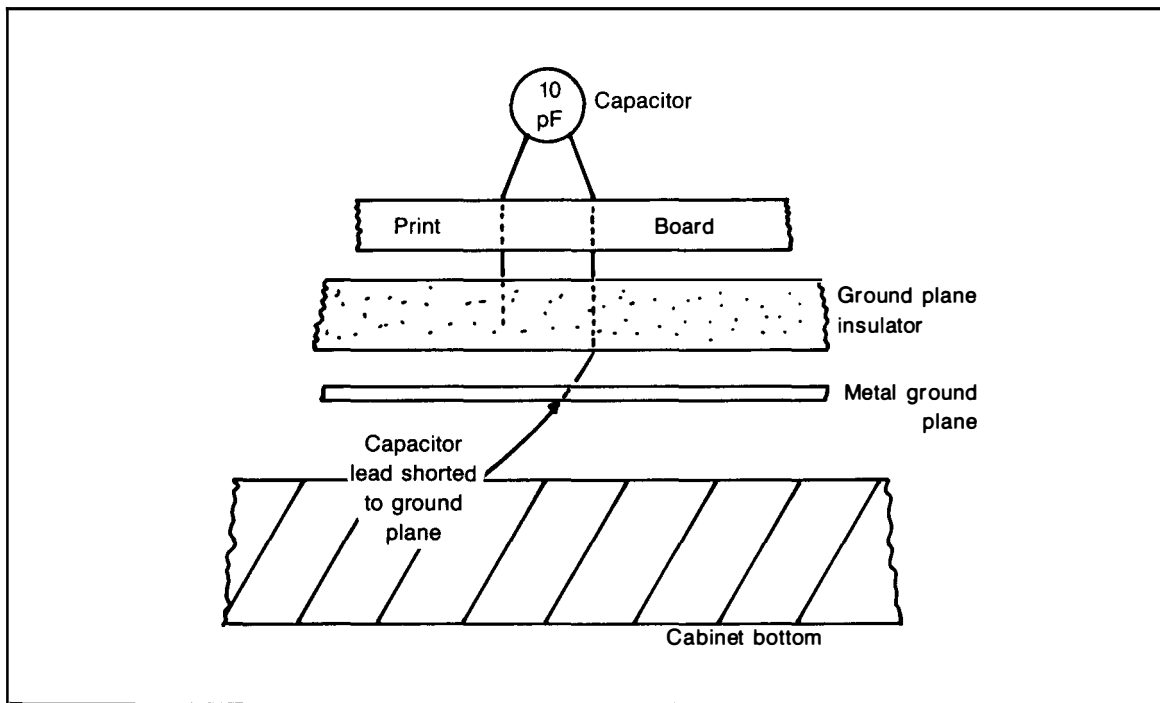


Fig. 2-9. The print board is the top of a sandwich. If a component lead should pierce the insulator, in the middle of the sandwich and contact the ground plane, the bottom of the sandwich, a short circuit could develop.

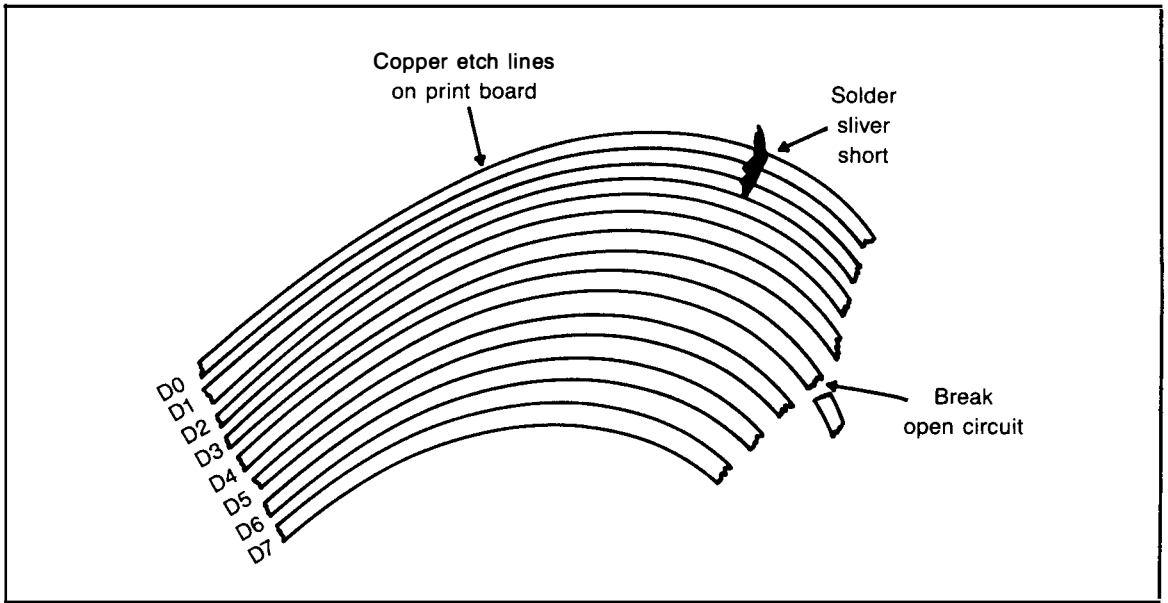


Fig. 2-10. The bus lines are copper-etched wiring strips that traverse the board. A short or open line will cause trouble.

mounted on. As shown in Fig. 2-9, the computer is put together in layers. The cabinet is at the very bottom. The metal ground plane is against the cabinet. It acts as a common ground for the entire board as well as a shield to ward off interference that might come up through the bottom of the machine.

The insulator is on top of the ground plane. Its job is to keep all the active connections in the board from contacting the common ground. The print board is mounted on top of the insulator. A ground plane short occurs when one of the print board active connections manages to puncture the insulator and make contact with the ground plane. You can often spot the short and clear it by snipping off the excess amount of the lead or by bending it over.

## Board Defects

Another quick check should be made at the chip sockets. Sometimes a socket pin can get bent under instead of being soldered into its board hole. It could also sneak by inspection as it works ok since the socket is attached firmly and the bent over pin makes a pressure contact. However, in your home,

after about a year, some corrosion builds up on the pin and the pressure contact no longer holds. You suddenly have either loss of computing or erratic performance.

This type of trouble can be seen with the bright light and magnifying glass. Once you locate it, you can gingerly move the pin into its appointed hole and apply a tiny drop of solder with a small iron.

Other types of board defects can also be found visually. The 16 copper address lines that travel around the board from chip to chip and the eight copper data lines that take almost the same routes as the address lines should be examined carefully. These two bus lines must be continuous. There cannot be any breaks. Also each line must be separate. They are not allowed to touch. If you find a break, then you have an open circuit. If any of the copper traces touch each other, then you have a short circuit. Figure 2-10 shows both problems. Either way, you are going to have trouble. Chapter 15 goes into these bus lines in detail. It is good practice to examine the address and data bus lines carefully before each repair. Odds are favorable that you might find a solder sliver short or a break and have a quick fix.



## CLEANING

Another service move that might complete a repair, but is useful in any case, is cleaning. While the computer is apart, you will be able to see how dusty it is. As a computer is used and time goes by, the inside collects dust. It can't be helped without taking extraordinary measures. Dust itself is not a problem. Ordinary dust is an insulator and, as such, won't short out the print board or its components. One problem, however, besides the fact that it is dirty, is ventilation. Dust enters through the ventilation slots. If enough dust collects, the circulation becomes restricted, which can cause some overheating. If dust gets into the keyboard area, the interface ports, or other moving part areas, it can clog up plugs and cause all types of annoying and erratic operation.

A good way to remove the dust is with the patient use of a thin, clean, dry paint brush. The dust removal should be done slowly and carefully. Be especially careful around the RAM chips since they are susceptible to static electricity. Don't dust on a day that is dry and static electricity is jumping off of you. It is a good idea to ground the brush in any case, as described in Chapter 4. Never use any water or cleaning solution on the board. The idea is not to make the board spotless. All you want is good air circulation and a clear view of the circuits.

## STATIC ELECTRICITY

The silicon chip has one serious drawback. It can be easily destroyed by ordinary static electricity. If you walk across a carpeted room on a dry day, reach for a chip, and a static spark flashes from your hand to the chip, odds are the chip has just been electrocuted. Oftentimes, you cannot stop the static build up of the electric charge on your body. How can you avoid this problem? There are ways. Let's examine the situation first before discussing the preventive measures you can take.

There are two important types of chips you'll encounter in the 64. One is called TTL, which stands for Transistor-Transistor-Logic. The second is the MOS, which stands for Metal-Oxide-Semiconductor. There is more detail on the con-

struction of these chips in Chapter 4.

The chips have a lot of electrical characteristics. One of them is the amount of static voltage a chip can take before it is damaged. This is called the Threshold Voltage for Electrostatic Damage. It is measured in volts. It has been shown that a TTL will be destroyed if a static shot of 300 volts or more is allowed to contact it. The MOS chips are likely to fail if hit by 250 volts or more. How much voltage is pushing a spark between you and the door knob on a low humidity day? It could easily be a jolt of 3000 volts! As you can see, it is quite easy to kill a sensitive chip.

### What Is It?

Static electricity involves the build up of electric charges on the surface of an insulator. The electric charges are electrons. The insulator will be charged if there is an excess or deficiency of electrons. An insulator with an excess of electrons is said to have a negative charge. When there is deficiency of electrons, a positive charge is present. The voltage developed gets larger as the charge increases. Either type of charge can cause a spark and a chip death.

Static electricity results from friction between different types of material. Your shoes on a carpet produce a lot of this friction. As you walk around the room, you charge up. The rug and your shoes are insulators. The charge is built up there and passed into your body, which is a conductor.

The charge on an insulator remains at the spot where it develops. The charge on a conductor, however, spreads itself uniformly throughout the conductor. That is how we are able to get rid of the charge. If we, as conductors, touch an earth ground, the charge is removed at once. However, keep in mind that the discharge to earth ground consists only of the electrons in our body. Any charges that are on our pieces of clothing, which are mostly insulators, will remain. Attention to chip handling must still be a concern even if you are grounded properly.

To alleviate the problem somewhat, wear clothes that are more conductive. For instance,

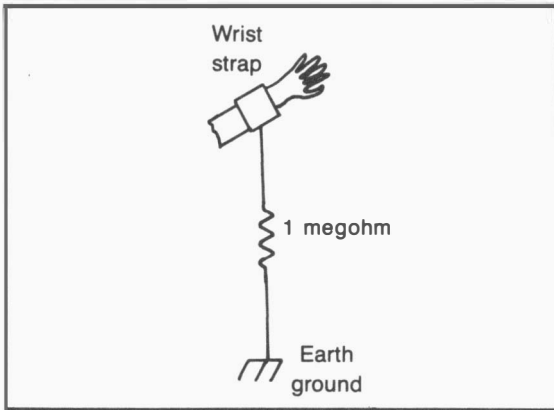


Fig. 2-11. In order to keep static electricity on your body at a harmless, low level, commercially available wrist strap systems can be used. They provide an escape path to earth ground for the unwanted charges.

leather soled shoes are better than rubber soled shoes, and cotton clothing is more conductive than nylon.

## Wrist Strap

If you are going to be doing any extensive chip handling, the professional way to deal with static electricity is with a commercially available wrist strap discharger. These can be bought in an electrical or electronic supply house. They consist of a wrist strap that is a conductor with a connector coming out of it. The connector contains a resistor, typically about a megohm, that connects to an earth ground. Figure 2-11 illustrates this grounding scheme. Any charge that builds up in your body will immediately be discharged to earth ground. Note

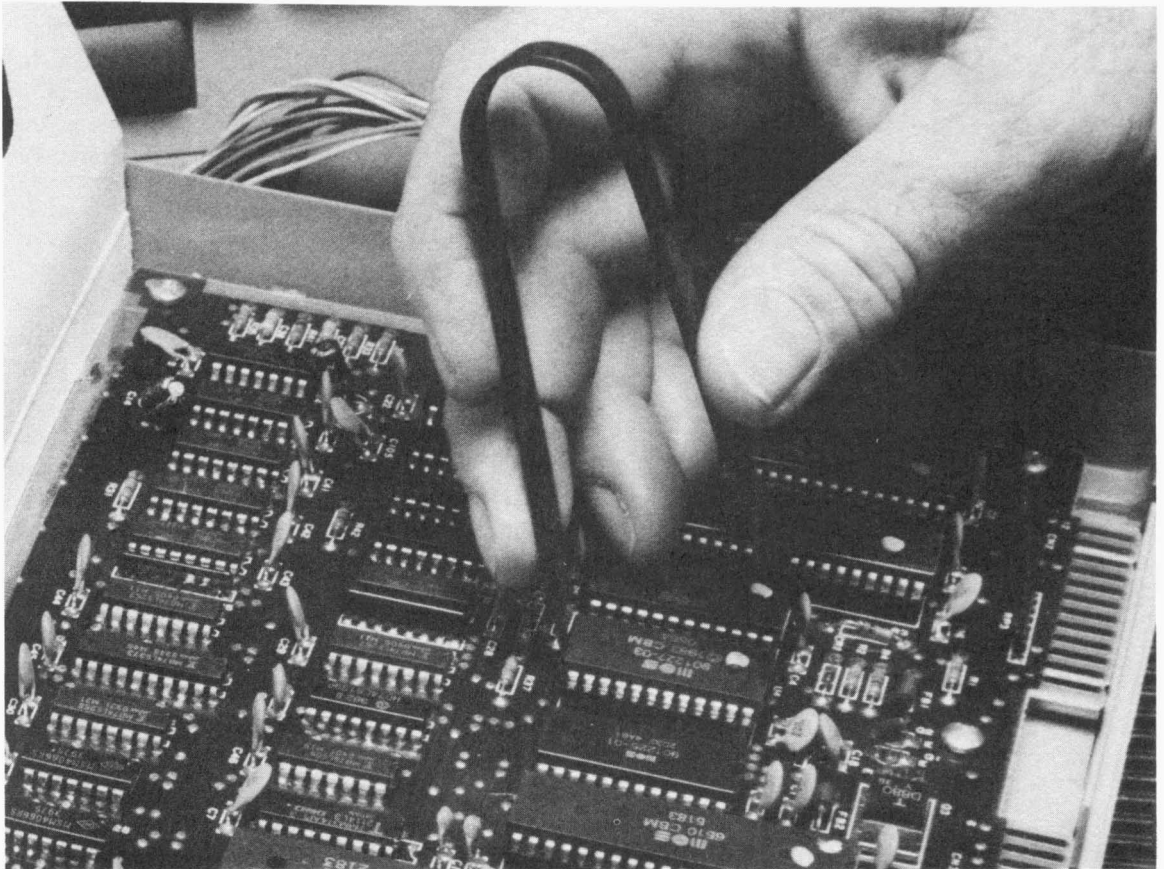


Fig. 2-12. This chip extractor has a hole in the top to connect a ground wire to it.

that I said earth ground. A cold water pipe is usually such a ground.

**This wrist strap is to be used only while you are working with equipment that is completely disconnected from any sort of electric power!** While wearing the strap, you are connected to ground. If you contact 120 Vac, you could get a nasty electric shock, although the one megohm resistor will help to limit current. To be safe, only connect yourself to the wrist strap during chip handling on disconnected equipment.

Put the wrist strap on before opening up the computer. Keep it on all during any chip handling. You are producing static electricity with every move you make. Be sure to remove the strap before you plug in the machine to the line voltage.

### **Additional Precautions**

The professional electronic technician often uses a lot of other static discharge techniques to avoid chip losses. First of all, work benches are usually insulators. They build up charges. You can't easily discharge the entire surface because a ground wire only discharges the point of the bench it

touches. To solve this difficulty, the technician will ground the print board he is working on. Being sure the board is disconnected from power, he will connect a jumper wire, with a one megohm resistor in series, from the board to an earth ground.

The replacement chips come packed in a conductive foam. He does not remove the chips from the foam till he is ready to install it into the print board. He will never take the chip out of the foam a long time before it is going to be installed and store it in a shirt pocket. That is asking for trouble.

If you can, it is a good idea to pull chips off print boards with a grounded extractor tool like the one in Fig. 2-12. The tool in the photo has a hole in the top to attach a ground wire to it. A companion tool is the chip inserter. It can also be conveniently grounded through the pin on its top.

You may think that I am overly cautious concerning these chip handling problems. It is possible and even likely that you could replace a chip without all these measures. However, if you do blow out a new chip with a static spark after waiting weeks for it to arrive, the feelings of frustration can be aggravating.

A black and white photograph showing a person's hand holding a pencil, pointing at a printed circuit board (PCB) of a Commodore 64 computer. The hand is positioned on the left side of the frame, and the pencil is pointing towards the center of the board. The board is a complex grid of components, with various chips and traces visible. The background is a plain, light-colored surface.

### 3. Chip Location Guide

**T**HE FIRST PIECE OF SERVICE INFORMATION a TV repairman looks for during a job is the TV's Location Guide. There is such a guide pasted on the inside or bottom of practically every TV. The guide is a rough layout of all the important landmarks in the TV. This includes any tubes, all the transistors and chips that the TV uses. The guide gives the location, name, and generic number of all the landmark parts. The servicer is usually able to complete most TV repairs with the aid of the guide coupled with his knowhow. I do not know of a single TV that comes purposely without such a location guide.

On the other hand, I have not seen a guide like this in any computer, including the Commodore 64. Therefore, since the last chapter provided the information on taking the 64 apart I will take the next servicing step and create a Chip Location Guide. I will place on the guide a lot more than just the 32 chips.

#### **DRAWING A GUIDE**

This book covers the Commodore 64. As time goes

by, the model number could change and the layout of the print board might vary. I will use my 64 as the basis for the guide I'll produce.

All that is needed are some simple school supplies: a pencil, a ruler, and some graph paper. Drawing a useful guide shouldn't take more than five minutes. You will easily save more than that during any 64 repair or cleaning job with such a guide. There will be no stumbling through a complex factory drawing or schematic when the repair only requires knowing where the chips are. In addition, if the repair does need more servicing information than the chip guide provides, you'll already be briefed since you did the chip guide first. It will be much easier to read the complicated service information that you will be studying later in this book.

To begin, take a piece of graph paper about 8 1/2" x 11" and draw a rectangle about 5" wide and 10" high. The 64's print board has an aspect ratio of approximately 2 to 1. Lay out coordinates by inches. My illustration in Fig. 3-1 shows 10" across the top of the rectangle and 5" up the left side. That

way ever chip on the board has a location. For example, suppose you want to know where the 6510 MPU is located. All you have to do is look on the chart that accompanies the sketch. It says the MPU is located at 3 1/2" across and 1 1/2" down. A glance reveals the 6510 on the location guide and the chip is quickly found on the print board in the same relative position.

Once you have the coordinates laid out, you can draw in the chips. Do not worry about being exact. The only reason for the guide is to give you a feel for chip locations. You should have a familiarity of the board layout of any computer you are repairing or cleaning. The better your feel of the board, the faster and safer the work will progress. It is all in the interest of eventual mastery over your 64.

The shape of the chips can be approximate. The location of the chips need not be precise. You do not have to put the manufacturers part number on the chip. You do not even have to put as much information on your guide as I've shown in order for the guide to save you lots of time.

I put the following information into the illustration. First of all there are four 40 pin chips. There are three out in the open and the VIC chip encased in a metal shield. The 40 pin chips require a bit more care. I have a little chart for them at the top of the page. The chart quickly identifies them by their coordinates. I also identify them by their corner pins, function, and generic part number. For example, the VIC chip is shown as a 6567 in Fig. 3-1.

The 40 pin chips are drawn onto the chart first. In this case I've drawn in the two CIAs, the MPU and the VIC. Since the VIC is under a shield I've drawn the shield as a dotted line. The rest of the chips are then sketched in by using the 40 pin packages as reference. There are the RAMs, ROMs, buffers, latches, decoders, flip-flops, gates, separators, video circuits, the rf modulator, and finally the power supply regulators.

I'm sure it will take you longer to read the last few paragraphs than it will take you to rough out a useful location guide for the 64. In case you want to avoid actually drawing the location guide, make a photocopy of Fig. 3-1 and paste it inside your 64

case. You'll miss out on one pass at becoming familiar with the print board though, if you do copy it.

The amount of information required for the quick checks is actually small, and you have all these valuable tidbits on the guide. They are the location of the chips, corner pin numbers, the function of the larger chips, the generic part number, and the dc voltages in the power supply. With only these tiny bits of information, you can complete a large percentage of repairs in the same way that a TV repairman does using a tube and transistor guide.

Between Fig. 1-9 and Fig. 3-1 you can quickly refer to the following information:

- The location of all the chips. If a chip has a paint dot or notch at one end, it is marked on the guide. Pin 1 will be found immediately to the left of the marking.
- The location of the chips under the metal shield. The dotted lines could serve as a reminder to replace the shield after the repair. It is not good practice to leave shields off.
- The location of any and all plugs, adapters, and other interface circuits.
- The location of external circuits mounted inside the case. The rf modulator is an example.
- A special marking to show which chips are socketed and which are not.
- The location of the power supply fuse.
- The function of the large chips and their generic part numbers.
- The voltages of the power supply regulators.

These two drawings illustrate what you actually see when you look down at the print board. It gives you the perspective of the physical computer. As you compare the guide to the board during troubleshooting and repair, you do not have to convert a theoretical schematic symbol to a physical layout as you must when the schematic is used for location information.

These guides, though, will only take you so far. They enable you to find a suspect chip so you can remove it from its socket and try a new replacement. They show you where the fuse and the

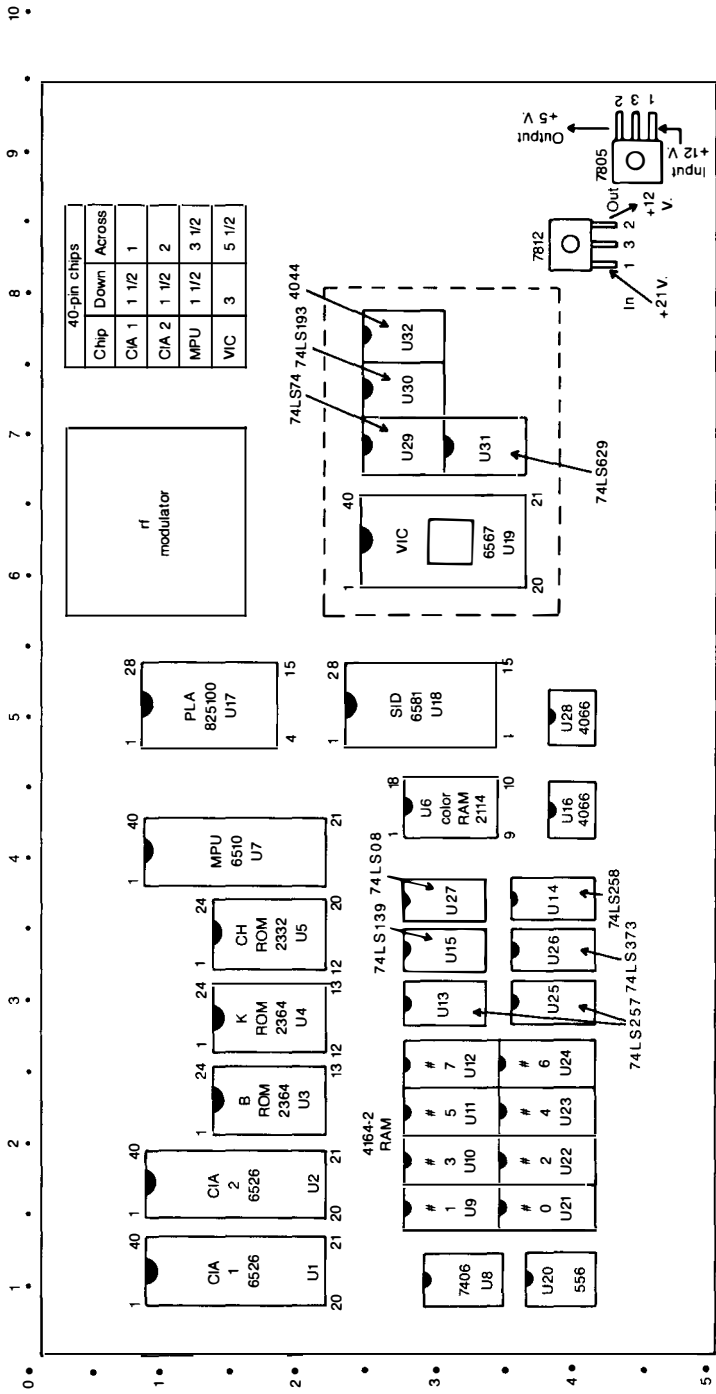


Fig. 3-1. The Chip Location Guide is the most used piece of service information during repairs.

various plugs are to check for visual indications of trouble. Fortunately, these few general checks will allow you to complete a large percentage of repairs.

When the simple measures fail and the repair does require the use of the schematic, the value of the location guide does not end. When you read the schematic, you are seeing symbols that represent the physical chips and other components. Yet the symbols and the actual components bear no resemblance to each other. You must be able to relate the symbols to the electronic parts on the print board. You'll find that the familiarity you gain with the location guide helps bridge the step of finding a part on the board after reading it on the schematic.

## CHIP SURVEY

To get you even more familiar with the circuit board, I want to introduce some of the major chips that you have drawn on your chip location guide. I will locate each chip with the coordinate system used in Fig. 3-1.

### Microprocessor

The microprocessor is located at 3 1/2 across and 1 1/2 down. The 6510 is a variation of the 6502, which has been one of the most used microprocessors in recent years. The important difference between the 6510 and its predecessor the 6502 is an extra I/O register in the 6510 that the 6502 does not have. The details of the additional register is discussed in Chapters 12 and 13. The register provides a tricky way to handle all the memory in the 64.

The MPU has the job of addressing all the residents of the memory map all around the board. It sends and receives data from the memory. It processes the data with the aid of its internal registers, its arithmetic and logic center, and its control section. The registers are temporary memory locations to store results in progress, incoming instructions, and memory map addresses that will have to be contacted during the data processing.

The arithmetic and logic center does the data processing. It is known as the ALU (Arithmetic

Logic Unit), and it performs all the calculations. It also selects, sorts, and compares the information according to the instructions it receives. The control section is a traffic cop in the MPU that keeps data traffic moving in the correct direction at the proper time.

### Complex Interface Adapters

The two 40-pin 6526 Complex Interface Adapters, or CIAs, are located at 1 1/2 × 1 1/2. The one closest to the left side of the board (CIA1) is used by the keyboard to enter data. The plug that the keyboard cable plugs into is next to the CIA, and you can see the copper etch tracks from the plug to the CIA pins.

The keyboard is an arrangement of rows and columns that intersect. Everytime you hit a key a row is shorted to a column as Fig. 3-2 illustrates. There are eight rows and eight columns. This provides 64 ways that the rows and columns can short. Each short is transformed into a code for the character or symbol on the key. There is more detail about this setup in Chapter 16. The CIAs also provide entry or exit of signals to or from peripherals like the joysticks, user port, cassette, disk, printer, and other items that are interfaced to the 64.

These CIAs also have other important abilities that are discussed in Chapter 16. They deal with shift registers and internal timing circuits, including a time-of-day clock. In addition, the CIAs are able to produce interrupts which enable the MPU to service different circuits.

### Video Interface Chip

Under the metal shield at coordinates 6 × 3 is the 40-pin 6567 Video Interface Chip. The VIC chip is the interface between the digital circuits and the analog video output circuits. The VIC produces many types of outputs that include variations of alphanumeric, semigraphics and pure graphics. One of the features of the VIC is its contribution in the creation of "sprites." A sprite is a high resolution programmable figure that can be put into a graphic display. It is used in sophisticated graphic

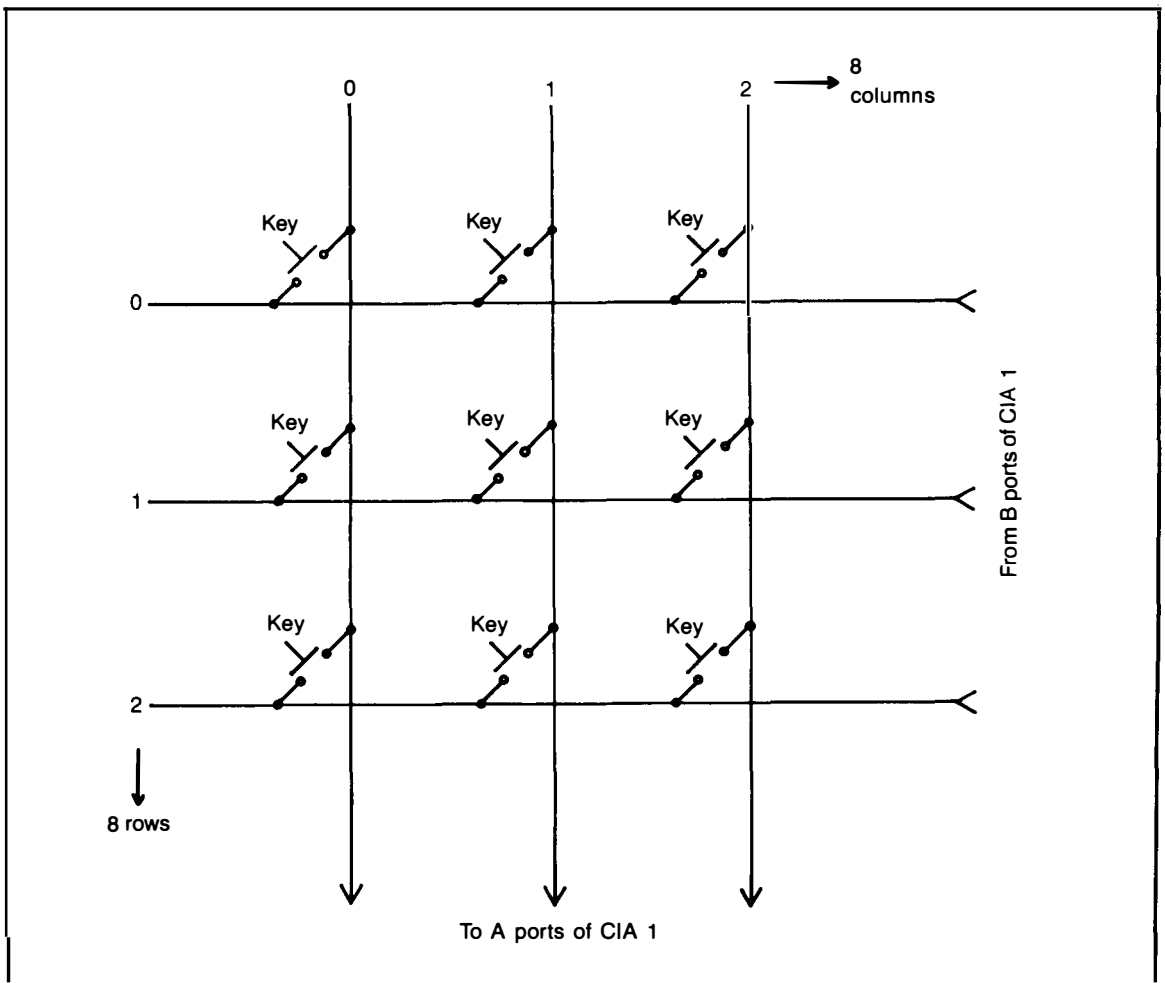


Fig. 3-2. The keyboard matrix is an arrangement of rows and columns. A row is shorted to a column when a key is pressed.

programs. A sprite can be formed in all types of conceivable shapes and be made to cavort freely around the TV display.

The VIC chip can act somewhat like an MPU. It has some addressing capabilities and other MPU features. The VIC chip is able, under some circumstances, to turn off the 6510 MPU and take over the operation of the computer during video processing. When the VIC chip, acting as copilot, is in charge, it conducts the addressing and control functions that the MPU normally performs. The details on the operation and servicing of the VIC chip is discussed in Chapter 17.

### Sound Interface Device

The 6581 Sound Interface Device, or SID is located at  $5 \times 3$ . The SID is also a device that takes a digital input and converts it to an analog output. The output is a fairly decent sounding audio. The chip is a 28-pin DIP (dual inline package), and it is quite complex. The details are covered in Chapter 18.

The SID is referred to as a three voice electronic music synthesizer. It is used to generate interesting and exciting sound effects to go with the graphics created by means of the sprites and other video creations. The SID has a wide range of fre-



quencies that it can produce. There is a high resolution control of the frequencies, harmonics, and volume.

There are four different types of waveforms that the SID can produce. They are triangular, sawtooth, rectangular, and white noise. The frequency of each waveform can be individually varied. The sound can then be fed into an envelope shaper circuit. The sustain level, attack, decay, and release rates are then arranged. With these variables you can get the 64 to sound like different musical instruments and other noises. For further information, turn to Chapter 18.

Besides being a digital-to-analog converter, the SID has inputs that can change analog inputs to digital signals. One example is the inputs of two paddles can be interfaced to the SID for use during games.

## Random Access Memory

Eight dynamic RAM chips, each with a 4164-2 number are arranged in two lines centered at coordinates  $2 \times 3 \frac{1}{2}$ . They are all 16-pin DIPs. Figure 3-3 provides a closeup of these chips. A more precise name for RAM is read/write memory. The RAM is the warehouse for the MPU. It is able to store data and transfer data back to the MPU when it is addressed. Chapter 6 goes into the actual mechanism of data transfer.

There are 64K memory locations in the Commodore 64. There is one byte consisting of eight bits in each location. The bits in a byte are numbered 7, 6, 5, 4, 3, 2, 1, and 0. There are eight chips in the RAM set. The chips in Fig. 3-3 are numbered 7, 6, 5, 4, 3, 2, 1, and 0. Each chip contains 64K bits. Chip number 7 contains all the bit number 7s. Chip number 6 contains all the bit

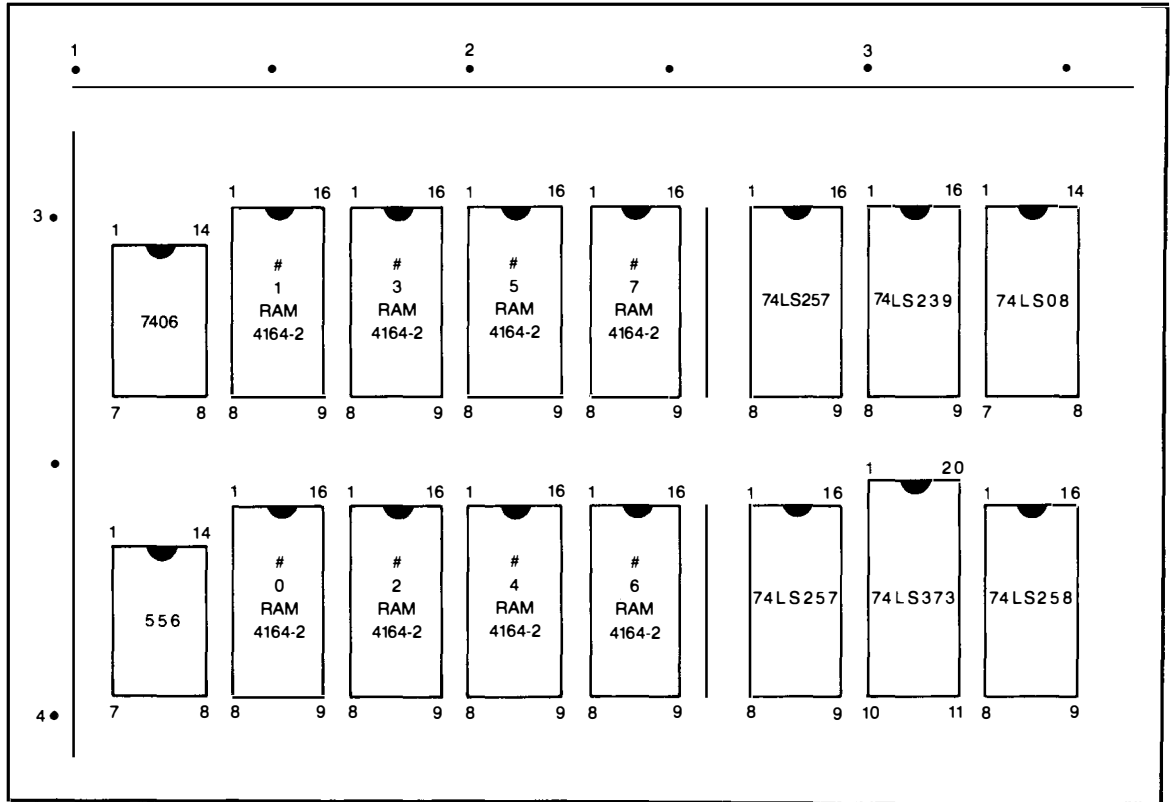


Fig. 3-3. The most congested chip area on the board is around the dynamic RAM and support chips. This is a closeup of that area.

number 6s, and so on. When the MPU addresses a RAM location it is actually addressing all eight chips at the same time. It gets one bit of a location from each chip.

The dynamic RAM stores its bits as charges or noncharges in a grid of tiny capacitances. The charge in the capacitance can only stay for a few milliseconds. It must be recharged every few milliseconds or it will leak off. The VIC chip has a special circuit attached to the RAM chips that constantly refreshes the charge so that it will stay as long as the computer is turned on.

## **BASIC Read Only Memory**

Located at  $2\ 1/4 \times 2$  is the BASIC ROM chip. It is a 2364A contained in a 24-pin DIP. All of the BASIC language is stored in memory locations on the chip. The chip is an interpreter for the BASIC commands. It is packed full of various routines that will cause the computer to fulfill the instructions in a BASIC program.

When a BASIC program is written and installed into a computer, it is passed through the BASIC ROM. The ROM translates a BASIC instruction into its comparable machine language. One BASIC command could cause literally dozens of machine language instructions to be executed. There is a machine language in the BASIC ROM for every single BASIC command that the Commodore 64 is able to handle.

The ROM is unlike the RAM in that it is a Read Only Memory. The ROM cannot store data that the MPU might send its way. It can only send data on to the MPU when it is addressed. There is more detail on the BASIC ROM in Chapter 7.

## **Kernal Read Only Memory**

At guide location  $2\ 1/2 \times 2$ , the next chip to the right of the BASIC ROM is the Kernal ROM, another 2364A chip. The Kernal is the operating system of the Commodore 64. When you turn on the 64, the Kernal takes over and controls all the input, output, and memory management of the computer.

The Kernal works with the BASIC ROM. Most

of the work a computer does is transfer data from the MPU to memory and back. Another vital job that must be done is verify the validity of the data that travels from place to place. The operating system has special load, store, and verify routines that the BASIC routines call upon to conduct the business of computing.

The Kernal is made with an eye to the future. As time goes by, the machine language of the 64 could and probably will be improved and upgraded. There is a special jump table in the Kernal that is designed to accommodate these changes. That way, the machine language routines that you develop will still work on the newer model operating systems that will inevitably replace present systems. There is a lot more about the Kernal in Chapter 7.

## **Character Read Only Memory**

In the same row of chips, to the right of the Kernal is the Character ROM. It is a 24-pin 2332A read only memory chip. It is an amazing little storage area. It contains all the shapes of the characters that are displayed on the TV screen.

Each character can be depicted on the screen with eight bytes of memory as shown in Fig. 3-4. The Commodore 64 has 512 characters to display. They are all in the Character ROM. If you multiply 512 characters times eight bytes, that means there must be 4096 bytes of storage in the Character ROM. There is. It is a 4K memory chip.

In a set of eight bytes, there is a pattern. The different patterns form all the letters, numbers, and symbols that you place on the screen. During machine language programming you must pay particular attention to the details that are needed to work with the Character ROM. When you are using BASIC, the BASIC ROM and Kernal handle the Character ROM.

## **Color Random Access Memory**

At location  $4 \times 3\ 1/2$  is the Color RAM. The 18-pin DIP is a static RAM chip in contrast to the dynamic 64K RAM that is used to store normal programs, video, and data. The Color RAM is used to

Memory map addresses	Bit positions								
	7	6	5	4	3	2	1	0	
	53255	L	L	L	H	H	L	L	L
	53256	L	L	H	H	H	H	L	L
	53257	L	H	H	L	L	H	H	L
	53258	L	H	H	H	H	H	H	L
	53259	L	H	H	L	L	H	H	L
	53260	L	H	H	L	L	H	H	L
	53261	L	H	H	L	L	H	H	L
	53262	L	H	H	L	L	H	H	L
53263	L	H	H	L	L	H	H	L	
53264									

Fig. 3-4. Character patterns are stored in a matrix of eight bytes. With eight row locations and eight column bits, any character can be formed by judicious placing of the highs and lows.

store the position of a color that a character on the screen is using. This static chip is needed since the dynamic RAM must hold so much information that it doesn't have any more convenient storage to contain the color position information.

The chip is a 2114-30L. It is organized in a  $1024 \times 4$  bit arrangement. This makes each location in the chip only four bits wide, a nybble, rather than a byte. Figure 3-5 contrasts ROM and this RAM. The color of a position on the TV face can be contained in a nybble.

The color RAM chip is in the memory map and is addressed by the CPU just as any other map resident is. The difference in this chip is that it also outputs to and receives data from the VIC chip. It works as a helper to the VIC chip. It holds the information of the color of each character position on the screen. There will be a lot more detail about

this RAM in Chapter 17.

## Other Board Landmarks

There are more circuit components that will be discussed in their pertinent chapters. One of these is the chip at  $5 \times 1 \frac{1}{2}$ . It is the Programmable Logic Array (PLA) that helps in selecting the chip that the MUP is to read from.

The rf modulator is located at  $6 \times 1$ . It places the composite color TV signal that the VIC chip creates into a channel 3 or 4 signal that your TV can use.

In the rear and on the right side of the print board are the ports that interface with the peripherals that are designed to work with the Commodore 64. In clockwise fashion they are the user port, the cassette port, the serial port, the audio vid-

eo plug, the cartridge port, and the control ports. These are discussed in Chapter 19. The ports are very important test points during troubleshooting.

### SOCKETS

Whether a radio, TV, or computer finds its way to a servicing bench, in the final analysis the trouble will turn out to be a defective component or connection. While the diagnosis in each can be different, radios have mostly sound trouble, TVs have picture problems, and computers have digital information complications. The root of the trouble in each case, however, is electronics. Basic repair techniques have a lot in common, regardless of what you work on.

One of the most common and important components that servicers look for are sockets. Vacuum tubes almost always use tube sockets. It is rare to find a tube that is wired into a circuit instead of being plugged in. The only tube I can recall that was wired in was an old miniature high voltage rectifier.

As a result of the almost universal use of tube sockets and the fact that most TV troubles were

caused by bad tubes, a large percentage of TV repairs could be performed by any mechanically inclined person. Technical training was not required to change a dead tube.

When transistors arrived on the repair scene, they were found in two ways. Some manufacturers used a lot of transistor sockets in their units while others simply saved money by soldering the solid-state devices in place on the print board. There are advantages and disadvantages to both methods. It is claimed that a soldered in transistor is much more reliable than a socketed one. Transistors are built with skinny, flexible leads that lend themselves to be soldered into place rather than poked into a tiny socket. In addition, unlike a tube that has a keyway to prevent plugging it in wrong, transistors have no such keyway. They can be inserted incorrectly, which could be disastrous.

On the other hand, during servicing, if transistors are wired in, a repair hardship occurs. First of all, the luxury of the quick direct replacement transistor test is not available. If a transistor is a suspect it must be tested in-circuit or desoldered

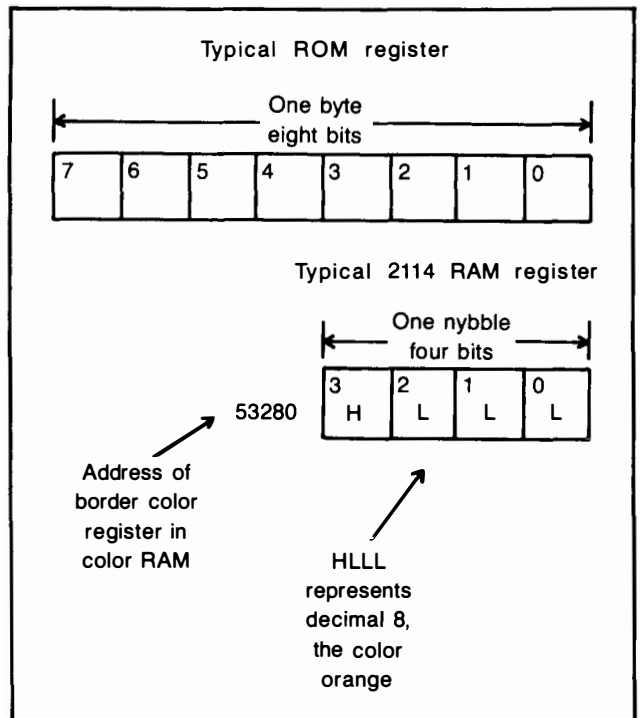


Fig. 3-5. The data registers in the memory are all one byte wide except for the color RAM, which is one nybble wide.

with a new one resoldered in place. Wholesale replacement, which often quickly effects a repair, is out. To add to the problem, a lot of soldering can induce additional problems that complicate an already poor situation.

The servicer sees the transistor socket as a very valuable test point. The transistor can be removed and all types of signal injection, signal tracing, and voltage and resistance tests can be made. When the transistor is wired in, to make the same tests requires desoldering the transistor with all its attendant difficulties.

Chips, from a socket point of view, appear somewhere between tubes and transistors. Chips are miniscule and not as easy to handle as a tube. However, chips have sturdy little feet and can be pulled out of and inserted into the tiny socket holes freely if you use the proper techniques.

The chip package is designed to mate with a socket. The reliability of a chip in a socket is almost as good as a chip that is wired onto a print board. If a manufacturer produces a microcomputer with the large chips soldered into place instead of socketed, its usually because he wants to save the price of the socket. He can't be faulted too much, because in production the savings of the cost of 10 or 20 sockets can come to a lot of money, and we are all rooting for him to keep the price of computers low.

The rule of thumb among designers concerning chip sockets has been the following. During design and breadboarding, while technicians are hand wiring a new board, sockets are used for practically every chip, large or small. During the experimental stages, techs are constantly removing and reinstalling all the components, including the chips. Once the project is finished and the board goes into production, the sockets are dispensed with, except perhaps for the 40-pin or larger packages, and the board ends up with mostly soldered in chips.

As a servicer, I would naturally like to enjoy the benefits the design breadboarder has with his board full of sockets. As a computer user though, I am also interested in buying my equipment at the lowest possible price. The manufacturers try to

make everyone happy. As a result, you'll find some computers replete with sockets and others with practically none.

Chips operate with very little electric current and have proved to be quite rugged. They do not seem to fail as often as tubes and transistors do. This fact tends to lessen the need for sockets. However, if a 40-pin or even a 24-pin package that is soldered in should fail, you have a considerable replacement job on your hands. That is not all. In addition, the testing of the chip in-circuit is quite a chore too. The socket question is knotty, but as time goes by, and millions upon millions of computers are installed in homes, I'd say more and more chip sockets will be used to make servicing easier and consumer repair bills tolerable.

In the Commodore 64 that I have sitting open in front of me, I see a number of sockets. The four 40-pin packages—the CPU, the two CIAs, and the VIC chip—are all socketed. Among the smaller chips, the SID, the Kernal, and the BASIC ROM are also in sockets. The RAM, including the eight dynamic and the single static Color RAM are not socketed. They are soldered right into the board. None of the other chips are socketed.

Handling the socketed chips and desoldering and resoldering unsocketed chips are techniques that require considerable briefing. Locating the bad chip is perhaps the easiest part of a repair job. The next chapter goes into the right way to do the replacement job.

## **USING THE LOCATION GUIDE**

Let's see how the guide works on a typical easy repair on the Commodore 64. Suppose you turn on your computer one warm summer day and it is operating fine. It continues working well for about a half hour. Then, all of a sudden, as you enter a program line, the screen quickly fills up with garbage. To make matters worse, as you try to straighten out the literal confusion, the machine will not respond to any of your key strikes. You have trouble.

As you analyze the condition you know that garbage is usually due to failure in the digital world area. This includes the circuits of the 6510 MPU,

the 6526 CIAs, the 6567 VIC, the PLA, the ROMS, the support chips and maybe one of the RAM chips.

The experienced tech would also pick out another symptom that is not so obvious. The trouble did not begin till the 64 was up and running for a half hour in the warmth of the day. This indicates that one of the chips or components is heat sensitive. It breaks down as it heats up. This heat breakdown lends itself to some heat testing techniques. The schematic of the 64 won't be needed, the location guide should be the only service sheet needed.

There is a quick freeze aerosol liquid that can be bought in a can at any electronic supply house. If you have the print board exposed you can, through a tube that comes with the can, spray the suspect circuit components, one at a time, while the trouble is happening. If you spray a chip that is failing due to heat prostration, and the trouble suddenly disappears, chances are you have pinpointed the

defective chip.

You must be careful where you spray the quick freeze liquid. You only want to place a fine spray on one component at a time and then observe the results. That way you will pinpoint the bad one. You also only want to spray prime suspects, not every part of the board. That is where the location guide is invaluable. It permits you to find the suspects and conduct the test.

In this particular trouble, taken from an actual repair, the condition disappeared when the CIA on the end of the print was quick frozen. The trouble reappeared in about a half hour as the machine was left on. A new CIA chip cured the problem permanently.

This is a typical example of the way that the location guide is used as servicing information during a repair. You'll find that a good guide will be used much more often than the schematic.



## 4. Chip Changing Techniques

**T**HE GERM SIZE TRANSISTORS IN THE CHIPS on the print board are both rugged and reliable as well as sensitive and fragile. While the chip is in its socket or soldered onto the board, it has a very high reliability rating. It is when the chip is not connected to the board that it loses its protective environment and becomes vulnerable.

The danger signs go up as a repair begins. During a normal bench repair there can be a lot of tests, desoldering, resoldering, handling, lead pulling, exposure to voltages, static electricity, dust, humidity, freezing, heating, and other life threatening experiences that the miniscule transistors must go through. Even though a chip is perfect at the beginning of a repair, there is always the possibility of it not being quite so perfect at the end.

Some types of chips are exceptionally sensitive to heat, voltage, and static electricity. Other forms of chips are not overly disturbed by these forces. In the 64, there are lots of these easily hurt chips and a few tough types that can withstand some punishment. As we discuss the various chips in

Chapters 5, 6, 7 and 8, you'll be able to tell which are which.

Even though some chips are more rugged than others, it is a good idea to treat them all with great care. This chapter covers the general techniques needed to handle the various forms of chips as they are manipulated in and out of sockets or desoldered and resoldered. With the correct techniques and tools, you will be able to change and test chips with as much safety as possible.

### TTL, DTL, AND RTL

The rugged type chips you'll find in the 64 are called TTLs. This stands for Transistor-Transistor-Logic. The predecessors of the TTLs were the DTLs, Diode-Transistor-Logic, and the RTLs, Resistor-Transistor-Logic. You won't find any DTLs or RTLs in computers anymore. They have been completely replaced by the more efficient TTLs.

The TTL is known as a bipolar integrated circuit. The bipolar name is used since the chip is built

with bipolar transistors. Without going into a long dissertation about bipolar transistors, let us say that the bipolar transistor has three connections called the emitter, base, and collector, e, b and c. Figure 4-1 shows a typical TTL gate with these connections marked. They are called bipolar because conduction takes place in two directions at the same time. The electrons travel in one direction between emitter and collector, while holes travel in the other direction. Figure 9-2 gives a graphic explanation. This type of conduction is in contrast to the Field Effect Transistor, or FET, discussed next. It has either electrons or holes on the move, but not both at the same time. If you desire more details on the bipolar transistor theory, please look it up in the many books available from TAB Books.

The TTL, DTL, and RTL all use bipolar transistors. While you won't see any DTL or RTL types in the 64, it's a good idea to review them since they were the predecessors of the TTL. The RTL was among the first forms of integrated circuits. It was

an inexpensive chip that was easily connected to larger discrete components like resistors and capacitors. However the RTL was highly susceptible to voltage noises and it had low fanout ability. *Fanout* is the characteristic a chip has to drive a number of parallel loads. The RTL can only drive a few loads. On the other hand, a TTL can drive a large number of loads which gives it a better fanout characteristic.

The RTL gate in Fig. 4-3 uses two resistors in the base inputs of two npn bipolar transistors. That is why it is called a Resistor-Transistor-Logic chip. This particular configuration is called a NOR gate. There will be more about NOR gates in Chapter 10.

The DTL gate in Fig. 4-4 uses diodes in the input circuit rather than resistors. This change of component makes the gate faster, gives better noise immunity protection due to diode clipping, and increases the fanout characteristic. In addition, the DTL permits a large fan-in. *Fan-in* is the ability of the chip to accept parallel inputs. The diodes are

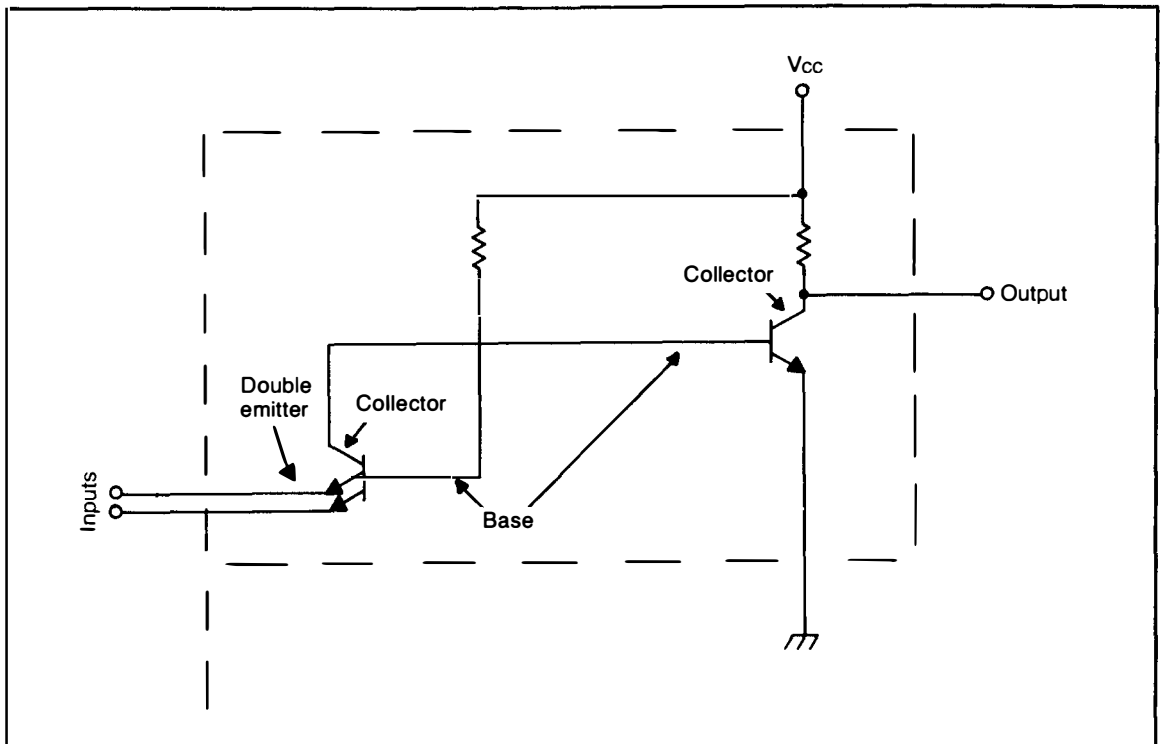


Fig. 4-1. The TTL (Transistor-Transistor-Logic) chips are based around the transistors that are endowed with double emitters.



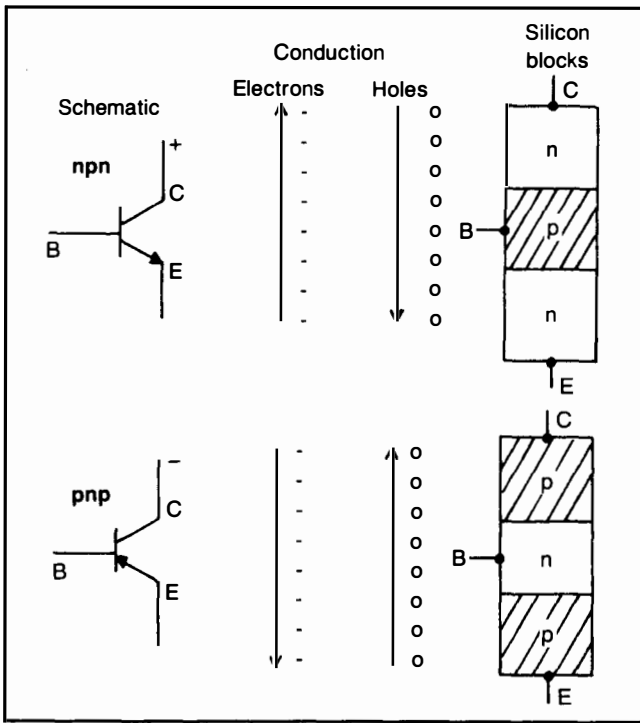


Fig. 4-2. Bipolar transistors have electrons traveling in one direction at the same time holes are moving in the other direction.

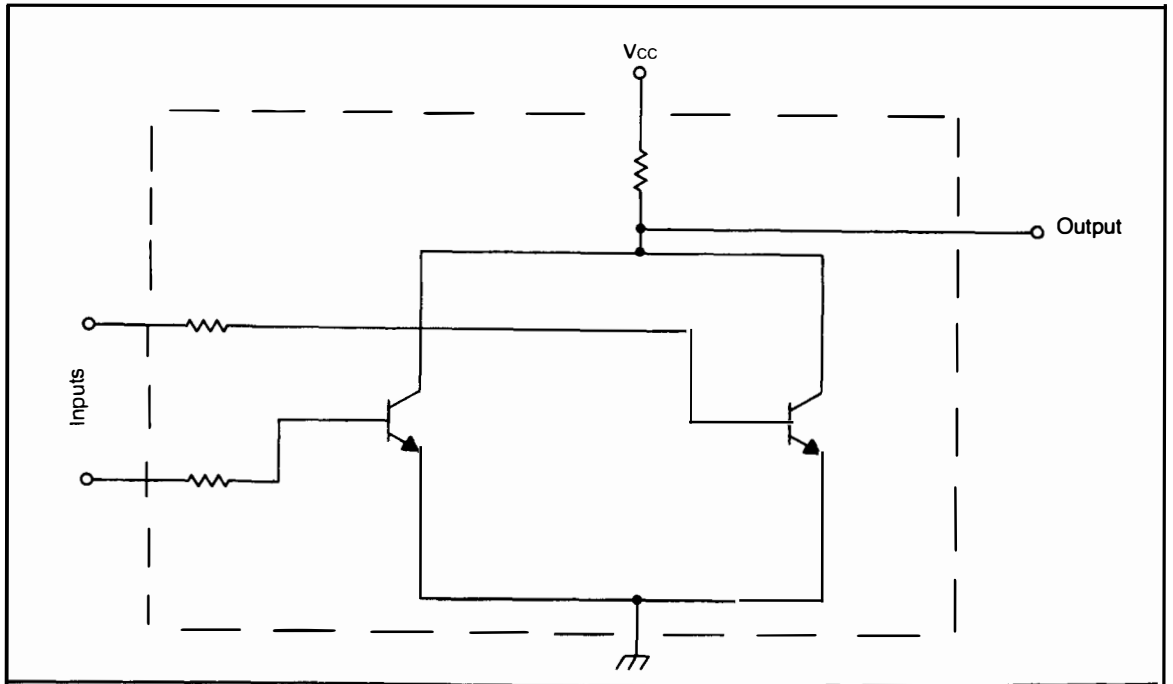


Fig. 4-3. The DTL (Diode-Transistor-Logic) chips are so called because the inputs are made through diodes.

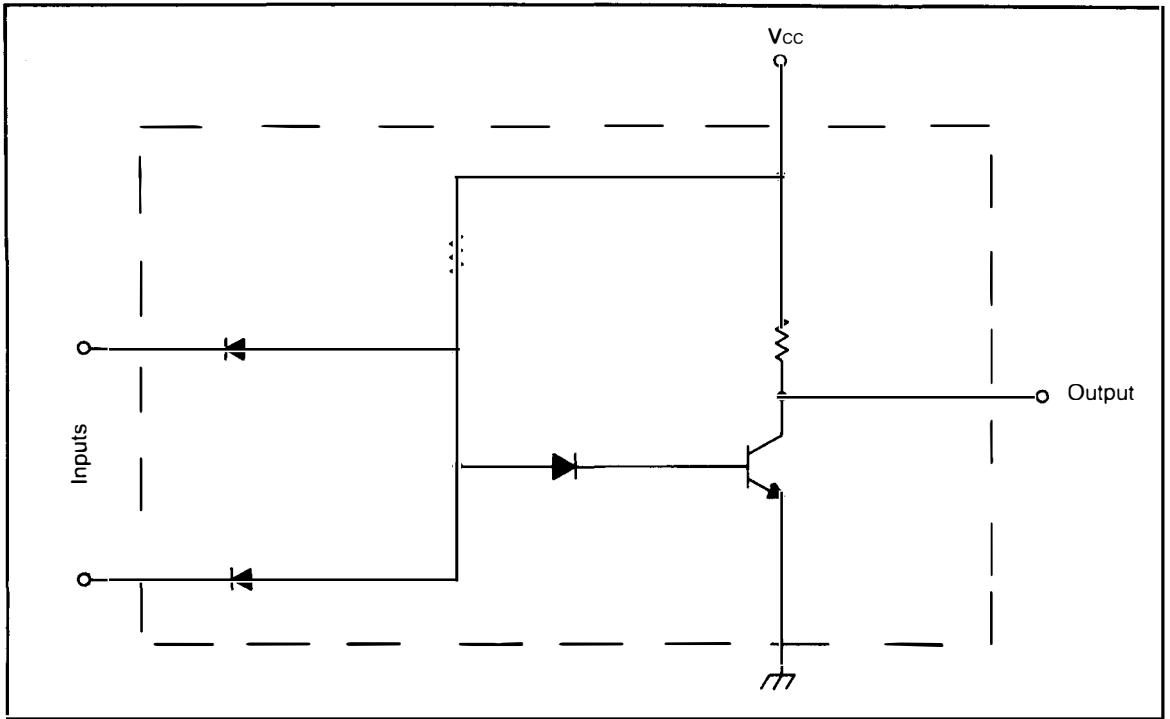


Fig. 4-4. The RTL (Resistor-Transistor-Logic) get their name from the input resistors that are used.

able to isolate the gate input circuits from the preceding stages and many parallel inputs can be connected without loading the input. The basic DTL circuit shown is called a NAND gate. There is more about NAND gates in Chapter 10.

The TTL evolved from these two basic circuits. In 1961, Thompson invented the TTL. It is like the DTL in that there is diode action in the input. The input is through the double emitters of the transistor. There are a number of emitters in the TTL in Fig. 4-1. The extra emitters are pn junctions, just like diodes. All the inputs to the chip can enter through the multiple emitters and be diode isolated in the same way that the DTLs are isolated. The circuit operation is very similar to the DTL.

There is a whole family of TTL chips. At this writing, they are approaching 200 in number. They have been given the generic series part numbers beginning with 7400. Typical TTLs in the 64 are 7406, 74LS08, 74LS74, 74LS239, and so on. All the TTLs will be discussed in various sections of the

book. While the numbers in the 7400 series indicates the chip is a TTL, the letters in the numbering, such as L and S have special meanings. The L stands for low power. When there is an L in the chip number, it means the chip uses 80% less power than a chip without the L designation. However, the lower power dissipation is at the expense of slower switching speed.

That is why there is usually an S accompanying the L in the nomenclature. The S stands for Schottky clamped diode. When the S is in the name, there is a Schottky barrier diode clamp in the base circuit that speeds up the switching action. It compensates for the slowdown caused by the low power characteristic. Therefore, if you are called upon to replace a 74LS type chip, use another 74LS and not a plain 74 type. While the two chips are functionally about the same, the exact replacement is always the best way to go. If you have no choice but to make a change, just be aware of it in case some other trouble symptom should suddenly appear. In

some cases, the substituted chip will not work properly.

### THREE-STATE TTL

There is another ability that is often built into TTLs. It is called TRI-STATE, three-state, or the third state. What is this third state? By this time, you should know that computers are in the business of processing two logical states. The states are known by such names as 1 and 0, high and low, off or on, true or false, +5 volts and zero volts, set and clear, set and reset, and other names. Whatever the names, they are only descriptions of the two logical states the computer goes to work on. Then there is the third state.

The *third state* is sometimes confused with the zero volt logical state. The third state condition, though, has no definable output. It is the condition

that the output of the chip exhibits when it is disconnected from the circuit. The TTL output gets shut off and assumes a high impedance condition. In contrast, the zero voltage state is a connected state and the chip output shows a voltage, only thing is the connected voltage is zero.

When a chip is three-stating, there is no definable output voltage. As a matter of fact, if you do happen to take a voltage reading with a vom, there will be a voltage present, around 2 volts, but it is a result of accumulated static electricity not a logical high or low. There will be more about testing the TTL chips in Chapters 10 and 11.

In the TTL with the three-state ability, there is a special input stage that is able to disable the TTL gate circuit upon command. Figure 4-5 shows a chip with the disable stage. There are three npn transistors and a diode in the disable stage of Fig. 4-5. They are Q2, Q3, Q5, and D1. When the disable

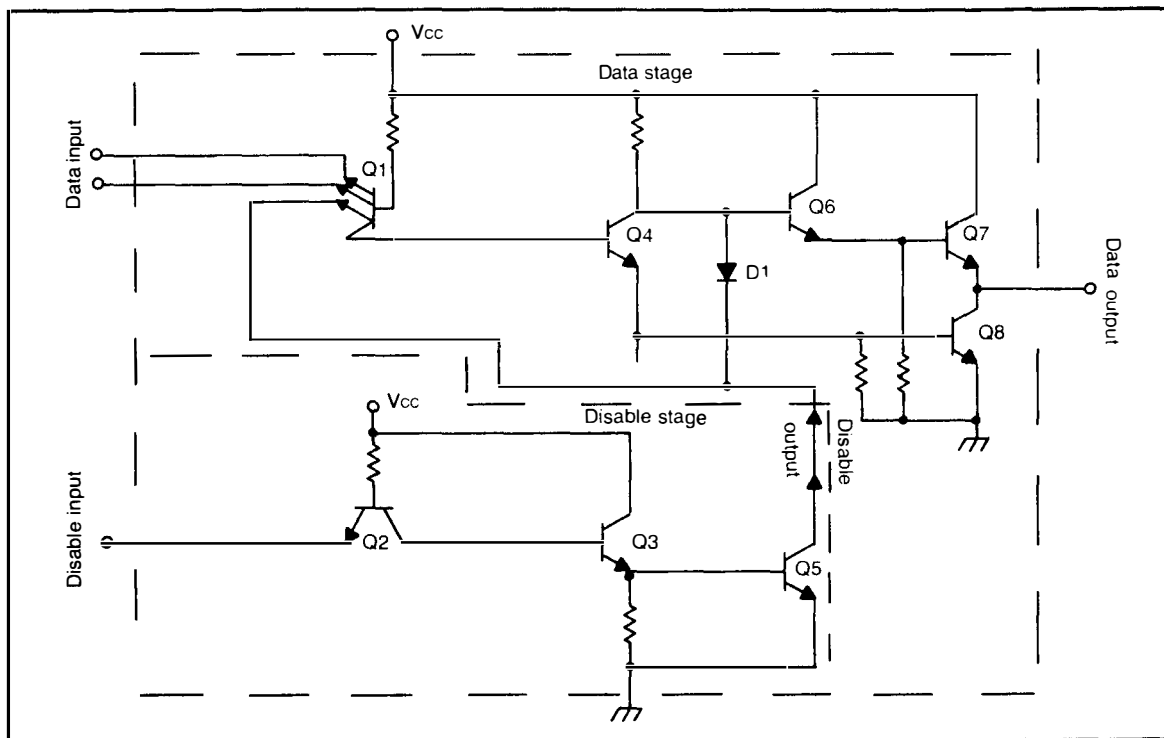
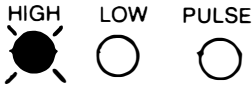




Fig. 4-5. This three-state NAND gate has two stages. The top stage is normal NAND gate and will process input H's and L's accordingly. The bottom stage is the disable. It acts as a switch for the NAND gate. It either lets the gate process normally or it shuts off the gate and puts it into a high impedance state.

**Table 4-1. Logic State Indications for a Vom and Logic Probe**

LOGIC STATE	VOM READING	LOGIC PROBE LED LIGHTS
HIGH	2.3V to 5.0V	
LOW	0V to 0.8V	
THREE-STATE FLOATING	0.9V to 2.2V	

input is low, Q2 will turn on fully and, as it is called, saturate. As a result of Q2 saturating, Q3 and Q5 will turn off. This turns off the output of the disable stage and the stage has no effect on the rest of the chip. The data stage of the gate continues to operate as if there were no three-state circuit in the chip. The normal processing of 1s and 0s continues unabated.

Should the input to the disable stage go high, Q3 and Q5 will conduct and kill the output current at Q4. This makes the output transistors of the gate, Q7 and Q8 stop conducting. Should you measure the voltage at the output, you'll find neither a defined high or low. The only voltage there will be undefined, somewhere between a high and a low.

During bench type troubleshooting, you can test the output of a three-state type chip at its output. You can also produce a defined output or a three-state output by injecting a high at the disable input to effect the logic condition or a low input to get the undefined performance.

The three-state effect can be used as a valuable servicing technique. Sometimes when a chip dies, it produces a three-state condition at its output. Other times, a three-state condition is present during normal operation. You can use the reading you obtain to figure out whether the condition is normal

or indicates trouble.

The TTL is easily tested with either a vom or a logic probe. The only difficulty is the tinyness of the chip feet. The feet are test points and it behooves you to have a bright light and maybe a magnifying glass on the light. You should take care as you touch down on each test point.

The vom, on a TTL, will normally reveal a logical 1, or a high, by reading a voltage between 2.3 volts and 5 volts. The logical 0, or low, will display itself on a vom by reading a voltage between 0 volts and 0.8 volts. During a three-state condition, the vom will read somewhere in between, from 0.9 volts to 2.2 volts. The logic probe has LED lights to show the logic conditions. The high lights the HIGH light and the low makes the LOW light shine. When the condition is a three-state one, the logic probe does not light up at all. Refer to Table 4-1 for a summary of the logic probe indications.

Most of the bench readings that is performed during servicing is either with the vom or the logic probe. The servicing consists of a search and seek expedition of examining the tiny test points.

In the 64, there are a lot of TTL chips. They are all the smaller support chips and are for the most part soldered onto the board without the benefit of having their own sockets. They are fairly rug-

ged and can take some careful soldering if need be. They are easily and safely tested with either the vom or the logic probe.

## MOS CHIPS

While the TTL chips are plentiful and are used extensively, in the past ten years or so, they have been receiving competition from the Metal Oxide Silicon chips known as MOS. There are three types of MOS chips. There is the NMOS that is based around pieces of n-type silicon, and the PMOS that has p-type silicon and the CMOS, (Complementary MOS), which has both n- and p-type silicon channels for electrons or holes to pass through.

When you look at a chip there is no quick way to tell whether it is a TTL or one of the MOS types. They are, for the most part, contained in what is known as a DIP. The term DIP stands for Dual-Inline-Package. The tiny wafer of silicon is wired into the familiar rectangular package and the leads emerge from the package out of both long sides of the rectangle.

While the basic transistors in the TTLs are bipolar transistors, either npn or pnp, the transistor in the MOS is the FET. Where the TTL elements are the emitter, base, and collector, the FET uses a source, gate, and drain. The main thing that the TTL chips and the MOS chips have in common is that they are both made out of pieces of p- and n-type semiconductor material.

The bipolar transistors in the TTL chip are constructed with building blocks. There were two types of blocks shown in Fig. 4-2. A block is made of p-material or n-material. A pnp is, as the name suggests, a sandwich. The n-material is in the middle between two blocks of p-material. The npn is also a sandwich with the p-material inside the two pieces of n-material. The top block is the collector, c, the middle block is the base, b, and the bottom block is the emitter, e. The transistor connections are made at c, b and e.

The bipolar transistor can roughly be thought of as an electronic set of gears. If you get a small current to flow between the emitter and base, that will cause a large current to flow between the emit-

ter and collector. This is called a current amplifier. The bipolar transistor deals in current amplification. This is different than the FET that deals in voltage amplifying. We'll get to that in a few paragraphs.

The FET is not made with the same building block layout. The FETs found on ICs are made with a channel. The channel can be thought of as a piece of either n-type or p-type silicon. Study Fig. 4-6. At one end of the channel is the connection called the drain, D. The other end of the channel has the source connection, S. In the middle of the channel is the gate, G. The gate is not connected directly to the channel like the source and drain. A piece of glassy silicon oxide is connected to the channel and the gate is attached to the other side of the oxide. The oxide forms an insulating barrier between the gate and the channel.

This insulation must be maintained. Should something happen to the oxide, such as the static electric short in Fig. 4-7, the insulation will be ruined and the chip would die.

Since there is an insulator between the gate lead and the channel, electric current cannot pass from the gate to the channel. In the bipolar transistor, the base, which is analogous to the FET's gate, is not insulated from the emitter-collector electron path. Therefore current can flow between the base and the emitter-collector path. That is why the bipolar transistor is called a current amplifier. The current from the base controls the emitter-collector current.

In the FET, current can't flow from the gate to the channel, but a voltage on the gate can affect the electrons flowing in the channel. The voltage can vary the number of electrons that flow. For example, if the voltage is high, it can stop the electrons flow. Should the voltage be low it can allow the electrons to flow at full strength. At any rate, the voltage on the gate controls the electron flow in the FET channel between the source and the drain. That is why the FET is called a voltage amplifier.

During troubleshooting and repair, the way the microscopic transistors are performing in the chips is abstract. There is no ordinary way you can test

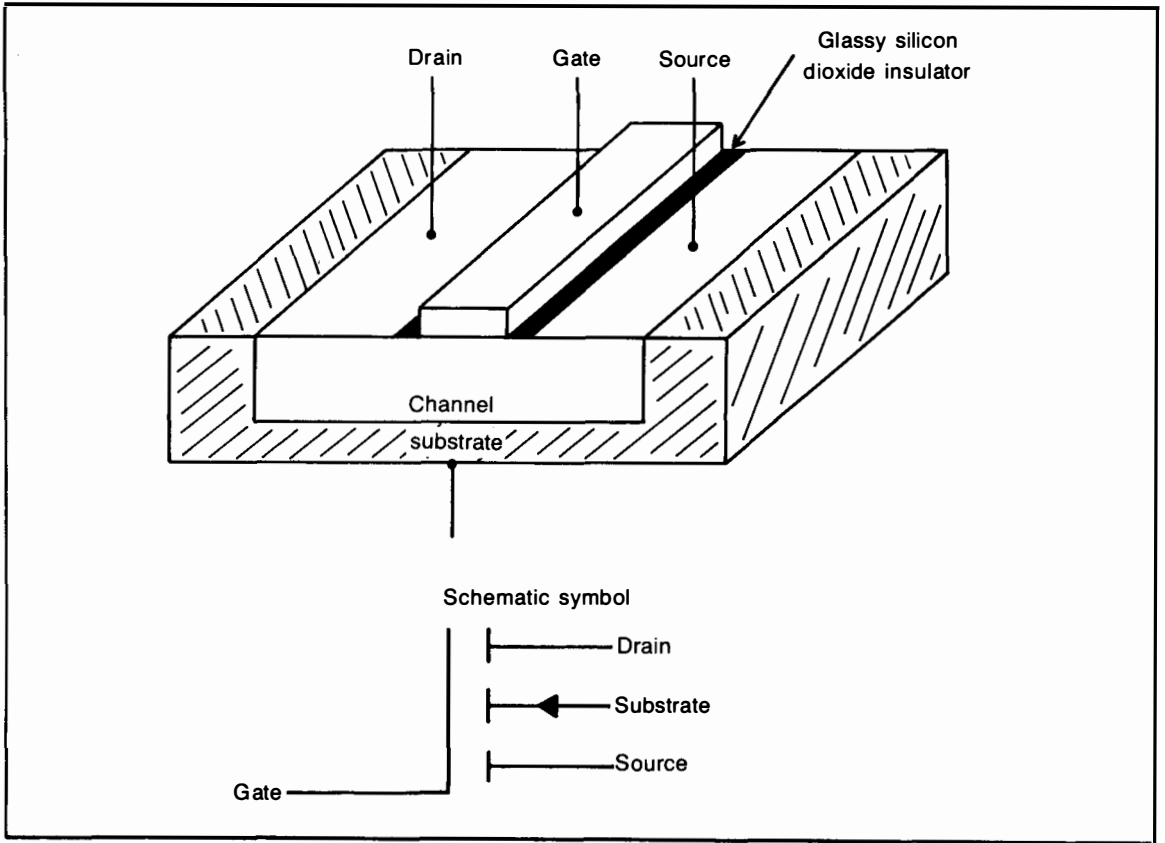


Fig. 4-6. The basic IGFET has four connectable sections. The source, gate, drain, and substrate.

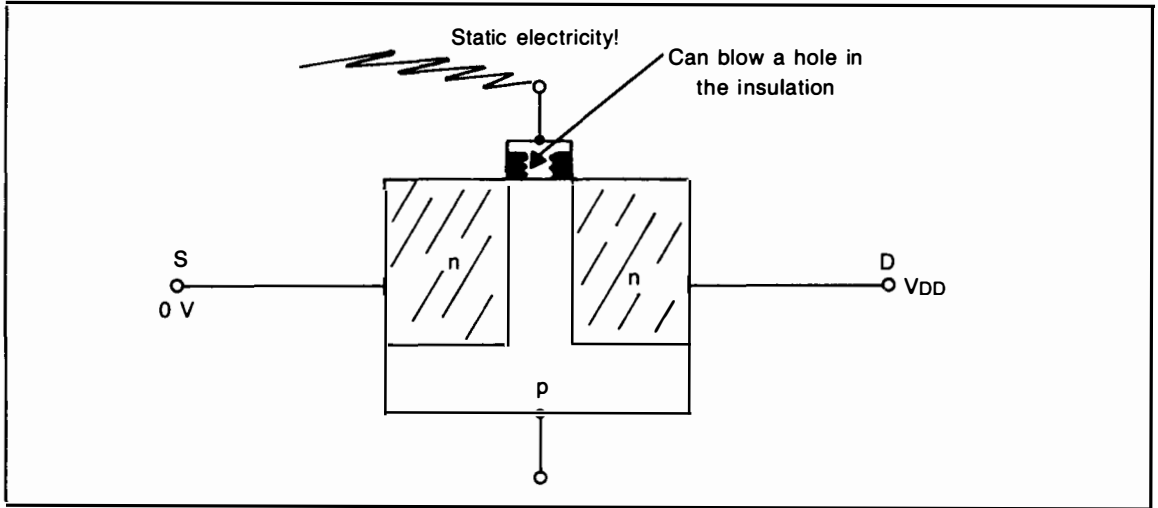


Fig. 4-7. The glassy oxide insulator between the gate and the channel is very fragile and will blow a hole if static electricity is applied there.

the individual TTL or MOS transistors. If you are desirous of learning more about the transistor activity, again, I refer you to other books.

All three types of MOS chips are in common use. The NMOS is used a lot in large scale integration (LSI). The LSI chips are those that have more than 100 individual gates on a chip. It is useful to know that the NMOS chips use a positive dc supply voltage. The NMOS is said to be a single channel, one polarity chip. The dc voltage applied goes to all the FETs on the chip at the same time. The ground return is also connected to every FET on the chip. The gates, sources, and drains though have their own configurations according to the job they are doing on the chip.

When the channel is made of p-material, the FET works in a similar way, except that holes move from source to drain instead of negatively charged electrons as in the n-material channel. In the n-channel, a + voltage is applied to the drain to attract the electrons from the source. In a p-channel, a - voltage is applied to the drain to attract the holes from the source. The gate still does the voltage controlling job except that it changes the intensity of hole conduction rather than electron conduction. You'll recognize a chip with a p-channel because the schematic shows a negative dc supply connected to the chip. Whatever the polarity, whether holes or electrons are on the move, the sensitivity of the MOS chip to gate oxide rupture remains the same. Great care must be taken while testing and handling MOS chips.

The CMOS chip is the most common chip in small scale integration (SSI). SSI is the term for a chip with less than 10 gates to a chip. The CMOS type is also used a lot in medium scale integration, (MSI). MSI chips contain between 10 and 100 gates.

Remember, a gate can contain a number of FETs. The NMOS chips are built with n-material channels. The PMOS chips are made with p-material channels. The CMOS chips are made with both n- and p-channels. Figure 4-8 shows how this is done.

Typically the supply voltage to a CMOS is of a + nature. Internal wiring takes care of applying the correct polarity voltages to the different type channels. The NMOS is thus able to propel electrons from source to drain. The PMOS is able to move holes from source to drain. The insulated gates are able to exercise control over the channel currents no matter whether electrons or holes are on the move.

Just as the TTL chips have been designated numbers in the 7400 or 74LS00 series, the CMOS small package chips are assigned numbers in the 4000 series. An example of a CMOS chip in the 64 is the 4066, a quad bilateral switch. This chip receives more attention later in the book.

## THE DIP PACKAGE

If you take a small magnifying glass and examine the chips on the 64's print board, you will see many neatly arranged DIPs. You know by now that the

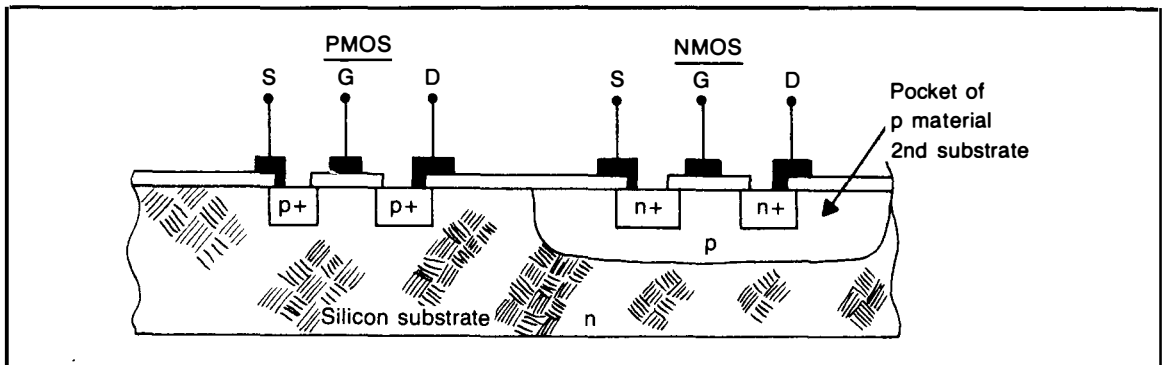


Fig. 4-8. Both NMOS and PMOS IGFETs can be installed on one substrate with this pocket scheme. The result is a CMOS.

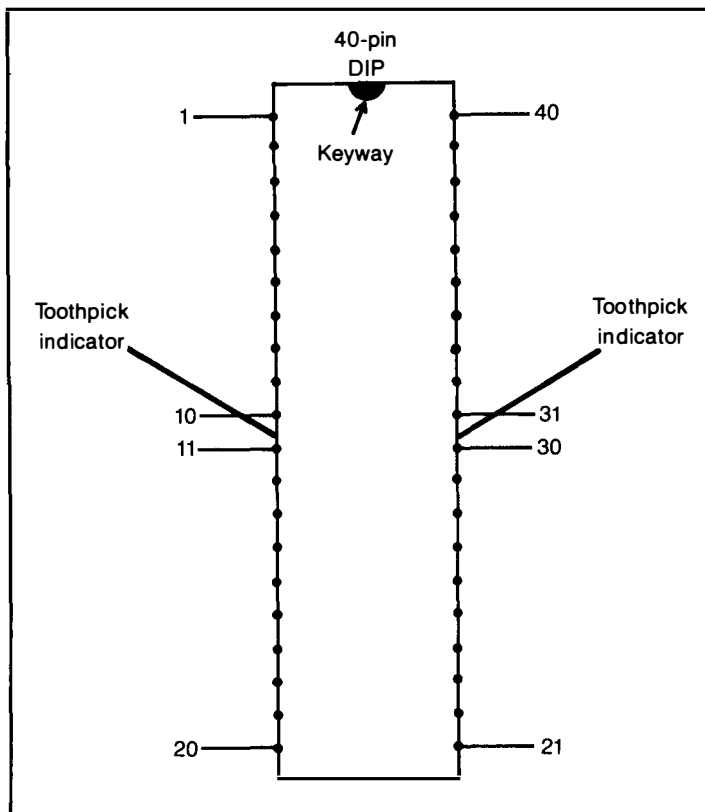


Fig. 4-9. Finding the pins on the large 40-pin DIPs is made easier by inserting toothpick indicators between 10-11 and 31-30.

word DIP stands for Dual In-line Package. On each chip, there are two in-line rows of tiny feet. The top view of the chip is a rectangle with a key designation at one end. The key is either a notch, a paint dot, or an indentation. The key is between pin number 1 and the last pin of the chip. The pins are counted starting with the first pin on the left and counting counterclockwise. The last pin is on the other side of the key across from the first pin.

When you replace a DIP, you must make sure that the key is placed in the same position. The key is only a visual indicator. The chip can be placed into the chip socket or print board holes incorrectly. Be sure not to do that. If you do remove a chip, it might be a good idea to make some sort of mark on the print board to indicate which way the key must point. Recall that I marked the location of the key on the Chip Location Guide in Fig. 3-1. You should do the same.

When you test the feet of a chip, you must be

able to read the numbers of the pins. Tests consist mostly of applying the vom for a voltage, touching down with a logic probe to see if a logic state or pulse is present, and reading the resistance of the pin to ground or to another test point with a low voltage continuity tester.

The fastest way to find a pin is by knowing at a glance how many pins are on a chip. In the 64, there are many size chips. Let's use the 40-pin chip in Fig. 4-9 as an example. Pin 1 is at the upper left of the rectangular top view. Pin 40 is opposite to pin 1. At the bottom left is pin 20. Across from 20 is 21. At the center of the chip on the left side are pins 10 and 11. Across from 10 and 11 are 31 and 30.

With a bit of concentration, you can quickly find all those pins. Once you are oriented, you can locate any other pin using one of these as a reference. If you have any difficulty remembering where 10-11 and 31-30 are, take a couple of



toothpicks and stick them into the gaps between 10-11 and 31-30. That way, if you are making a lot of careful readings you won't accidentally touch down on the wrong pin.

Other chips you might have to handle have different numbers of pins emerging from the package. Some SSI DIPs are packaged with 14 or 16 pins. Other packages have 18, 24, and 28 pins. The largest DIPs in the 64 have 40 pins. No matter the number of pins the general packaging arrangement and numbering are all the same.

Printed on the DIPs can be all sorts of useful servicing and replacement information. Note on Fig. 4-10 the logo of the manufacturer. Near the logo is a code date that is needed to exercise warranties. The code date is placed on the chip in a difficult to read configuration. Often, only the manufacturer is able to decipher it. Look at the illustration for some hints.

Next on the chip are part numbers. First there is the generic part number. This is the standardized industry designation. The generic number allows you sometimes to replace the part with a DIP from a replacement manufacturer rather than the original. As a word of warning, it is always best to

use the original manufacturer's parts, if possible, and not a replacement.

TTLs as mentioned earlier have generic numbers in the 7400 or 74LS00 series. CMOS chips are found in the 4000 series. In the 64, there are other numbers not easily found in generic lists. First of all there is the 6510 MPU. This is a special microprocessor that is used by the Commodore 64. It is part of the 6502 MPU family. This makes it compatible with 6502 type machine-language software, but it has an extra I/O register that is not found on the 6502 MPU. This is discussed in detail in Chapter 12.

Next there are the 6526 CIAs. They are special I/O chips that operate with the 6510 MPU. Then there is the 6567 VIC II chip, the 6581 SID chip, and the RAM and ROM DIPs. They are all discussed in detail in their own chapters.

## SOCKETED CHIPS

The 64 has a lot of the chips in sockets. All the 40 pins chips are in sockets. This includes the 6510, the two 6526s and the VIC chip under the metal shield. Two of the ROMs are also in sockets. The SID chip and the PLA chips are given sockets. This

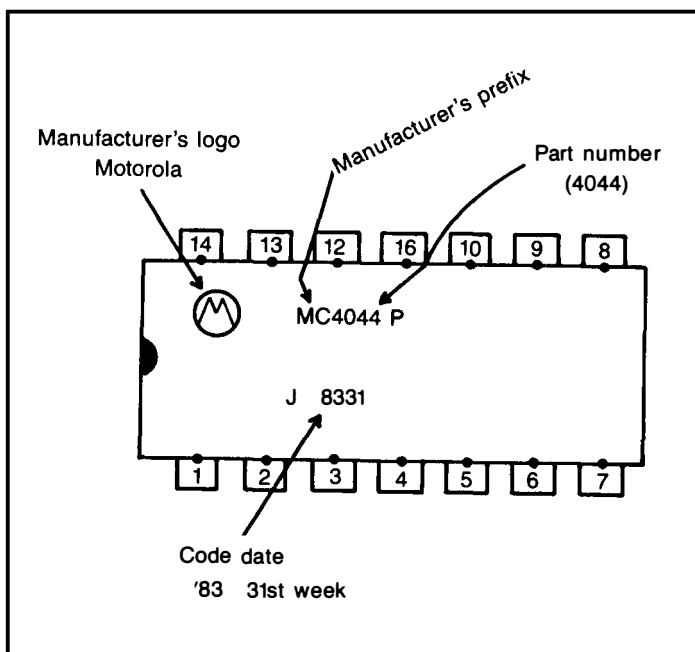


Fig. 4-10. While a lot of the markings on the various chips will remain a mystery, you can decipher some of them in this manner.

totals eight sockets. All the rest of the chips are wired into the print board.

The removal and replacement of socketed chips is relatively easy. There are a few guidelines that can make the job safe and easy.

### Chip Removal

You probably won't be able to help yourself, but a good rule to try and abide by is, never touch a chip with your hands, body, or clothing. TTL chips can be touched safely. They are not as sensitive as the MOS chips, but I'd use the rule for both. On occasion, you might pick up what you think is a TTL only to discover that it was a sensitive dynamic RAM MOS. My reasons for the antiseptic approach is fear of chip electrocution by static electricity. The weak, vulnerable time of a MOS chip is when it is loose. Any static discharge into the chip at that time could very easily burst some of the insulated gates of the tiny FETs.

Small chips, those with 24 or less pins, can be extracted and handled safely with the DIP extraction tool shown in Fig. 4-11. The tool is nothing but a tweezer formed of steel. It is built so that two little lips can be placed under the two ends of the chip. This allows you to gently tug the chip out of its socket. Once out, the chip can be placed on a conductive grounded surface that shorts all the pins to the surface. With all the pins shorted to ground, no static potential can build up and kill a FET gate.

The DIP extractor tool comes with a small hole at the top. This is to be able to screw a grounding strap onto the tool. The grounding strap is then attached to earth ground through a one megohm resistor. One other important word of caution. Do not remove or insert chips while the computer is plugged in to ac, whether it is on or off.

The extraction tweezer works easily with chips of 24 pins or less. It can be used with the larger 64 pins too, if you take some extra care. The longer DIP bodies, especially on the 40 pin chips, will be placed under strain if you use the same technique as you did with the smaller chips. The way around that problem is not to pull the chip all at once. Take one side at a time. Gingerly rock the chip out of

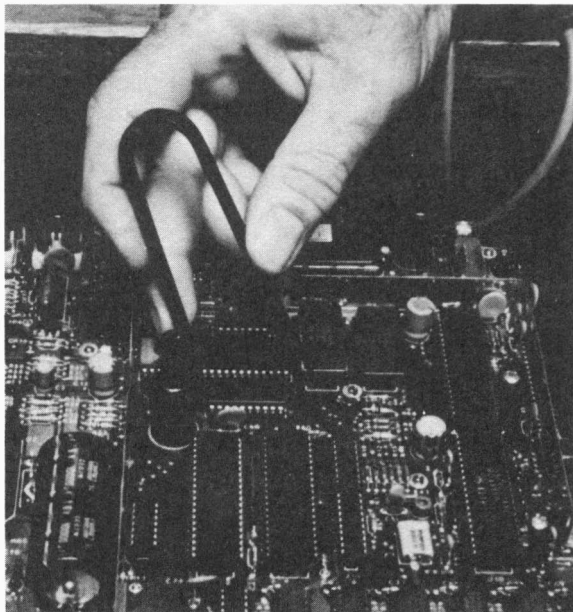


Fig. 4-11. The chip extractor is the safest way to remove chips.

the socket, first tugging one side and then the other. That way the holding ability of the socket is gradually relaxed and the large chip will not experience undue strain. Once again, after the chip is free, place its feet on a conductive surface.

The most sensitive of all the chips are the eight 4164 dynamic RAMs and the 2114 color RAM. They are all NMOS chips, and the gate insulators are easily shorted. On the RAM instructions, you are told to ground yourself and keep the chips in a conductive tube or foam. If you use the grounded extraction tool, you will be relatively safe.

### Chip Replacement

During troubleshooting and repair, chips are usually in place when you begin. The preceding section went through chip removal from a socket. There comes a time in the repair that chips must be inserted back into the socket. Either a new replacement or the old, proven good, chip is to be reinserted. Just as much care should be exercised during the chip insertion as the extraction.

The chip to be inserted should be standing on

a conductive grounded surface. You could use the grounded extractor tool if you are surehanded and careful. There are pitfalls though, as the little feet are fragile and getting all the legs into the socket at the same time, without bending a leg or two under, is tricky. Therefore it is advisable to use the chip inserter shown in Fig. 4-12.

The insertion tool in the illustration is typical of the devices. Note the conductive post sticking out of the top of the tool. That is where you attach the grounding strap. The post goes all the way through the tool. The post ends at the two flat sided metal holders at the bottom of the tool. If you pull the post, you can see the way the holders operate. The holders are able to grasp a chip or let it go as you pull the post up or down. On the side of the gadget, is a locking button. This can lock the holders and the post in place. That way, while the holders are hanging onto a chip, it can't be accidentally dropped during the action.

On one side of the inserter is a pin straightener. The pin straightener is also grounded to the post.

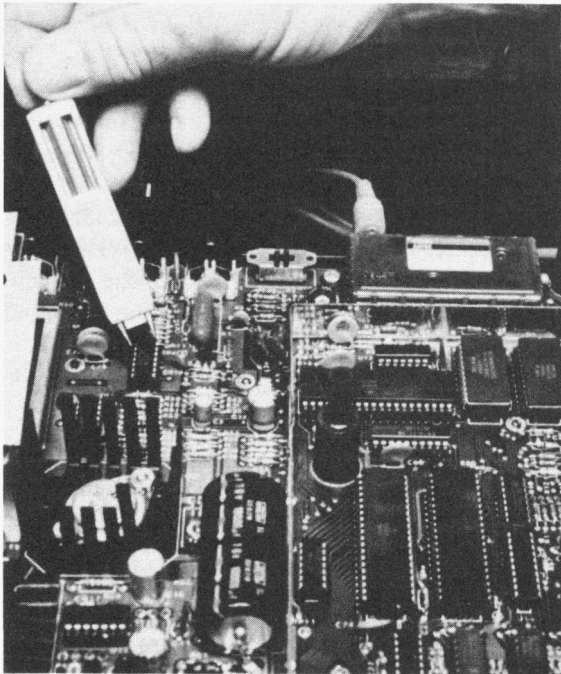


Fig. 4-12. Chips are best inserted with a gadget that can hold all the feet at ground potential at the same time.

Often the little legs of a chip can go out of line. The straightener fixes that problem promptly. Let's go through the procedure once and see how a fragile chip is inserted into its assigned socket.

The chip should be standing on its own feet on a conductive surface. In this position the entire pinout is at the voltage level of the surface, which is zero volts or ground. Look closely at the pins. Are they all nicely lined up? Are any of them bent?

If any are, pick up the chip with the grounded extractor tool. Push the chip feet first into the grounded pin straightener. Rock the chip, feet first into the grounded pin straightener. Rock the chip gently till the pins are all in their correct inline spacing. Then place the chip back on the conductive surface. Do not use the extractor to insert the chip. The inserter is better suited for the job as you'll see.

Pick up the insertion tool. The post should be attached to an earth ground. Pull the post out. The twin holders will retract. Place the tool over the subject chip and release the post slowly. The holders will now come down snugly around the chip and ground all the pins. Note that the extractor had held the chip by its insulated ends. The inserter holds the legs instead and grounds them at the same time. The chip is as safe as it can be under the replacement circumstances.

Next, place the chip over its socket and put all the pins into their correct holes. Observe the keyway on the chip and socket so that the chip is placed in the socket with the pins in the proper locations.

Once the pins are all in their correct holes pull the post up carefully. The holders will release the chip. Then with the post still held up press the inserter against the chip to seat it firmly. Do not press too hard, just enough for proper seating. Then remove the insertion tool. The idea is to keep all the pins of the chip grounded as long as possible. Notice that the only time during the insertion that the pins were off ground was an instant when the chip was released into its socket.

The extractor doesn't keep the chip pins at ground level. It grasps the chip at the ends. It holds the plastic packaging material of the DIP. The inserter on the other hand grounds the pins most of

the time. Grounding is the trick to keeping MOS chips intact during handling. If you must move or manipulate the chips out of circuit, make sure you are grounded. Attach grounding straps to your wrist. Should you have to transport chips from place to place, even across the room, keep the chips in some sort of grounded condition. It is very trying to order a chip, have it arrive after a week or so, and the lose it to a shot of static electricity as you walk across a carpet on a low humidity day. When you keep the chip at ground level, it is safe from static electricity.

## **SOLDERED-IN CHIPS**

Of the 32 chips in the 64, 24 of them are soldered directly to the printed circuit board. This presents a problem when one or more of them must be removed or reinstalled. The desoldering and resoldering of chips has been described as a job for an artisan, not the ordinary person.

Perhaps this is true if you want to reproduce the same professional finished look that is accomplished in a factory. However, as much as you take pride in your work, the polished look of a finished job does not mean a thing to the computer. The only important thing is replacing the chip accurately so that the computer begins operating correctly once again.

## **Desoldering**

Once the decision is made to desolder a chip, whether it is known bad, just suspected as defective, or must be removed for a test, the potential for inducing additional problems becomes a large factor. It is bad enough to have one trouble in a computer. Great care must be taken to avoid causing a second and third complication.

The first step is to reach for the right soldering iron. Only the right one will do. Don't place a hot 100/400 watt bench gun against the print board. It is much too hot. Use the lowest wattage iron you have. One that will just about melt the solder. Thirty watts is the absolute maximum and if you have an iron with less wattage use it. The iron should

be one specifically designed to be used for chips and sensitive transistors. They are on the market, some are battery operated and others use dc operating schemes. These are good to use when replacing MOS chips because the use of dc eliminates all the 60 Hz line voltage sine waves you get out of the house sockets. However, if you do use conventional ac house current with a low wattage iron, just ground the iron like you did your wrist.

The skill required to use solder on a chip is the control of heat. Too much heat, even momentarily, or prolonged heat even from a low wattage iron can kill a chip. Too little heat does not let the solder connect properly to the lead and a cold solder joint is the result. All surfaces that receive solder must be clean, and the tip of the iron should remain tinned all during the operation. That way connections are made in the fastest possible time.

Heatsink techniques are a must. The best heatsink is to grasp the lead between the connection and the body of the chip with a long nose pliers although this heatsink is not always possible. A piece of lamp cord connected to clip leads can also serve as a heatsink. The heat will always take the easiest path to dissipation, which is the fat plier nose or the lamp cord rather than the skinny lead to the chip. It is good practice to have the pliers grounded too. This prevents the connections from developing any undesirable static buildup.

Solder tips are easily kept clean by wiping it with a paper towel. Naturally, wipe quickly so the paper doesn't get a chance to char or burn. The solder should be rosin core 60-40 (tin to lead). It melts at 371 degrees F.

Desoldering a chip is not too difficult especially if you know it is bad and you do not care if it is damaged. Even if you do care the technique, when performed properly, is easy. The first steps are to be sure the print board is free and clear at the top and bottom. A good bright bench lamp with a magnifier will be extremely helpful. Get into a comfortable position so you are not straining as you work. Then the job can begin.

Attach the lamp cord-clip lead heatsink between the connection to be desoldered and the chip. Touch the hot iron to the connection. Most of the

heat that is melting the solder flows into the lamp cord. The technique is to slowly apply heat to the print board connection, jiggle the pin with a solder pick and wipe the connection. Keep doing that patiently till the pin is free. It could take a number of picks and wipes. Most of the solder in the print board hole will be removed. Once the pin is free go on to the next connection. Pin after pin is freed in this way. Keep working till each individual pin is free. Then lift the chip off the board.

Commercially available solder suckers can speed up this process. They are expensive, but they work. Most electronics suppliers carry them.

When you are sure the chip is defective and you just want to get it out of there, you can use less care. Then you can apply heat and pry faster. After awhile you'll pop the chip out. Just be careful not to disturb the board and the nearby components.

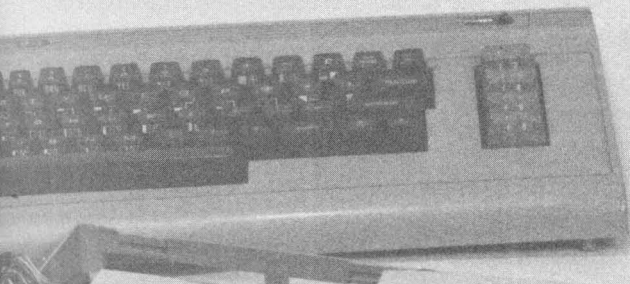
## **Resoldering**

Once the old chip is out, clean up the holes in the print board. Use a bit of heat, some wiping and pick out all the excess solder. The new replacement can then fit nicely into the holes. With the new chip

in place, apply a drop of properly heated solder to each connection. After checking the connections for possible shorts or opens, the job is done.

As you can see, the tough part of the job is the patient loosening of each lead during the desoldering process. It is good technique to never have to desolder the same chip twice. Often a particular chip is the victim of a manufacturer's design mistake. The chip will, over the life the computer, fail time and time again. You will do a lot of breath muttering if you have to desolder and resolder the same chip over and over again. What can you do? There is an easy solution. You can avoid the unpleasant chore after the first replacement. Make it a rule in your repertoire of soldering techniques to always install a socket whenever you have to remove an unsocketed chip. That way you'll never have to resolder that particular chip again.

Follow the same resoldering instruction for installing a socket. Of course, you need not worry about damaging the socket like you would a chip. You still want to use as little heat as possible, however, since the plastic body of the socket can melt if too much heat is conducted up the pins.



## 5. LSI Chips

**I**F YOU LOOK DOWN AT THE EXPOSED PRINT board of the Commodore 64, you'll see three large 40-pin DIPs and a 28-pin DIP. Should you remove the metal shield with rows of ventilation holes, you will see one more 40-pin DIP. These five chips are in the class of large scale integration. LSI consists of those chips that contain at least a thousand individual microscopic circuits. Once these circuits have been installed on a chip, there is no easy way to get a test probe into any of the circuits. Even if you could somehow run a voltage or scope test and find a bad circuit among the thousand or more, there would be no feasible way you could remove the defective component and install a new replacement.

The only practical way to gain access to the minute circuits, is the 40 pins that stick out of the little package. The pins connect internally to the circuits. There are only 40 pins that attach to over a thousand circuits. It follows that, on an average, one pin connects to at least 25 circuits and each circuit could have at least 10 input-output leads, which makes each leg on the DIP a connection to 250 in-

ternal nodes. While this statement has many variations and complications, the point is that the circuits inside the chips are in a different world. During troubleshooting and repair, you have to forget the internal circuits of the LSI chips and consider the chip as a black box with 40 test points available to your vom, logic probe, scope, and continuity tester. Only the original designer and manufacturer knows what is inside the black box down to the finest detail.

Fortunately, to service the Commodore 64, it is not necessary to learn all there is to know about LSI chips. You do not need to know the detailed construction secrets of the 6510, 6526, 6567, and 6581 chips. The servicing skills are in general those techniques that are used to repair any electronic circuits. The servicing techniques in specific terms are those that apply to the specific chips in the 64, as shown throughout the book. Let's start the exploration with an overview of the 64's five LSI chips.

### 6510 MICROPROCESSOR

The heart of the 64 is the 6510 MPU. It is a

member of the 6502 family. The 6502 is found in the VIC 20 computer. Commodore chose the 6510 for the 64, so a lot of the software that is developed for their computers will be compatible in the different machines.

The reason the MPU is called the heart of the computer is because it connects to all the rest of the chips. It is Grand Central Station to all the chips in the digital area of the computer. The MPU has four types of lines that go to and come from all the rest of the chips. The 6510 has address lines, data lines, control lines, and special I/O lines. These lines are grouped together in Fig. 5-1. The 6502 is almost identical to the 6510 as far as the address, data, and control lines go, but the 6502 does not

have the I/O lines. The I/O lines are used for memory control and will be discussed in detail in Chapter 12.

The MPU in all computers handles the vital computing chores. Without an MPU, a computer is not a computer. You can probably design a computer and leave out any of the other chips. You cannot leave out the MPU and still have a computer. At this point, let me introduce the general duties of the MPU. They might not have much meaning at this stage, but the duties will be clearer as we go.

The MPU, first of all, has the job of providing and requesting data. Figure 5-2 illustrates both of these jobs. The reason it requests data from the rest of the computer is to process it. The data is ob-

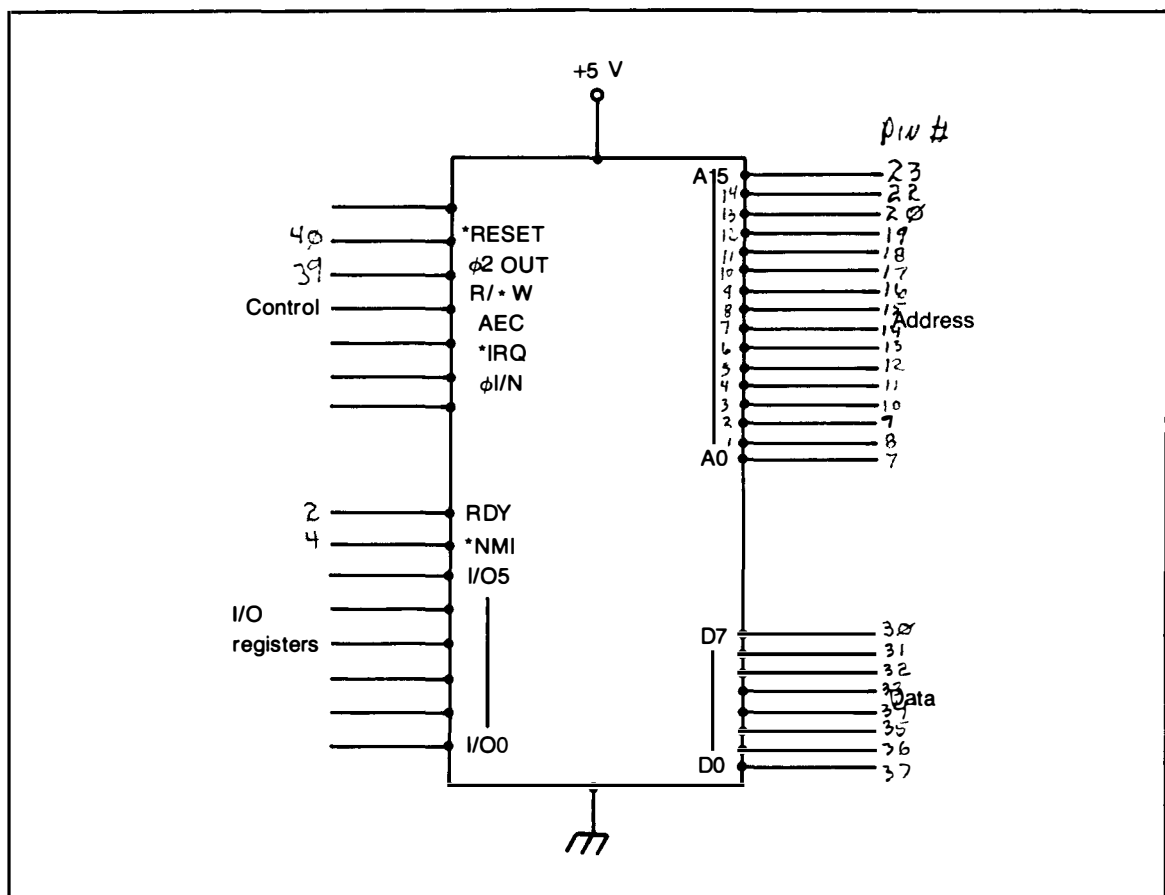


Fig. 5-1. The 6510 has four types of lines that connect to the rest of the computer chips. They are address, data, I/O, and control lines.

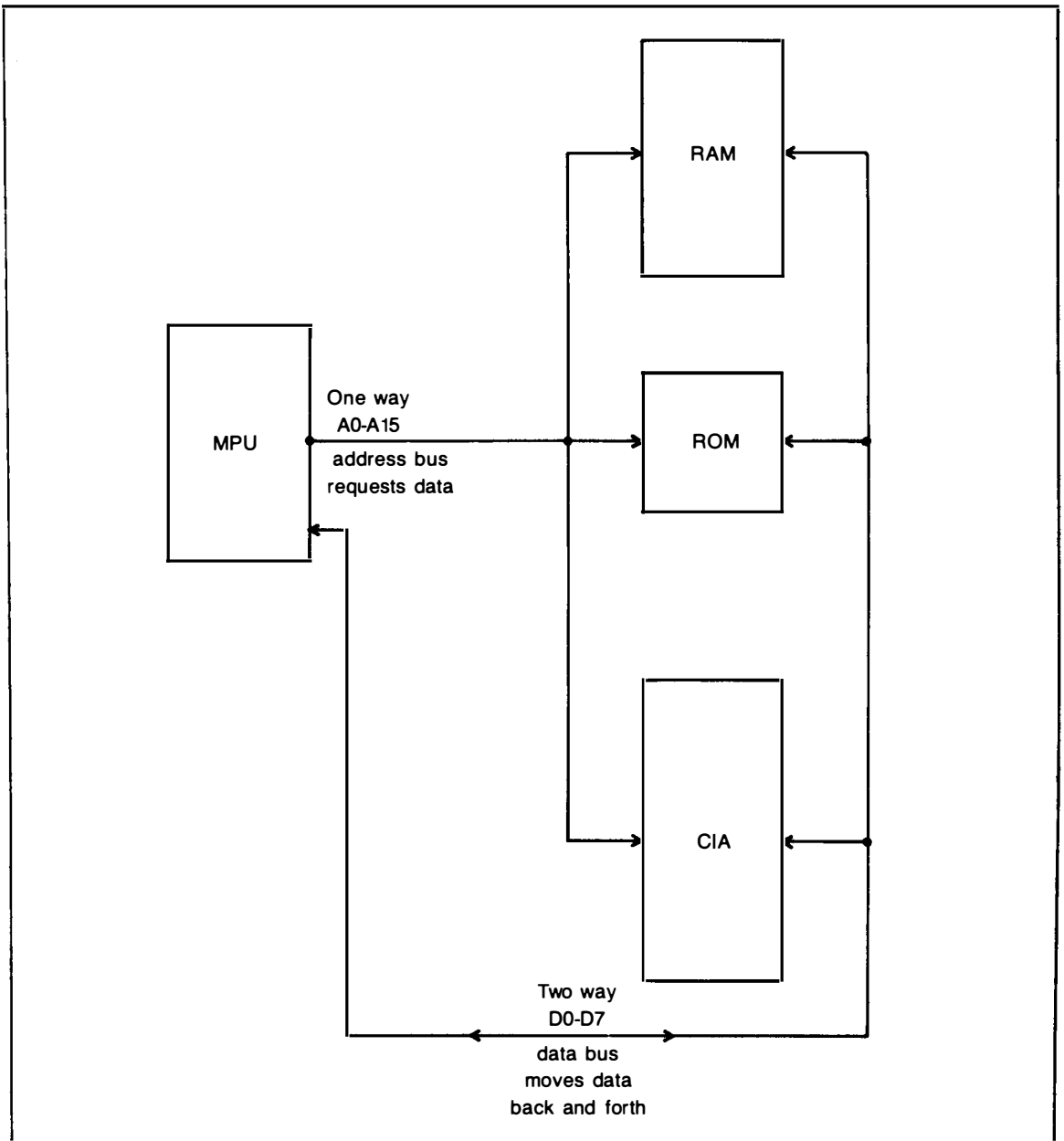


Fig. 5-2. The address lines have the job of going to a location and requesting access to the data in the location. The data lines are the wires that do the actual transferring of the data.

tained from places like the RAM, ROM, and the CIAs. The MPU will send a message to one of these places and request data. The data is sent back to the MPU over the data lines. The MPU can then

process the data and send it back via the same data lines to one of these storage or I/O places just mentioned. How does the MPU alert a data holding area that it wants data?



The message that the 6510 sends is an address. The address goes out over 16 address lines called the address bus. These are shown in Fig. 5-3. The 16 lines carry the address. The address is composed of 16 bits of information. One address bit can be transmitted over each address line. That way, different combinations of bits can form different addresses.

The procedure is quite like telephoning. If you want to call somebody you dial their number on the telephone. Everyone in the telephone book has a different telephone number. You contact your party by dialing their assigned number. The telephone lines are like the address lines that connect the MPU to the rest of the computer. The MPU is connected to every major digital section. With the 16

address lines, the MPU is able to contact the individual addresses 0 through 65,535 in the Commodore 64.

Therefore, when the 6510 needs data, it outputs an address on the address bus line. Like a dialed telephone number the address goes right to the correct location in the computer and opens up that address. As soon as the address opens, the data contained at the address is placed on the data lines. Then it travels to the MPU. The MPU accepts the data through the eight data lines shown in Fig. 5-4.

The MPU is capable of doing some limited mathematical and logical manipulating of the data. The nature of this work is covered in greater detail in Chapters 10, 11, and 12.

Once the MPU has processed the data, it is

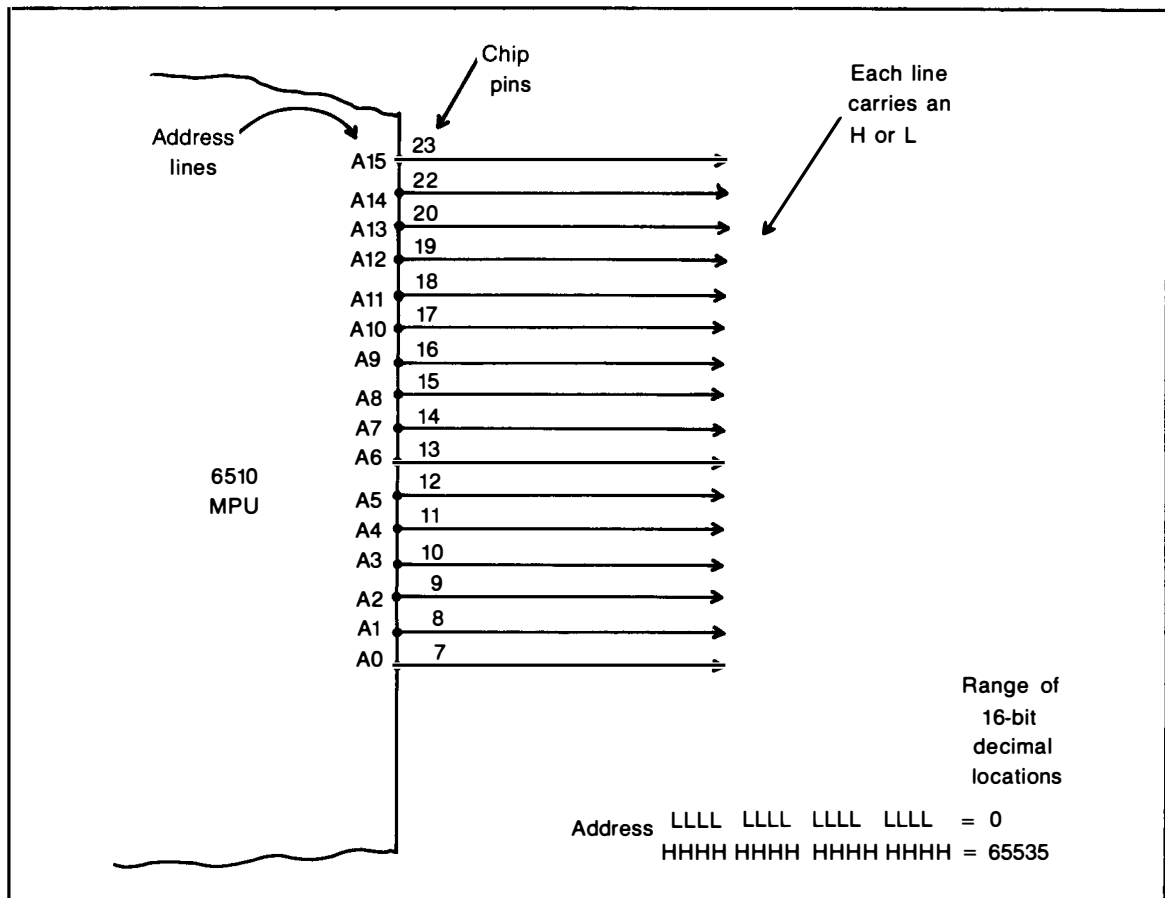


Fig. 5-3. The 6510 has 16 address lines that are able to form 65536 different binary bit combinations.

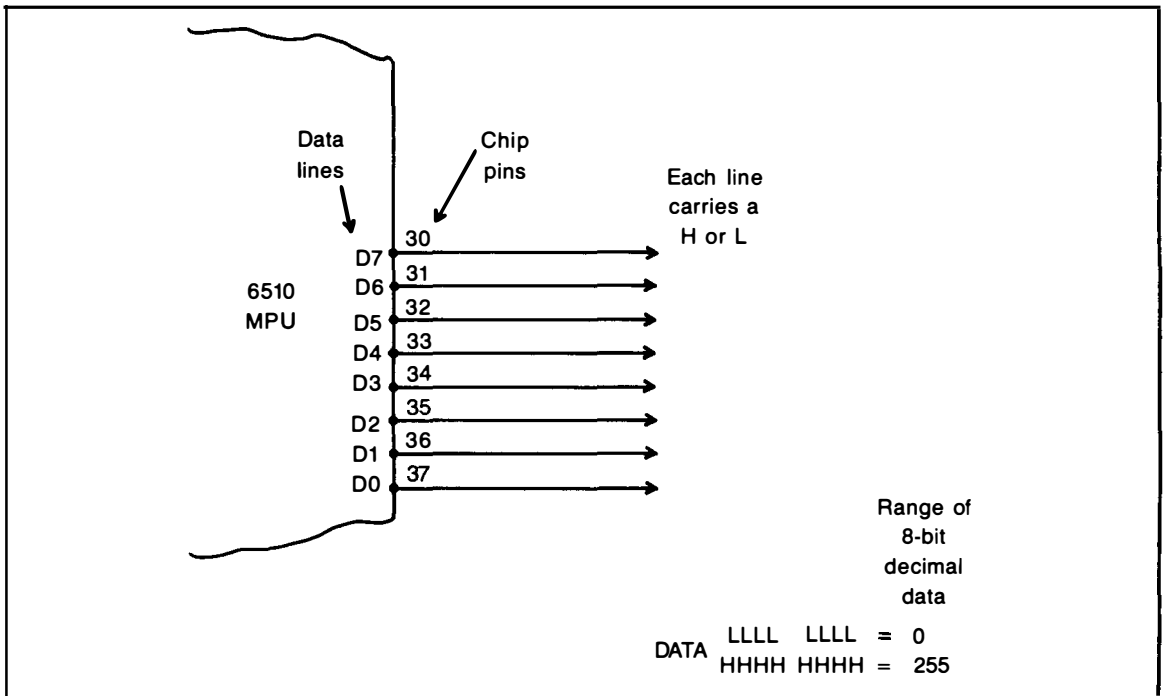


Fig. 5-4. The 6510 has eight data lines that are able to transport 256 different binary bit combinations.

ready to send the data back to storage or to an output peripheral. The MPU addresses the location where the data is to be sent. The addressed location is activated and opened, and then the MPU outputs the data to the location on the data bus. The location accepts the data and, if the location is RAM, it stores the data. If the location is a CIA, the data is readied for output to a peripheral.

This explanation describes in brief what the MPU does as it works through a program. In computerese, this is known as the *fetch and execute* cycle. When the cycle breaks down, the computer then needs troubleshooting and repair services. It is a necessity for you to understand the hardware layout and general operation to apply the fix to most repairs. Your ability to write programs will not aid very much here.

Besides the 16 address lines and the eight data lines, there are 16 more connections in the total of 40 on the package. Two of the last 16 are the +5 volt and ground connections shown in Fig. 5-1. These pins power the chip. The remaining 14 pins

do various controlling jobs that are described in Chapter 12.

## 6526 COMPLEX INTERFACE ADAPTER

The two CIAs are prominently displayed next to each other at the left top of the print board. They have the job of connecting, or interfacing, the MPU with external devices. The CIAs are both input and output ports. They interface input devices like the keyboard and joysticks, input-output machines like the cassette player, and output units like the printer.

The CIAs connect to the MPU through the same data lines that go to all the other data locations in the computer. The CIAs have their own addresses. When the 6510 desires to access the CIA, it outputs the CIA address, and it is immediately connected.

The CIA closest to the end of the board is almost exclusively devoted to the keyboard. When you hit a key on the keyboard you are shorting out a row and a column. Internally the keys are wired

in a block composed of vertical rows and horizontal columns. There are eight rows and eight columns. This produces 64 row-column intersections. There is one intersection for every keyboard character. When you strike a key, one of the intersections is shorted, which causes that character to be generated inside the computer. The keyboard CIA, as shown in Fig. 5-5, is the device that transfers the shorted intersection identity to the MPU so that the character can be generated.

The second CIA performs this same type of duty on other external devices. For instance, the second CIA connects to the terminal strip called the user port. The user port is able to connect up to a printer, a Votrax Type-N-Talk, a telephone modem or even a second computer. With some additional software, the CIA can send or receive from a large number of other devices.

The CIA can also be connected to more than one device at the same time as long as only one device is operating at the same time. For example, the CIA that is handling the keyboard can also be connected to the joysticks simultaneously. This arrangement is shown in Fig. 5-6. The keyboard and the joysticks are both input only devices. However, when they are not being used they are both open circuits. As long as they are both not in use at the same time, the connections can remain in place.

The 40 pins of the CIA are shown in Fig. 5-7. There are 24 pins to handle data. There are eight data pins that connect to the internal computer side. There are 16 data pins that connect to the user port. The 16 external pins are actually from two different CIA I/O ports. These are two sections in the CIA that operate almost exactly alike. They will be discussed in Chapter 16.

The rest of the CIA pins are used to address the chip and control it. There are only five lines into the CIA for addressing purposes. The CIA only contains 16 locations, unlike the RAM and ROM that could have thousands of locations on a chip.

In addition to the normal I/O duties that the CIA performs, there are some other abilities that the chip can be programmed to use. First of all there is a 24 hour Time Of Day (TOD) clock. The clock can give you the time of day in either the AM

or the PM. It also has an alarm. You can set the alarm with a few program lines.

For some sophisticated computer operations, special timing mechanisms are required. In the CIA, there are two such circuits called 16-bit interval timers. They are said to be independent and linkable. These are characteristics needed when these timers are used. If you'd like more details about these timers or the Time Of Day clock mentioned in the last paragraph, you'll find them in Chapter 16.

Another feature of the CIA is a special 8-bit serial I/O shift register. The term I/O, of course, means input/output. It was mentioned that the CIA has two I/O ports between the chip and the interface connector called the keyboard connector. The keyboard uses the two I/O ports to communicate the row-column short location to the computer. The two I/O ports each have eight connections to handle the keyboard information.

The two 8-pin I/O ports all send a bit of information at the same time. Since all eight bits are moving at the same time, or abreast, they are said to be moving in a parallel fashion. This is in contrast to bits moving along in single file along one of the eight lines. The single file movement of bits is called serial. No doubt you have heard the terms parallel and serial concerning the transfer of digital information. Figure 5-8 contrasts this type of data movement.

Anyway, besides the CIA having the two parallel I/O ports, it has one serial I/O port. The port uses one pin. Inside the chip, attached to the pin is an 8-bit serial register. The 8-bits in the register leave or enter through one end and exit or enter the chip one bit at a time, in single file, through the one pin. There will be more about the serial and parallel ports in Chapter 16.

The CIA has the capability of conducting an 8-bit or 16-bit handshake for both read and write operations. The handshake is a complicated procedure whereby the computer can either receive information from a peripheral (read) or send information to a peripheral (write). The handshake is very important and is conducted in machine language. If you conduct a handshake in BASIC,

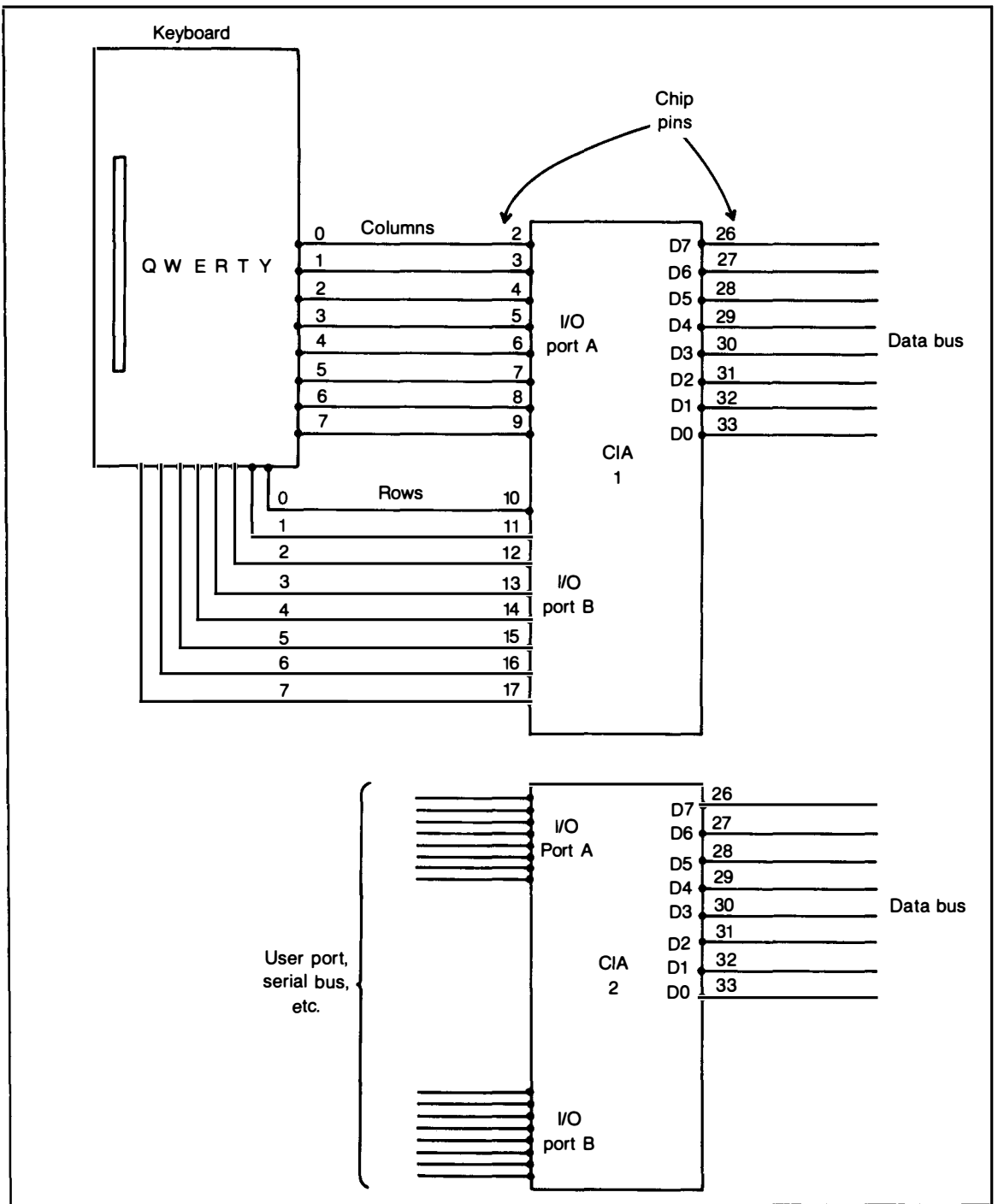


Fig. 5-5. CIA1 is devoted almost exclusively to interfacing the keyboard rows and columns to the MPU. CIA2 uses its A and B ports to interface external devices to the MPU.

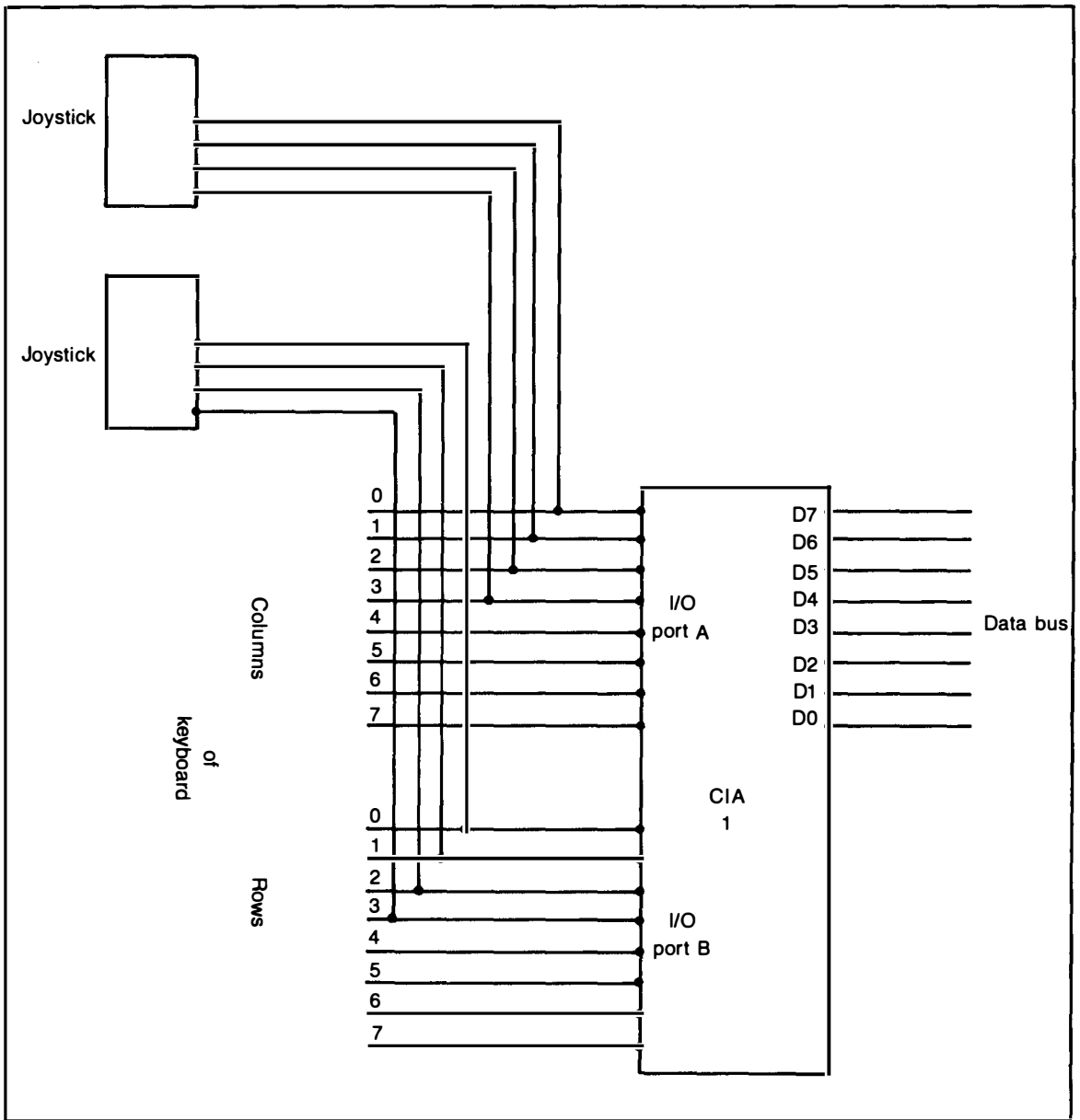


Fig. 5-6. CIA1 is able to interface the joystick positions through the same pins the keyboard uses.

it is done automatically for you.

A *handshake* is simply a check and doublecheck between the computer and the peripheral to be sure the information is transferred correctly. If we endow the computer and peripheral with human characteristics, the handshake will proceed as

shown in Fig. 5-9. Suppose you have a peripheral that is attached to a CIA with information. The peripheral says in digital code, "Hi there 6510. I have some data for you."

The 6510 hears the message and says, "I hear you calling. All's clear, send the data." The

peripheral promptly sends the data over the parallel lines and waits. The data enters the CIA, passes through the CIA, and exits onto the data bus. The data flashes over the data bus and enters the MPU. The MPU notes the data entry into its registers and calls to the peripheral, "Ok there peripheral, the data is received, I'm ready for more."

The peripheral thus sends information to the MPU and is able to continue sending until all its

data has been transferred correctly. While the characterization of the two devices is exaggerated, the procedure is not. The CIA chapter has a description of the electronics of the handshake procedure.

### 6567 VIDEO INTERFACE CHIP

The VIC chip is the place in the 64 where the TV picture originates. It receives a lot of digital infor-

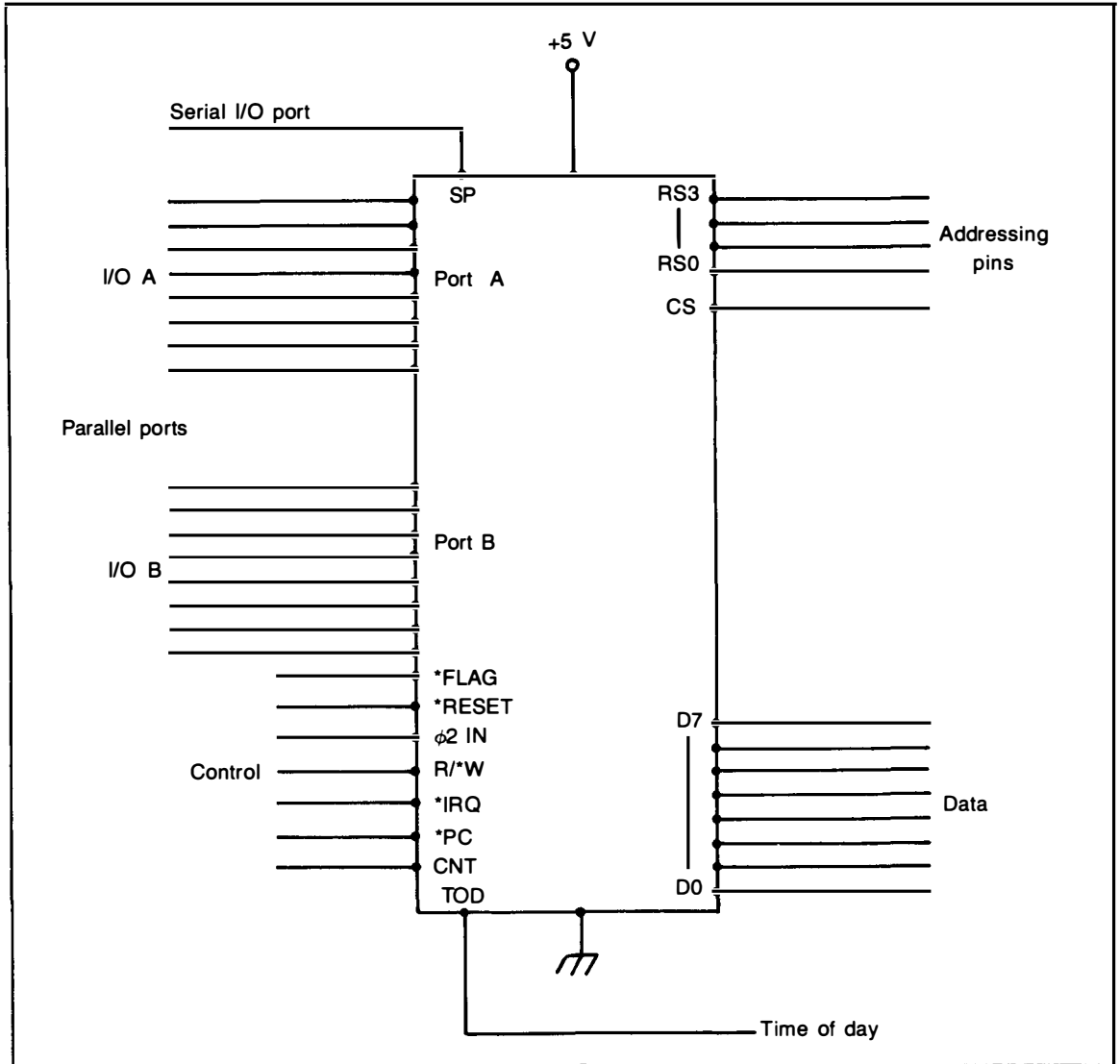


Fig. 5-7. The CIAs have address and data bus connections, control lines, and the A and B port pins.

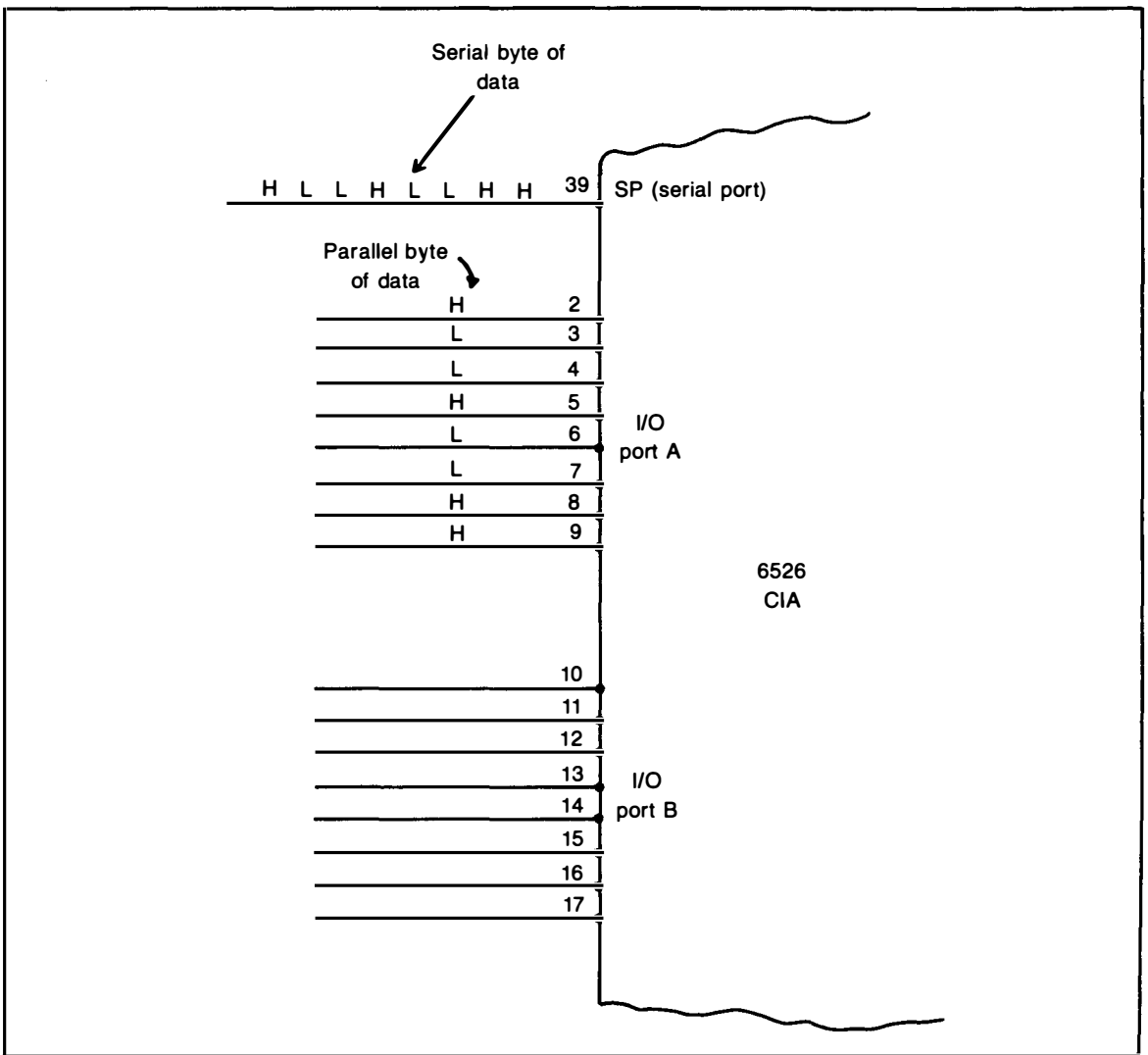


Fig. 5-8. The CIAs have the parallel-type A and B 8-pin ports and a single-pin serial I/O port.

mation. It takes all these bits of information and assembles them into the composite color TV signal that appears on your TV screen or computer monitor screen. Figure 5-10 identifies the functions of the VIC's pins.

Inside the VIC the various digital signals are assembled and converted into analog video signals. The VIC then outputs a sync signal, a luminance signal and the compatible color signal. These three components are then combined in conventional col-

or TV video amplifiers and applied to the TV display machine. The electronic details are covered in Chapters 17 and 18.

The analog signal that the VIC outputs appears on the TV screen. It is the window into the computer. What you see when you turn on the 64 is an arrangement of little lighted blocks. There are 40 blocks across and 25 blocks down. As Fig. 5-11 shows, this produces 1000 blocks on the screen. The blocks are graphic locations. Each block has

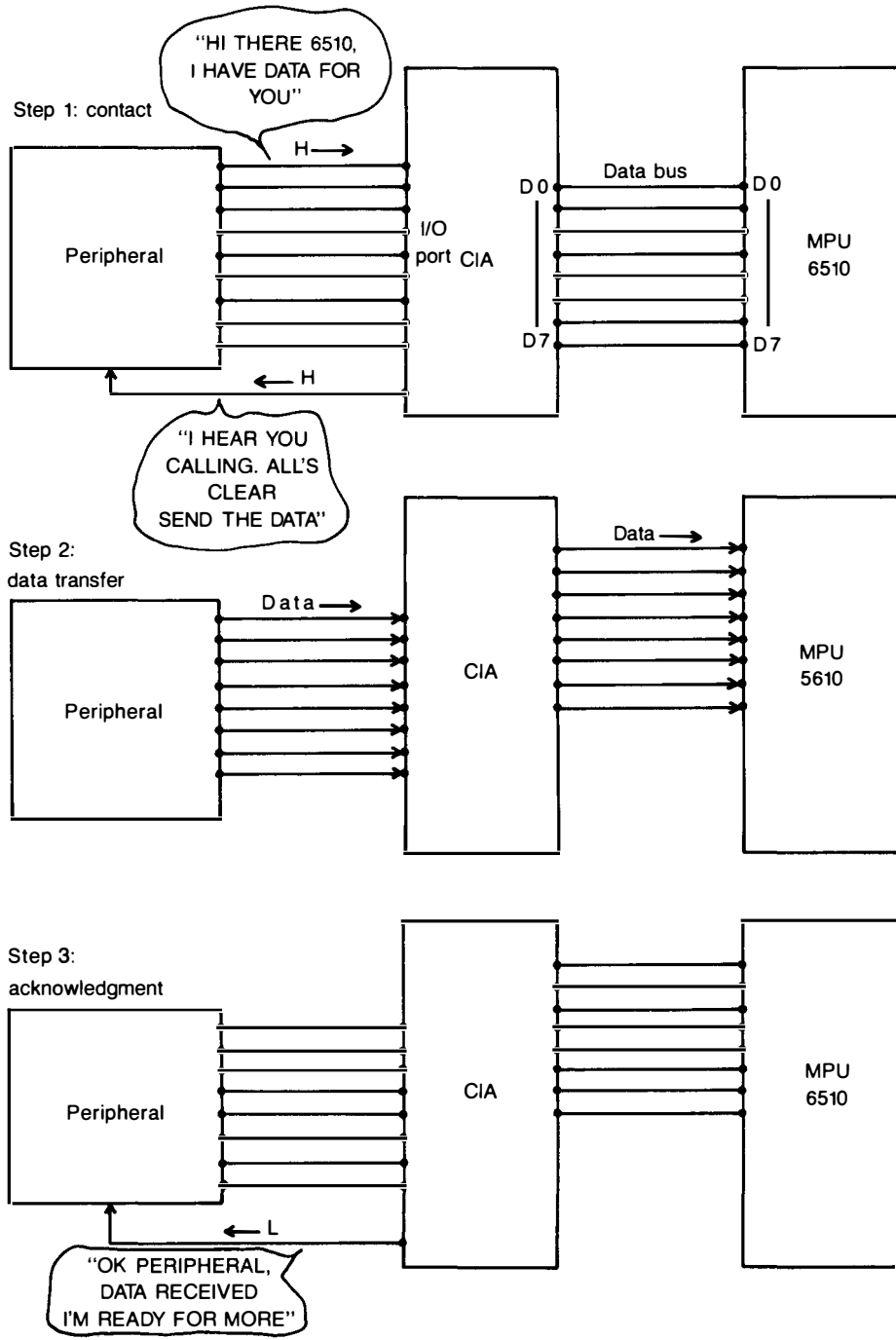


Fig. 5-9. The CIAs are able to conduct a handshake operation between the MPU and a peripheral device.



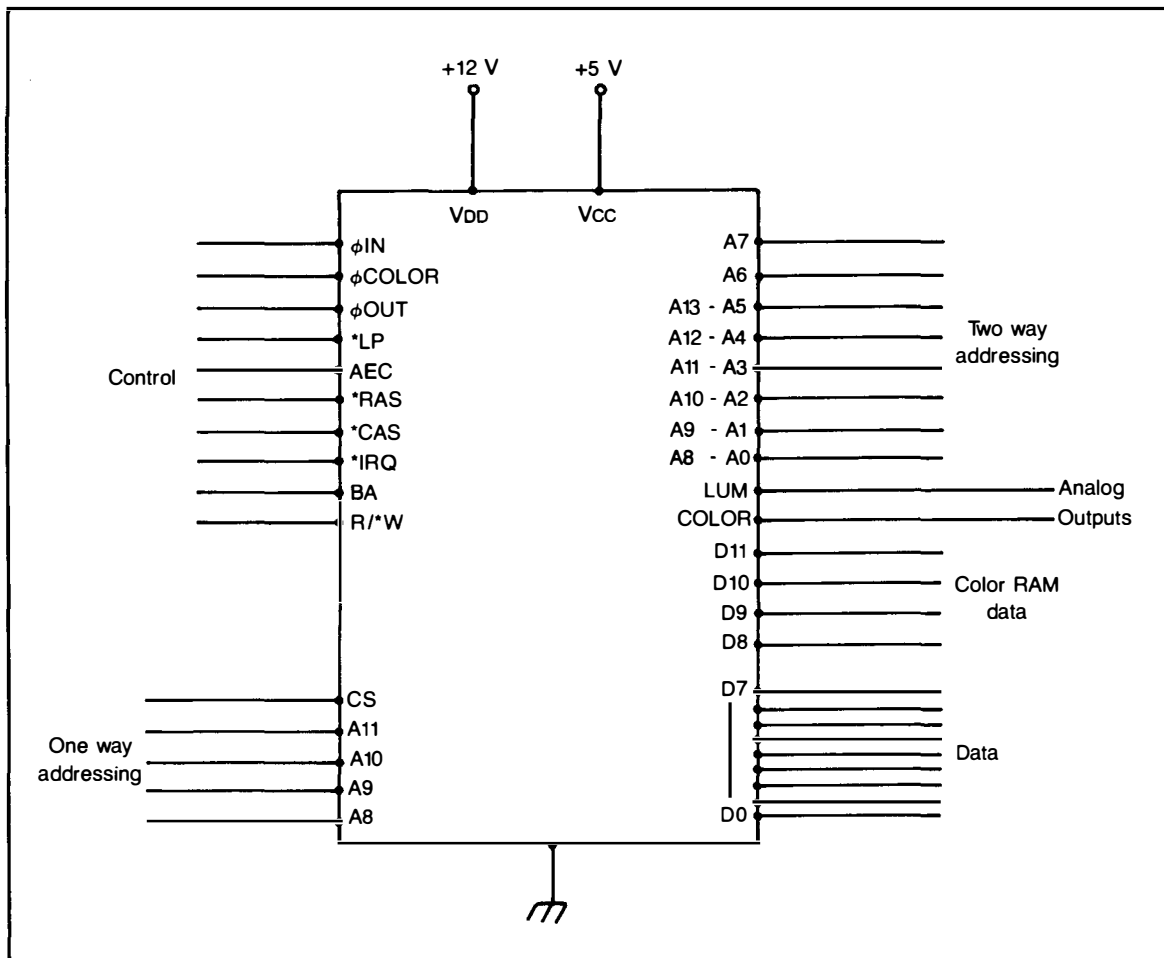


Fig. 5-10. The VIC has two addressing modes. One is the normal addressing from the MPU by means of the chip select, CS, and the address lines. The second mode allows VIC to do the addressing through the two-way addressing lines.

a corresponding address in 1000 bytes of the video RAM. Figure 5-12 demonstrates this correspondence.

Each block on the screen consists of 64 dots. The dots are arranged as eight dots across and eight dots down. The computer is able to light or darken each dot individually. Also, the computer can color each dot with one of 16 different colors.

The Commodore 64 is able to form colored letters, numbers, symbols, and graphic characters in each of the 1000 blocks. It does this by sending a digital pattern of the character that is to appear in each block to the VIC. There are 256 patterns for

characters stored in a Character Base in the Character ROM chip. The computer calls specific characters and sends the patterns to the VIC. Figure 5-13 shows a portion of ROM and the resultant block that it produces on the screen.

In order to color the characters, the use of the Color RAM is needed. It sends a pattern to the VIC that causes a choice of colors to light up the character. All of this input information is in digital form. The VIC in turn changes all this information into sync, luminance, and color outputs in an analog form. The TV circuits take it from there.

During servicing the inputs of the VIC are

## 6581 SOUND INTERFACE DEVICE

tested with digital techniques while the output testing is nothing more than old TV repair procedures. You'll find the methods in Chapters 17 and 18. These chapters are very important since a large percentage of the Commodore 64's breakdowns have been in the video input and output circuits.

The VIC II is a very powerful chip. In some ways, it can operate without needing any help from the MPU. These independent operations stem from the fact that the VIC is used as a video controller in video game applications besides appearing here in the Commodore 64.

The VIC has 47 control registers that attach to the computer's data bus. It is also able to access 16K of the machine's 64K of RAM memory. The registers and the memory that the VIC can use are all needed to place the display information into the TV picture.

The 6581 Sound Interface Device (SID), is the 28-pin DIP found in the center of the print board. Its pin functions are identified in Fig. 5-14. The SID and the VIC are, relatively speaking, new to the computer scene. In the early days, before the advent of the micros, computers were large and very expensive. Only the government and giant corporations were able to afford their use. As a result, the applications of computers were few. Aside from the military using it for ballistic calculations and corporations doing complex business figurings, the computer had little use. When the micros came along and the small computers dropped in price from around \$25,000 to under \$1000, new applications of computers proliferated.

One of the most popular uses quickly became arcade/home video games. The VIC and the SID

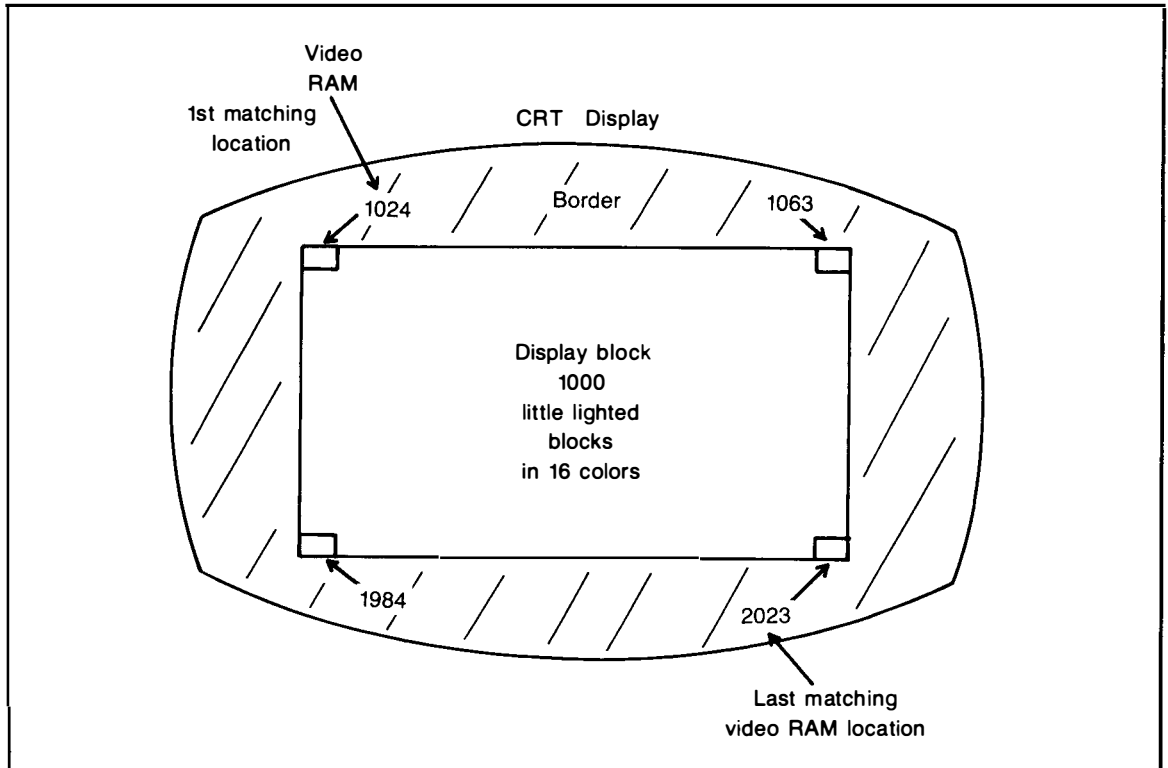


Fig. 5-11. The VIC lays out the display into 1000 little lighted blocks. There are 1000 8-bit dynamic RAM locations assigned to service those blocks. There are also 1000 4-bit static locations assigned to color those blocks. Each block, therefore, needs 12 bits in order to show a character in color.

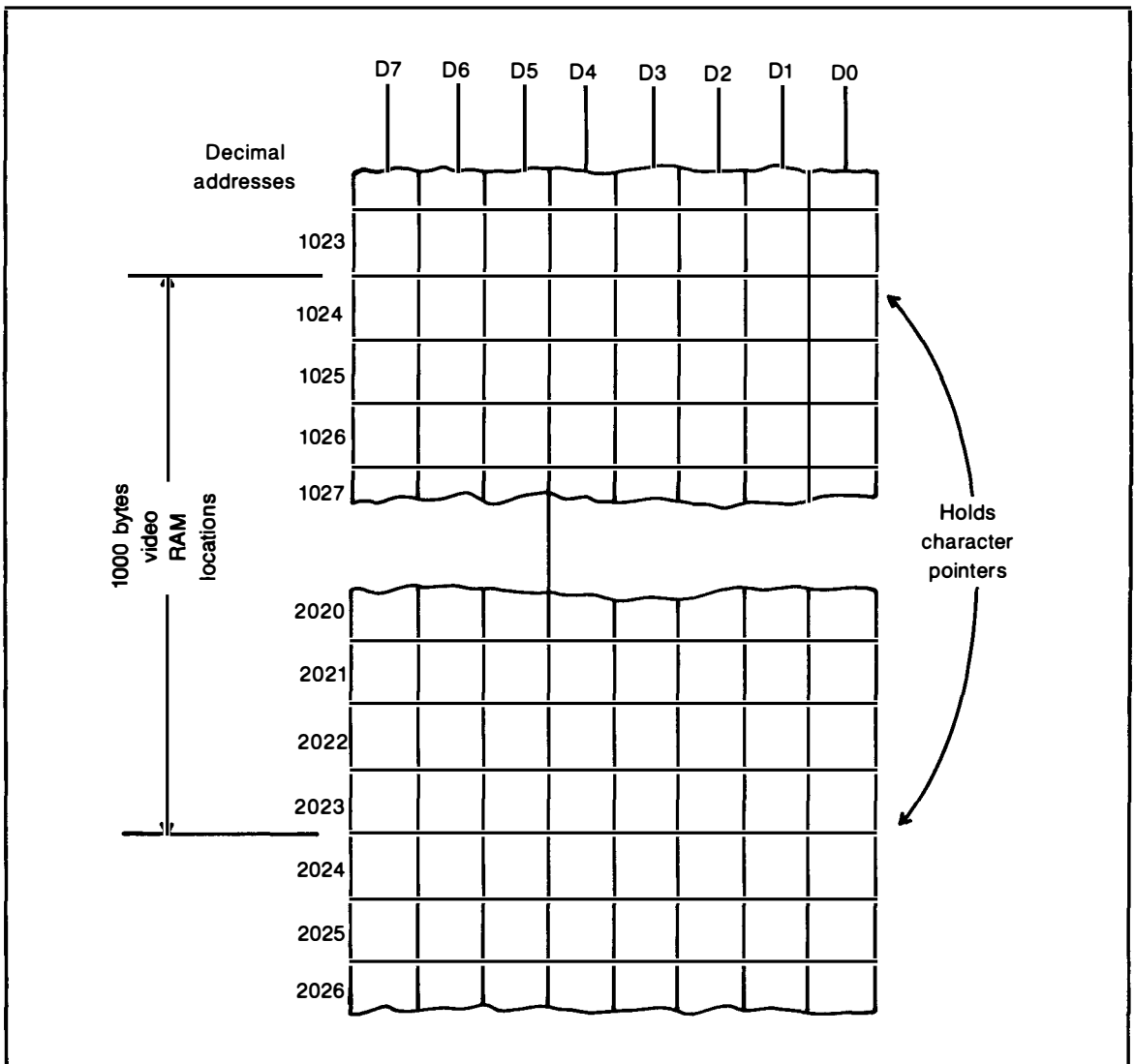


Fig. 5-12. The addresses 1024-2023 are called video RAM. They are filled with an address called a pointer. The pointer tells the VIC the address of a character that is stored in ROM.

are both designed to fit the game scene. This is a far cry from tracking an ICBM. While the VIC is needed to control the TV picture, the SID is used to create the sounds that come out of the TV. If you are using a monitor that does not have any audio output, the SID would be useless to you unless you hook the SID's output to a compatible audio arrangement.

The SID is a sound effects generator. It is

designed to be driven by circuits in the 6502 MPU family. Note that its number is 6581. This computer's 6510 MPU is one of the group. The SID is one of the most advanced computer music synthesizer and sound effects chips around.

If you are musically knowledgeable, you'll recognize the following qualities. First of all the SID is able to provide a wide range, high resolution control of pitch. Pitch is produced by controlling the

frequency of an audio output. There are three audio oscillators in the SID. They are called *voices*. Each voice can be used by itself or in combination with the others. By controlling the frequency of the audio oscillators, you can control the pitch of each voice.

In addition to pitch control, the SID permits control of tone color. This is accomplished because the oscillators produce four waveforms at the tuned frequency. Each waveform has its own special harmonic content. By judicious choosing of

the individual waveforms, you can control the tone color of the sound.

Besides controlling pitch and tone color, you can instruct the SID on the volume level of the audio. Each voice has an amplitude modulator circuit on the chip. Each voice also has what is called an envelope generator circuit. When the program instructs it to, the envelope generator creates an envelope waveform that controls the amplitude modulator. All this creates the desired amount of

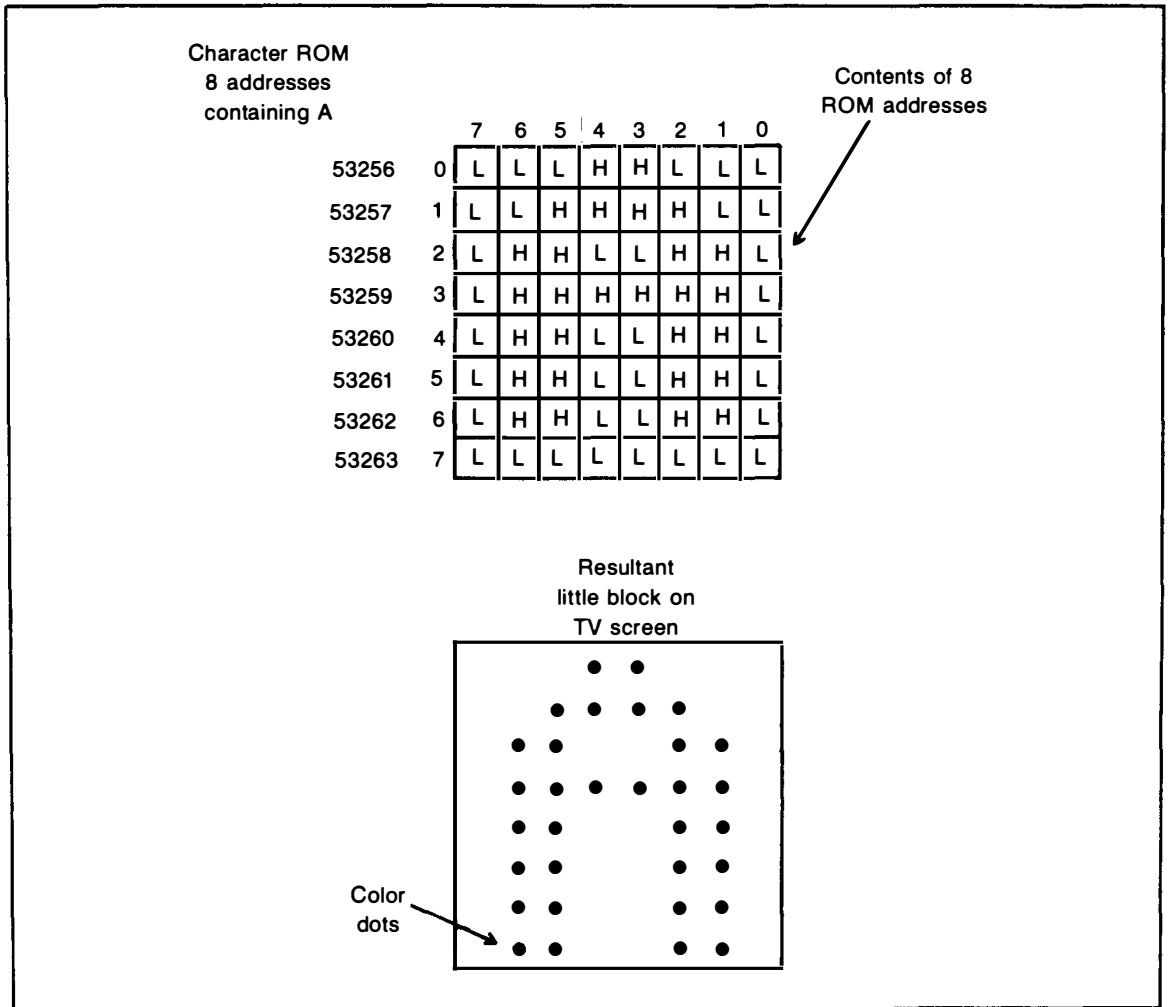


Fig. 5-13. The little blocks on the TV screen consist of an 8 × 8 matrix of 64 color phosphor dots. Each dot can be turned on by an H or turned off by an L that is passed through the VIC. The character ROM contains the necessary Hs and Ls in eight locations. Here is how the letter A is stored in addresses 53256-53263. The VIC is able to fetch the eight bytes and put them into a TV block.

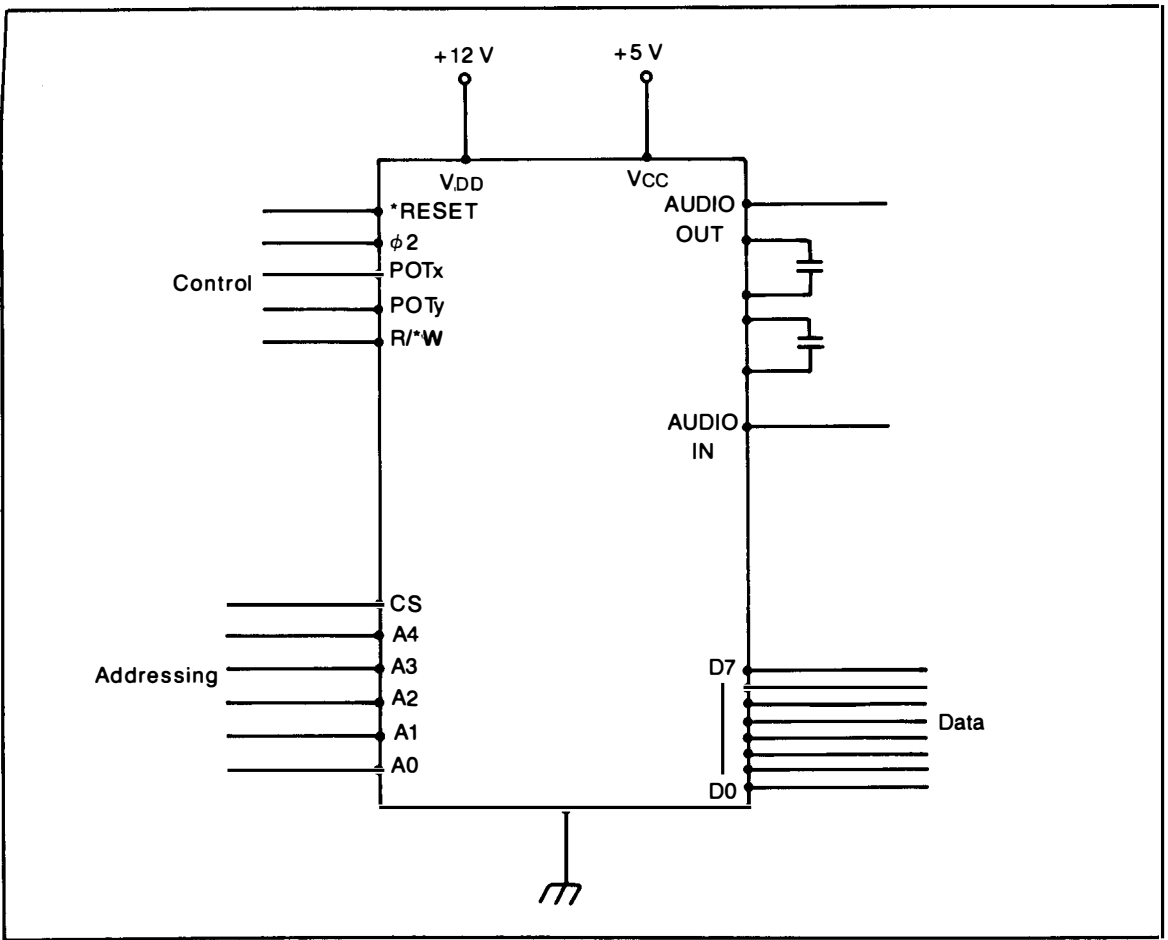


Fig. 5-14. The SID has the usual address and data lines. It has some special control lines as well as audio output and input.

volume. Lastly there are programmable filter circuits that further generate complicated, interesting tone colors.

The electronics that runs the SID is discussed

in greater detail in Chapter 18. The chapter explains the way the SID uses 29 8-bit control registers to produce the arcade quality sound effects.



## 6. Random Access Memory

**W**HEN THE MEMORY IS DESIGNED FOR A computer application, the engineer can choose static or dynamic. Dynamic is much more economical than static. It takes up less room on a chip and costs less. However, there is the refreshment problem to contend with. Static is more expensive but is very stable and requires less support circuitry. As a result, you'll find static memory being used where small amounts of memory is needed while dynamic is utilized for large memory demands. That is why the 2114 static RAM chip is used for the Color RAM with its small demands, and the 4164 dynamic RAM chips are used for the main RAM with its large demands. In this chapter, I will describe static and dynamic memory and how the 64 memory is organized.

The 64 in the name Commodore 64 refers to the amount of random access memory that the system has. Specifically, there are eight 4164 dynamic RAM chips in the machine. There is also a 2114 static RAM chip.

Each dynamic chip contains 65,536 bit holders.

With eight chips, the machine can store 65,536 bytes of data since there are eight bits to a byte. It is said that the 65,536 bytes are 64K bytes. The K, of course, stands for 1024.

If you multiply 1024 by 64, you arrive at the 65,536. The reason the 1024 number is chosen is because computers do all their figuring in powers of 2. The closest binary multiple to 1000 is  $2^{10}$ . The numbers go like this: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024, which is  $2^{10}$  and is called 1K. If you continue the progression, you'll encounter a lot more important computing numbers, 2048 (2K), 4096 (4K), 8192 (8K), 16384 (16K), 32768 (32K), and 65536 (64K). Table 6-1 summarizes all these numbers for you. It is a good idea to remember these landmark numbers up to 64K as you could be using them often during troubleshooting and repair.

The term RAM for random access memory is a holdover from before the microcomputer. It is not really an accurate description. You'll find that computer people think of it as read/write memory in

**Table 6-1 Good Numbers to Remember.**

Powers of 2	Decimal Count	Bits Required
2	0-1	1
4	0-3	2
8	0-7	3
16	0-15	4 (nybble)
32	0-31	5
64	0-63	6
128	0-127	7
256	0-255	8 (byte)
512	0-511	9
1024(1K)	0-1023	10
2048(2K)	0-2047	11
4096(4K)	0-4095	12
8192(8K)	0-8191	13
16384(16K)	0-16383	14
32768(32K)	0-32767	15
65536(64K)	0-64535	16 (2 bytes)

comparison to ROM which is accurately called read only memory. ROMs are covered in the next chapter.

I use the kitchen memory board to describe RAM workings. You are probably familiar with the

type of kitchen memory pad that has a plastic transparent face. When you lift the plastic, the number you jotted down disappears and the memory board can be used again for more reminders. The memory board can be used over and over again till it wears out, which is usually a long time.

The board starts out as a blank page. When you want to, you can write on it. Again, at your leisure, you can read the information it contains. After you are completely finished with the information, you can lift the plastic sheet and the information disappears as if it was never there. The board is then ready to receive more information.

RAM, in principle, acts in an analogous way to the kitchen memory device. The RAM chips though do the job by means of electronics. The eight dynamic RAM chips in the 64 are 16-pin DIPs. The eight DIPs are a set. They comprise the stated 64K in the name of the computer.

## STATIC RAM

In addition to the eight dynamic RAM chips, there

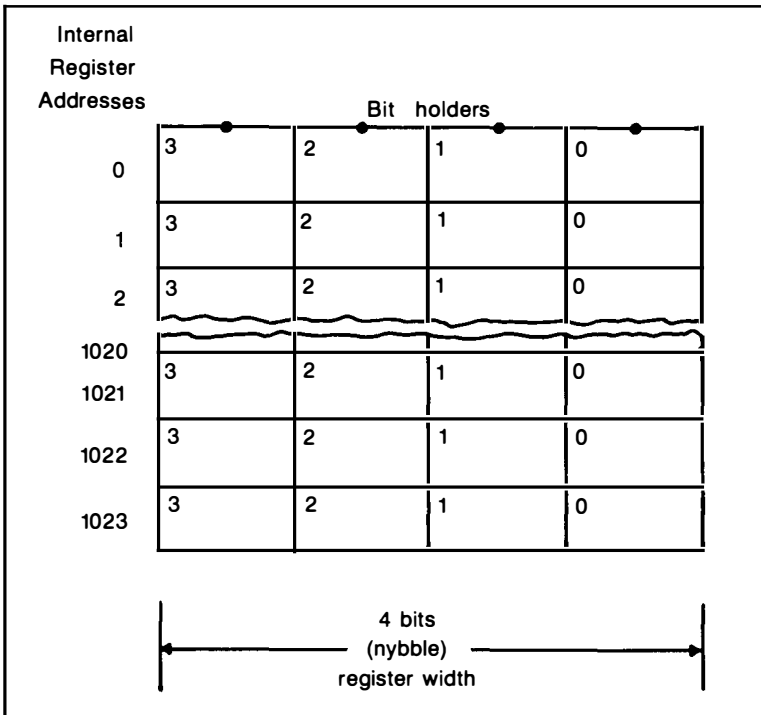


Fig. 6-1. The Color RAM 2114 chip is organized in nybbles. The internal address of each 4-bit register ranges from 0 to 1023.

is one more RAM chip, but it is called static. It is the 2114 Color RAM. Static RAM, although it stores data like dynamic does, works on different principles. Let's examine the differences between static and dynamic RAM. We can begin with the 2114 static chip.

The 2114 is an 18-pin DIP. It is made as an NMOS. It is said to be organized in a  $1024 \times 4$  format. This means there are 1024 individual locations built into the chip. The 4 means that each location has four bit holders. Four bits is half a byte and is accordingly called a nybble. Figure 6-1 illustrates the arrangement as a tall skinny 4-bit wide structure. Each nybble location has an internal address. The 1024 addresses are numbered 0 through 1023. 0 is the first address and 1023 the last.

Each bit holder (four to each address) is able

to store a high voltage or a low voltage. The highs and lows are code for 1s and 0s. The highs and lows are stored in flip-flop circuits. Each bit holder is a flip-flop circuit in a static RAM. The flip-flop circuit itself is covered in Chapter 11. In fact, this is the difference between static and dynamic RAM. Static RAM uses flip-flops for bit storage while dynamic RAM uses another method.

This tall skinny memory layout is called the memory matrix. Each location is called a register. The address bus is connected to the rows of addresses from 0 to 1023. The data bus is connected to the columns for four bit holders. The bit holders could be called D3, D2, D1 and D0. All of the 1024 locations have their D3 flip-flop connected to the D3 data bus line. The D2 flip-flops are all connected to the D2 data bus line, etc. The signal on the ad-

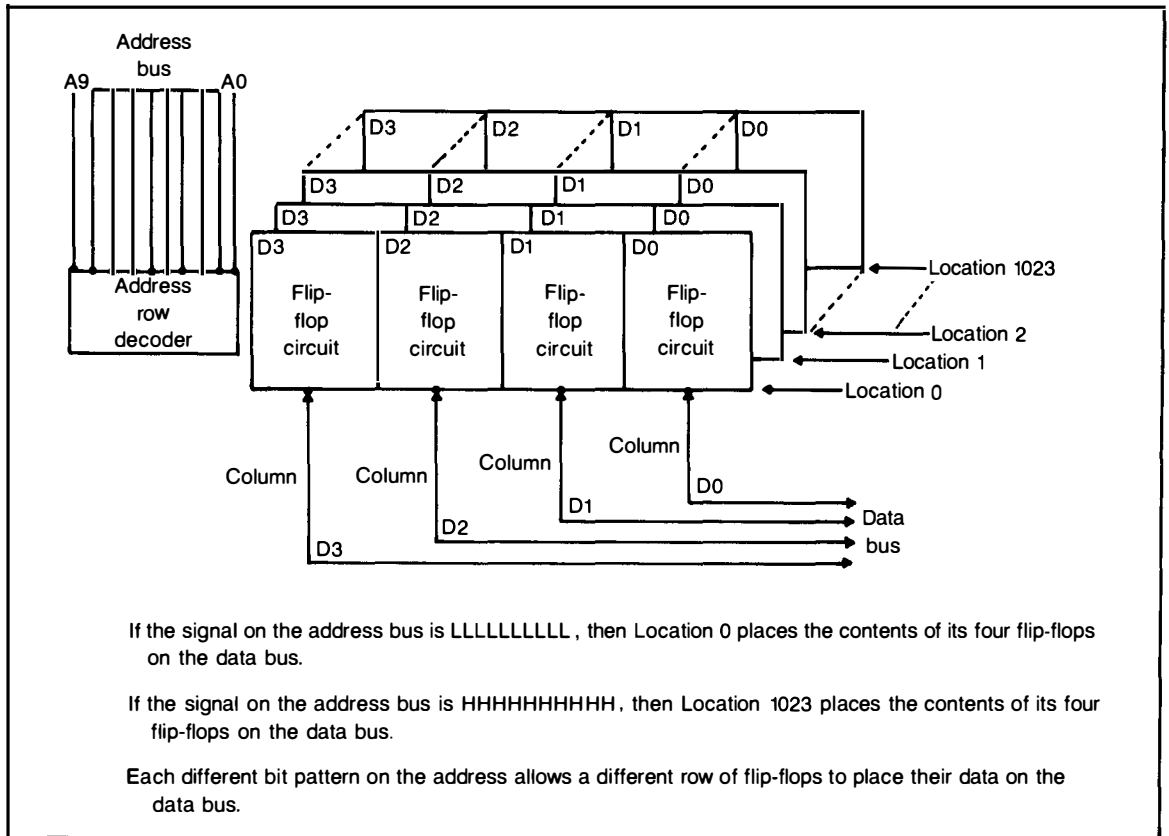


Fig. 6-2. Each register in the 2114 is able to store a high or low voltage. The high is the electronic representation of a 1 and the low indicates a 0. The register addresses are arranged in rows. The data bits are in columns.



dress bus will determine which row of four flip-flops can place their data on the data bus. Figure 6-2 illustrates this idea.

When one of the registers is addressed it activates the connections between the bit holders and the data bus lines. A copy of the highs and lows in the bit holders will pass over the data lines.

If you look at functional arrangement of the pin connections of the 2114 in Fig. 6-3, the first thing you'll note is ten address lines A9 through A0. Each address line is able to carry a high or low to the chip. It so happens that 10 address lines are able to bring 1024 possible combinations of highs and lows to the chip. Since there are exactly 1024 registers in the 2114, the ten lines are able to bring

a separate address for each location in the chip. That is how the memory matrix of the chip can be addressed.

Figure 6-4 is the actual arrangement of pins on the chip. At pin 8 of the chip is the notation \*CS. The CS means chip select. The asterisk in front of the CS means pin 8 is held at a high voltage most of the time except when you want to turn it on. If you desire to turn it on, then you send a low voltage to pin 8. The change in voltage from a high to a low selects the chip for action. When the chip is turned on it will respond to the ten address lines.

If the asterisk was not there, it would mean the CS pin is being held low while off and would turn on if a high arrived.

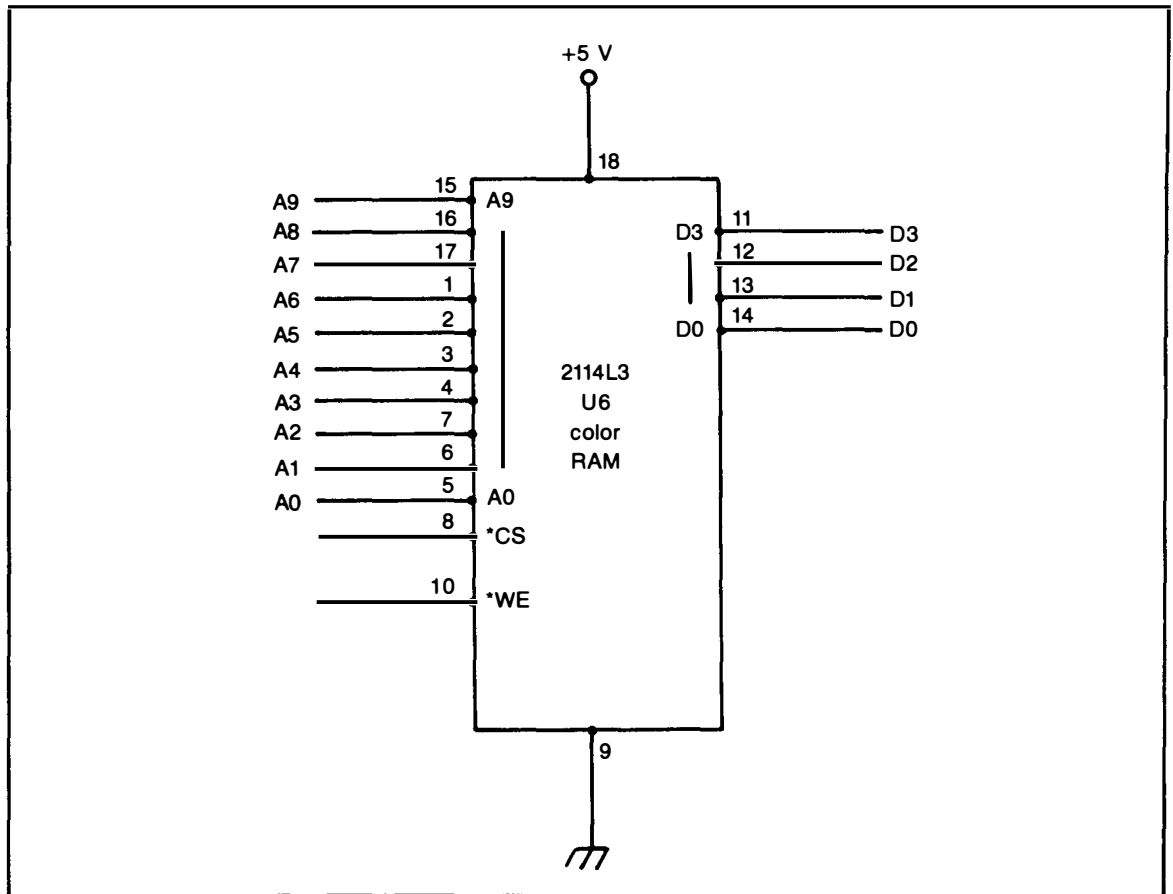


Fig. 6-3. The ten address lines, A9-A0 are able to bring one of 1023 separate addresses to the 2114. \*CS is able to turn on the chip. \*WE sets up a read or a write. The four data lines pass the contents of the addressed register.

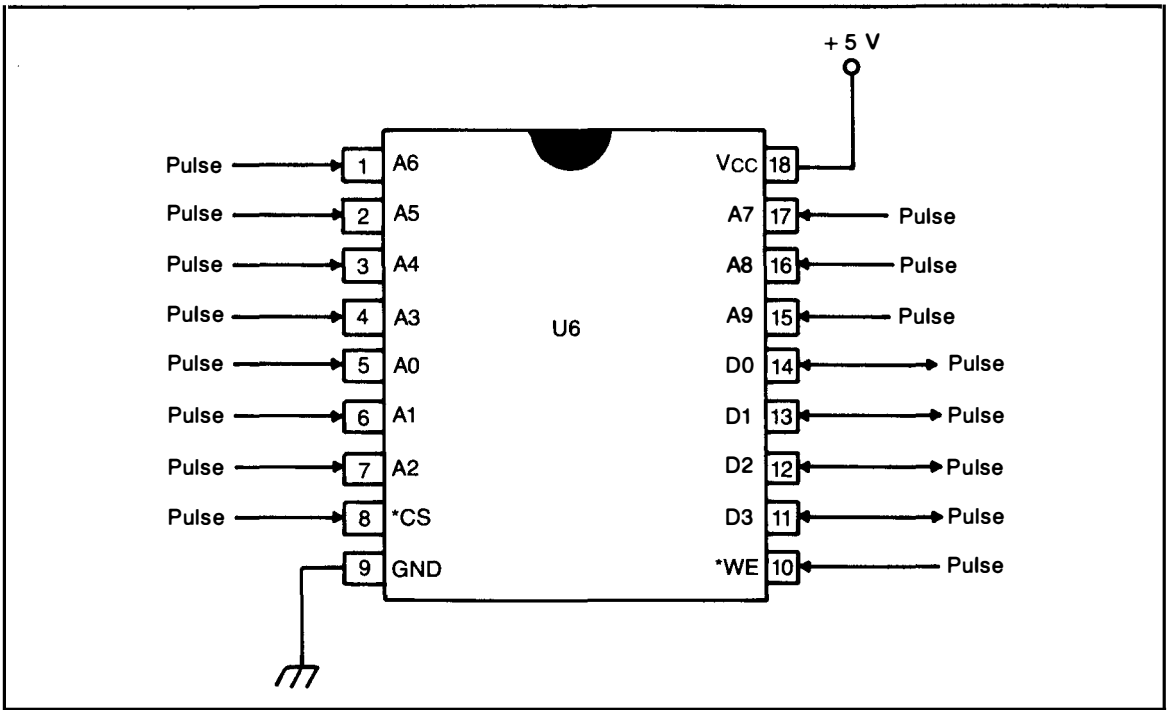


Fig. 6-4. The 18 pins on the 2114 are its inputs and outputs. These are the test points you examine with a logic probe to troubleshoot the chip.

The 4-bit data to or from the 2114 travels via pins 11-14, which are connections to the data bus, D3-D0. As the chip is selected with a low into \*CS and the internal location is addressed by the 10 address pins A9-A0, the 4-bit holders D3-D0 can either be read and give up a copy of their contents, or be written to and receive a new set of highs and lows to store.

How does the chip know whether to output the register contents or input a new set of highs and lows? Pin 10, called \*WE, makes that decision. \*WE means write enable. The write enable determines if the chip is to be read or written to. It is the read/write input of the chip. The asterisk means the pin is held high. It will read into its registers while in that standby condition. If it is necessary to write the chip, a low will be sent to the pin and change the mode from a read to a write.

## DYNAMIC RAM

One of the most striking differences between static

and dynamic RAM is the organization of the bit holders. As we just discussed, the static 2114 RAM chip is organized in a  $1024 \times 4$  layout. Other static RAM or ROM examples are organized as  $1024 \times 8$  and  $2048 \times 8$ . These are a common 1K byte chip and a 2K byte chip. The arrangement of a 2K byte chip is shown in Fig. 6-5. In all of these chips, the nybble or the byte sized registers are contained on one chip. The 2114 has nybble sized registers, and the other two examples have byte sized registers. The layout of the dynamic RAM is entirely different.

One dynamic 4164 chip has a layout of  $65,536 \times 1$ . This means there are 65,536 bit holders in the chip and each one has its own address. Refer to Fig. 6-6. They could be loosely referred to as 1-bit registers in contrast to the 2114's 4-bit registers, or the other examples with their 8-bit registers. The Commodore 64 is an 8-bit computer with 8-bit RAM addresses. The set of dynamic RAM chips are all identical, and they are connected in parallel as

Internal  
Memory  
Address

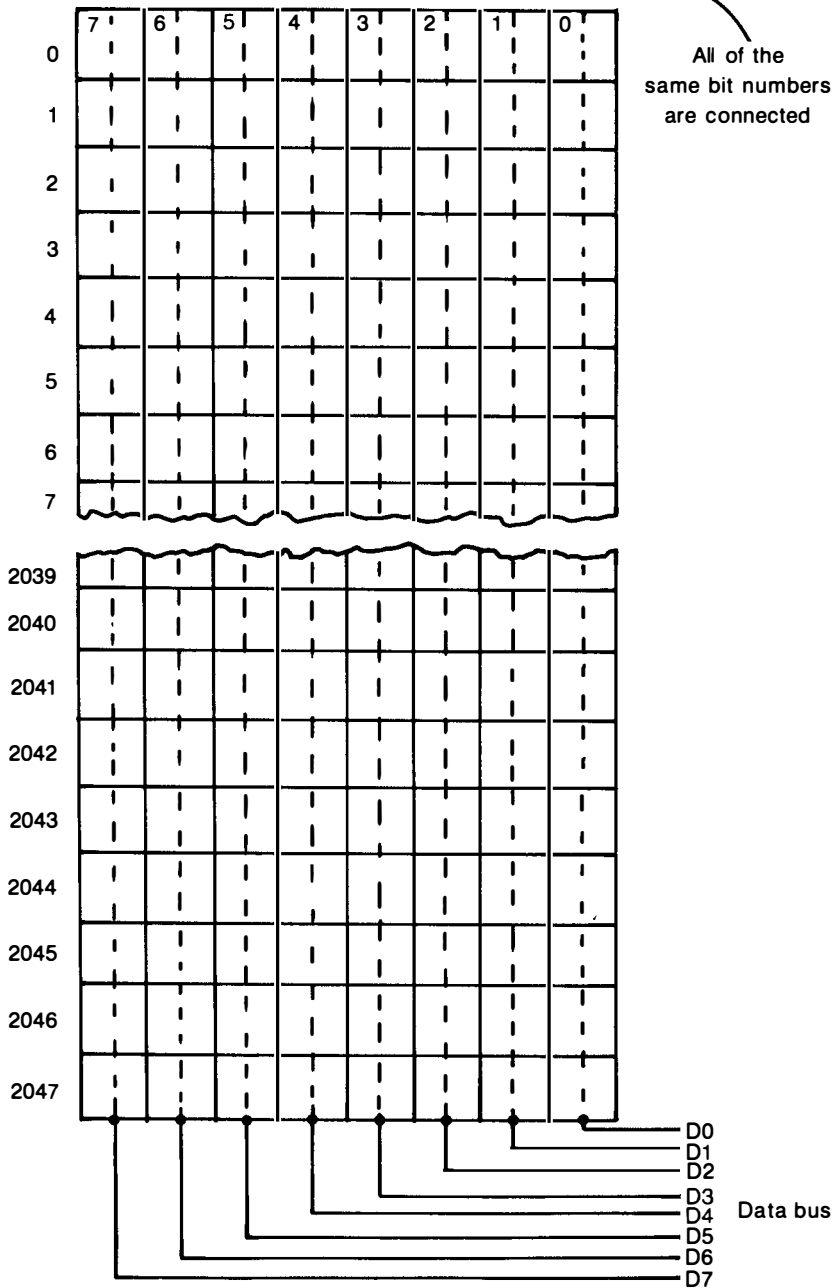


Fig. 6-5. Other static RAM chips are organized like the 2114 but with 8-bit holders, 7-0, instead of only four, 3-0. The addresses in this example have a row of 2048 addresses and eight columns for the data.

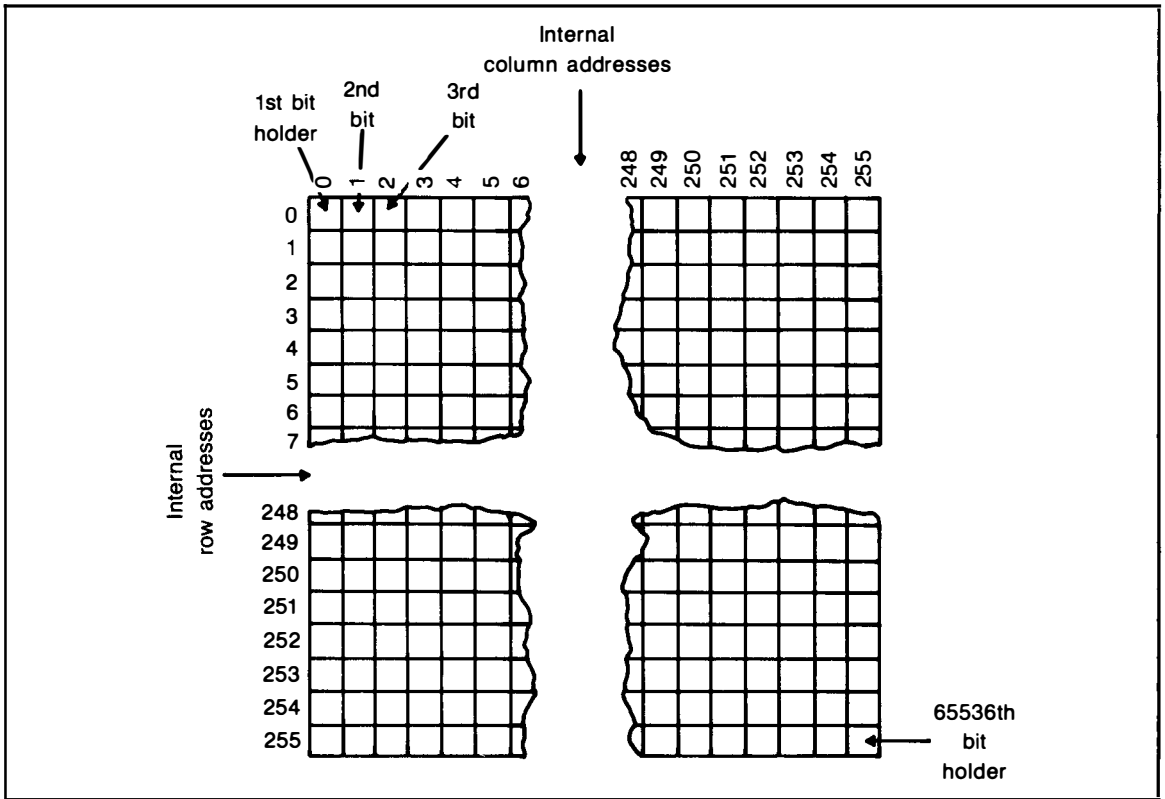


Fig. 6-6. The dynamic RAMs are organized in an entirely different manner than the static RAMs. There is only one bit in a register rather than four or eight.

shown in Fig. 6-7. Each chip has the same number of bit holders. That comprises 65,536 total addresses. With the chips in parallel, all the first bit holders of each chip combine to become an 8-bit register even though each bit of the register is on a different chip. Figure 6-8 illustrates this type of memory organization. All the second bit holders are a register, all the third bit holders are the next register and so on. Figure 6-8 illustrates the 8-bit register at location 1024.

If you look at the pin connections in Fig. 6-9, you'll see that the address pins are all connected together. If you dial up an address, for example 1024, you are connected to all eight chips simultaneously and can receive or send data in bytes. There are two data bus lines to each chip. They can receive or send the single bit to the desired address. One pin is to read data out of the

chips and the other is to write data into the chips. If you write to the RAM, a byte of data travels the length of the bus and the byte is stored, one bit to a chip. Should you read from the RAM, one bit will emerge from each chip and the net result is the desired byte of data.

The bit holders are not flip-flop circuits as in the static RAM. The bit holders are the capacitors that are formed in the MOS gates. Refer to Fig. 6-10. The gate is separated from the rest of the MOSFET by that sensitive glassy insulator. The gate becomes one lead of a capacitor and the rest of the FET becomes the other lead. The insulator takes the part of the dielectric. In this way, a tiny capacitance is formed. The capacitance, like any capacitor can be made to hold a charge.

In order for the charge to represent code for the binary data, the following convention has been

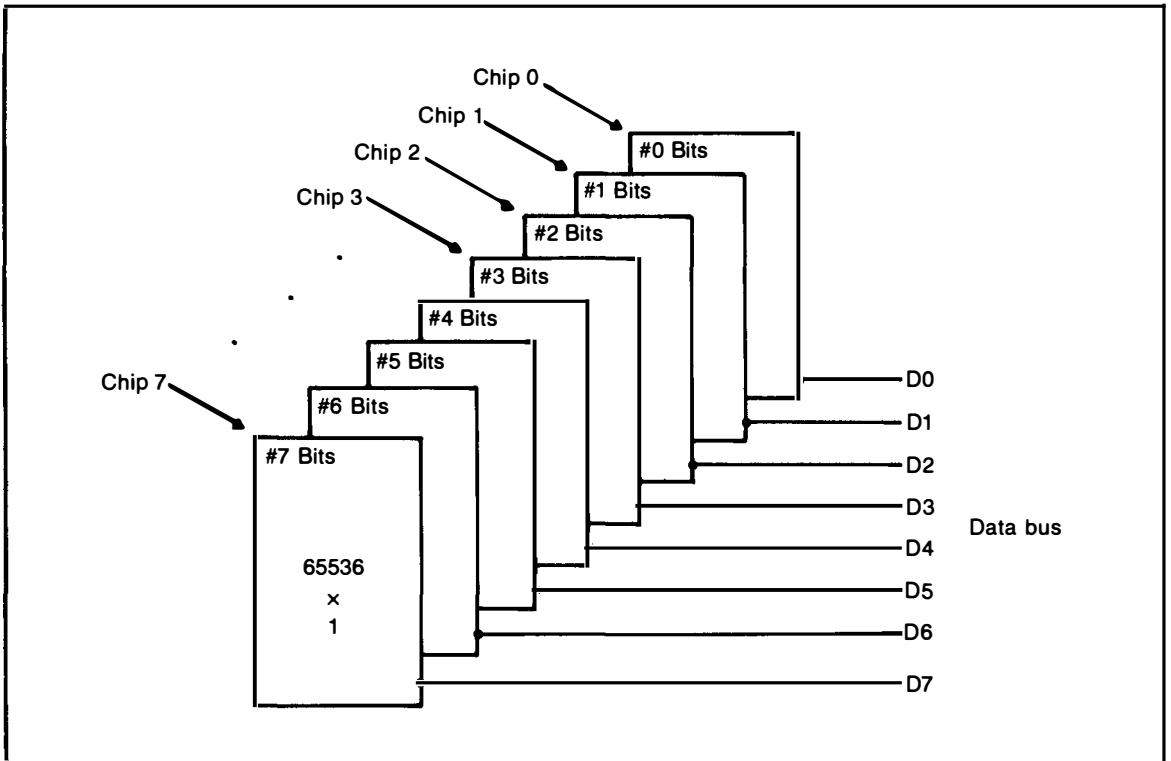


Fig. 6-7. Each RAM chip is attached to one of the data bus lines. All the #7 bits are contained on the same chip. All the #6 chips are on the same chip and so on. Between the eight chips, the 64K bytes can be formed.

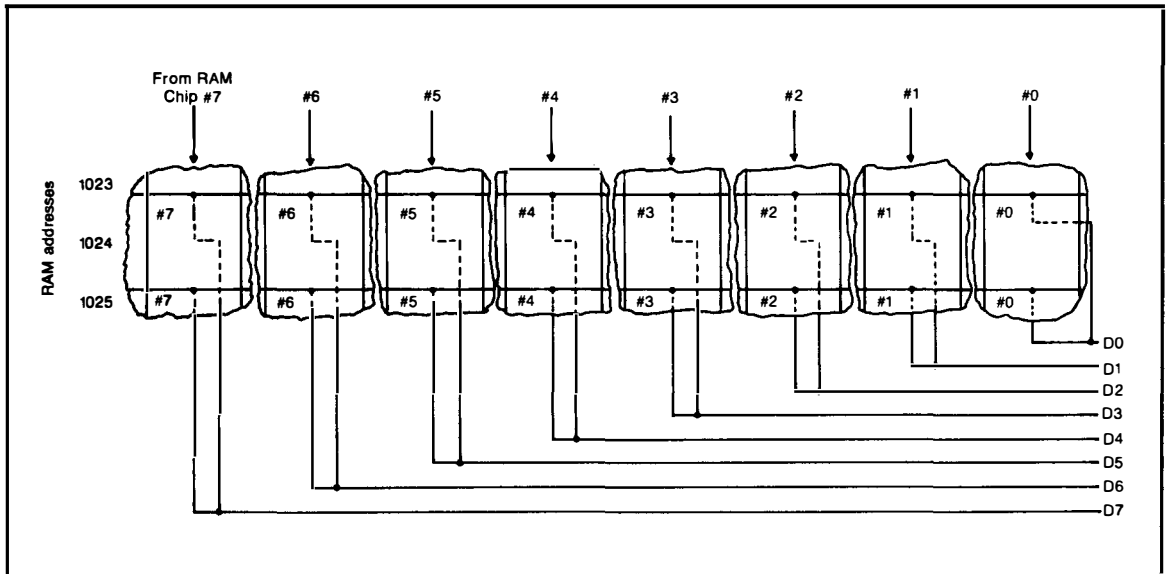


Fig. 6-8. RAM address 1024 is shown as one bit torn out of each RAM chip.

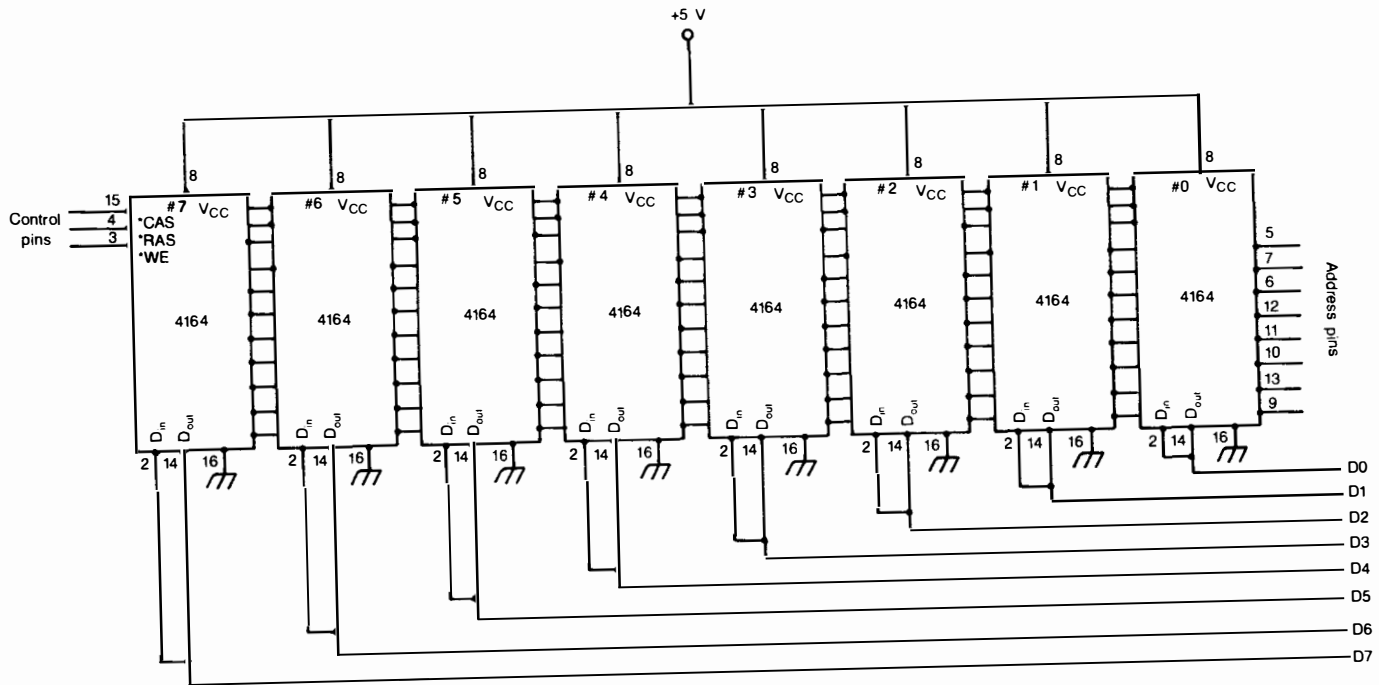


Fig. 6-9. While there is only one data bus line to each chip, all of the address bus lines go to all of the chips simultaneously.

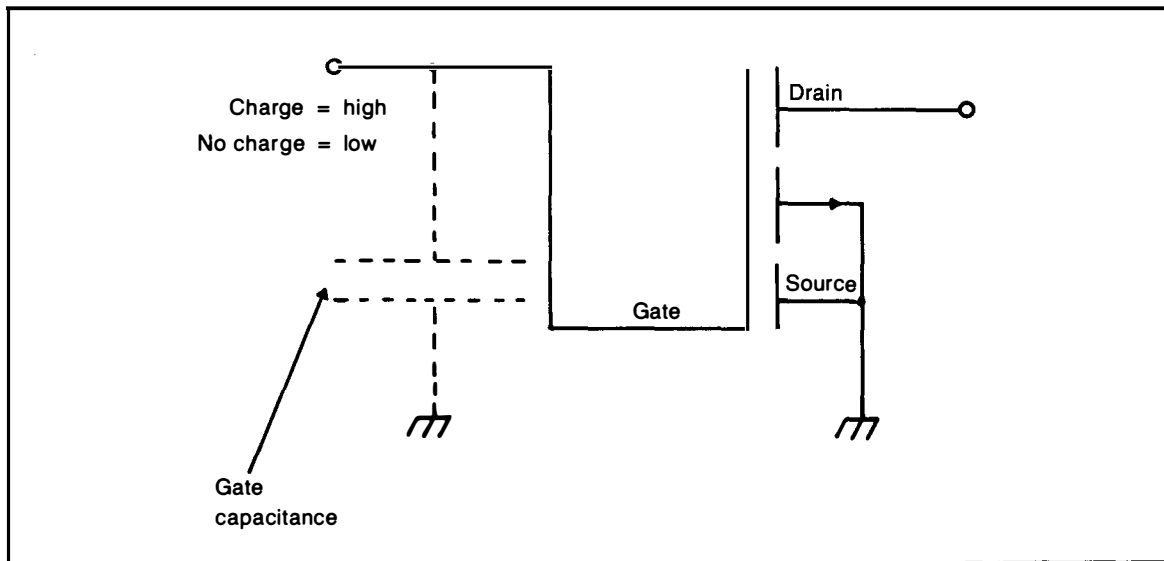


Fig. 6-10. Dynamic RAM does not use flip-flop circuits as bit holders. The gate capacitance of the FETs on the chip hold the highs and lows.

arranged. If the capacitor is charged, it represents a high or binary 1. When there is no charge, the capacitor is storing a low or binary 0. The capacitors have a tiny but adequate capacitance. The memory matrix on the dynamic 4164 chip is really nothing more than a clever circuit around a grid of tiny capacitors.

## Memory Refresh

The 2114 static memory chip, with its nybble sized registers, have four flip-flop circuits in each register. The flip-flop circuits are dc stable. That is why the chip is called static. If you place the circuit into a high state or low state (representing a 1 or 0), the flip-flop will hold that state as long as the power is on. Nothing more has to be done. If you want to read the state of a memory register, you address it. You can make as many copies of the data in a static register as you want without losing the stored state.

This is not the case with the dynamic RAM. The 4164 chips are storing their states as capacitor charges, not stable flip-flop transistor conduction. The capacitors contain tiny charges. The capacitor can only hold the charge for a short period

of time. To be exact, the capacitor charge will fall to a useless voltage level in a little more than 3.66 milliseconds (thousands of a second). The capacitors in the 4164s must be refreshed at least once every 3.66 ms.

As Fig. 6-11 shows, the VIC provides the refreshments. Pins 18 and 19 of the VIC output signals called \*RAS and \*CAS. These signals are applied to pins 4 and 15 of all eight 4164 chips at the same time. These signals from the VIC are designed to recharge the grids of the capacitor bit holders in the memory matrix.

In any computer using dynamic memory, some sort of refreshing system must be used to keep the memory full of data. There are all sorts of schemes. The 64 uses the VIC with its special refresh circuit. The VIC is able to refresh five 8-bit row addresses every raster line.

## Memory Layout

As Fig. 6-7 illustrated the bit holders are laid out in a grid of 256 rows and 256 columns. Actually, as Fig. 6-12 shows, the layout is divided in half with two grids of 128 × 256 with a band of sense amplifiers inbetween. But for all practical purposes,

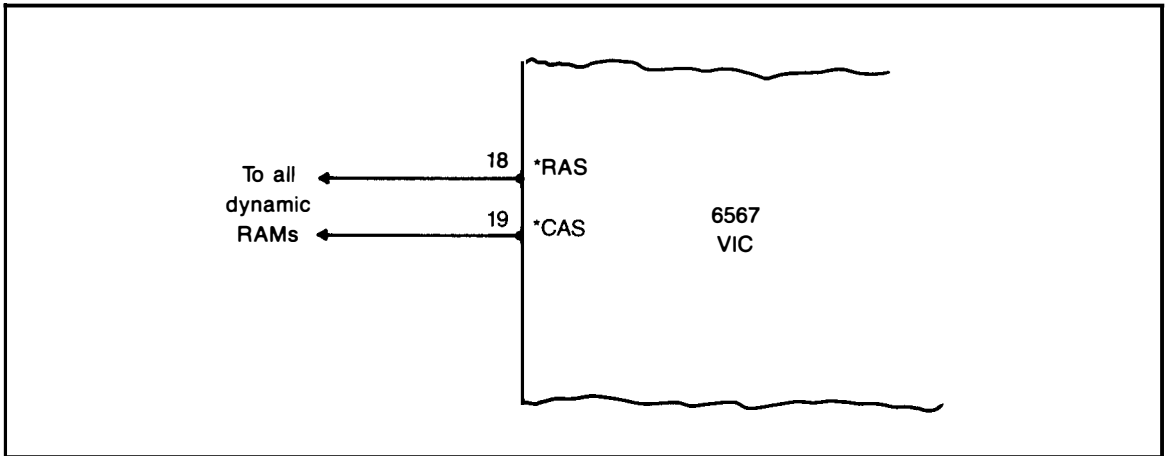


Fig. 6-11. Pins 18 and 19, \*RAS and \*CAS, are the signals that constantly keep recharging the tiny capacitance bit holders in the dynamic RAM.

you can think of the grid as  $256 \times 256$ . Consider the total grid as a single component.

Figure 6-13 shows a functional block diagram of the inside of a 4164 RAM chip. Attached to the bottom of the grid is a column decoder circuit. It is able to contact every one of the 256 columns. Connected to the side of the grid is a row decoder circuit. It is able to contact every one of the 256 rows.

The column decoder circuit is the data input/output circuit. When a bit in the grid is addressed and becomes activated, the column decoder is there to help out. If the bit is being read, the column decoder receives the high or low contents of the bit at one of its connections. Should the bit be the one that is being written to, the column decoder will be the circuit holding the data and will transmit the data to the addressed bit.

The row decoder circuit is part of the bit holder addressing mechanism. The column decoder circuit is also part of the addressing system. The column decoder helps out with the addressing in addition to handling the data in and out.

The dynamic RAM chip is addressed differently than the static RAM. From the minds eye, a static RAM is thought of as a high stack of individual byte locations. A dynamic RAM chip is a grid of bit-sized locations. Each location has its own address. The 4164 chip has 65,536 locations, but

they are arranged in a  $256 \times 256$  grid.  $256 \times 256$  equals 65,536.

A single bit is addressed by using two addresses: a column address and a row address. The desired bit holder is the location where the two addresses intersect. Therefore, the address sent out by the MPU over the address bus is in two parts. Eight of the address bits are to locate one row address, and the other eight bits find the column address. The address bits go to all the RAM chips

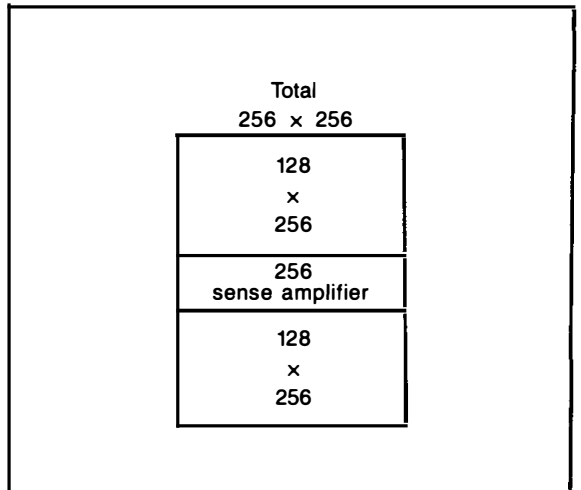


Fig. 6-12. If you could see the microscopic insides of the 4164, you'd find two grids with a set of sense amplifiers between them.



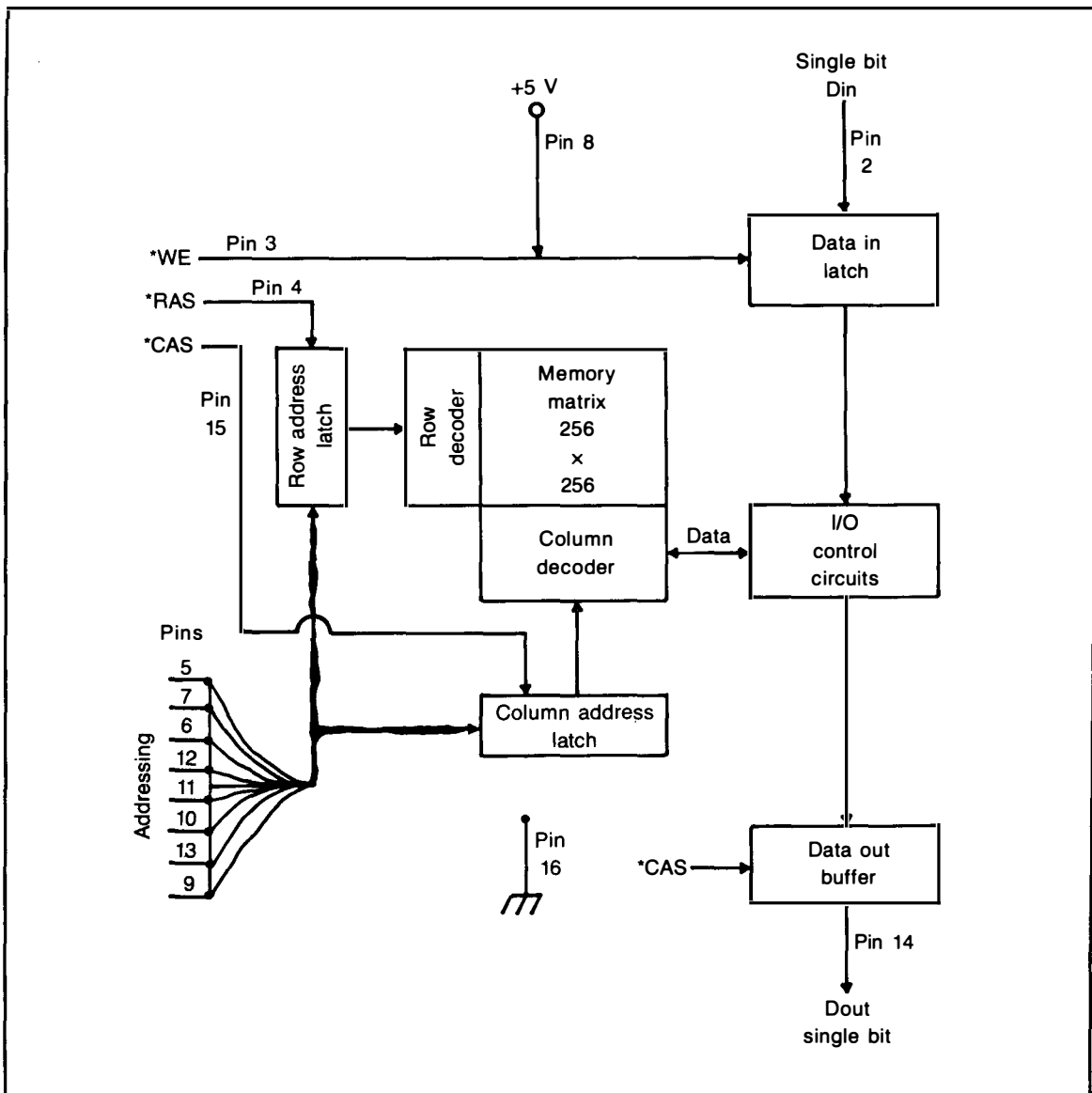


Fig. 6-13. An address in the chip is located by addressing the 256 rows of addresses with eight bits. Then the 256 columns of addresses are addressed with the remaining eight address bits. Where the addressed row meets the addressed column is the desired bit. That bit can be read or written to.

at the same time. The same location bit, on each chip is addressed. The contents of each bit that is addressed can then enter or exit each RAM.

The RAM chips in the 64 are addressed through two 74LS257 multiplex chips. The multiplexer is needed to help out because the ad-

ressing gets complicated. There is more about this addressing in Chapters 12 and 17. The internals of the 74LS257 chip are discussed in Chapter 8.

### Operation

Figure 6-14 shows the pin definitions for the

4164 dynamic RAM chips. The 16-pin DIPs are powered with +5 volts at pin 8 and the chip is grounded at pin 16. There are eight input address lines, MA0-MA7. The eight lines are derived from the 16 address lines that are connected to the multiplexers, the 74LS257s. These eight address lines enter all of the chips in the 64K byte array in a parallel manner. That way all eight chips have their bits addressed simultaneously.

There are 16 bits needed to address the 64K bytes in the array. Eight bits address the 256 rows and eight bits address the 256 columns. The multiplexers handle the arrangements along with the two inputs \*RAS and \*CAS.

At pin 4, the \*RAS signal enters. At pin 15 \*CAS arrives. RAS stands for Row Address Strobe. CAS means Column Address Strobe. The row ad-

dress bits approach the multiplex-memory array on eight address bus lines. The column address bits arrive at the array via the remaining eight address bus lines.

The two multiplex chips receive the two sets of bits, as Fig. 6-15 shows. The multiplexer then feeds the two bit sets alternately to the array. The multiplexer performs this bit feeding in sync with the signals \*RAS and \*CAS. During \*RAS the multiplexer feeds the row address bits and during the \*CAS it feeds the column address bits. That way all 16 bits are injected into the eight 4164 address lines, eight bits at a time.

The array accepts the row addresses during \*RAS, and then over the same lines, accepts the column addresses during \*CAS. This multiplexing scheme uses only eight pins. The saving of eight

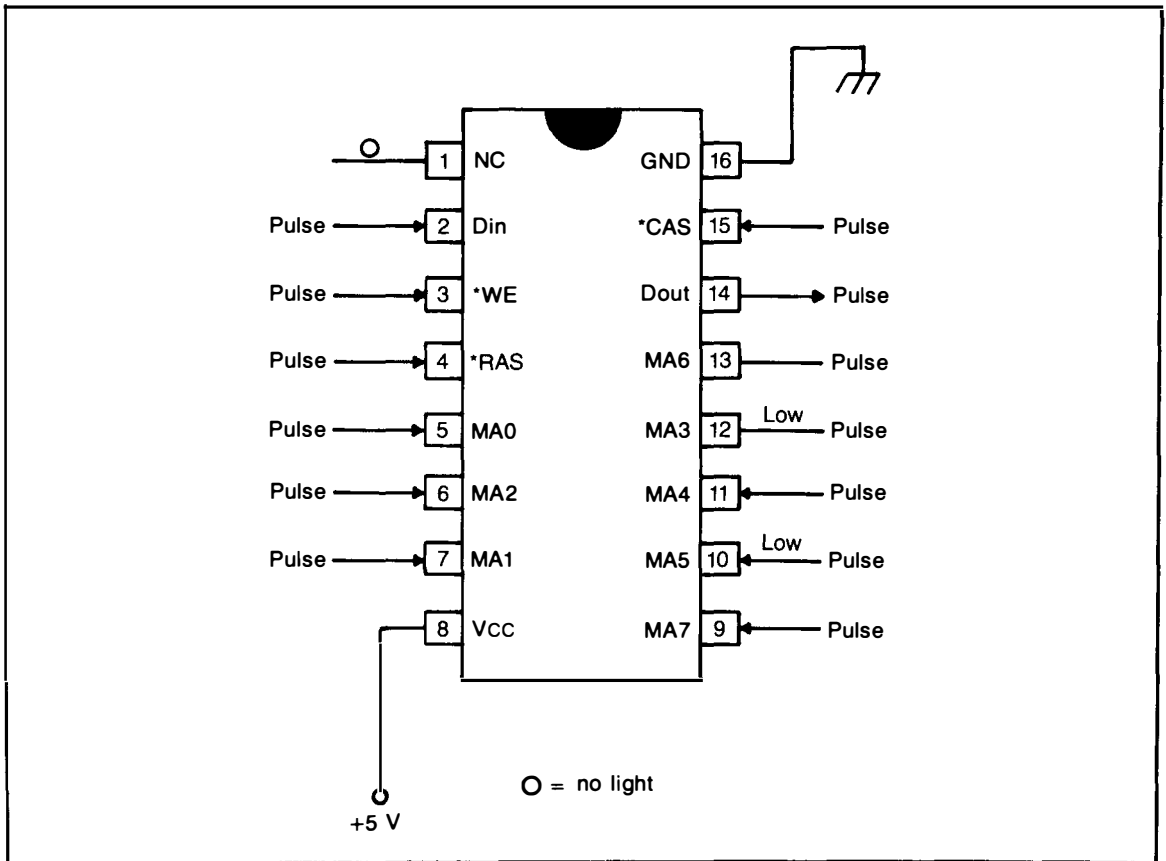


Fig. 6-14. This test point chart is for all eight chips. They should all show the same readings if they are operating properly.

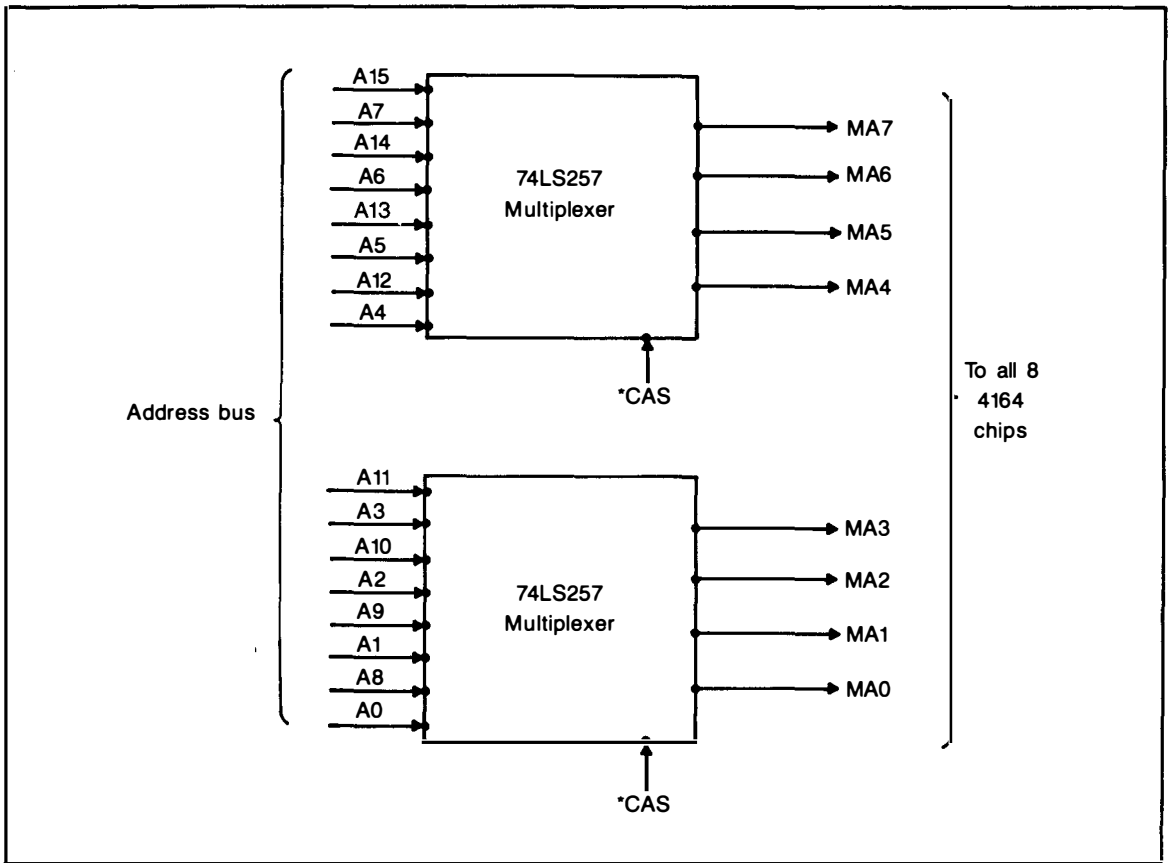


Fig. 6-15. The address bits are multiplexed through a pair of 74LS257s before application to the 4164s.

lines allows the 4164 chips to be placed in a 16-pin package.

Recall from Fig. 6-9 that the 4164 has two data pins at 14 and 2. Note the data pins connect together and then attach to one of the data bus pins. Chip 7 attaches to data bus line D7, chip 6 attaches to data bus line D6 and so on. The two lines are for input and output. Pin 2 is data-in and pin 14 is data-out.

Internally the two pins work with the column decoders. Refer to Fig. 6-13. The data-in connection is attached to a data latch. The data latch is a temporary holding register for the incoming data. The data latch is also connected to pin 3, \*WE, the write enable pin. During a write operation, \*WE goes low. When it does go low it activates the latch and lets the latch transfer the data to the column

decoder. The decoder will then install the data into the addressed bit holder.

When data is leaving the chip it is passed from the bit holder to a data output buffer stage. The buffers are connected to the \*CAS inputs. When \*CAS strobes the data output buffer permits the outgoing data to leave the chip and head out over the data bus.

### Timing

The timing diagram in Fig. 6-16 summarizes how the memory array is controlled by the three signals \*RAS, \*CAS, and \*WE. During addressing, the \*RAS strobes in the eight row address lines. Eight lines can have 256 different bit combinations. The combination of bits chooses one of the 256 rows of bits. A row address latch in each

chip accepts the row address and feeds it to its address decoder. That row in each of the eight chips is activated.

The \*CAS then strobes in the eight column address lines. A column address latch in each chip accepts the column address and feeds it to a column decoder. The decoder chooses a column address. The corresponding columns in each chip are chosen at the same time.

Meanwhile the \*WE signal sets up either a read or a write condition for the eight selected bits, one on each chip. A low at \*WE causes the data-in latch to pass data written to the array to be installed in

the memory bits. A high at \*WE keeps the data-in latch shut and lets the chip output the addressed data to the data bus.

As you can imagine, the timing of the three signals are critical. However, during routine servicing the timing is not normally a factor requiring testing. The timing is an important consideration during design. The important part that timing plays from a repair point of view is its feature during a chip replacement. You'll see the timing in the spec sheet of a new part referenced as *access time*. For instance, the 4164 could come in three timing versions. The access time on different 4164 chips could

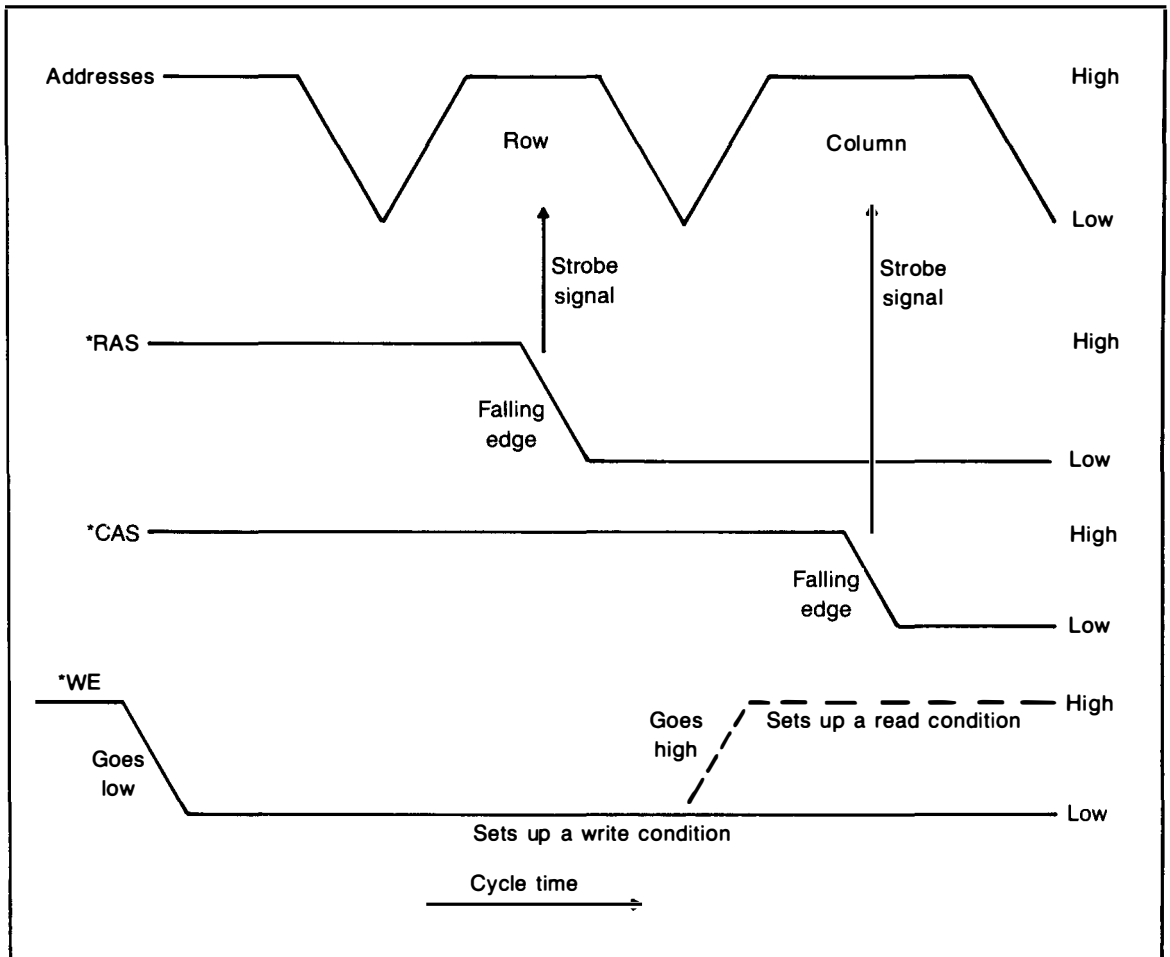


Fig. 6-16. The 4164 chips are controlled by the three signals \*RAS, \*CAS and \*WE. During a cycle, the row addresses are strobed and then the column addresses are strobed. During that time \*WE sets up a read or a write condition.

be 150 nanoseconds, 200 nanoseconds or 300 nanoseconds. What does this mean?

Different computers run at different speeds. The Commodore 64 runs at a clock rate of 1 MHz. This means that the 64 completes one million full cycles in one second. One cycle takes a millionth of a second, or as it is called, a microsecond. In one microsecond, there are 1000 nanoseconds. A nanosecond is one billionth of a second. When a 4164 chip is said to have an access time of 200 nanoseconds, that means it uses 200 ns of the 1000 ns in one clock cycle. The clock cycle comprises one full execution of the MPU.

During the 1000 ns execution cycle, the MPU must address the 4164 array, send the signals to control it, open up the addressed bits, either write to the bits or read them, and then prepare for the next cycle.

A 200 ns array is fairly fast and can easily receive the control signals, open the addressed bits, and complete a write or read. In fact, even the 300 ns chips are also easily accessed. If on the other hand, you had some sort of chips that had a 1000

ns access time, you would be hard put to get them fully accessed with a 1 MHz clock. You'd have to get a slower clock or add additional circuitry to gain proper access.

## Refresh Timing

Now that we got into timing, a question might have arisen about refreshment time. Specifically, there are 64K bytes in the Commodore 64 memory array. It was mentioned that every bit must be recharged at least every 3.66 milliseconds or it will lose its information as the charge drops below a certain voltage. Incidentally, notice that the refresh time value is milliseconds, a thousandth of a second in comparison to the micro and nanoseconds that I just mentioned.

In order to recharge the tiny capacitances, all you have to do is address each one. The electrical action of the addressing restores the charge. This means every address must be contacted within the 3.66 millisecond period. There are 65,536 addresses. At the 1 MHz clock rate of the 64, it is clearly impossible to contact every address at 200

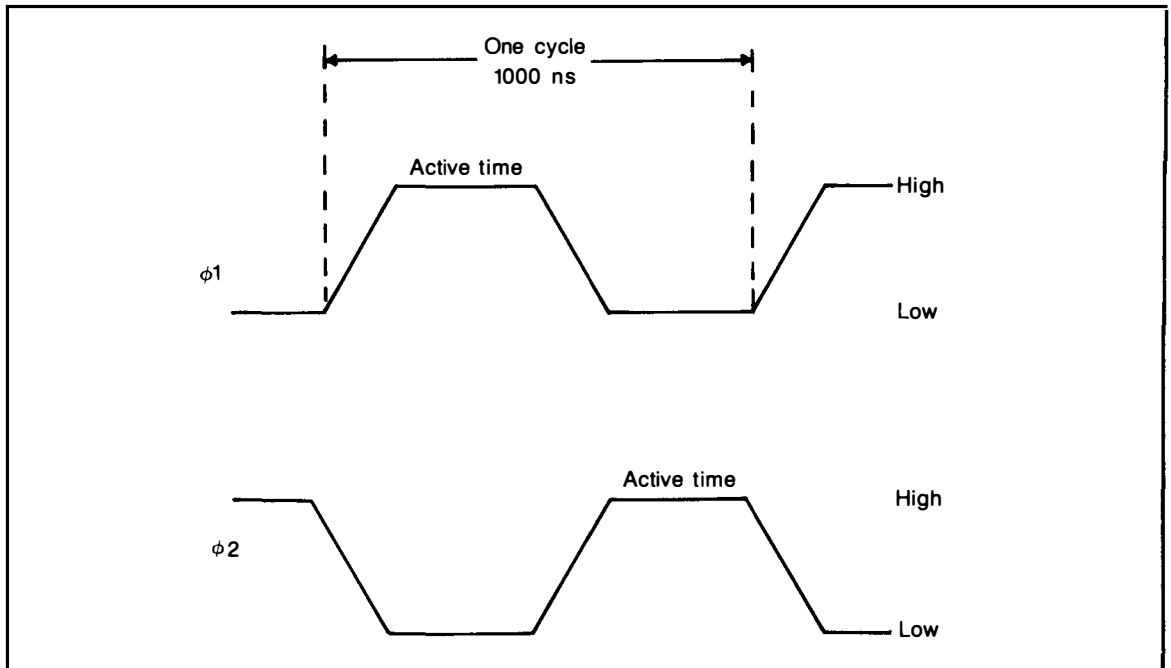


Fig. 6-17. The 64 has a 1000 ns cycle time. About half the time the signal  $\phi 1$  is active and the other half  $\phi 2$  runs the operation.

ns each and still be able to do the rest of the computing. It would take more than 13 ms. Fortunately, there is no need to contact every address. All that is required is to contact every row. There are only 256 rows.

A dynamic RAM refresh controller in the VIC is able to refresh all 256 rows in under 3.66 milliseconds. The refresh circuit uses the \*RAS signal to address the rows. It does the job while the computer is busy doing other jobs, not during the time the RAM is being accessed.

The 6510 is controlled by two clock signals shown in Fig. 6-17. They are called phase 1 ( $\phi 1$ )

and phase 2 ( $\phi 2$ ). These will be discussed in depth in Chapter 14.  $\phi 1$  is a signal that drives the internal circuits of the MPU. It is active about 500 nanoseconds of the 1000 ns total cycle time.  $\phi 2$  is a similar signal that is active the other 500 ns of the 1000 ns cycle time. It drives the rest of the computer, external to the MPU. The 4164 array is accessed during  $\phi 2$ .

During  $\phi 1$ , while the MPU is busy internally and the 4164 array is waiting to be accessed, the VIC goes ahead and refreshes the 256 rows in the array.



## 7. Read Only Memory

**T**HE COMMODORE 64 CONFIGURATION CAN handle four ROM chips. It has three ROM chip sockets and one cartridge expansion port. All four entrances are connected directly into the memory map. This means the connections of the sockets and port are wired to the address bus, data bus, and control lines from the MPU. There is no need to use an I/O interface like the CIAs. The ROMs are plugged directly into the thick of the digital world.

The three sockets on the print board accommodate the three ROMs that come with your computer. They are the 4K 2332A Character ROM, the 8K 2364A Kernal ROM, and the 8K 2364A BASIC ROM. The two 2364A ROMs, even though they have the same number, have thousands of differences.

In the last chapter, it was shown that RAM is a warehouse for binary bits. The bit can be installed, removed, and reinstalled continually. ROM on the other hand is hand wired. The bits in the ROM are burnt into the bit holders and are there on a permanent basis. Once a bit is burnt onto the

chip there is no removing it. To change the information at a location, the ROM must be physically changed.

The two 2364A ROMs house the Kernal and BASIC operating systems. They started out life as identical chips. They were built to hold binary bits in byte sized locations like a RAM. However a RAM chip has either flip-flop circuits or capacitances to hold bits while the computer is activated. A ROM chip has neither. In each bit holder, a component is installed. For example, the component could be a diode as shown in Fig. 7-1.

The ROM, in true memory fashion, has rows and columns. The rows are attached to the address bus and the eight columns are connected to the data bus. Each row has its own address. Each column is connected to every address. Between the rows and the columns, a diode could be wired in. There is one microscopic diode between each row and every column. In Fig. 7-2, row address 0010 has eight diodes attached. The cathodes of the diodes are connected to the row side. The eight anodes of

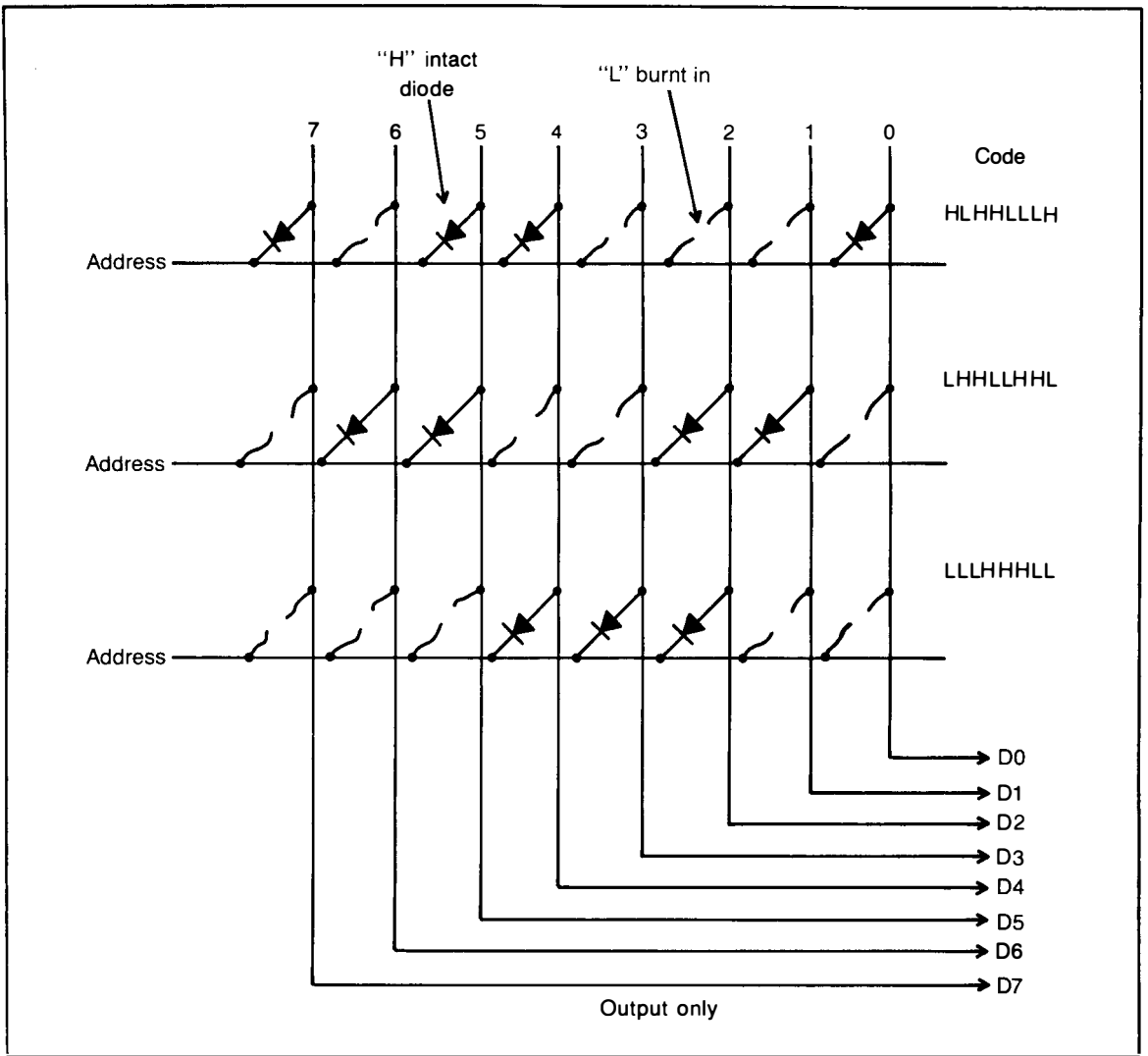


Fig. 7-1. A byte of bits can be burnt into an address with diodes. An intact diode can output an H voltage. A blown diode can't pass any voltage, which is an L.

the diodes are wired to the column side. While all eight cathodes are connected in common to row 0010, the anodes are attached separately to the eight columns.

The columns are in turn attached to the data bus. When address 0010 is dialed up, the eight diodes will all conduct, cathode to anode. The columns D7-D0 will all be applied with a high voltage. This is the binary signal representing 1. The col-

umns will then output 11111111 to the data bus. From there, the 1s will go to the MPU.

Suppose you are the manufacturer of this ROM. You do not want address 0010 to output 11111111. You want address 0010 to output binary 10011100. In order to effect the change, you must alter some of the diodes to produce the change in signal. An analysis of the change shows you want to alter bits D6, D5, D1, and D0. Bits D7, D4, D3,



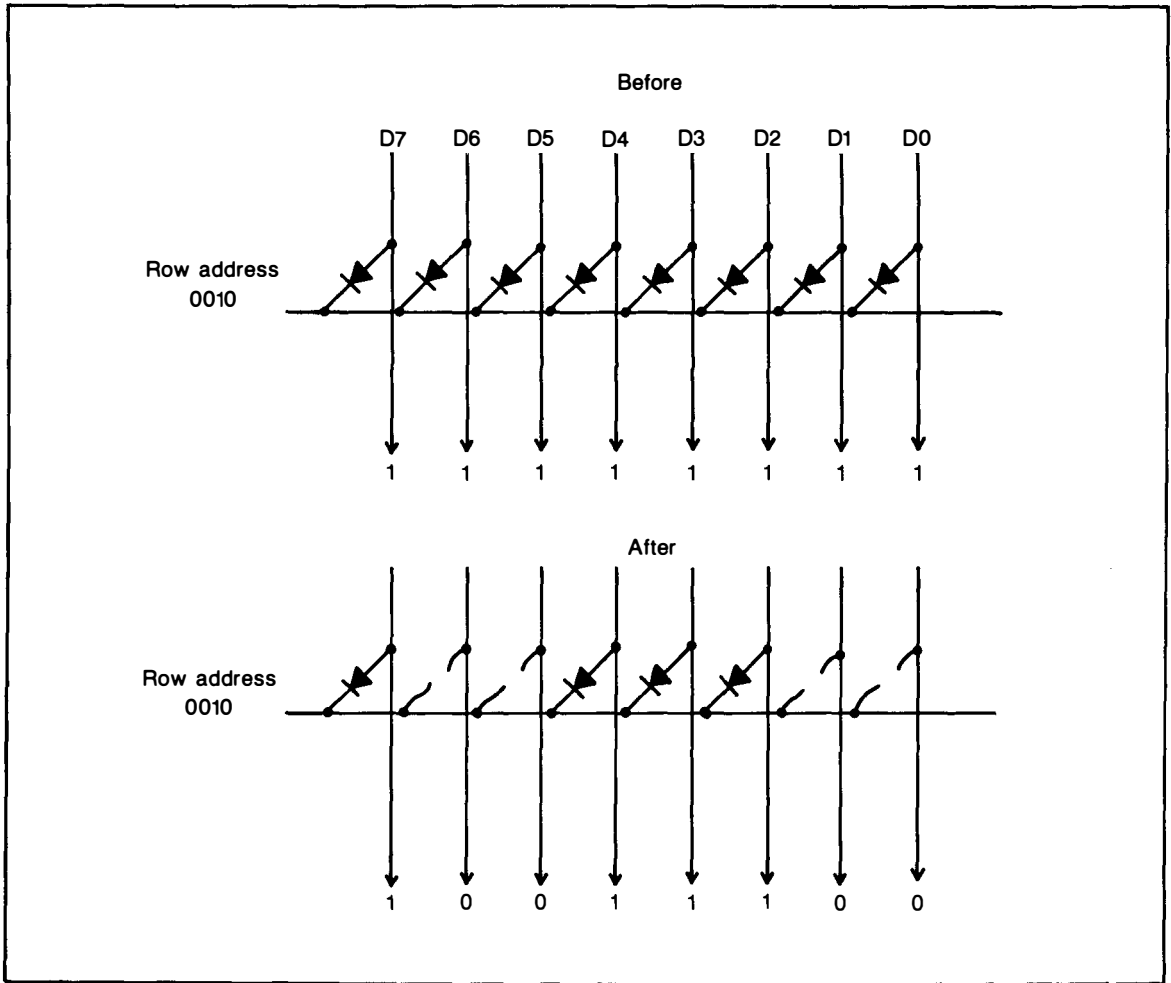


Fig. 7-2. A row address on a ROM can start out life with eight intact diodes. It will output eight 1s, representing 1s. Each bit position that then has its diodes blown out, outputs 0s, representing 0s.

and D2 should be left alone since they are already outputting 1s. But you want 0s installed in bits D6, D5, D1, and D0.

The technique is to place a large voltage onto the diodes where the 0s are to be installed. The diodes from the row address to columns D6, D5, D1, and D0 are thus blown. They become open circuits. They can no longer output a high voltage. With the conducting diodes representing 1s and the blown diodes representing 0s, the data at address 0010 becomes the desired 10011100.

While this procedure will work, during actual

ROM manufacturing nothing so slow is used. When a ROM is designed, large printed circuit patterns are made. The junctions that are going to contain a diode or transistor are clearly defined on the pattern. They become the row-column locations that will be outputting 1s. The row-column locations that are to designate 0s are left open. Once the patterns are produced, they are reduced photographically and used to manufacture the ROM.

So while the 2364s started out life as identical, once they have their individual patterns burnt in, they become different entities. In the 64, the two

2364s become the Kernal and BASIC ROMs.

## BLOCK DIAGRAM

There are four main parts to a typical ROM besides the power in and the ground. For example, let us look over the 2332A, the Character ROM. The pinout of the 2332A in Fig. 7-3 shows 24 connections. The connections all lead to microscopic circuits that are completely inaccessible to you. Although the circuits are too tiny to work directly with, you can make tests accurately from the pins on the chip.

In the block diagram in Fig. 7-4, note the memory matrix. The memory is organized as  $4096 \times 8$ . This means that there are 4096 byte sized locations in the ROM. The locations are figuratively stacked in a tall skinny pile with location 0 at the

very top and location 4095 at the bottom. Actually, the matrix is laid out in a grid fashion like the dynamic RAM, but it is wired as a tall pile, like the static RAM, so we should consider it in that way.

In order to address 4096 individual locations, an address bus with 12 lines is needed. There are 4096 separate combinations that can be made with 12 bits. In order to address one of the internal locations, the 12 lines can choose whichever one of the group it desires. For instance, if you want to address location 185, you'd send bits 000010111001 out over the 12 address lines. Those highs and lows would activate location 185 and leave all the rest of the locations shut down.

The 12 address lines A11-A0 are the connections made to the pins 18, 19, 22, 23, 1, 2, 3, 4, 5, 6, 7, and 8. The lines enter the package and are

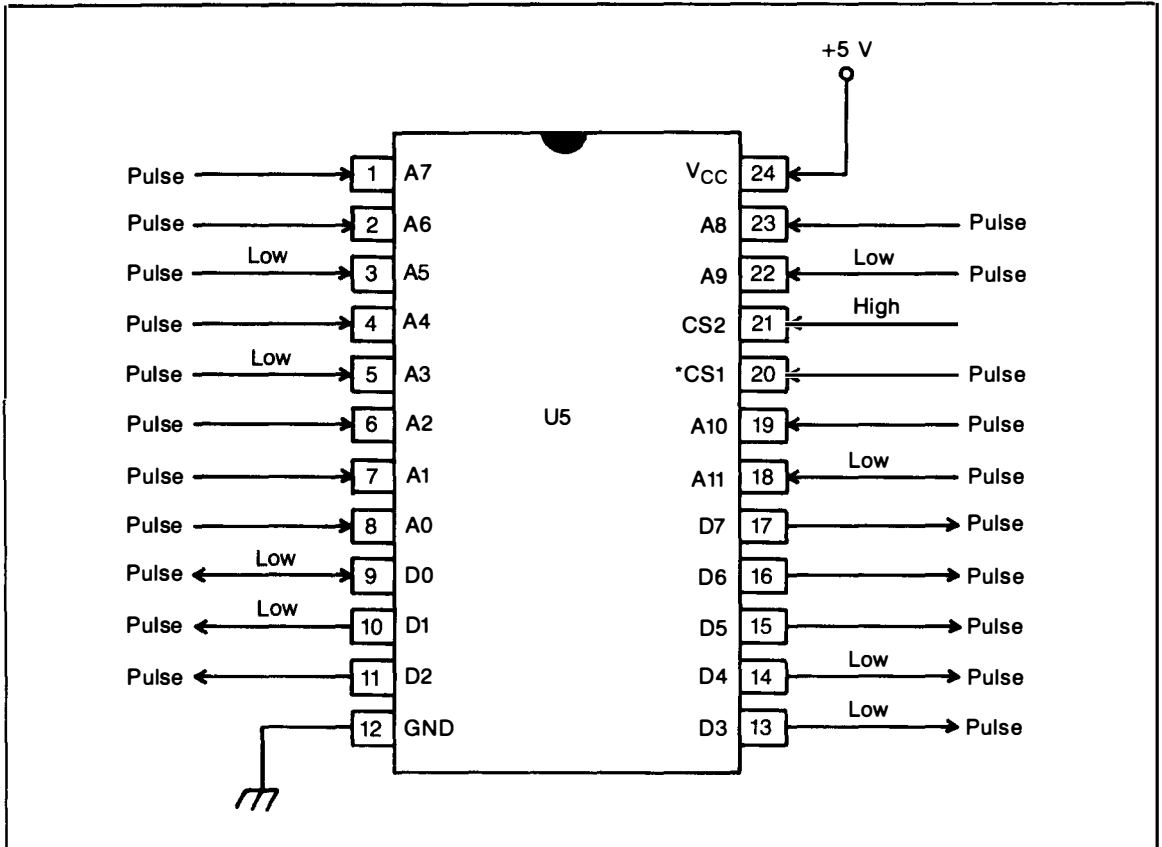


Fig. 7-3. The 2332 Character ROM will read these highs, lows, pulses, and voltages on a logic probe.

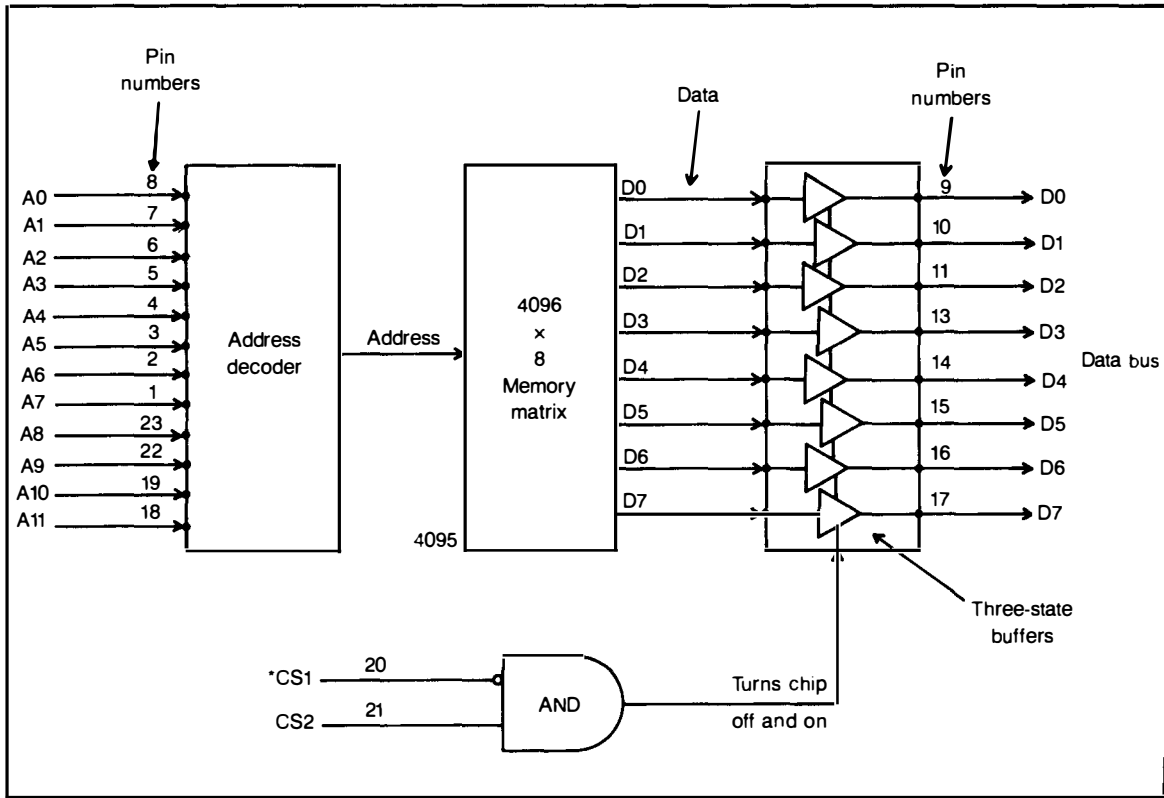


Fig. 7-4. The ROM is turned on with the two CS pins. The internal addresses are turned on with the address lines. At that time, data from the location can exit through the buffers and out of the data pins.

connected to an internal circuit called the address decoder. In the decoder, the highs and lows are evaluated and control voltages are produced. These voltages are processed into the memory matrix and activate the single location that was desired.

Once the location is accessed, the voltage applied at the row address follows the wire pathways that Fig. 7-2 demonstrated. The paths are through components, like diodes, that are connected from the row line to the column lines. The pathways with intact components pass the high voltage. The pathways that were blown in manufacturing do not pass the high voltage, but instead, maintain a low voltage. The byte of highs and lows travel out the columns to the next stage. These bit holders can only transmit highs and lows, or be read. They cannot receive highs and lows, or be written to. The bit patterns are permanently burnt into the mem-

ory matrix.

The next stage in the ROM is a set of three-state buffers. The buffers can be turned on and off by another circuit in the ROM, which we'll discuss next. Then the buffers pass the highs and lows from the eight matrix output columns to the eight connections that attach to the data bus, D7-D0. They are pins 17, 16, 15, 14, 13, 11, 10, and 9.

Between the incoming 12 address lines and the eight data lines, 20 of the pins are accounted for. Pins 24 and 12 are +5 volts and ground. This leaves two pins unaccounted for. They are pins 21 and 20, called CS2 and \*CS1.

The asterisk indicates that the pin is activated when a low voltage is applied. Another computerese word that means activated or turned on is *enabled*. Therefore the asterisk is said to indicate a condition whereby a pin is enabled when a low

voltage is applied. \*CS1 is enabled by a low voltage. A pin name without an asterisk usually indicates that the pin is enabled if a high voltage is applied. The low voltage is typically thought of as near zero volts while a high voltage is considered roughly as being near +5 volts.

These two pins \*CS1 and CS2 are chip selects. The chip will be enabled only if \*CS1 receives a low and CS2 receives a high. Otherwise, the chip will remain dormant. To sum up, the chip is enabled by one out of four possible conditions. The conditions are shown in Table 7-1.

The two chip select pins are connected to an internal AND gate (Refer to Fig. 7-4). AND gates are discussed in detail in Chapter 10. This particular AND gate circuit will only be enabled by the application of a low through \*CS1 and a high through CS2. When this combination of correct voltages are applied, the AND gate conducts. The gate is in turn connected to the three-state buffers in the ROM output. The AND gate conducts when the correct combination of a low and high are applied at pins 20 and 21, thus enabling the three-state buffer. The three-state buffer allows the addressed location to output its data contents to the data bus. In this ROM, the data contents are part of a character that is to appear on the TV screen.

CS2 at Pin 21 requires a high to be enabled. On the circuit in Fig. 7-5, you can see that CS2 is tied directly to the +5 volt power line. The +5 volts in this case is performing two separate jobs. It is powering the entire chip at pin 24. There is a 0.47  $\mu$ F bypass capacitor also connected there

help filter the +5 volts. Besides the powering duty, the +5 volts is also meeting the requirements of CS2. It applies the +5 volts to CS2 as the enabling high voltage. This connection is permanent. The +5 volts holds the CS2 pin high all the time. In this computer, CS2 is really extra baggage. It is not needed for the chip select operation. To keep it out of the way, it is held high and always enabled. \*CS1 does all of the chip selection. In a different computer, both CS2 and \*CS1 might be needed, but in the 64 only \*CSI is needed for the chip selection chore.

\*CS1, pin 20, is connected to pin 16 of the PLA chip, which is covered later in the chapter. Pin 16 of the PLA is outputting a signal called \*CHAROM. The \*CHAROM signal stands for Character ROM. It is the signal that enables the Character ROM. When the \*CHAROM signal goes low the Character ROM will turn on and output the contents of the location of the address bus. The data bus will receive a character from the ROM.

A lot of the time \*CHAROM is held high and the chip stays off. The \*CHAROM signal originates at the MPU. It is routed to the 2332A chip via the PLA. The full name of the 2332A chip is Character Generator ROM. The 4096 locations in this chip contain all the shapes of the characters you might want to display on the TV. The letters, numbers, punctuation symbols and the other shapes you see on the keyboard. When you program the 64 to display those shapes the 6332A is enabled with a low on line \*CHAROM and the ROM character patterns are output from the burnt in bits in the matrix.

Each displayed character is composed of a grid of dots in an 8  $\times$  8 matrix of light. In order to see one full character, you must address eight ROM locations. There is a set of eight bytes for each character. Each byte is one row in the pattern. Each bit in a byte has control over one dot. A high from a bit holder causes a dot of light to go on. A low from a bit holder makes a dot of light go off.

The output of the Character ROM leaves the chip through the data lines in the data bus, D7-D0. The output goes to D7-D0 of the VIC. The input to the VIC receives the character shape information and processes it onto the TV display.

**Table 7-1. 2332 Rom Chip Select**

	Input Pin 20 *CS1	Input Pin 21 CS2	Chip Reaction
Four possible conditions	1 L	L	OFF
	2 L	H	ON
	3 H	L	OFF
	4 H	H	OFF

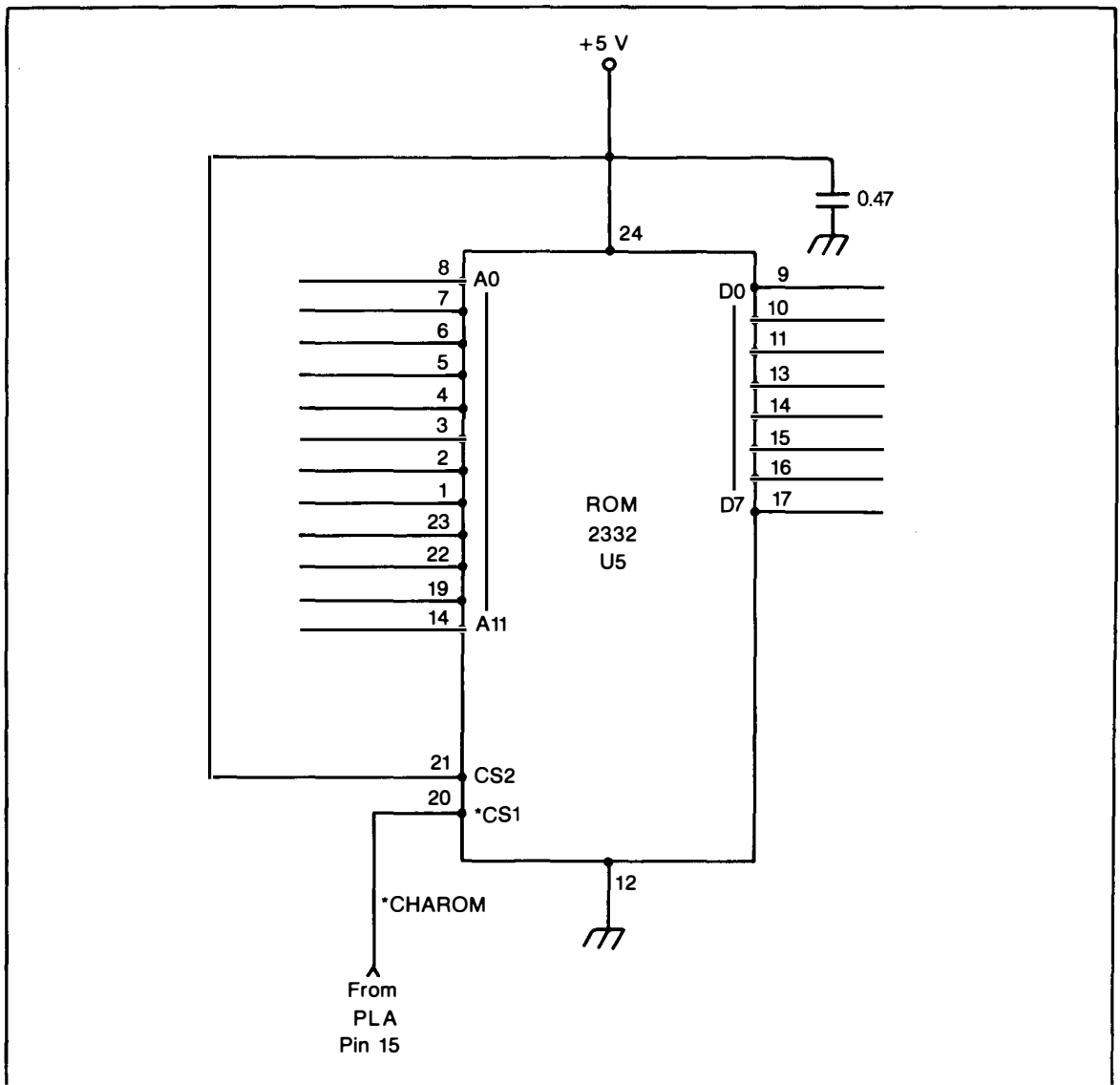


Fig. 7-5. Pin 21 is always held high because it is permanently attached to +5 volts.

## CHARACTER ROM

The Character ROM has a memory of 4096 bytes. Internally they are addressed from 0 to 4095. In the 64K computer memory scheme, the ROM is addressed from location 53248 through 57343. Each eight bytes contains the dot information for one complete character. The first character in the

ROM, found at locations 53248 to 53255, is the symbol @. @ is the first character code, and it is called code number 0. Code number 1 is the second character in the set. It is located at 53256 through 53263 and is the capital letter A. Recall from Fig. 5-13 how the binary contents of the locations and the resulting A formed in one block on the TV screen.

In the 4K ROM, 512 characters can be stored. Since each character takes up eight bytes and  $512 \times 8$  equals 4096, it is easy to see how the storage is accomplished. The 512 characters are the total of two sets with 256 characters in each set. The first set contains all the capital letters and graphics. The second set has a full repertoire of upper and lower case letters. When you first turn on the 64, it comes up in the standard capital letter character mode. This is the mode used for programming. The screen

display codes of the Character ROM are shown in the User's Guide that came with your Commodore 64.

## KERNAL AND BASIC ROMS

A computer is helpless without an operating system (OS). An *operating system* is a program that the manufacturer installs in the computer to run things. The operating system can be placed into RAM or ROM. It doesn't much matter as long as the con-

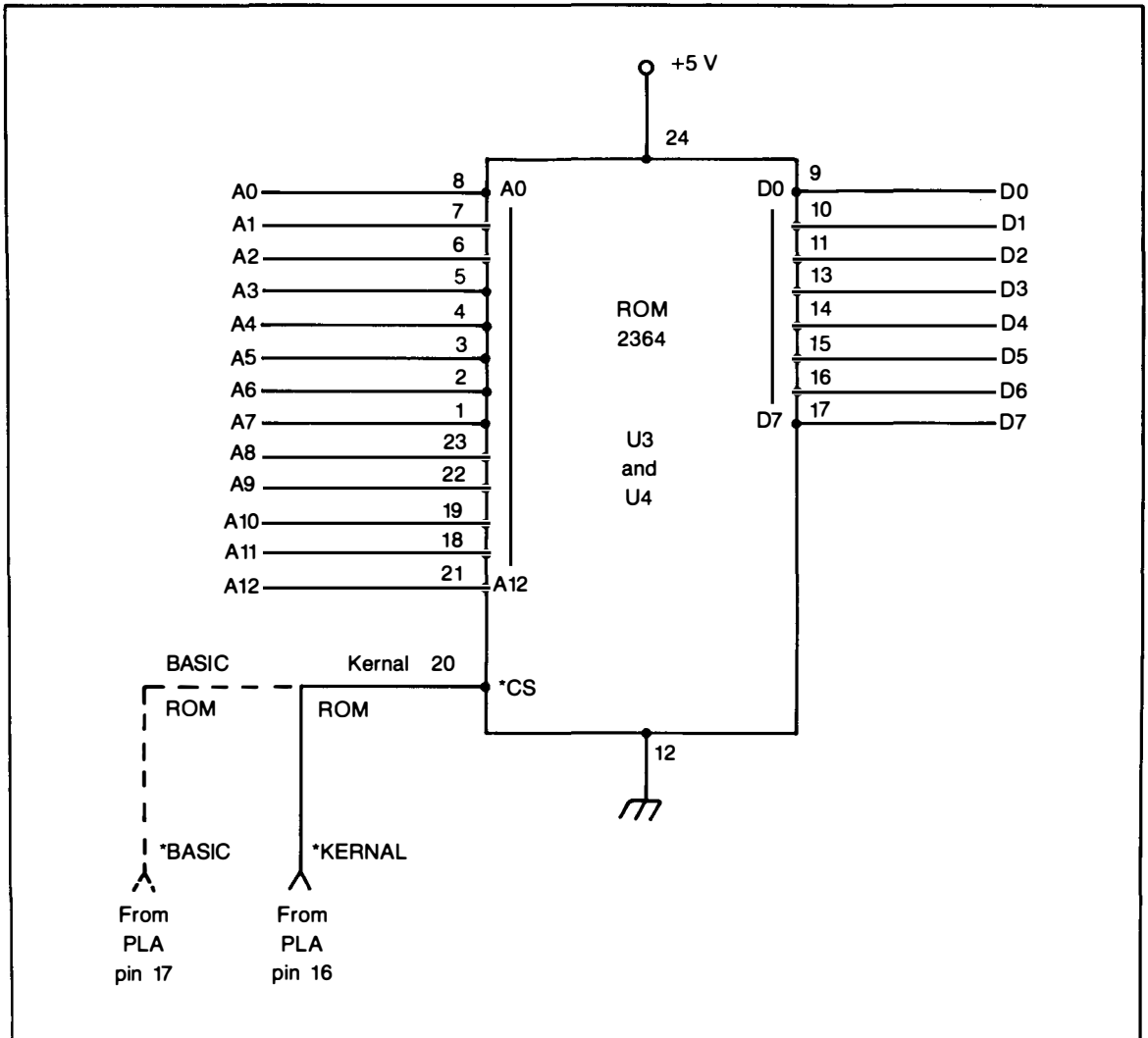


Fig. 7-6. The schematic drawings of the BASIC and Kernal ROMs are identical except for PIN 20, the chip select. The only internal differences are in the data burnt into the bit holders.

trolling system is present. In the Commodore 64, the operating system is found in the Kernal and BASIC ROMs. They are both 2364As. Their 8K each of burnt in memory runs the operation of the 64.

In Fig. 7-6 the two ROMs look quite alike and even resemble the Character ROM. They are both 24-pin DIPs. They are powered with +5 volts at pin 24 and returned to ground at pin 12. They both attach to the data bus D7-D0 at pins 17, 16, 15, 14, 13, 11, 10, and 9. At this point in the description, there are some changes. They also interact with each other, as you will see.

First of all, there are thirteen address line connections, A12-A0. The Character ROM only had A11-A0. That's because the Character ROM only had to address 4K. The Kernal and BASIC ROMs each have 8K internal locations. It takes thirteen bits to address 8K. There are 8K combinations that the thirteen bits can assume.

The next difference is the chip select arrangement. There is only one chip select on the 2364A. It is on pin 20 and is labeled \*CS. This means pin 20 is held high while the chip is inactive and is en-

abled by a low. The Kernal ROM is turned on if a low signal called \*KERNAL arrives from pin 16 of the PLA. The BASIC ROM is activated when the signal \*BASIC comes to its \*CS pin from the PLA.

The internal block diagram of the Kernal and BASIC ROMs are similar to the Character ROM. Refer to Fig. 7-7. They each have a memory matrix deep in the chip. The matrix is laid out in an 8192 x 8 organization. The matrix is connected to a decoder stage.

The decoder has an input from the address bus A12-A0. The address is decoded, and the desired memory location is activated. The location outputs a byte of bits to a three-state buffer stage. The buffers connect to the data pins, D7-D0. They in turn attach to the data bus.

The single chip select does not need any gates since there is only the one select. The chip select connects directly to the three-state buffers. When the \*CS is in its high state during standby, it keeps the buffers off. If a low enters the \*CS pin, the buffers go on and pass the byte of data that was ad-

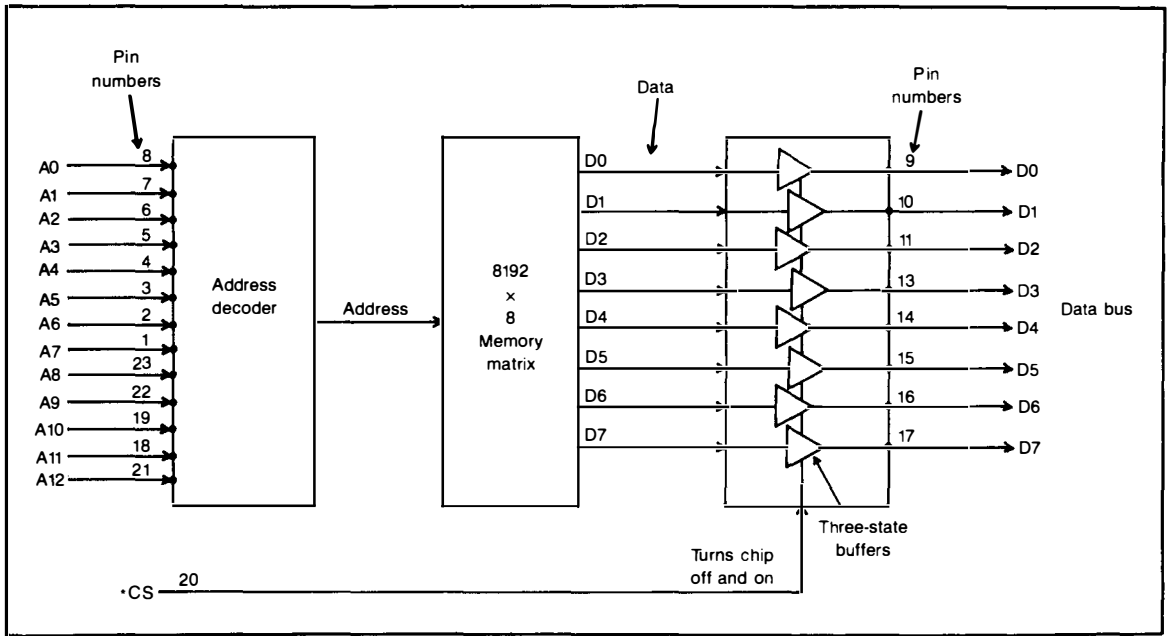


Fig. 7-7. The internal layout of the 2364s are similar to the 2332. There are more address lines since the memory matrix has more locations. The chip does not have to AND chip selects since there is only one—\*CS on Pin 20.

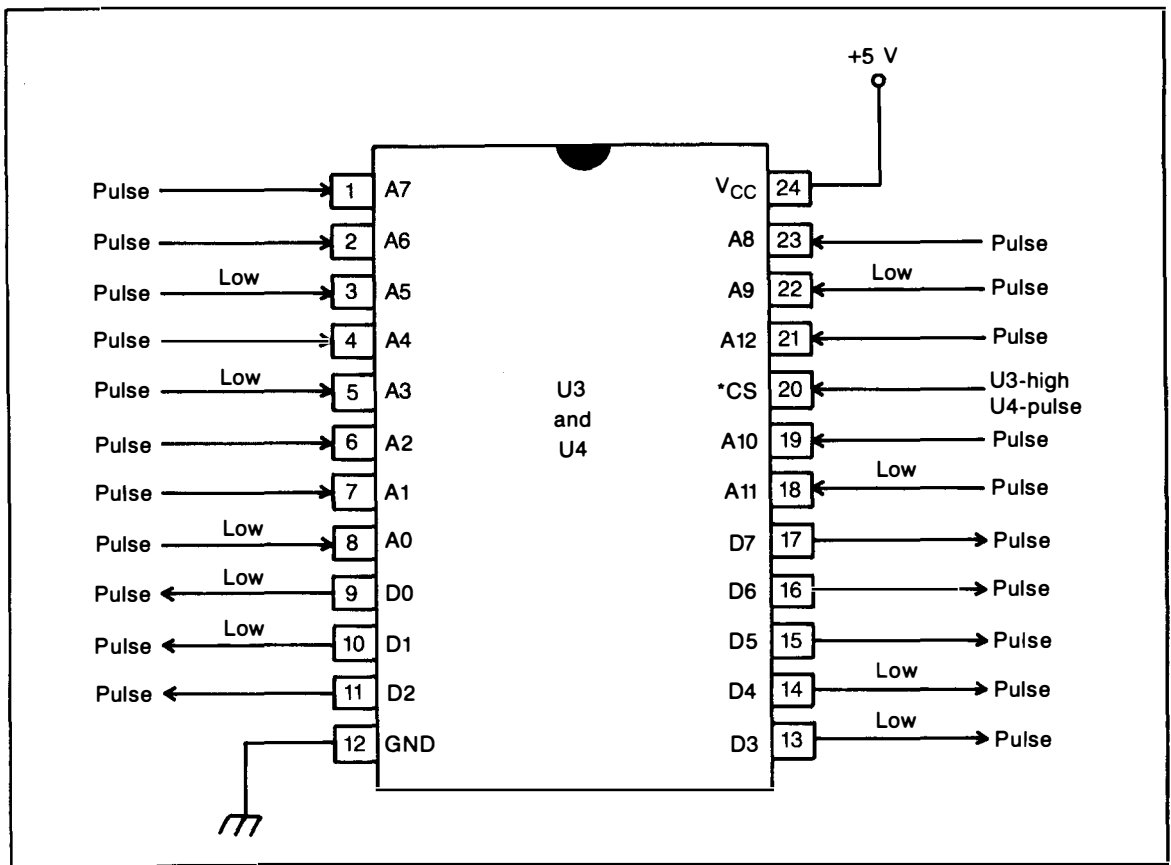


Fig. 7-8. The testpoints on both the BASIC and Kernal ROMs show the same logic probe or vom results, except for pin 20.

ressed. The Kernal and BASIC ROMs are identical except for the permanent data that Commodore has installed in them. Figure 7-8 shows the actual arrangement of the pins.

### Basic ROM

The BASIC ROM is also called the BASIC Interpreter. It is a translator that turns the BASIC code that you type into the 64 into the highs and lows that are stored in RAM. The computer has no idea what anything means unless it is a state of high and low voltages. The voltages are the only entities that RAM is able to store in its flip-flops or capacitances.

Burnt into the BASIC ROM bit holders are little programs and data. For instance, suppose you

want to clear the TV screen. You press the CLR/HOME key. A logical state is formed and the system addresses a location in the BASIC ROM. At that location is the start of a little program of bytes of highs and lows. The addressing of the first byte starts the program running. The program outputs the highs and lows. The screen clearing procedure then promptly takes place.

On the BASIC ROM are hundreds of small machine language programs that respond to all the BASIC statements and all the requests for calculations and data manipulations. The ROM is activated by about 65 special keywords. The keyboard upper and lower case characters plus the digits 0-9 are used to produce these 65 keywords. Some of the other symbols are also used to produce the BASIC keywords.



The keywords are the BASIC Commands you are familiar with: for instance, CHR\$, DATA, GOTO, etc. When you enter these keywords, the BASIC ROM is addressed and runs a little program that performs the chore that the keyword is designed to do. The keywords and other details are found in your User's Guide.

## Kernal ROM

While the BASIC ROM is a collection of a few hundred little programs that execute the keyword commands of the BASIC language, the Kernal contains the programs that perform the rest of the operating system's duties. The Kernal, first of all, sets up the machine when you first turn it on so that all the registers on all the pertinent chips are initialized for action.

Secondly, the Kernal activates the Screen Editor in your computer. This lets the 64 look at the characters you type on the keyboard and place them in their blocks on the TV screen. The Screen Editor gives you control over the cursor. It also responds to commands like delete and insert.

Thirdly, the Kernal provides access to a myriad of subroutines by means of a jump table. The jump table gives the computer a considerable amount of longevity. As time goes by, the Commodore 64 will not tend to become obsolete as quickly when new routines are introduced. The jump table allows your machine to quickly adapt to lots of new subroutines.

When you first switch on the Commodore 64, the Kernal goes into its initialization sequence. It sets a section in RAM called the *stack*. Then it clears the decimal mode of operation. Next, the Kernal checks the ROM cartridge holder circuits. If there is a ROM cartridge plugged in, the control of the machine is switched to the cartridge system. If the holder is empty, then the Kernal retains control and continues its initialization procedures.

The Kernal then turns its attention to all the Input/Output devices and their registers. We'll discuss them all later in Chapter 16, 17, 18, and 19. The serial bus is initialized. The CIAs are set to constantly scan the keyboard. The 60 Hz timer in the CIA is turned on. The SID chip is cleared and

the BASIC memory map is selected. The cassette motor is turned off.

The Kernal also contains a special diagnostic routine that exercises the RAM. The Kernal runs this routine at the beginning to make sure the RAM chips are intact. Once the RAM checks good, the Kernal sets the memory pointers for the top and bottom of RAM. Then the first page in RAM, page 0, is initialized. The tape buffer circuit is then set.

The Kernal then puts some special addresses in RAM at specific addresses. These addresses are used during some I/O activities. Another little jump table is installed by the Kernal in the low section of RAM. The TV screen is cleared. The screen editor has its variables reset.

All of these steps take care of the original housekeeping duties that the Kernal must perform before you begin your BASIC programming. Once all this is accomplished, BASIC is addressed and you are READY. The Screen Editor Program, in the Kernal will now work with you to display your keyboard input.

## Jump Tables

As you can see, the operating system of the 64 is detailed and complicated. It has been carefully burnt into the 8K memory matrix of the 2364A ROM chip. The layout is critical. If one vital bit in a byte becomes defective, it is possible that the entire 8K of programming could become useless.

Like the BASIC ROM, the Kernal is filled with many operating programs. The Kernal's programs though are not activated by BASIC keywords. The Kernal's programs are turned on by machine-language code numbers. However, both the BASIC and KERNAL program outputs to the rest of the computer are machine code. Their machine codes are instructions and data that the computer can understand and process.

The Kernal's control programs are located in the beginning addresses of the chip. The Kernal has 39 input/output routines and a number of other management programs. These routines take up most of the addresses in the chip. The jump table code is located near the end of the address list. The

*jump table* is a bunch of addresses with each address full of data, as shown in Fig. 7-9.

The data in each address is another address. The addresses stored in the jump table are starting addresses of the various routines stored in the beginning of the Kernal ROM. When you write a machine-language program, and you want to obtain the services of a Kernal routine, you address the appropriate location in the jump table. As the jump table location is addressed, it is activated. It's contents, the starting address of the desired routine, is output. That starting address is, in turn, put on the address bus and the routine is contacted.

The question becomes, why bother with this indirect way to contact the routine? Why not just address the Kernal routine directly? The answer is that the Commodore 64 designers want the pres-

ent models to be compatible with and operate with future models.

As time goes by, improvements in the operation of the 64 will continue to occur. Even a small change could make a newer operating system useless in an older model. In the interest of having new OSs work in older machines, or as it is called, to make the OS *portable*, the jump table scheme is utilized. The jump table is designed to standardize the input, output, and memory management routines of the 64 for the following reasons.

During expected changes, the actual routines are not expected to change much. Even if they do, they should still run the 6510 MPU and its dynamic memory without undue complication. What is expected to change are the addresses of the routines. If the addresses of the routines change, literally

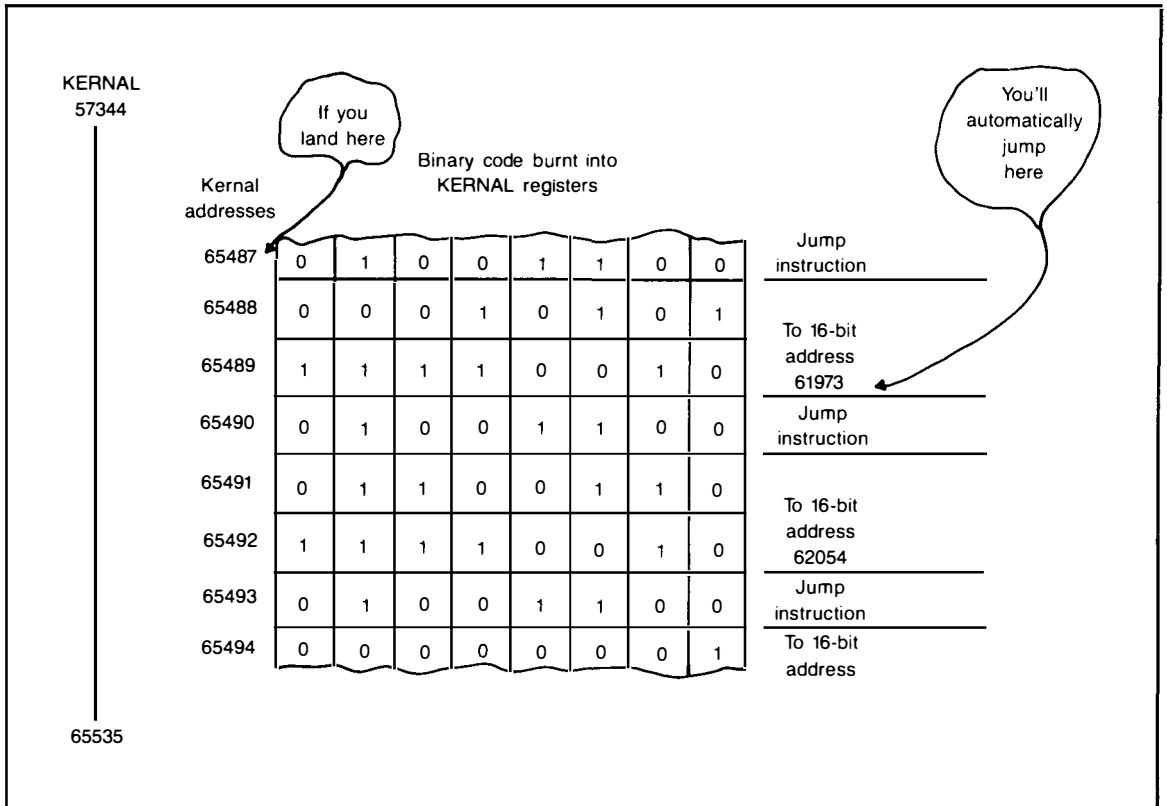


Fig. 7-9. At the highest addresses of the Kernal ROM is a special jump table. The table consists of a number of 3-byte data groups. Each three bytes is made up of a one-byte jump instruction and a two-byte address. If you address one of the data groups, you are instructed to go to the stored address.

thousands of programs on tape or disk will become obsolete. A lot of your hard written programming won't work with these changed OSs. That is where the jump table comes in.

When a new Kernal chip is manufactured, the jump table will be located at the same chip places as its predecessors. The contents of the locations, which are the addresses of the routines, can be changed. That way your program will still address the same old places in the jump table. The new address contents will then point to the new address of the routine. You won't even know the difference as the program is run.

I did not mention that the jump table holds some program addresses not found in the Kernal. These off chip addresses are located on the BASIC ROM chip. That is because the BASIC ROM chip contains a lot of programs that are very useful. Instead of a programmer writing some of those routines, he just puts an address in a program, that goes to the jump table. The jump table, in turn, addresses a program on the BASIC ROM. The program is then run. The programmer, as a result, saves the time it would have taken to write and incorporate the same thing in his overall program.

## PROGRAMMABLE LOGIC ARRAY

There is one other ROM in the Commodore 64. It is not usually referred to as a ROM since it is not used as read only memory, but it does a decoding job. It is the 82S100 PLA chip. It is a 28-pin DIP shown clearly on the chip location guide.

A PLA is a chip that has certain types of programming built into it. You've heard of AND and OR gates. They are covered in detail in Chapter 10. We'll touch on them here in a block diagram fashion. A PLA is a combination of two ROMs. One is an AND ROM, and the other is an OR ROM. Both of these sections are shown in Fig. 7-10. The AND ROM receives the chip inputs. The OR section delivers the chip outputs.

The AND gates at the PLA input area receive the input signals that need decoding. An AND gate can't output a signal, unless all the inputs to the gate itself are high. The AND gate can have two or

more inputs. It only has one output. It will output a high when all inputs are high. The AND gate is like an electronic combination lock. It turns on when all the high inputs are present. Otherwise it remains off.

The OR gate is so named because it will output a high if any of its inputs are high. The OR gate can have two or more inputs. It only has one output. It will output a high when one or more inputs are high.

The OR gate is like an electronic lock too. It turns on when a high is inserted into any one of its input leads. The difference between the AND gate and the OR gate, at first, does not seem world shaking, but it is. Because of the difference in logic, the computer is feasible. The PLA is only one of the applications.

In the PLA, the AND gates are able to distinguish between a lot of inputs and only output the desired signals to OR gates. The OR gates are then able to output when the desired signals appear.

## 82S100 Pinout

I mentioned earlier in this chapter that the BASIC, Kernal and Character ROMs were enabled by some chip select signals called \*BASIC, \*Kernal and \*CHAROM. The asterisks in the front of the signal name indicate that the enabling signals are lows. The three ROMs remain dormant while a high is at their chip selects, \*CS, and activate when a low appears. The three enabling signals are coming from pins 17, 16, and 15 of the PLA. These PLA output signals are shown in Fig. 7-11.

There are five more output signals coming out of the PLA. At pin 18, \*CASRAM is exiting. This is a form of the column address strobe that is used by the dynamic RAM during addressing. The \*CASRAM goes directly to pin 15 on all the eight RAM chips. It passes through a 33 ohm limiting resistor on its way.

From pin 13, the output signal GR/\*W emerges. It is the enabling signal to turn on the 2114 Color RAM. It goes to pin 10 of the Color RAM, \*WE, write enable. Pin 10 is held high till you decide to change a color in a character block

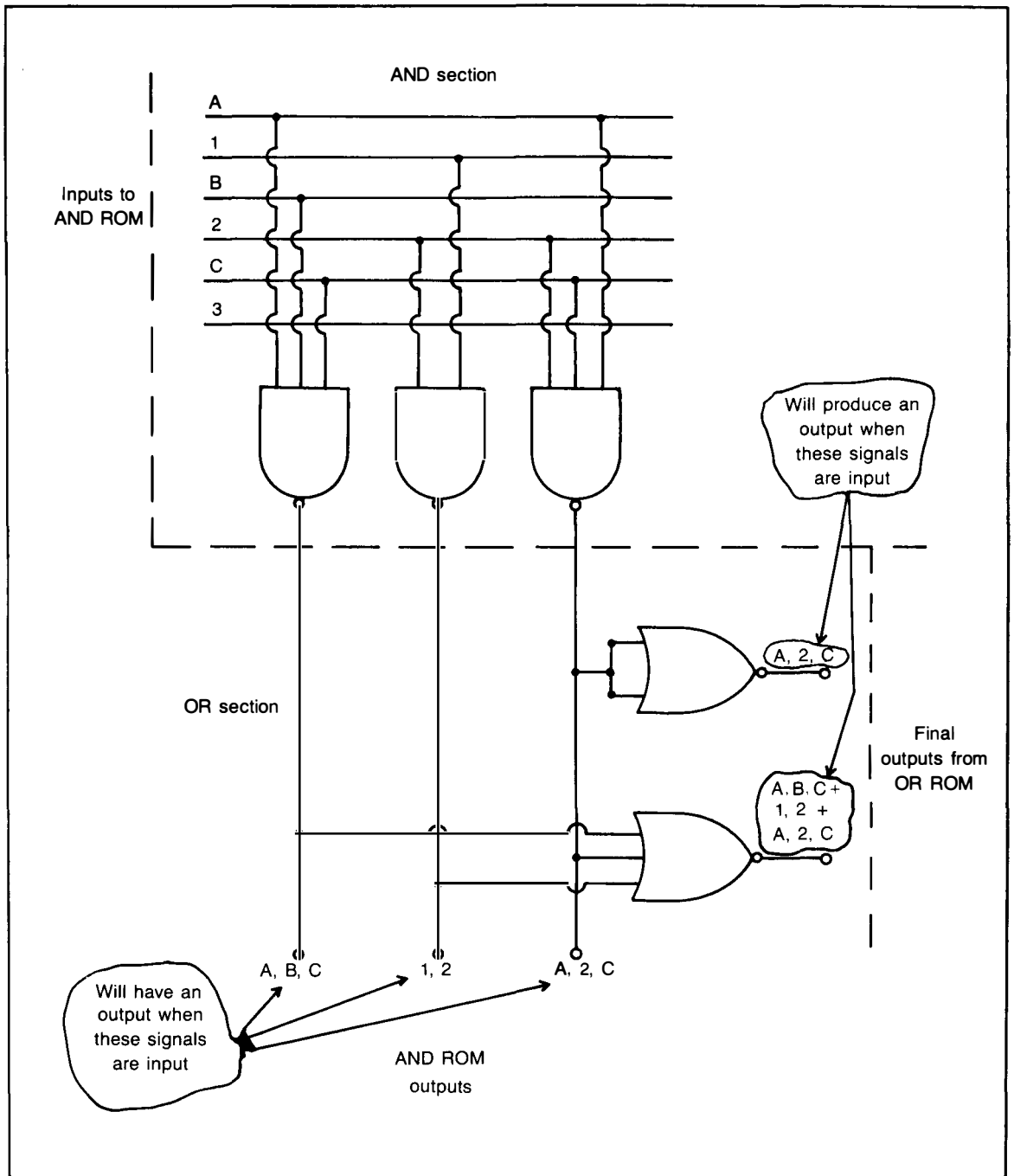


Fig. 7-10. A Programmable Logic Array is a set of AND gates receiving an input and a set of OR gates producing an output. By careful design a particular type of input will predictably always result in a predictable output. This is an expensive way to produce a ROM chip. The PLA in the 64 though, is not used as a ROM. It is made to operate as a decoder and produce chip selects and other signals.

on the TV screen. When you do and send the instruction, the GR/\*W signal goes low, and the \*WE pin is turned on. The Color RAM chip is addressed and you send the color you desire over the four data bus lines, D3-D0.

At pin 12, the PLA outputs \*I/O. It is one of the input signals to \*EN 74LS239 decoding chip that is discussed in the next chapter. Here again the \*EN 1 pin is held high. A low from the \*I/O line enables \*EN 1.

Pins 11 and 10 are called \*ROML and \*ROMH. They are two PLA outputs to the Cartridge Expansion 44-pin female plug. These two outputs connect

into a ROM cartridge that might be plugged into the computer. The ROM cartridge that might be installed is like the BASIC and Kernal ROMs.

A ROM cartridge could be a game, an assembly program, a bookkeeping system, or what have you. When it is installed, it takes over the operation of the machine. The BASIC or Kernal might still be used but probably in a subsidiary manner. If you recall, when the Kernal did its housekeeping duties during startup, it checked to see if a cartridge was installed. If it was, the Kernal would have turned control over to the cartridge.

If the ROM cartridge is in place, these two con-

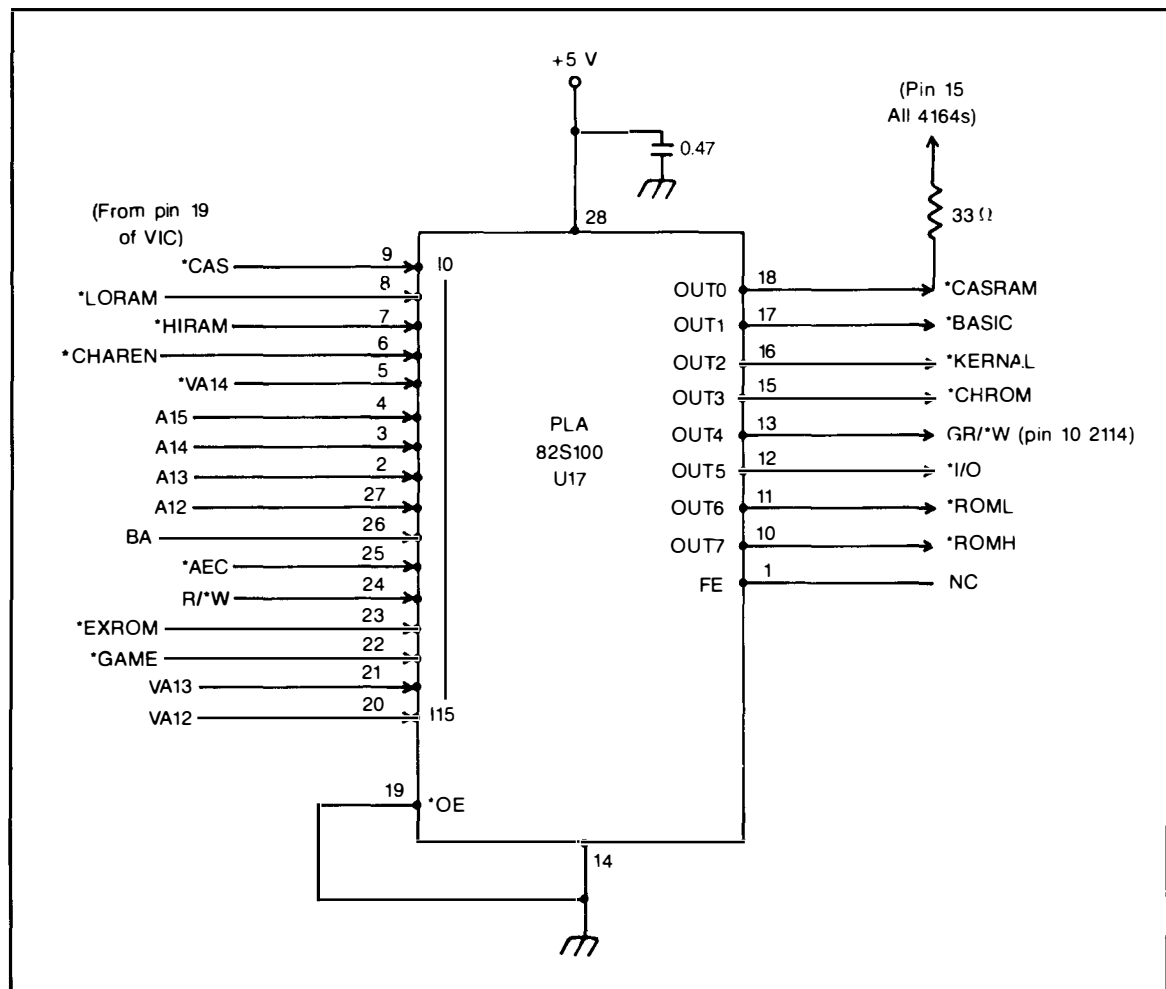


Fig. 7-11. The PLA has 16 input pins and eight outputs. Note \*OE is tied to ground, which keeps the chip enabled at all times.

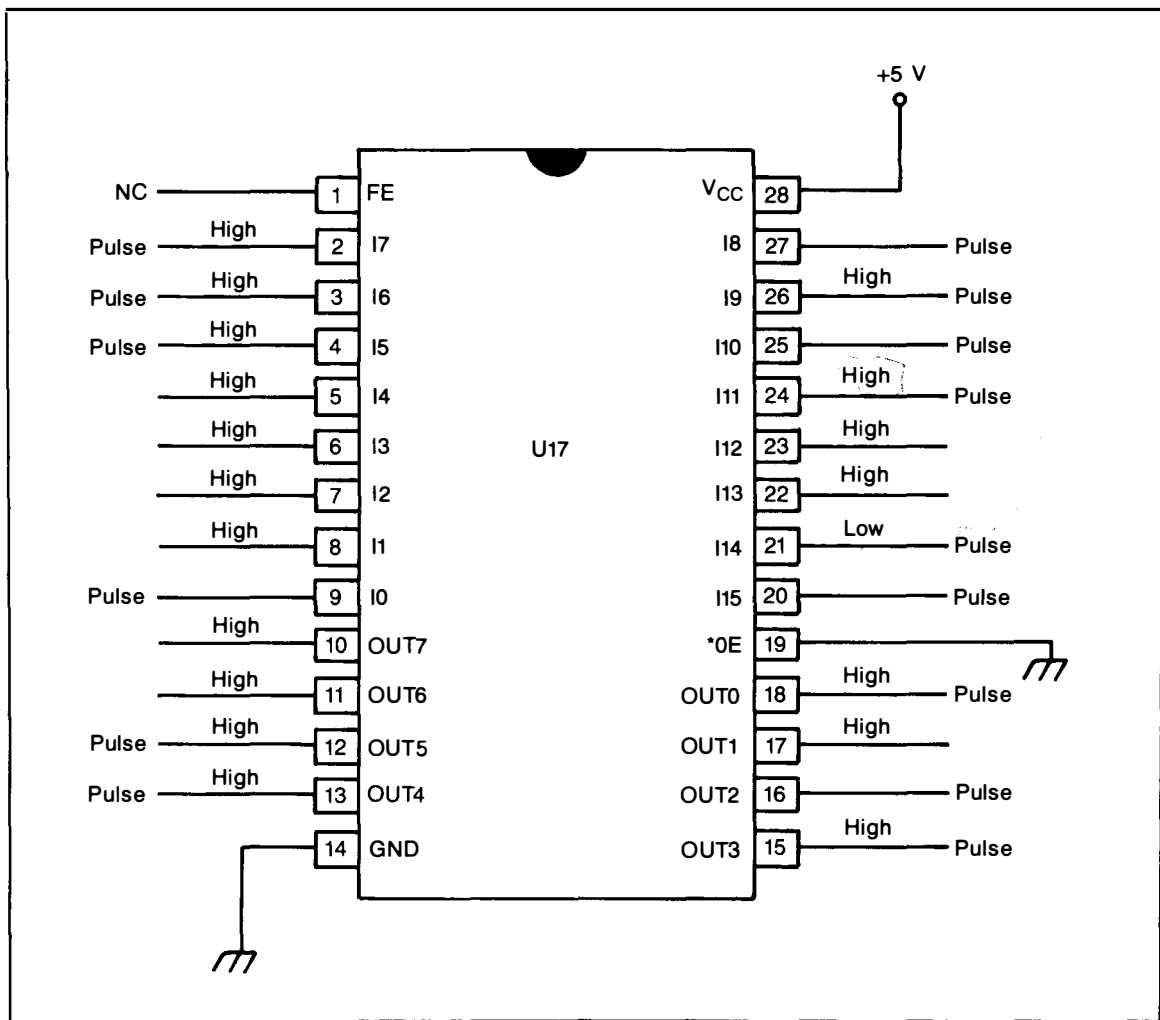


Fig. 7-12. These readings should be on the PLA when it is first turned on and is waiting for inputs.

control lines \*ROML and \*ROMH would be placed in use. They help in the assigning of addresses to the cartridge. \*ROML gives the cartridge addresses in a lower numbered area of the memory map. \*ROMH gives the cartridge addresses in the higher numbered section of the ROM address list.

The outputs of the 82S100 have pins named O0-O7. The inputs have the names IO-I15. The sixteen inputs, as they pass through the AND and OR gates, are thus able to be coded into eight outputs. Let's examine the inputs and see where they are coming from.

### PLA Inputs

At IO, the input signal \*CAS enters the PLA. \*CAS, the column address strobe for the dynamic RAM addressing, is generated in the VIC II chip along with its companion \*RAS, the row address strobe. \*CAS exits pin 19 of the VIC and goes straight to IO of the PLA. Inside the PLA, it enters an AND gate. There are some other connections to the AND gate. One of the resultant outputs of that AND gate is the \*CASRAM signal out of O0, that goes to all the dynamic RAM chips.

At pin 8 of the PLA, the \*LORAM signal

enters. \*LORAM is produced in the 6510 MPU. At pin 7, the \*HIRAM enters. \*HIRAM is also produced in the MPU. Also coming from the MPU is the signal \*CHAREN. It enters pin 6 of the PLA.

These three signals are used to transfer data from the MPU to a peripheral device and back. They come from a special I/O port located right inside the MPU. They are discussed in more detail in Chapter 12. They are open drain circuits and are held high by +5 volts through some pullup resistors.

At pins 4, 3, 2, and 27 are lines from the address bus, A15-A12. These are the most significant address lines. They are needed to aid in the selection of the various ROMs. \*VA14 is coming from one of the output ports of a CIA. VA13 and VA12

are two address lines from VIC. Note \*VA14 has an asterisk which means that it is held high while inactive and goes low to operate. The other two connections, both without an asterisk, are held low and must go high to be active. Their operation is discussed in the VIC and CIA chapters.

\*EXROM and \*GAME are both connected to the Cartridge Expansion Plug. They work with the expansion ROM and game cartridges.

This leaves the two control lines R/\*W and \*AEC. R/\*W is the read/write control line that travels from the MPU to a lot of the chips in the digital circuits. It is input to the PLA and is needed to turn some of the gates on and off during operations. The R/\*W line originates in the MPU. The PLA is only one of its destinations.

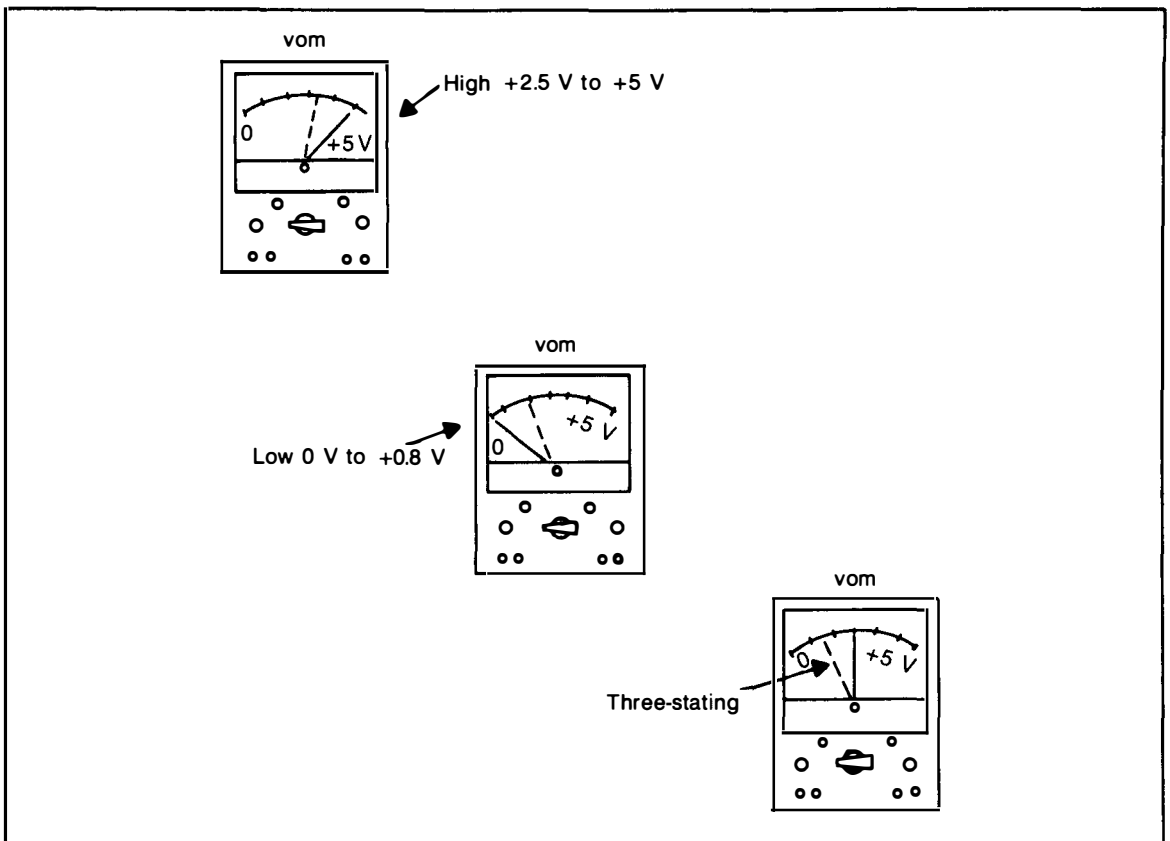


Fig. 7-13. I find my analog vom is more useful to service logic circuits than a digital vom. Highs are up near +5 volts while lows are down around 0 volts. Voltages in the middle are usually indications of three-stating. The only drawback is that the vom will not show pulses.

The \*AEC line, address enable control, enters pin 25 at I10. It is provided by the video circuits. It is needed to turn off all of the address bus lines from the MPU while the VIC chip is addressing the various memory areas. The PLA gets portions of its internal circuits disabled by the AEC.

Lastly the PLA has inputs from the power supply. Pin 28 connects to +5 volts and is bypassed to ground with a 0.47  $\mu$ F capacitor. Pins 19 and 14 are tied together and returned to ground.

## SERVICE CHARTS

All four of these ROMs, BASIC, Kernal, Character and PLA, are composed of bipolar transistors. As a result, they are fairly rugged. When you first turn on the computer, they all receive power and the kernel automatically goes through its start up routine where it initializes the pertinent registers and the other housekeeping duties. Then it gets BASIC to display the READY sign along with the blinking cursor. At this point, the digital circuits are in a standby loop waiting for keyboard input. At that time, you can run voltage or logic probe tests at all the ROM pins. There are three ROMs with 24 pins and the PLA with 28 pins. Figures 7-3, 7-8, and 7-12 show what voltages and logic probe results should be present if the computer is operating normally.

During trouble, as you make each pin test, it can be compared to the expected results. Should the reading be different than the one that is called for, that could be a clue indicating trouble. In order to decide if the reading is indicating a fault, you should be able to see which pin is not right and try to deduce, by having an idea of the pin's function, whether the pin's operation could be related to the trouble.

If there is a relationship, then you can decide if the trouble is in the chip you are testing or the circuits feeding or being fed by the chip. If the incorrect pin is an input, then chances are good that the circuit that is feeding the chip is causing the poor reading before the signal ever gets to the chip. Should the wrong reading be at an output pin, it is likely that the chip is defective. The signal is not able to get out of the chip. Of course, these are sup-

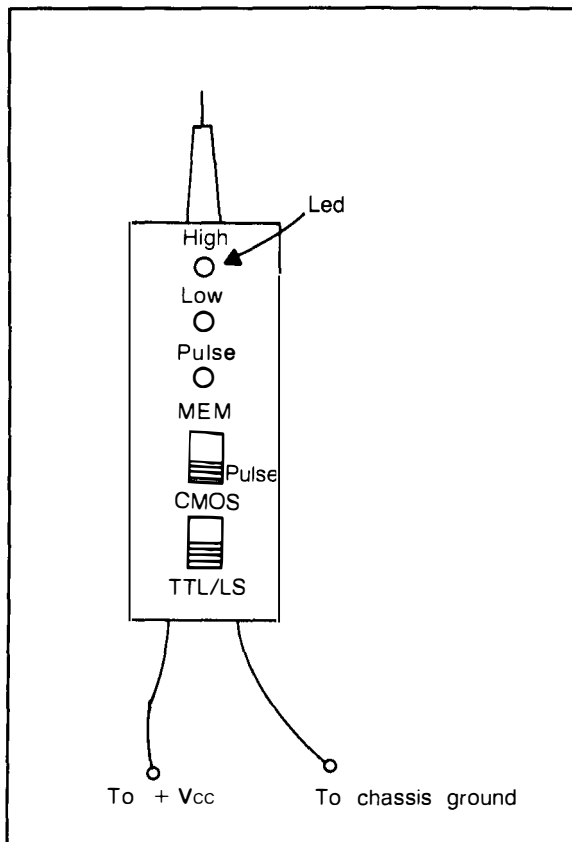


Fig. 7-14. The logic probe is a must during computer servicing. LEDs show highs, lows and pulses clearly. The only drawback is that the probe has no three-state indication.

positions and the reverse could be true, but this is a good way to start.

All the readings in these chips are quite the same. There are highs, there are lows, and then there are pulses. In addition, there could be three-state readings. Both the vom and the logic probe reveal the same conditions. They simply exhibit the results in different ways.

My vom is an analog device. It shows dc voltage readings on a meter like those in Fig. 7-13. In these circuits the highs are somewhere near +5 volts. The lows are around 0 volts. The three-state reading caused by static noise makes the needle rise up near 2 volts. The vom has no way to depict a pulse, its readings are dc only.

The logic probe reveals the voltage levels as



LED lights. In Fig. 7-14 the lights are labeled just as they appear on the probe. There is one for a high, one for a low, and one for the pulse indication. The logic probe has no light for the three-state noise level. It is best to have both a vom and a logic

probe to explore these chips. The vom does not show pulses. The logic probe will. The logic probe has no three-state indication. The vom does. Between the two, you can determine the logic states on the pins in an accurate manner.



## 8. Other Integrated Circuits

**T**HERE ARE 32 INTEGRATED CIRCUIT CHIPS in the Commodore 64. We have looked over the large chips: the eight dynamic RAMs, the MPU, two CIAs, three ROMs, Color RAM, the VIC, the SID, and the PLA. There is a lot more to be covered on these chips. Some have entire chapters devoted to them later on in the book. You have read about 18 chips so far. There are 14 more in the machine. These are not main landmark chips. These support and interface with the larger chips. Let's determine what they are, the jobs they do, and how to run tests on them. The test point charts in this chapter give the condition of the chips at the READY prompt.

### 7406N HEX INVERTER

This hex inverter is the simplest chip on the print board. It is a 14-pin DIP. The word hex stands for the number six. There are six little inverters built into the chip. What is an inverter?

As you'll learn in Chapter 10, an inverter is sometimes called a NOT gate. If you inject a high into an inverter, the circuit changes the high to a

low, and a low will emerge from the gate. Should you input a low into the inverter, it is changed to a high. Refer to Fig. 8-1.

Besides inverting the logic state, an inverter can act as a current amplifier and buffer to the incoming signal. It can raise the current level of the logic state and match it to the subsequent circuits.

The 7406N is a TTL with six of these inverter stages. Figure 8-2 shows that each inverter stage uses two pins. There is only one input to a NOT gate and one output. The six inverters require 12 of the chip's 14 pins. The other two pins are used for the +5 volt power and ground.

The six inverters are all on the same chip and all use a common power input, but they are installed in different circuits throughout the computer. Three of them are used in the output circuits of a CIA. They change the logic state of the pin output. A fourth and fifth inverter is needed at the outputs of a 556 timer. The last inverter is utilized by a feedback circuit from the VIC to the dynamic RAM area.

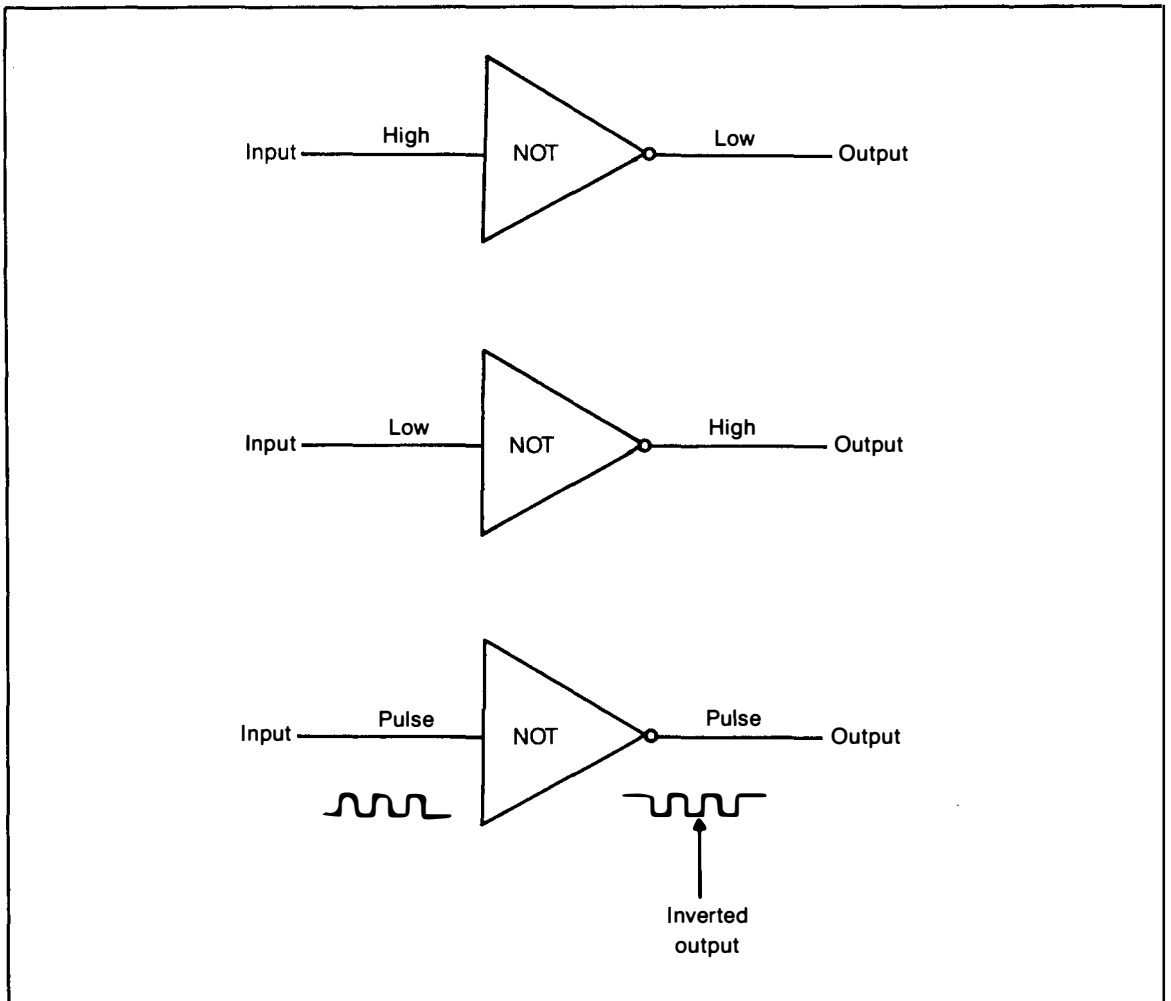


Fig. 8-1. Besides changing highs to lows and vice versa, the NOT gate will invert a pulse that is high going to one that is low going.

With the aid of Fig. 8-2 and a vom or logic probe, you can check out the hex inverter quickly. Each inverter has an input and an output. The only type of signal that should be present at the inputs and outputs are logic states. The 7406N has no three-state pins so there should not be a three-state condition on any pins under ordinary circumstances. The input pins are 1, 3, 5, 9, 11 and 13. The respective outputs are 2, 4, 6, 8, 10 and 12. Pin 14 and 7 are power.

The first check with the vom should be pin 14. There should be +5 volts present. Pin 7 should

read 0 volts at ground level. If either voltage is incorrect, it is a clue. For example, suppose there is no voltage at pin 14. If there are +5 volts on the surrounding chips but not on pin 14 of the hex inverter, chances are the copper trace on the print board to pin 14 is broken or has a corroded connection. When the quick check at 14 shows power is present, then the individual inverters come under scrutiny.

Test the voltage at each input. A high on this chip is considered +2.0 volts or better. If the input voltage is a high then the output pin should have

a low. The low on this chip is considered + 0.8 volts or less. When the input voltage is a low, then the output should have the opposite state, or a high. All six of these inverters should follow the same logical approach: Input high, output low; input low, output high. If the input is a pulse input, the 7406 should produce an inverted pulse output.

If there is a discrepancy from this pattern, you have a clue. The inverter with the wrong output voltage could be shorted or open. A dead short in the stage would place the input voltage on the output. An open circuit would place the surrounding voltages on the output. This could be any voltage including a three-state condition.

In the 64, this chip is fully occupied. In other computers, you might find the chip with some of

its inverters unused. In those cases, the unused inputs are usually tied to + 5 volts to keep the unused gates from interfering with the normal processing.

### 74LS08 QUAD 2-INPUT AND GATE

The Commodore 64 has a 74LS08 AND gate performing several jobs. It is a 7408 type chip with LS extras. The L stands for low power dissipation at the expense of switching speed. The S stands for Schottky. A Schottky diode is known for its high speed switching. By using both in a gate, the advantages work together.

The quad part of the name means four. Figure 8-3 shows the four little AND gates on the chip. The 2-input means each gate has two inputs in contrast

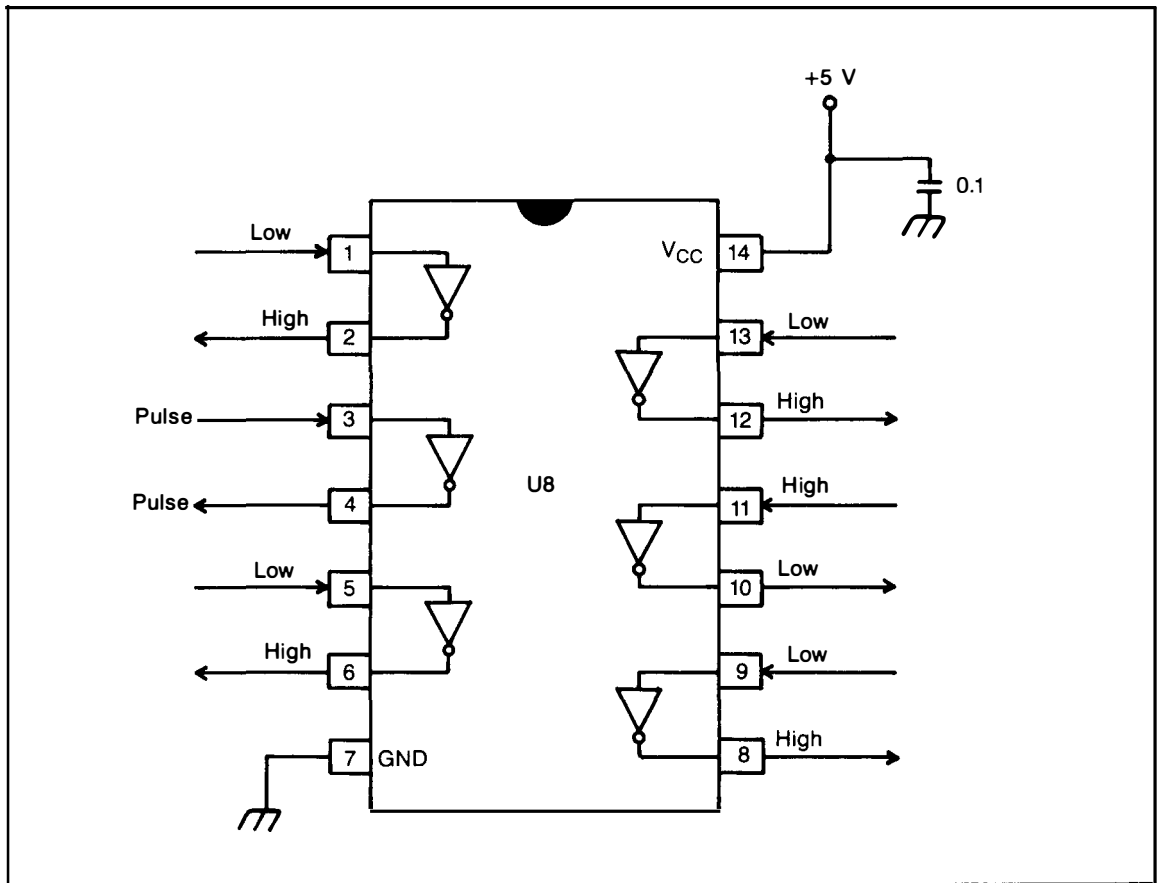


Fig. 8-2. The simplest chip on the board is the 7406. There are 14 test points you can probe.

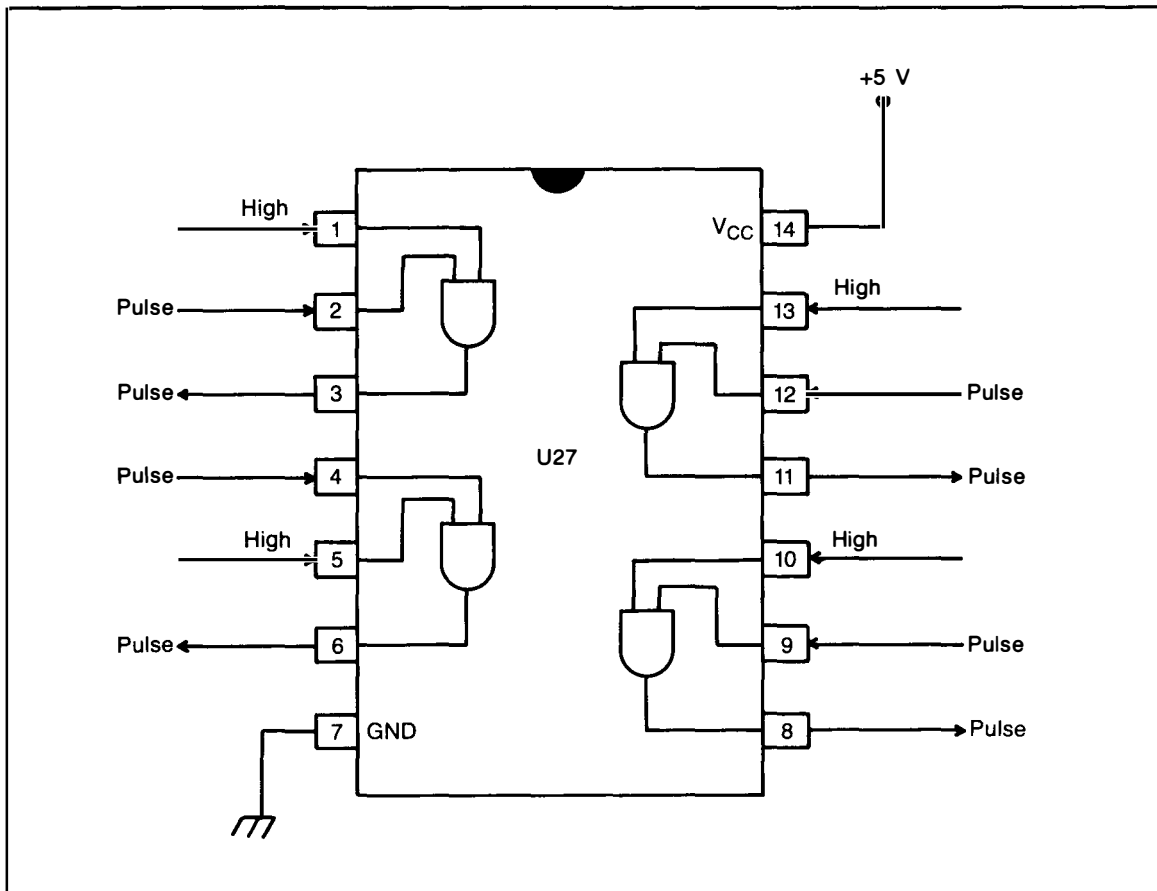


Fig. 8-3. The four AND gates in the 64 each receive a high and a pulse, which causes a pulse type output.

to the NOT gate that only uses one input. There is still only one output. Each AND gate requires three pins on the chip. There are four gates. 12 of the 14 pins are used up as inputs and outputs to the gates. Pins 14 and 7 are attached to the supply voltage and ground exactly like the 7406N.

The AND gate is built so that it will output a high if, and only if, both inputs are highs. Should one or both of the inputs be low, the AND gate does not output a high. It simply acts like an open circuit. Refer to Fig. 8-4.

The four AND gates on this chip are installed in four different circuits in the Commodore 64. Each performs a job of outputting a high if the condition of two input highs is met. The AND gate connected to pins 9, 10, and 8 of the 74LS08 has its inputs

9 and 10 attached to BA, bus available, and \*DMA, direct memory access. The output, pin 8 connects to the RDY, ready, line of the MPU.

The gate that connects to pins 4, 5, and 6, have the inputs attached to AEC, address enable control, and \*DMA. The output goes to the AEC line of the MPU.

Another gate is connected to pins 12, 13, and 11. The inputs are \*COLOR and AEC. The output connects to the \*CS pin of the Color RAM. The fourth AND gate, pins 1, 2, and 3, outputs to the Time Of Day circuit on a CIA. One input comes from the power supply and has 9 volts ac, and the other input is tied to a perpetual high at +5 volts dc. Refer to Fig. 8-5. Every time the 9 Vac passes through +5 volts the AND triggers TOD on the

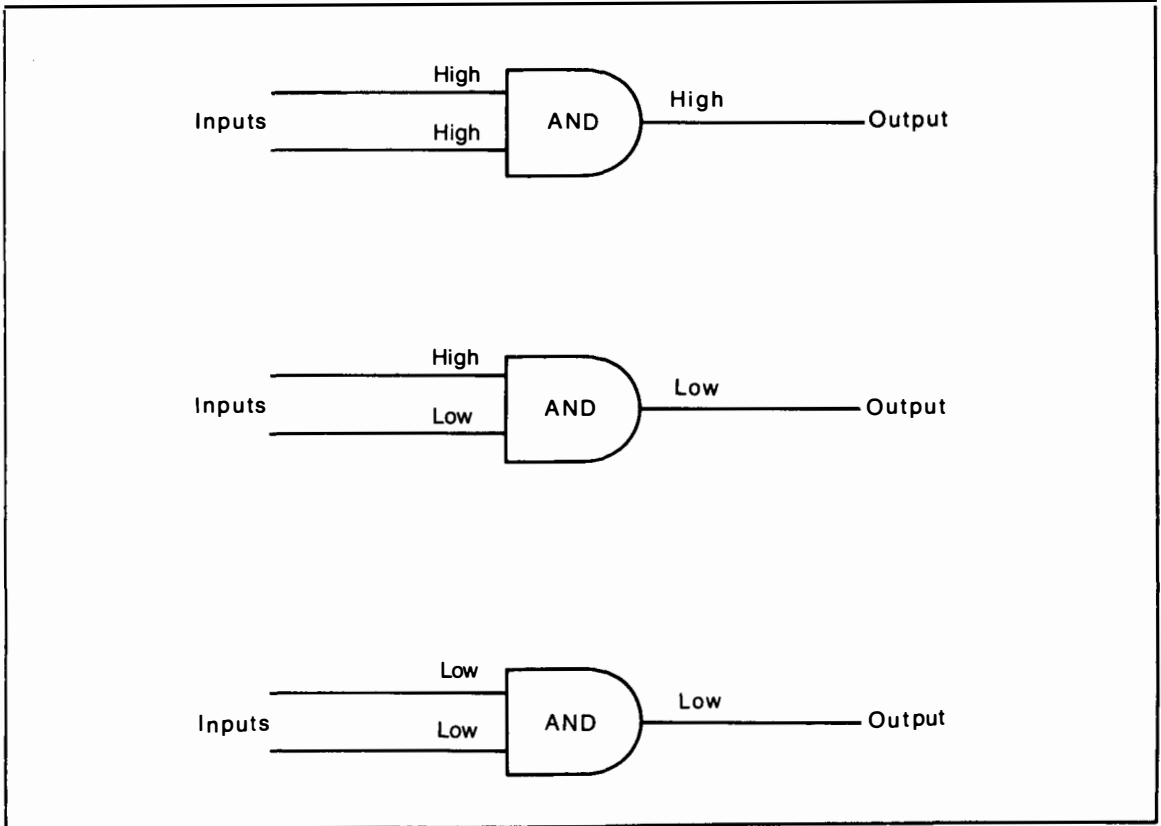


Fig. 8-4. The AND gate, when good, has these predictable outputs with these inputs.

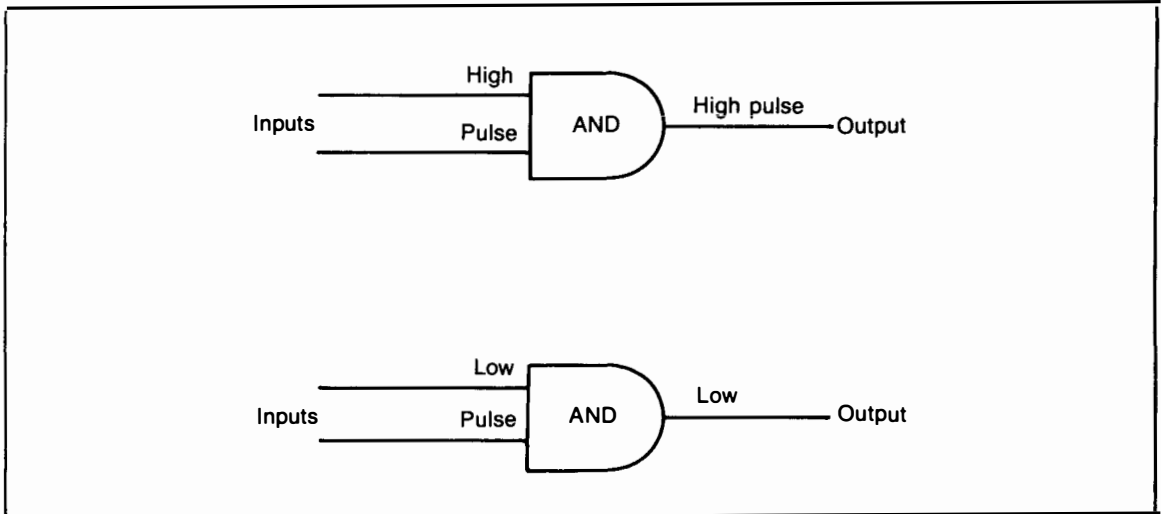


Fig. 8-5. When a pulse is applied with an input high, a high pulse is output everytime the input pulse goes high. When a pulse is applied with an input low, the gate never turns on and continually outputs a low.

CIA. Since the 9 Vac cycles 60 times a second, you have yourself a very accurate clock ticking away at terminal TOD.

The AND gates are used in this machine as handy control devices to send a high to various terminals to turn chips and circuits on and off. The conditions to send the triggering voltage highs must be met at the inputs by supplying two highs there.

### 74LS74 DUAL D FLIP-FLOP

The clock circuit of the Commodore 64 uses a 74LS74. The clock is discussed in detail in Chapter 14. The flip-flop acts as a momentary storage device in the clock circuit. It is a 14 pin DIP. The dual means there are two identical flip-flops on the chip. Figure 8-6 shows the physical layout of the chip. There are six connections for each flip-flop and two terminals, 7 and 14, for ground and supply voltage.

Figure 8-7 shows the way Commodore is using the flip-flops. Flip-flop one has four pins (1, 2,

3 and 7) shorted together and grounded. \*Q has no connection. Only preset and Q are connected in the circuit. They go to the 74LS193 chip, a binary counter.

The second flip-flop has the D and \*Q connections shorted. The CK pin is connected to the counter and Q goes to a phase detector network. Don't worry about the way a flip-flop works, at this time. It is covered later in Chapter 11.

This 74LS74's two sections are doing two jobs in the clock circuit. The top flip-flop is aiding the frequency selection network. The clock is capable of producing one of two frequencies. One is for the American NTSC and the second for the European PAL. America uses one type of TV frequency while Europe uses another. To make the computer able to function with an American TV or a European TV the clock produces a frequency of 14.31818 MHz for an American TV or 17.734472 MHz for a European TV.

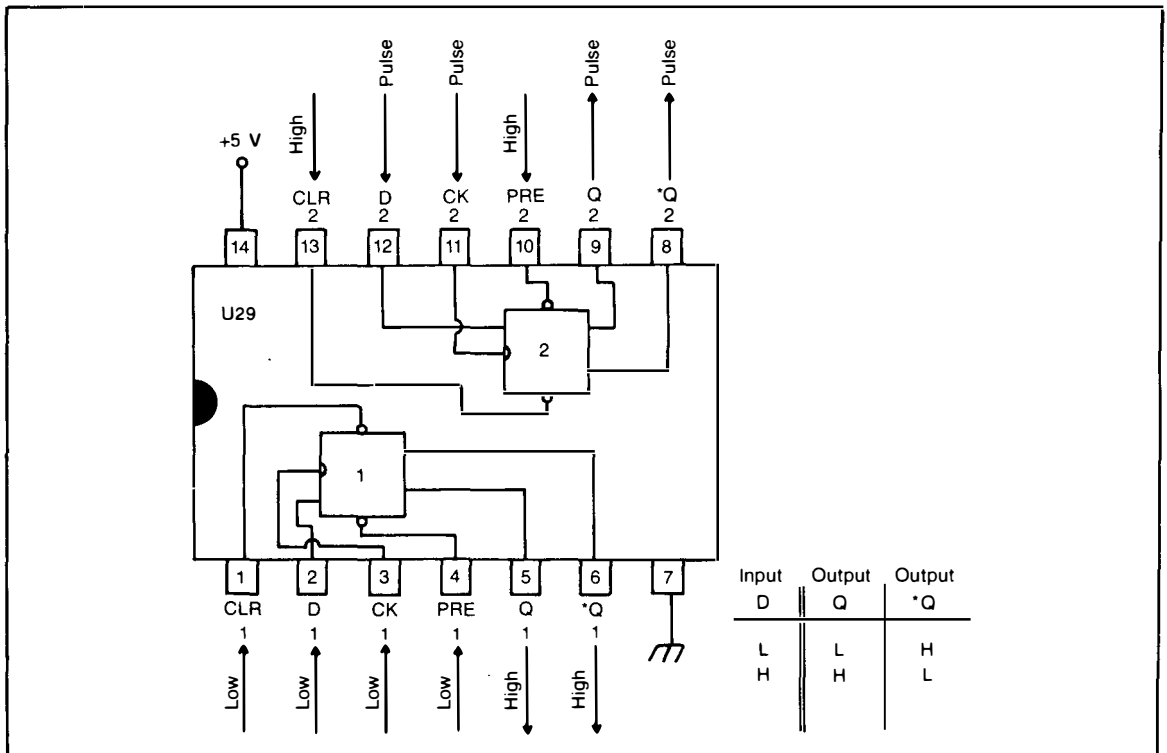


Fig. 8-6. The 74LS74 chip has two D flip-flop circuits. Each FF uses six testable pin connections.

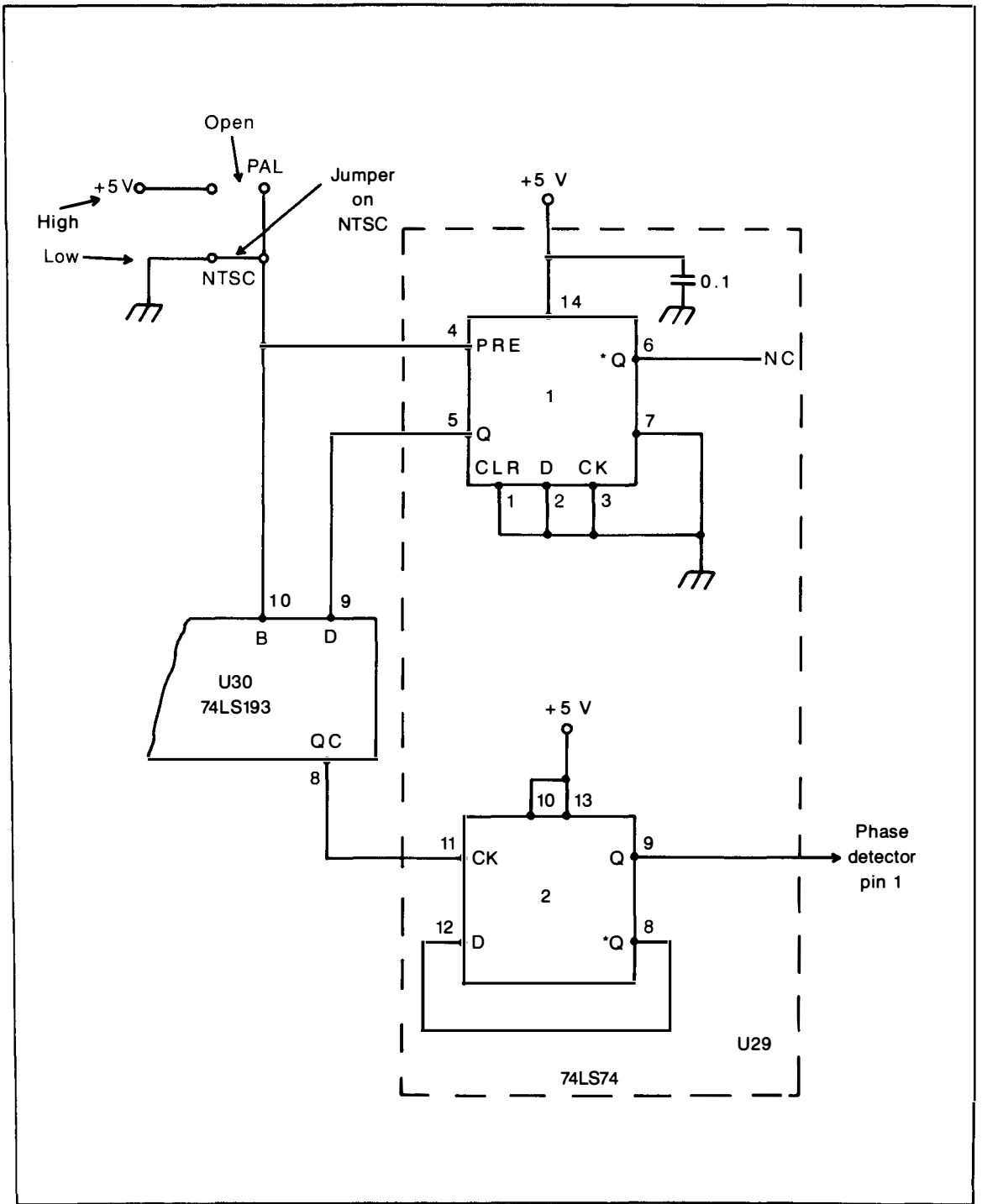


Fig. 8-7. Pins 1, 2, 3, and 7 are made permanent lows by tying them to ground.



A jumper type switch selects the frequency. A connection to a high gives you the PAL frequency and a low attachment forces the circuit to produce the NTSC. The top flip-flop aids in that arrangement.

The bottom flip-flop is a coupling circuit between the binary counter, the 74LS193, and the phase network. The phase network in the MC4044 chip has one input from the flip-flop but two outputs. Refer to Fig. 8-8 and 8-9. There is one output at pin 3 of the phase detector below the input at pin 1 and a second output at pin 5 of the charge pump to the base of the npn transistor. These are two frequencies originating from the single input. The frequencies are discussed in the Clock Chapter.

The flip-flops used here in the clock circuit are just one of the many ways the 74LS74 can be used. To test the flip-flop, a logic probe is useful. The test point chart in Fig. 8-9 shows what states or pulses should be on the pins after the 64 has been

turned on and the cursor is blinking near the READY signal.

## 74LS193 UP/DOWN COUNTER

The central chip in this clock circuit is the 16-pin DIP, the 74LS193. It is called the Synchronous Up/Down Counter with Dual Clock. All it really does is divide the incoming 14.31818 MHz into lower more useful frequencies. For instance, if you divide the incoming frequency by 16 you get a result of near 1 MHz. This is the frequency that exits at pin 3 of the phase detector. It is sent to the MPU to drive the computer activity.

The 74LS193 is a 4-bit binary counter. The details of a counter register are discussed in Chapter 11. The four bits are contained in four flip-flop circuits. Each flip-flop can hold a high or a low. The logic state in a flip-flop is determined by what input is placed into each flip-flop. The word *syn-*

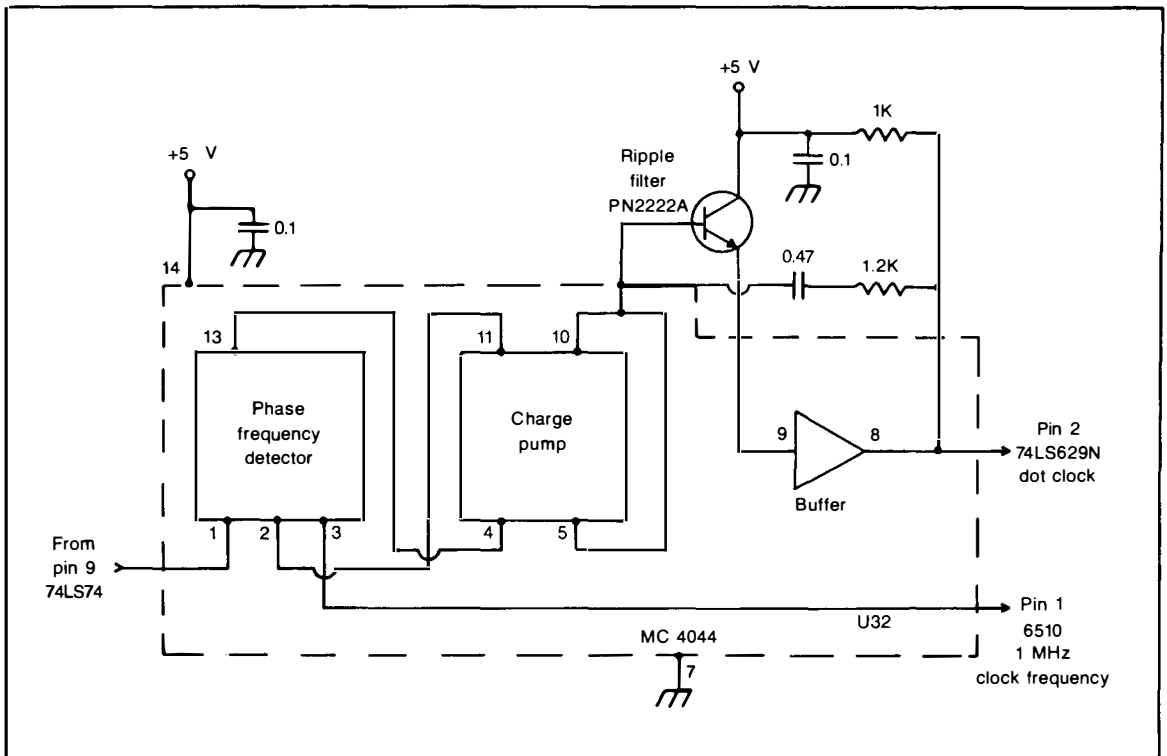


Fig. 8-8. The MC4044 chip has one input from the flip-flop that it splits up into two separate outputs.

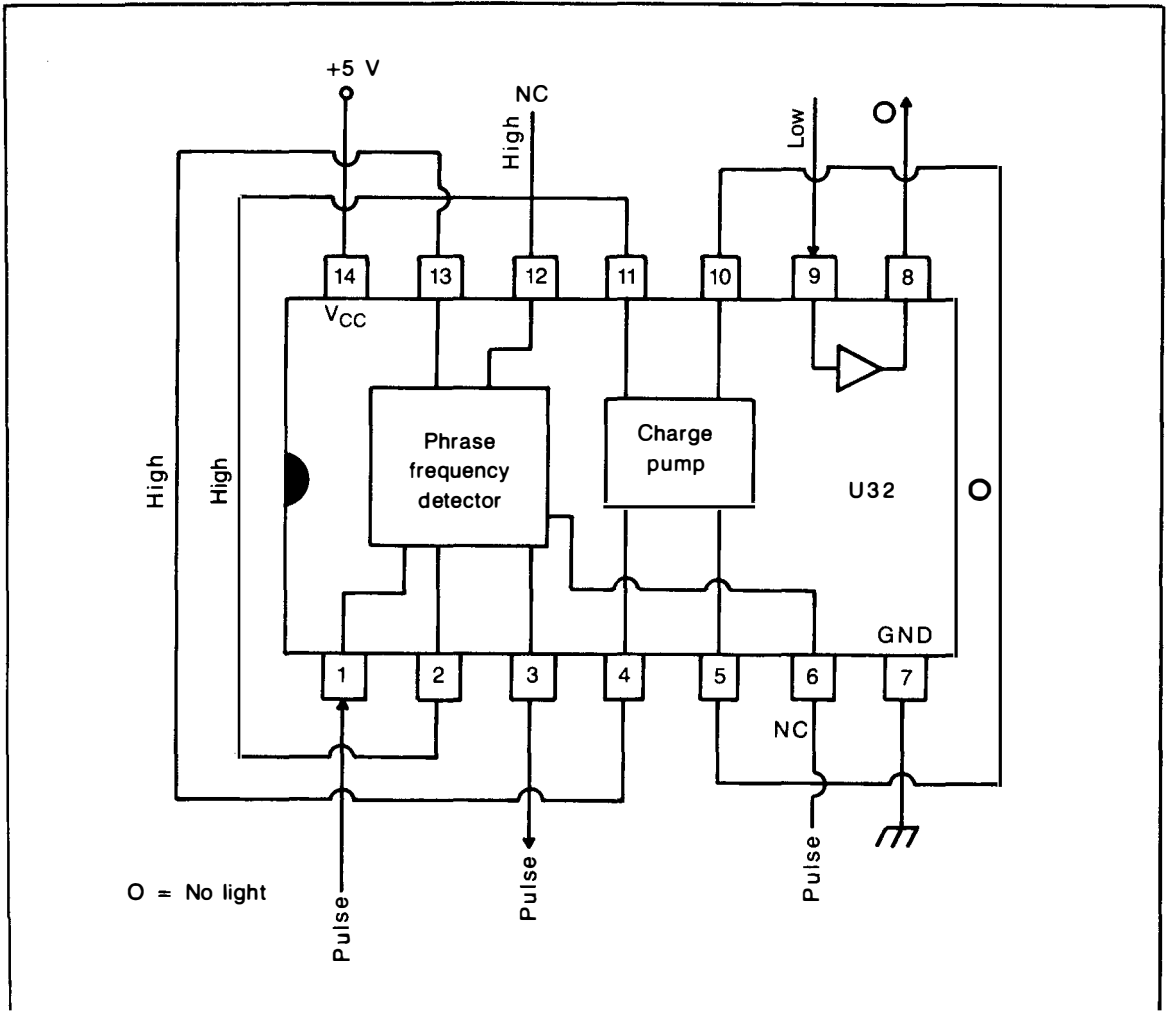


Fig. 8-9. The 4044 should show these test results when probed.

*chronous* means that all of the flip-flops output at the same time.

This chip is able to count from 0 to 15 in binary. This is an upward count. The chip can also count the other way from 15 to 0. This is a downward count. That is why the chip is called an up/down counter. Not all binary counters can do that. Some can only effect an upward count.

For this application, the 74LS193 is using on-pins 5, 6, 9, 10, and 1 for inputs and outputs. In other types of applications other pins could be used. In this case though, the other input/output pins

are unused. To keep them from interfering with the counting in the chip, they are rendered harmless. Pin 15, a possible input type is held high by tying it to +5 volts. Pin 12 and 11 are bypassed to ground through an 82 pF capacitor. Pin 10 is held low at ground and pin 4 is held high at the supply voltage. The supply of +5 volts is connected to pin 16 and grounded at pin 8.

The counter is receiving an input from the adjoining 74LS629N oscillator chip. The frequency it receives is either the NTSC or PAL frequencies as determined by the jumper select. The 74LS193

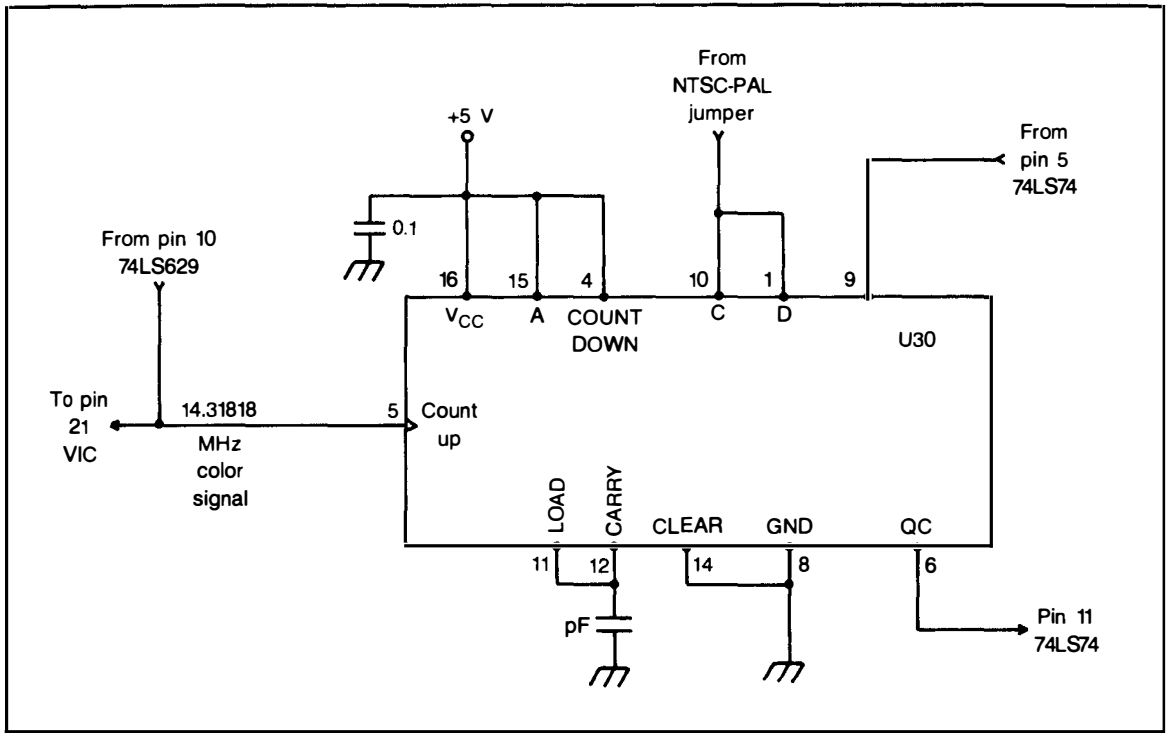


Fig. 8-10. The 74LS193 Up/Down Counter receives a sampling of the 14.31818 MHz signal at pin 5 and divides it into other useful frequencies.

then counts upwards and fills the flip-flops with highs and lows. When the count goes from 0 to 15, the chip outputs a pulse. This output pulse is 1/16th of the incoming pulse. The counter has divided the incoming pulse by 16.

The changed frequency is then transferred to the phase detector network. The phase detector outputs two frequencies. One is sent to the MPU, and the second to another section of the 74LS629N chip. All of this activity is discussed in more detail in Chapter 14.

Quick checking a suspected 74LS193 chip can be done with a logic probe. Figure 8-11 will show the states and the pulses. This illustration shows what should be present on the active pins. In order to read the actual frequencies at the input and output pins, an expensive lab scope is required because of the high frequencies involved. Fortunately, it is rarely necessary to look at the waveshapes and determine their frequencies for ordinary servicing.

It is usually only required at the factory during research and design.

### 74LS139 2-4 DECODER

This 2-4 decoder is a 16-pin DIP. Other chips are 3-8 decoders and 4-16 decoders. Notice the number relationships. In decoders, it takes four inputs to choose one out of a possible 16 outputs. In order to output one of eight outputs, you need three binary inputs. When there are four possible outputs, two logic state inputs will force one of them to output. The 74LS139 is a dual chip and has the 2-4 relationship. Each section has a pin called \*EN that provides additional control. Refer to Fig. 8-12. The 2-4 decoding won't perform unless the \*EN, which is normally held high, receives a low to turn on the chip section.

The inputs to the chip sections are address lines A11 and A10 for pins B1 and A1. Address lines A9 and A8 connect to pins B2 and B1. The address

lines can have the following four logic state situations at either A and B input pins. They can be low-low, low-high, high-low or high-high. If they are low-low, they will activate the V0 output. A low-high will turn on the V1 output line. A high-low enables V2. A high-high forces V3 on. Of course, there won't be any output unless \*EN also has a low input.

The decoders are part of the addressing scheme. The schematic in Fig. 8-12 shows the outputs are all going to chip select pins. In the number

1 section of the chip, each two bit address drives one of the V outputs low while the rest are held high. For instance, V0 is the VIC chip select. When address lines A11 and A10 input two lows while \*EN is low, V0 outputs the VIC enabling signal, \*VIC, a low. The signal is applied to pin 10 of the VIC, \*CS. This enables the chip select of the VIC. The VIC chip activates and lets the rest of the address, on the rest of the address bus open up a location inside VIC.

The same thing occurs at the SID chip. When

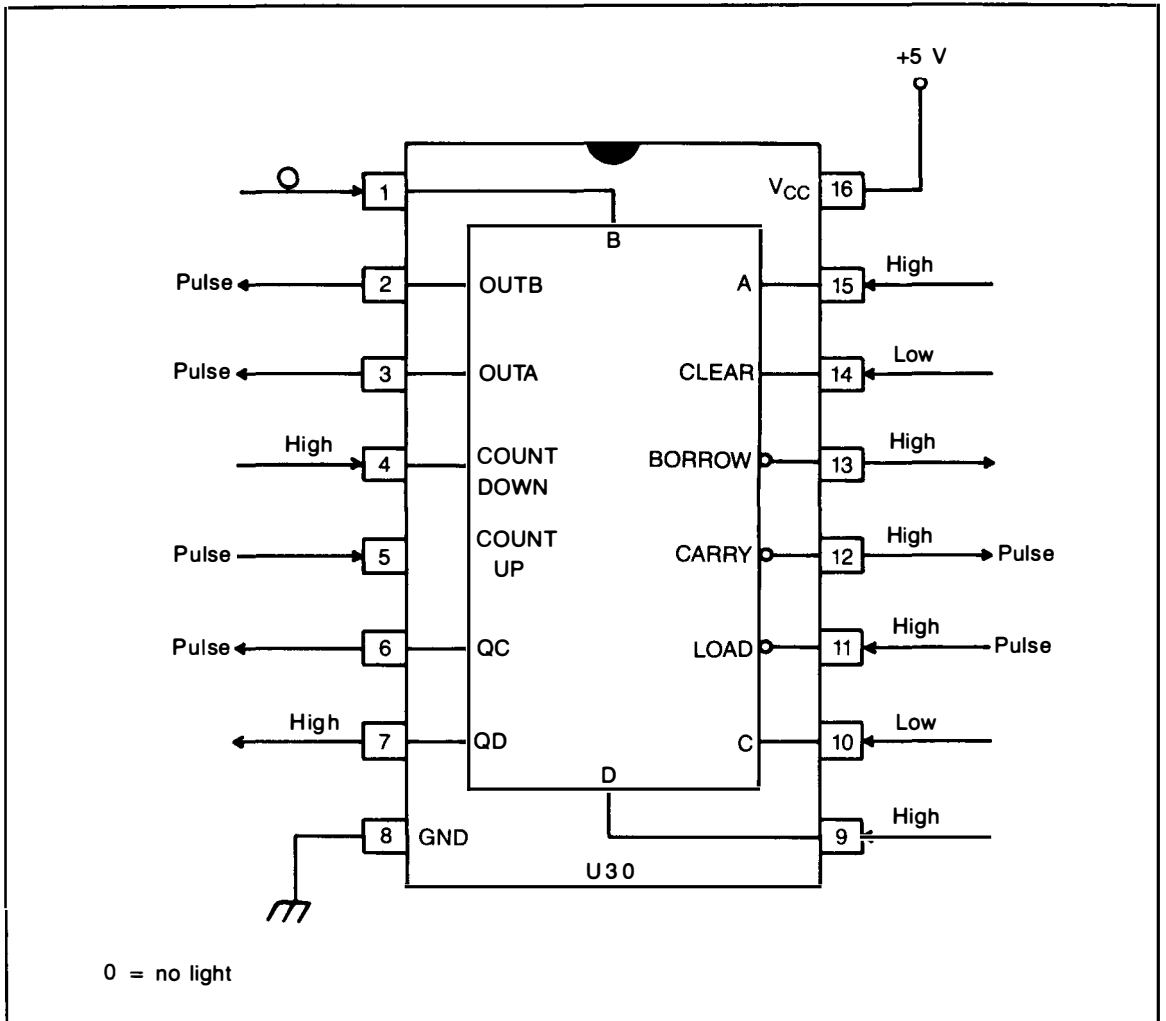


Fig. 8-11. Quick checking the 74LS193 with a logic probe should yield these inputs and outputs. If any are missing or incorrect, trouble is indicated.

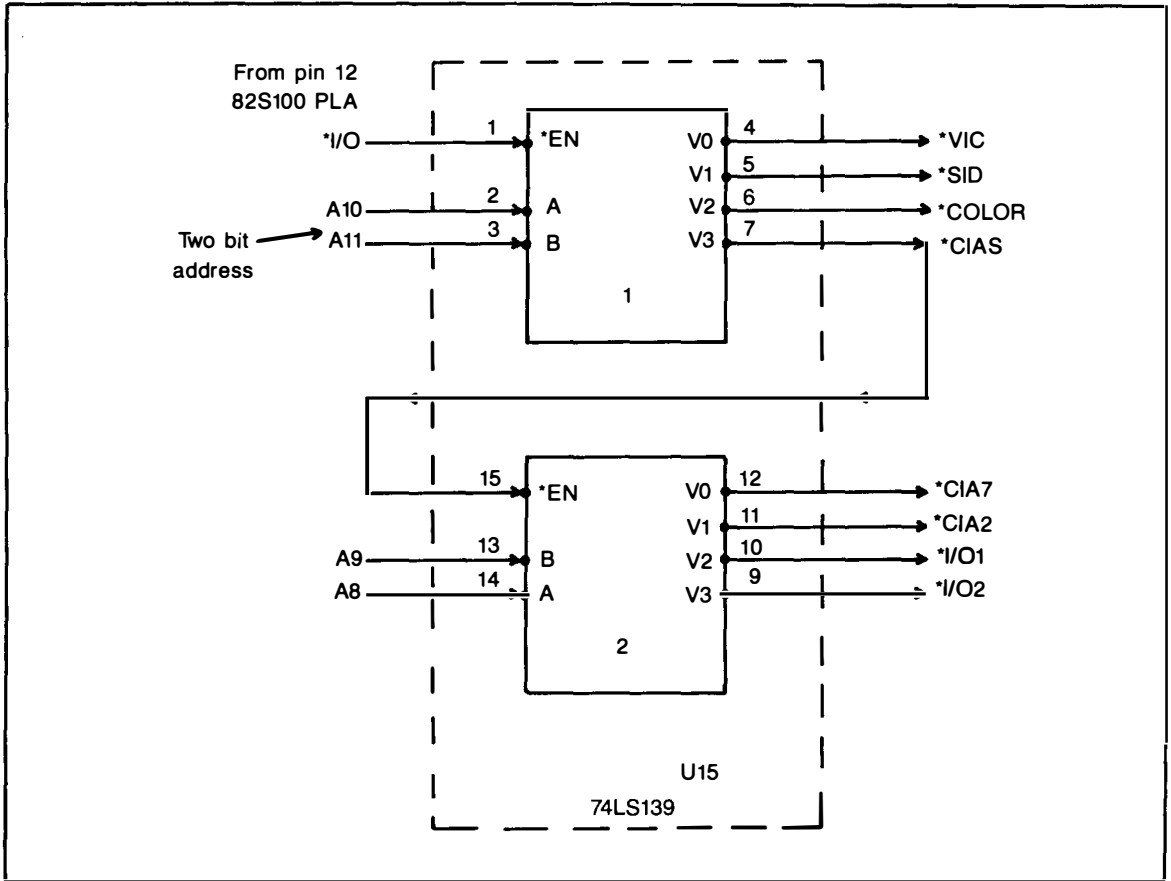


Fig. 8-12. The 74LS139 has two decoding sections that aid the PLA perform chip selection duties.

A11 and A10 place the two bit address low-high into the decoder chip, V1 goes low and outputs the low signal, \*SID. It enters \*CS of the SID chip, and the SID is selected, but the rest of the chips remain dormant.

When the two bit address to choose \*COLOR enters the decoder, as in Fig. 8-13, a low goes to one of the 74LS08 AND gates. If \*COLOR arrives at the gate at the same time as the high signal AEC, the AND will shut off. A low and a high shuts down an AND gate. Otherwise, the AND gate is on since two highs enter, a high exits and a high is applied to \*CS of the Color RAM. When the high from the AND gate is on \*CS, the chip is off.

When \*COLOR arrives at the AND gate though, the gate turns off which places a low on

\*CS of the RAM chip. As the AND gate turns off, the Color RAM is selected, and it turns on. All these offs and ons can be very confusing, but get used to it, for that is what computing is all about.

The situation is very logical but gets even more confusing, so follow it slowly. V3 puts out a low called \*CIAS. \*CIAS is chosen when the two bit address is high-high. \*CIAS is connected to \*EN of the second requirement of getting the number 2 decoder section to operate.

The next requirement is installing the two bit address from A9 and A8. When low-low and low-high are applied, V0 or V1 is chosen from the four possible outputs. V0 goes to \*CS of one of the CIA chips while V1 goes to \*CS of the other CIA chip.

In order to select a CIA chip, first you must choose \*CIAS and then one of the V signals from this second decoder section.

The last two decoder selects are attached to the Cartridge Expansion female plug. One can choose a disk connection, and the second selects the Z-80 card peripheral.

The 74LS139 decoder is a chip select helpmate to the 82S100 PLA chip. The \*EN on the first section of the decoder is controlled by the \*I/O output, from pin 12 of the PLA. Figure 8-14 provides a test point chart for checking out the 74LS139.

### 74LS257 QUAD 2-INPUT MULTIPLEXER

The 74LS257's stand between the dynamic RAM

array and the address bus. Refer back to Fig. 6-18. The 16 address lines are connected to the two chips, eight lines to a chip. The address arranging appears strange. The top chip receives address lines A15, A14, A13, A12, and A7, A6, A5, A4. The bottom chip accepts A11, A10, A9, A8 and A3, A2, A1, A0.

Each chip has four output lines, V0-V3. All of the output lines go to the same address inputs on all eight chips. The four output lines from the top 74LS257 go to MA4-MA7. The bottom lines go to MA0-MA3.

Note in Fig. 8-15 that pin 1 is \*SELA, the chip select. This select comes from the VIC. It is one of the \*CAS branches. Pin 15 is \*OE. It is the output enable and controls the three-state off and on ability of the chip.

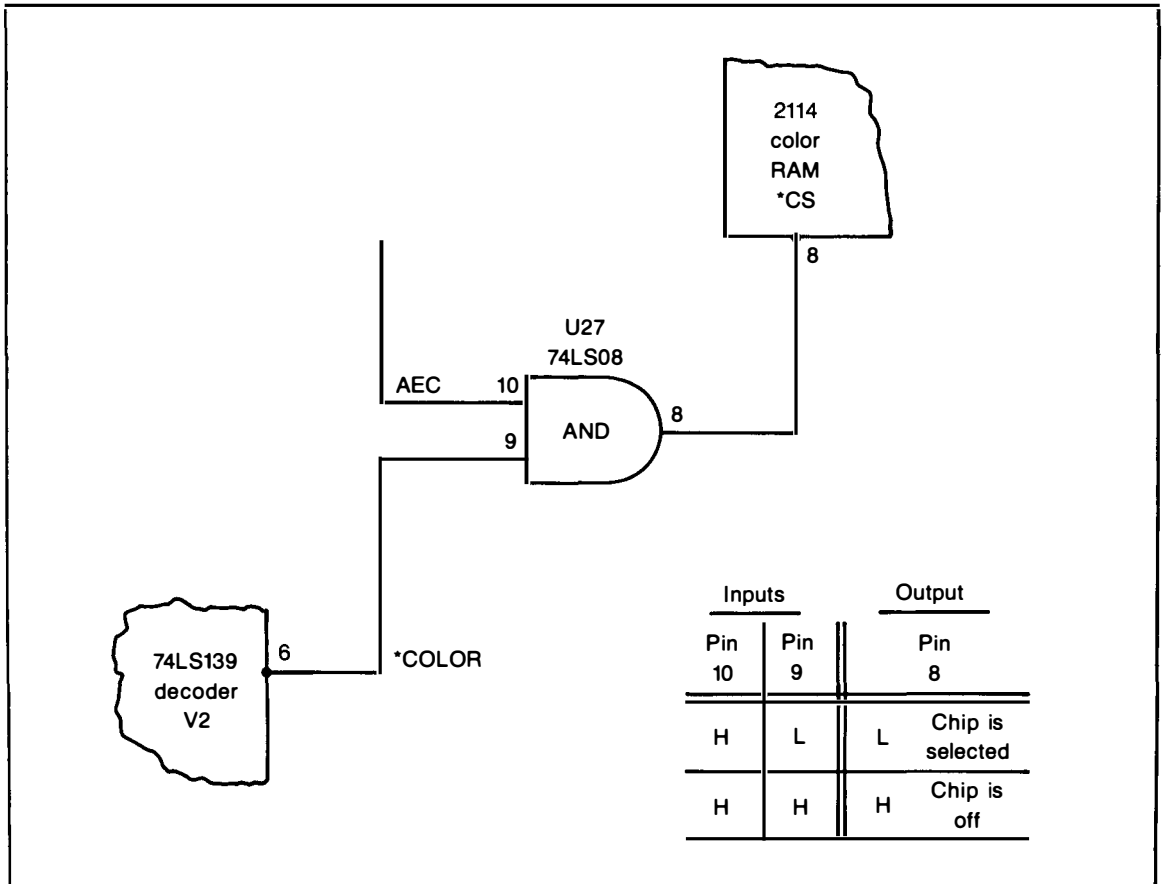


Fig. 8-13. The decoder provides the signal \*COLOR to an AND gate for Color RAM chip selection.

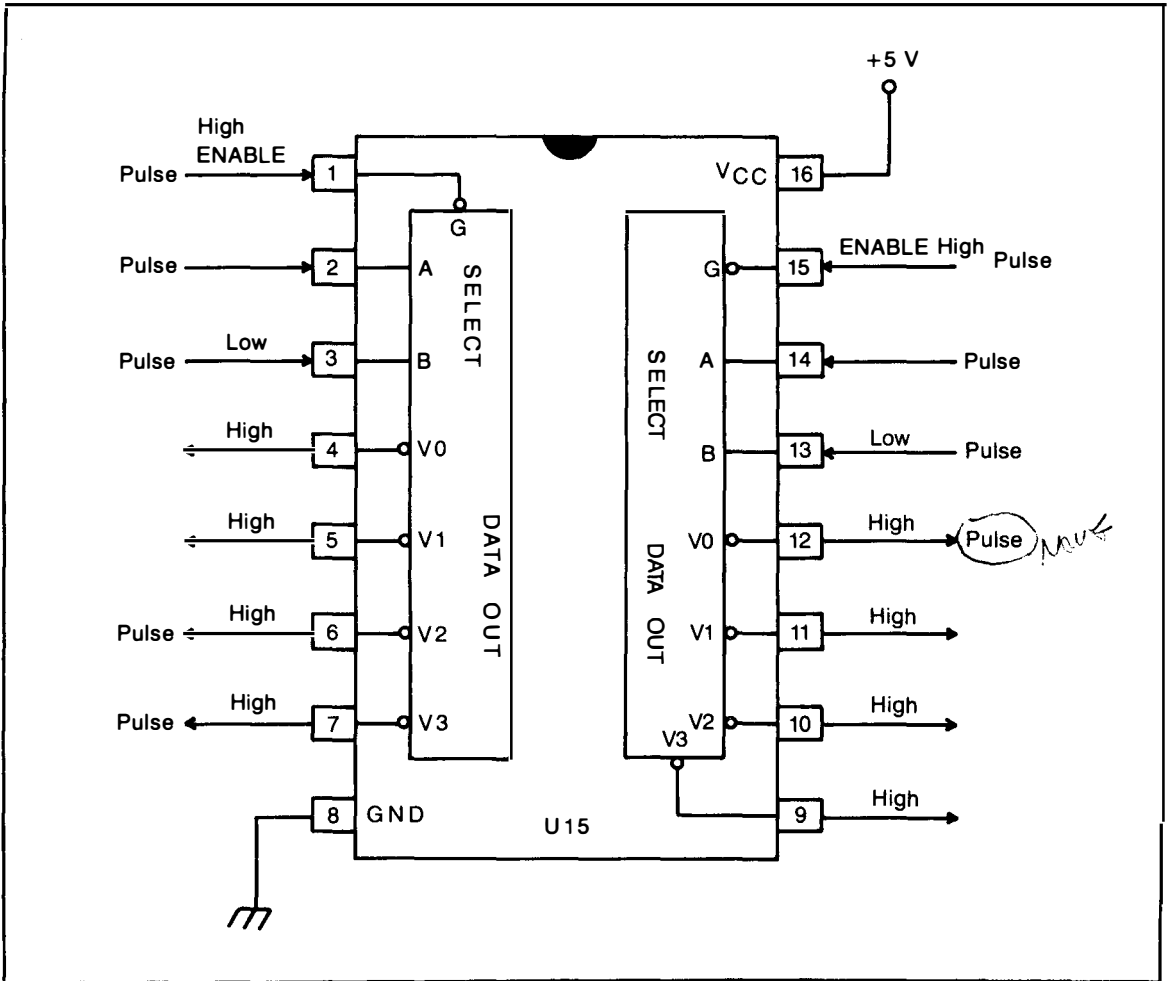


Fig. 8-14. The 74LS139 test points can be quickly probed and should yield these results if it is ok.

The chip is a multiplexer and it has four sections. There are two inputs and one output to the four sections. If you recall, the dynamic RAM chips have the matrix laid out in a grid of 256 rows and 256 columns. To address the columns requires eight addresses and to address the rows requires eight addresses. This takes up all 16 address lines. There are eight assigned to each chip. The chip select \*SELA and the three-state control \*OE act like two more address lines.

The multiplexers are able to organize the address lines to point out the rows and the columns. The actual scheme is covered in Chapter 13.

Pin 16 on each chip is powered with + 5 volts. Ground is at pin 8. The chips are quickly tested with a logic probe. The test point chart in Fig. 8-16A and B shows the highs, lows, and pulses that should be present at sign on.

### 74LS258 QUAD 2-INPUT MULTIPLEXER

There is a third multiplex chip in the 64 with the generic number 74LS258. It is almost identical to the 74LS257's. Besides the fact that it dissipates 35 mW and the 74LS257 dissipates 50 mW, the two chips are considered identical. The chip is installed as a workmate with the VIC. The actual wir-

ing is discussed in Chapter 17. Meanwhile let's take a look at the internal wiring of these multiplexer chips.

The test point illustration in Fig. 8-17 shows a block with leads to the pins. Pin 15 is the output three-state control. Pin 1 is the chip select. You can turn the chip on or off with these pins. A high on either pin can turn the chip off. It takes two lows, one on each pin to turn the chip on. All three chips use the same AEC and \*CAS for this purpose. All three chips get selected and freed from a three-state condition in the same way, but not at the same time. One of the 7406 NOT gates takes care of that, as Fig. 8-18 shows.

However, the two chips that are multiplexing the address lines to feed the dynamic RAM are using the bits coming in off the MPU's address bus. The chip working with the VIC is using address lines out of the VIC.

Note there are four input pairs and four single

outputs from the chip shown in Fig. 8-19. The internal wiring shows the circuits behind the scene. The input pairs lead to a pair of AND gates. One input lead goes to each AND gate. The single output line comes from an OR gate. Each of the four sections are wired with two AND gates and one OR gate. Also keep in mind that each gate is composed of microscopic bipolar transistors. The insides of the gates are discussed in Chapter 10.

With one lead on each AND gate being occupied by an input pin, the second lead is not accounted for. The drawing shows that the second leads are connected to the outputs of two NOT gates. All the A named AND gates are connected to the output of one NOT gate. The B named AND gates are all connected to the output of two NOT gates in series. These common connections are all coming from pin 1 which is the recipient of the signal \*CAS.

The two AND gates in each section attach to

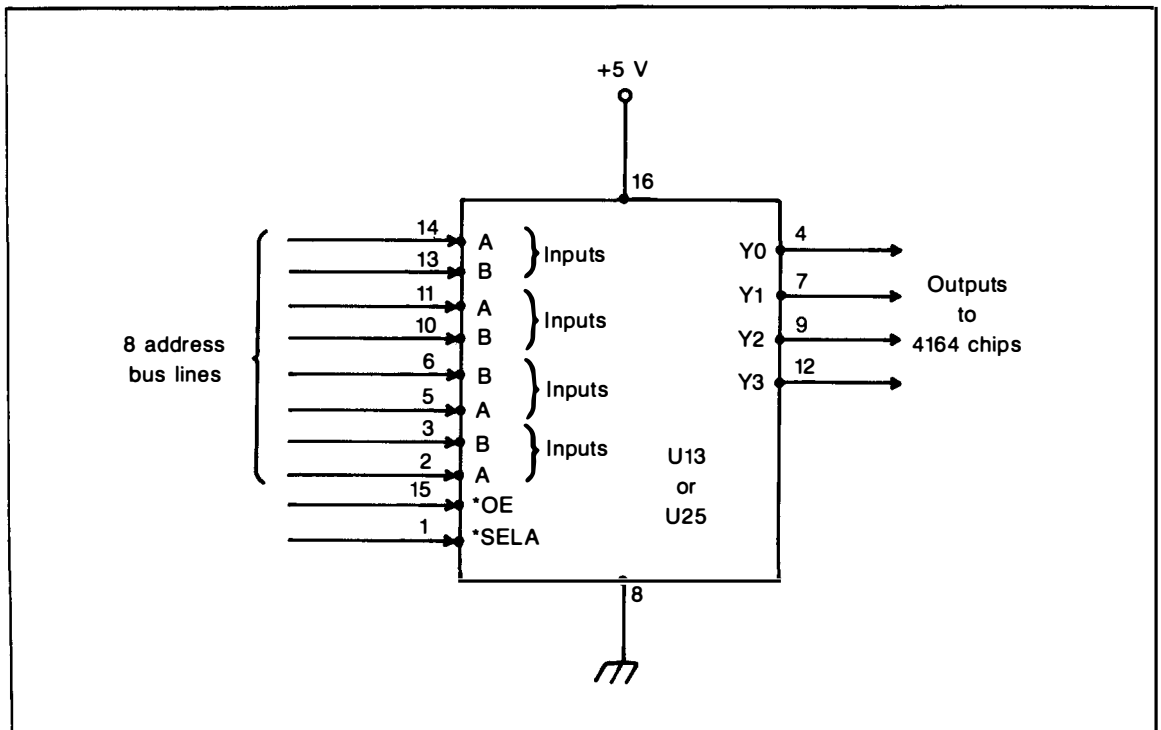
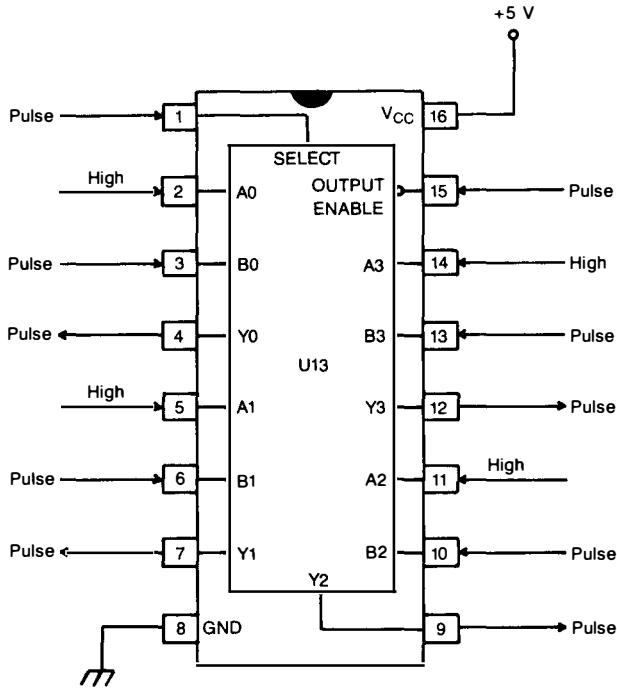


Fig. 8-15. The 74LS257 chips perform the multiplex addressing for the eight dynamic 4164 RAM chips. There are eight pins connected to the address bus and four output lines on each chip.



**A**



**B**

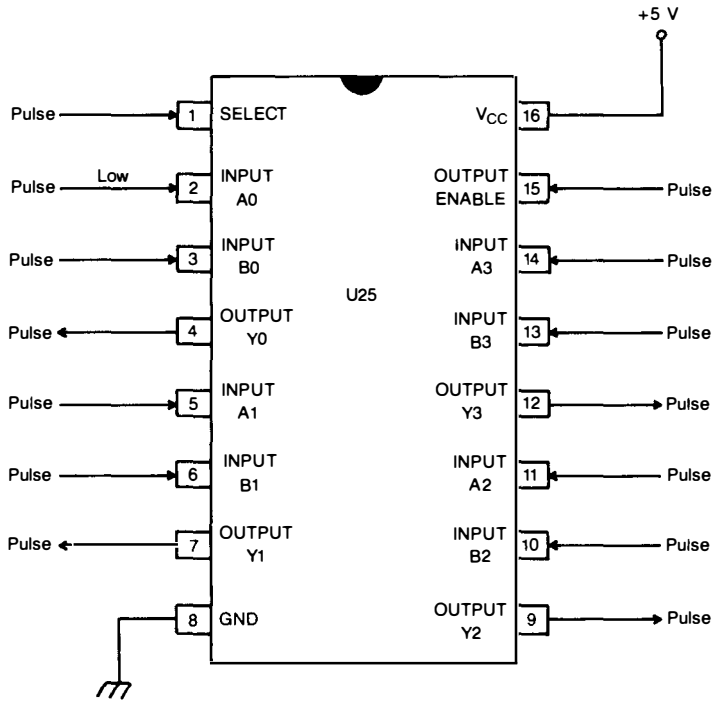


Fig. 8-16. The 74LS257 test points.

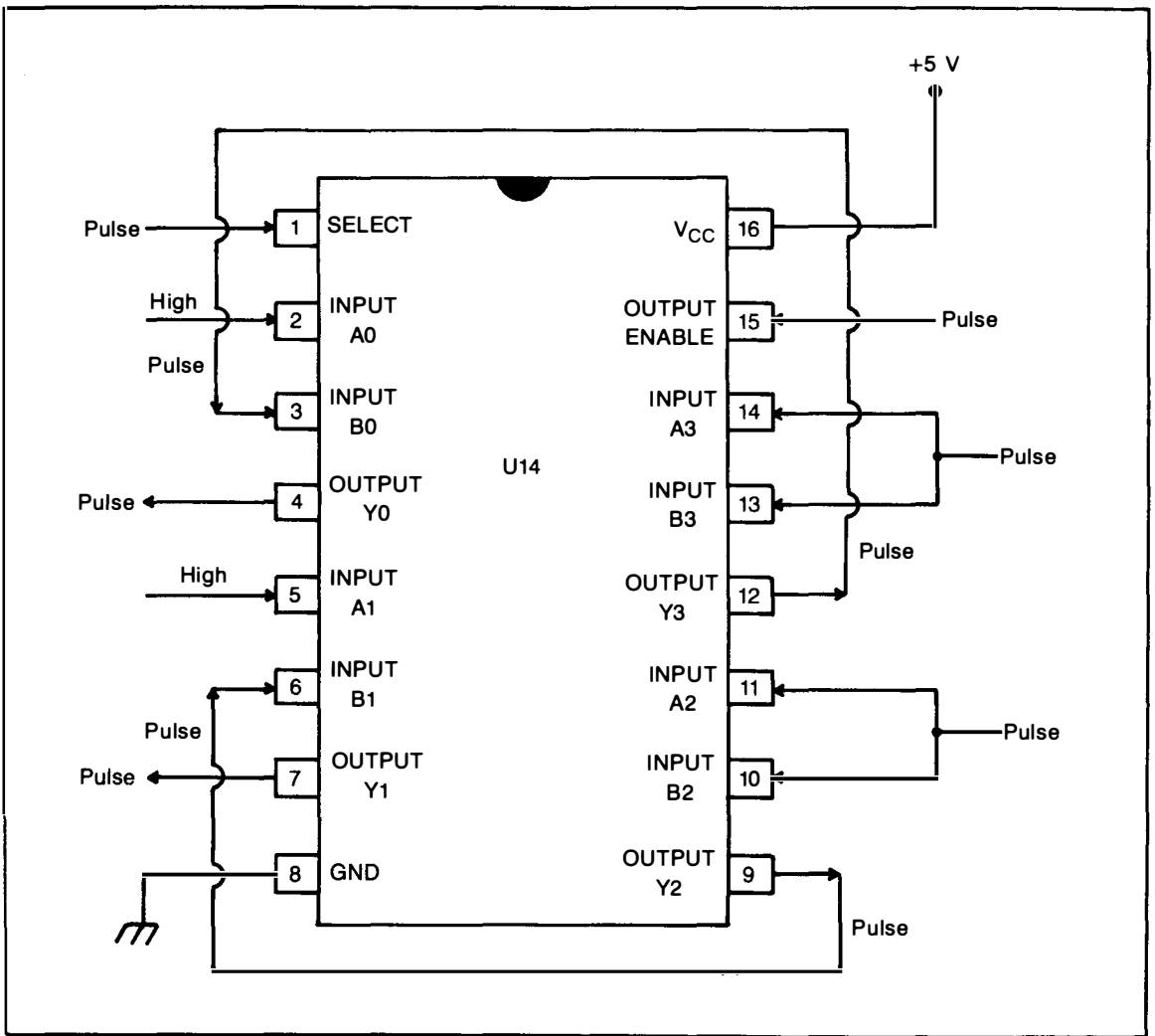


Fig. 8-17. The 74LS258 test points.

the inputs of the individual OR gates. The OR gates are three-state types and have a third connection for that purpose. These three-state connections are all wired together and go to pin 15 the three-state control. Pin 15 has the job of receiving the AEC type signal.

As mentioned, the AND gates will only output a high if both inputs are high. By using two AND gates in a section, data can be selected from two inputs, even though both inputs are arriving at the same time. For instance, suppose high address bits

from A15 and A7 arrive at this chip. They are injected into 11 and 10. You want to select the bit from A15 for a row addressing chore while selecting the bit from A7 to address a column.

You send a low signal \*CAS into pin 1. The low passes through the two NOT gates. After passage through the first NOT gate, the low becomes a high. The high is attached to the AND gate that connects to pin 11 with its incoming high. The two highs on the AND gate make it output a high.

Meanwhile, the \*CAS signal passes through

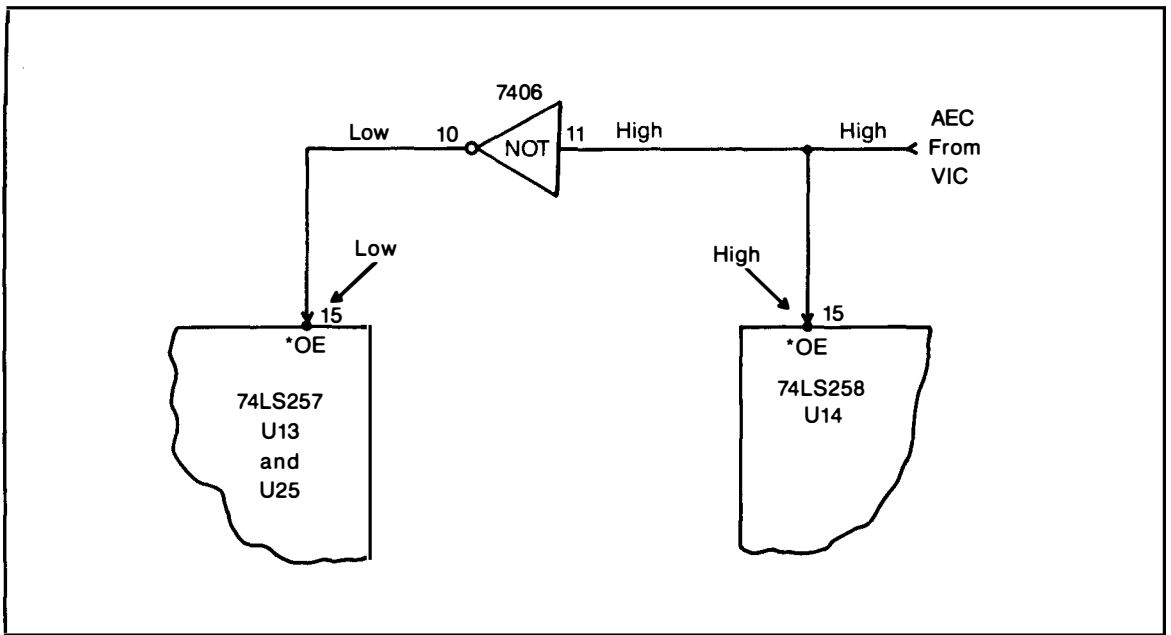


Fig. 8-18. Pin 15, \*OE, on both chips is separated with one of the 7406 NOT gates. By means of the AEC signal, this turns off one of the chips when the other one is on. They cannot interfere with each other.

the second NOT gate. It becomes a low again. This low is connected to the AND gate at pin 10. The high and low input stops the AND gate from outputting anything. As a result, the pin 11 AND gate outputs the A15 high and the pin 10 AND gate lies dormant.

At the next instant, \*CAS turns off. It becomes a high. The output of the first NOT gate becomes a low. The pin 11 AND gate turns off. The output of the second NOT gate becomes a high. The pin 10 AND gate now has two highs input. It outputs a high. The result is the high from A7 now passed through the chip. This choosing of data from one AND gate to the next is called multiplexing.

When the outputs leave the AND gates, they are injected into the OR gates. The OR will output a high when either one or the other of the inputs is high. That is what happens. First one of the inputs is high and then the other. First a row address is strobed into the RAM and then a column address is strobed in.

The OR gates can be turned off and put into a high impedance state by means of a low into pin

15. This three-state effect makes the chip appear to be completely out of the circuit. There is virtually no leakage of current or signal while the three-state control is exercised.

When either a high or low is passing through the chip, the chip has a low impedance. When the three-state control is on, the chip develops a very high impedance. This can be confusing if you are using a vom to check voltages. The vom will read about 2.5 volts at an output during three-stating. The voltage is all built up static noise levels and has no meaning to the computer operation. The logic probe simply does not light any of its LEDs during the three-stating.

If you are servicing a trouble, an open chip could appear as a three-state device. If you do encounter three-state evidence, it is possible that it is happening because the chip is faulty. Keep in mind that this is a common symptom of trouble.

### 74LS373 OCTAL D-TYPE LATCH

The 74LS258 chip outputs to a couple pins on the 74LS373 octal D-type latch. The latch also connects

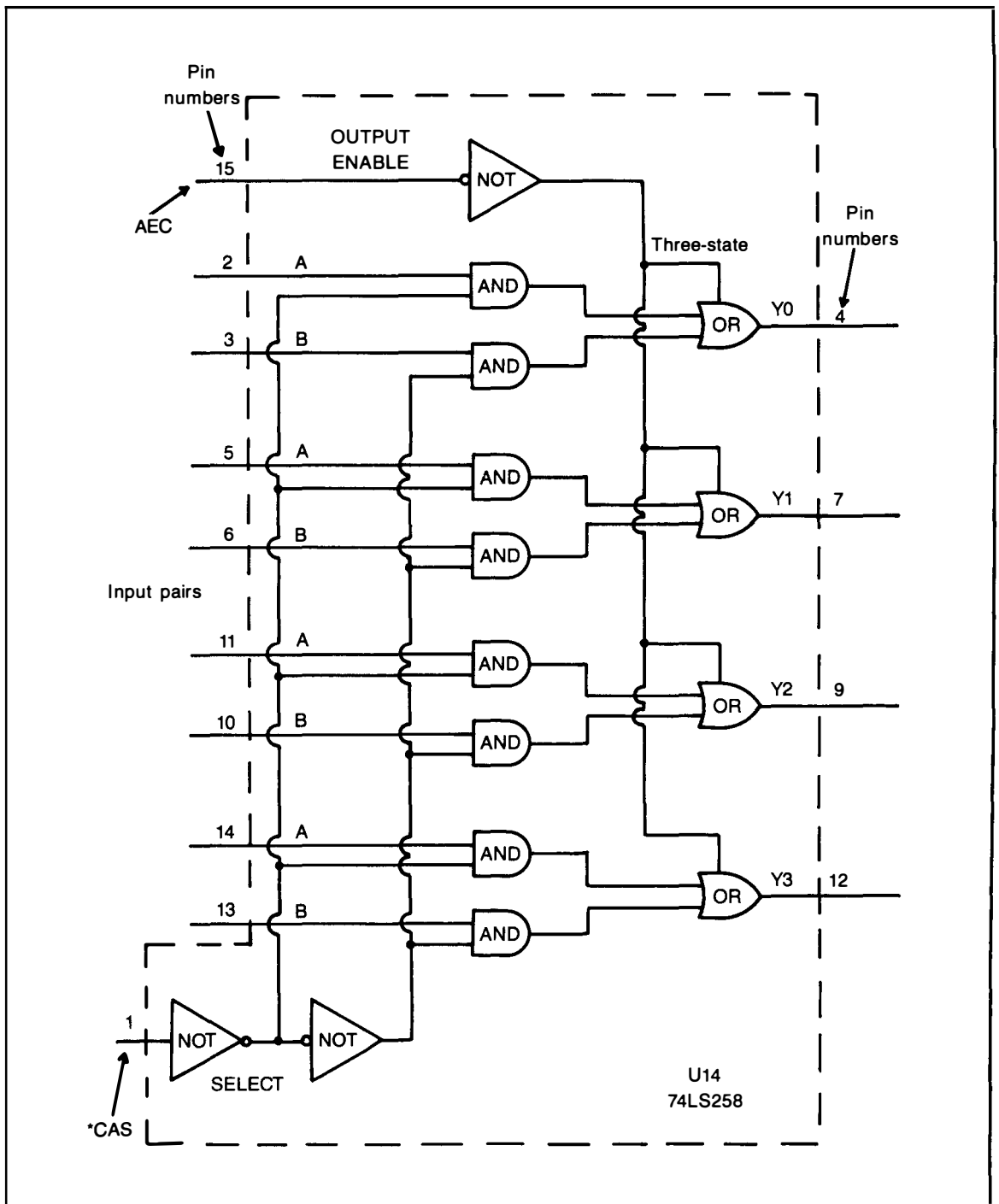


Fig. 8-19. The internal wiring shows the way the inputs and outputs are connected. There is one OR gate operating with each pair of AND gates.

between the VIC and the dynamic RAM. The other side of the latch interfaces with the ROMs and the Color RAM. It is a busy fellow and transfers all sorts of signals to and from these chips, as Fig. 8-20 demonstrates.

There are eight flip-flop circuits inside the latch. They are said to be "transparent". This means the output immediately follows the input as long as the chip is enabled. The enabling signal is a high.

A signal goes right through the chip, from input to output when pin 11, enable, is high, but when it is low, the chip does its latching. When enable is low, the eight highs and lows that enter the D

inputs get latched or stored in the flip-flops. Flip-flops are discussed in Chapter 11. When the enable subsequently goes high, the bits that are stored are then transferred to the output pins named Q.

Pin 11, the enable, is strobed by the VIC signal \*RAS. When \*RAS, a low, arrives it unlatches any signal that might be stored in the flip-flops. Pin 1 gives the chip an additional three-state control. It is connected to the system AEC line. While AEC is held high, the three-state condition is in effect, and the chip goes into a high impedance state. If AEC goes low, the chip is free to conduct its data transferring duties.

The 20-pin DIP latch is a staging area for the

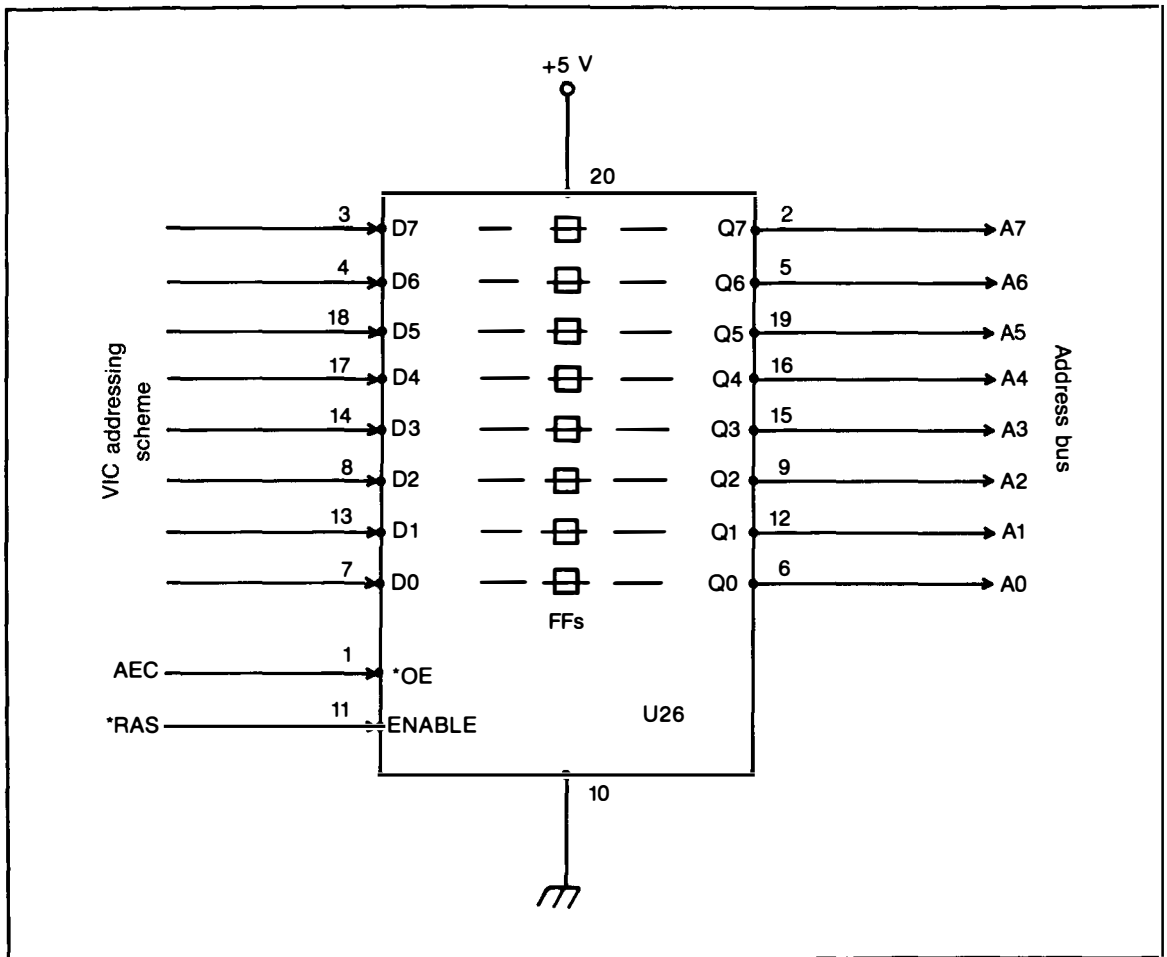


Fig. 8-20. The 74LS373 chip is a collection of eight flip-flops, each with an input and an output.

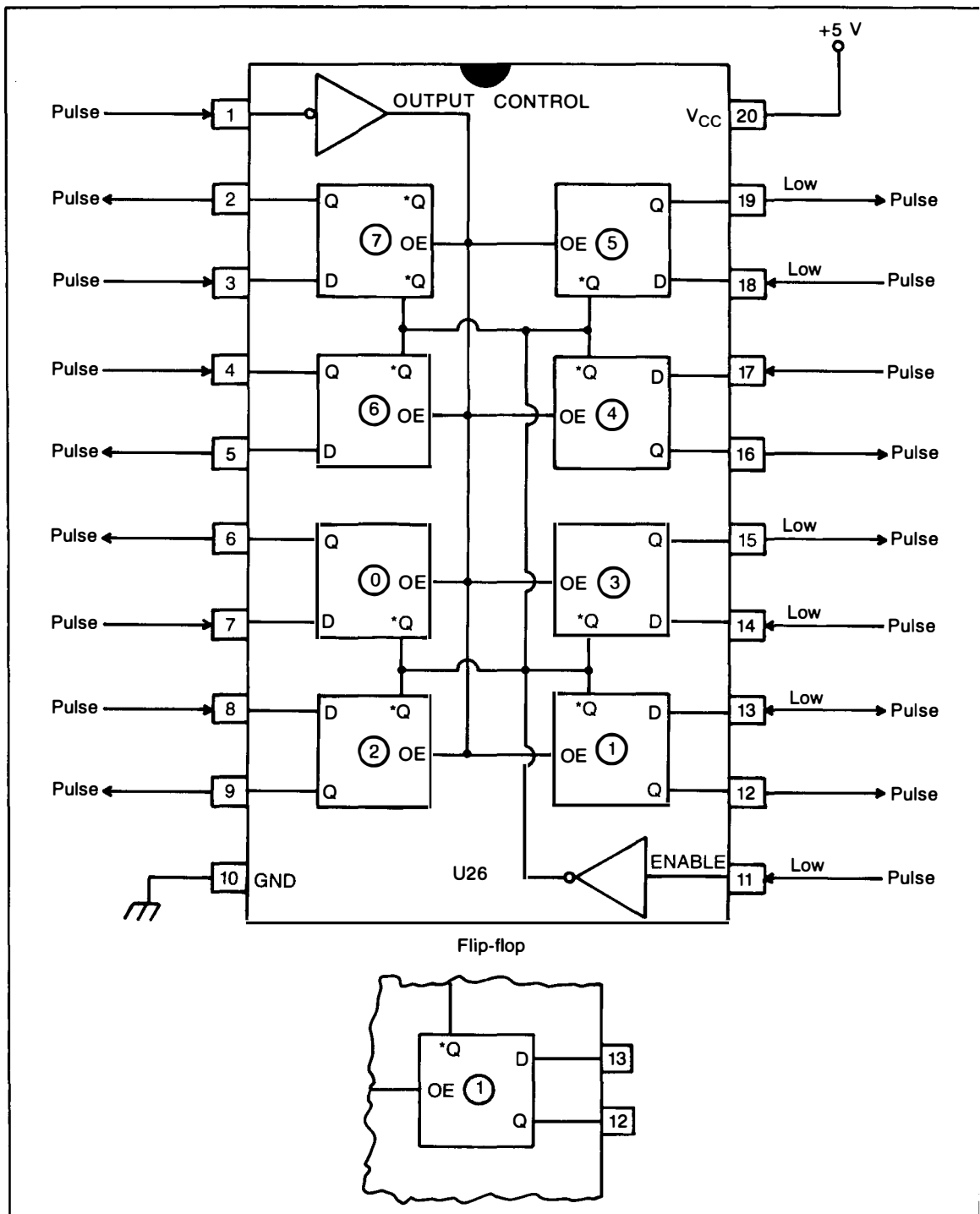


Fig. 8-21. The flip-flops have inputs at D and Q. They are each enabled at  $\bar{Q}$  and controlled at OE.

address bits that enter the eight lowest address lines of the Color RAM. It aids in the choosing of colors for the character blocks in the TV display. Figure 8-21 shows the pin connections and the test points.

### 4066 QUAD BILATERAL SWITCH

There are two 4066s in the Commodore 64. The first one, shown in Fig. 8-22, is located off the data lines at the VIC. The second one, shown in Fig. 8-23, is connected between the Serial Bus female plug and the two Control Ports. As their name implies there are four sections to the 4066 and the sections are switches. They are four off-on switches, electronically controlled. They are handy to control data bus lines or select data in other ways.

Each of the four switches have three connections. There are first of all two I/O lines. Signal can flow in these lines either way. The I/O lines for the A section are pins 1 and 2. B section are pins 3 and 4. C section are pins 8 and 9 while D section are 10 and 11. There is nothing mysterious, these are the off-on switch connections.

The third connection is the off-on control. When you want to close a switch, you connect its control connection to a high. If you want it permanently closed, attach it to the supply voltage pin 14. This puts +5 volts on the control, the switch stays closed.

If you want to open a switch, you connect its control connection to a low. Should you want a permanent open switch, connect the control pin to the

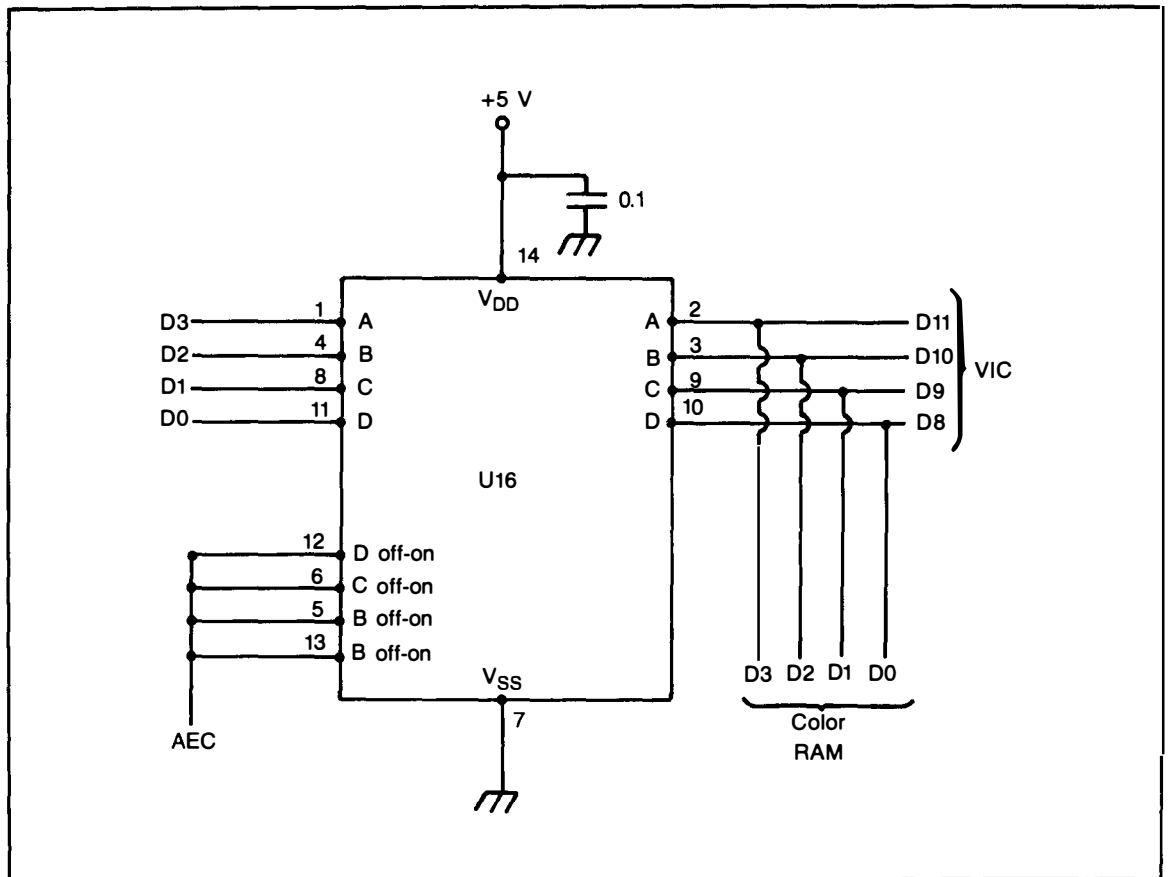


Fig. 8-22. The 4066 chip is a set of four electronic off-on switches. One 4066 is stationed in the data bus to work with Color RAM and VIC.

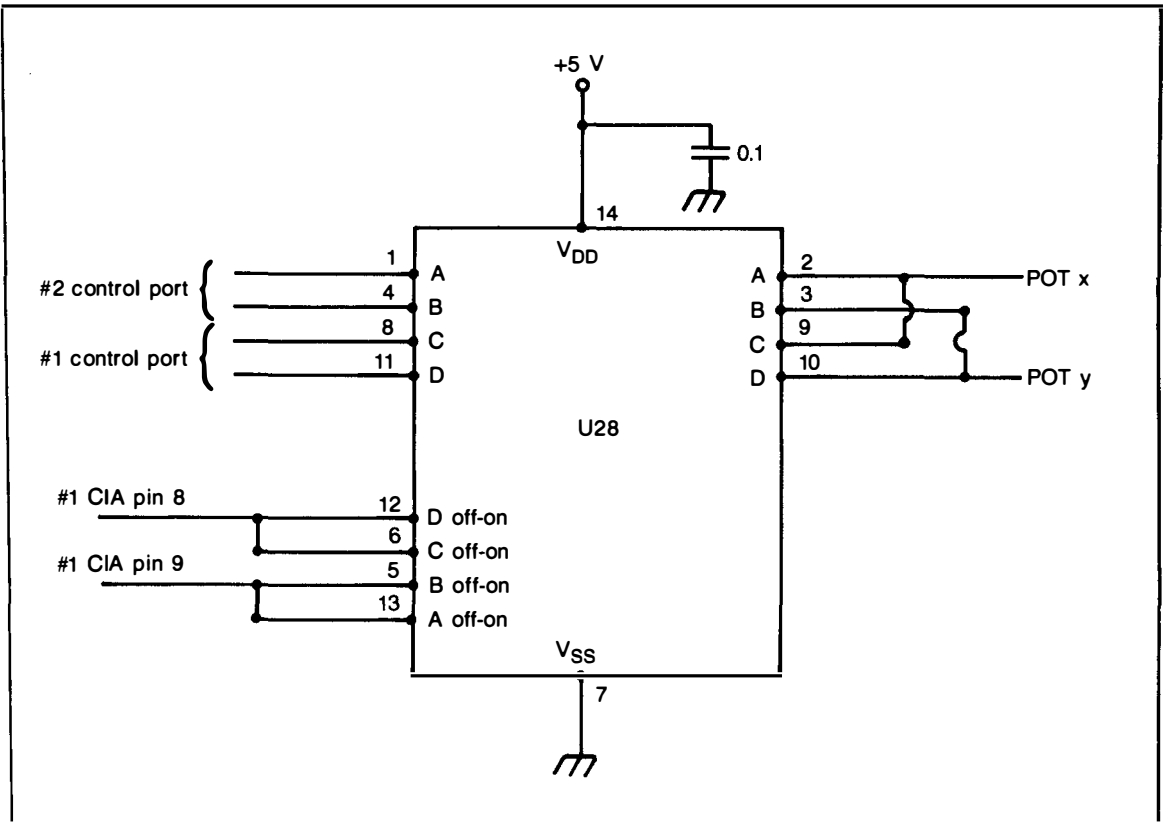


Fig. 8-23. A second 4066 chip is installed to work with the two Control Ports and the x and y pots.

ground of the chip, pin 7. This low puts 0 volts on the control and the switch will not close.

The control pin for the A switch is pin 13. The control pin for B switch is 5, the C switch is 6, and the D switch is 12.

These 4066 switches are fairly rugged. They can take a supply voltage range up to + 15 volts, although the 64 uses + 5 volts. The danger with switches in sensitive digital circuits is the static electricity that can build during the switching. The Commodore design is excellent, but during circuit operation that is other than ideal, the switching can develop troubles and the chip can fail.

They are easily tested with the test point charts in Fig. 8-25.

## 556 DUAL TIMER

There is one 556 chip in the computer. On the chip

are two separate timers. One of the timers is in the \*RESET circuit while the other is in the \*NMI arrangement. They are connected as shown in Fig. 8-26. The reset circuit and the non-maskable interrupt circuits are discussed in later chapters. Figure 8-27 provides the test point data.

The timers are used to clock out cycles as the reset or interrupt is controlled. These timers are able to time events ranging from microseconds to hours. They are versatile devices and used in many applications besides in the 64.

Each timer is based around a flip-flop storage circuit. The single flip-flop can flip-flop once or a large number of times according to the control applied. The control is given through two comparator circuits. The comparator is discussed in Chapter 11.

When a section of the 556 is made to flip-flop once, it is said to be acting as a one-shot timer. If



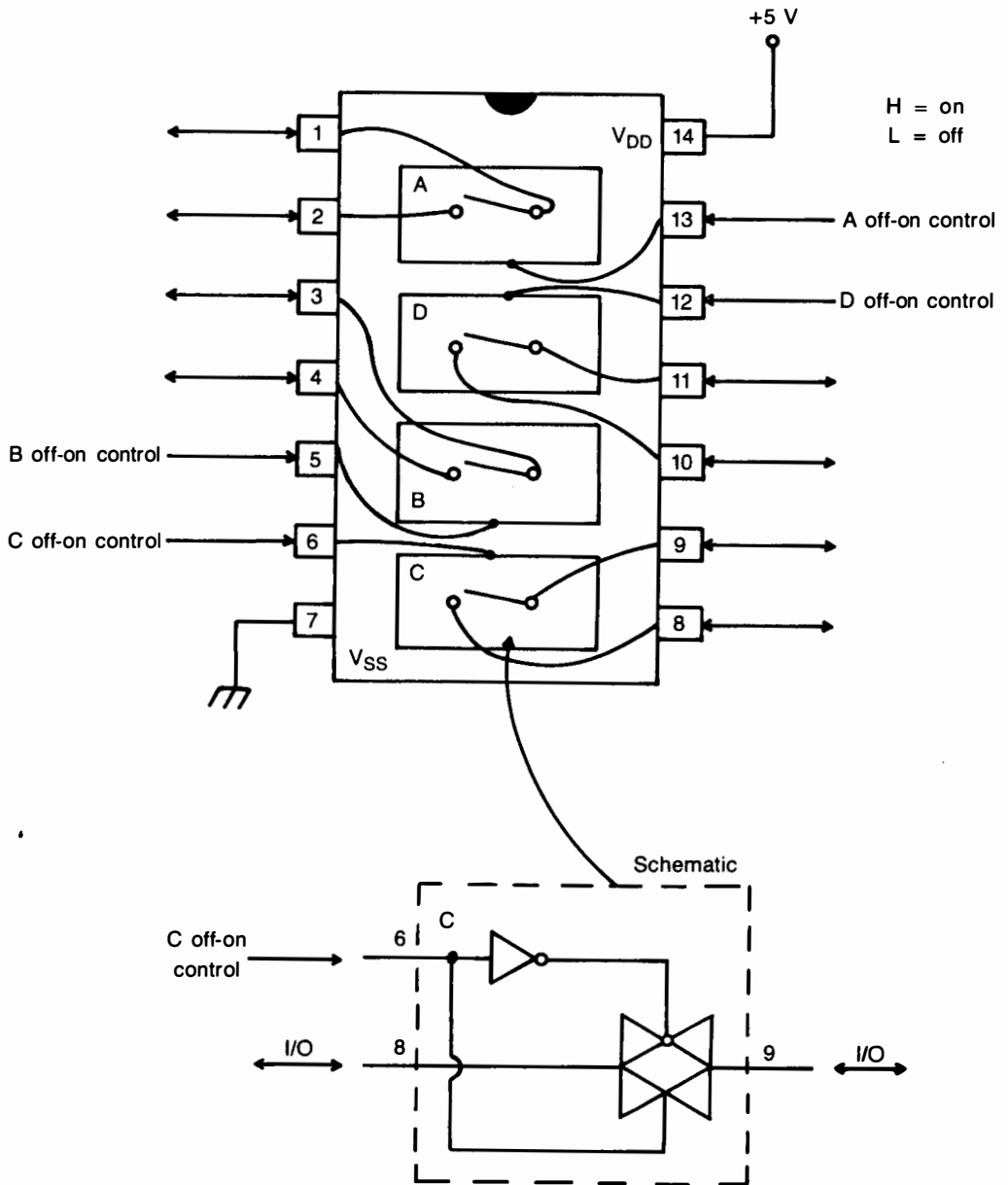


Fig. 8-24. Each of the switch sections have two input/output pins and an off-on pin.

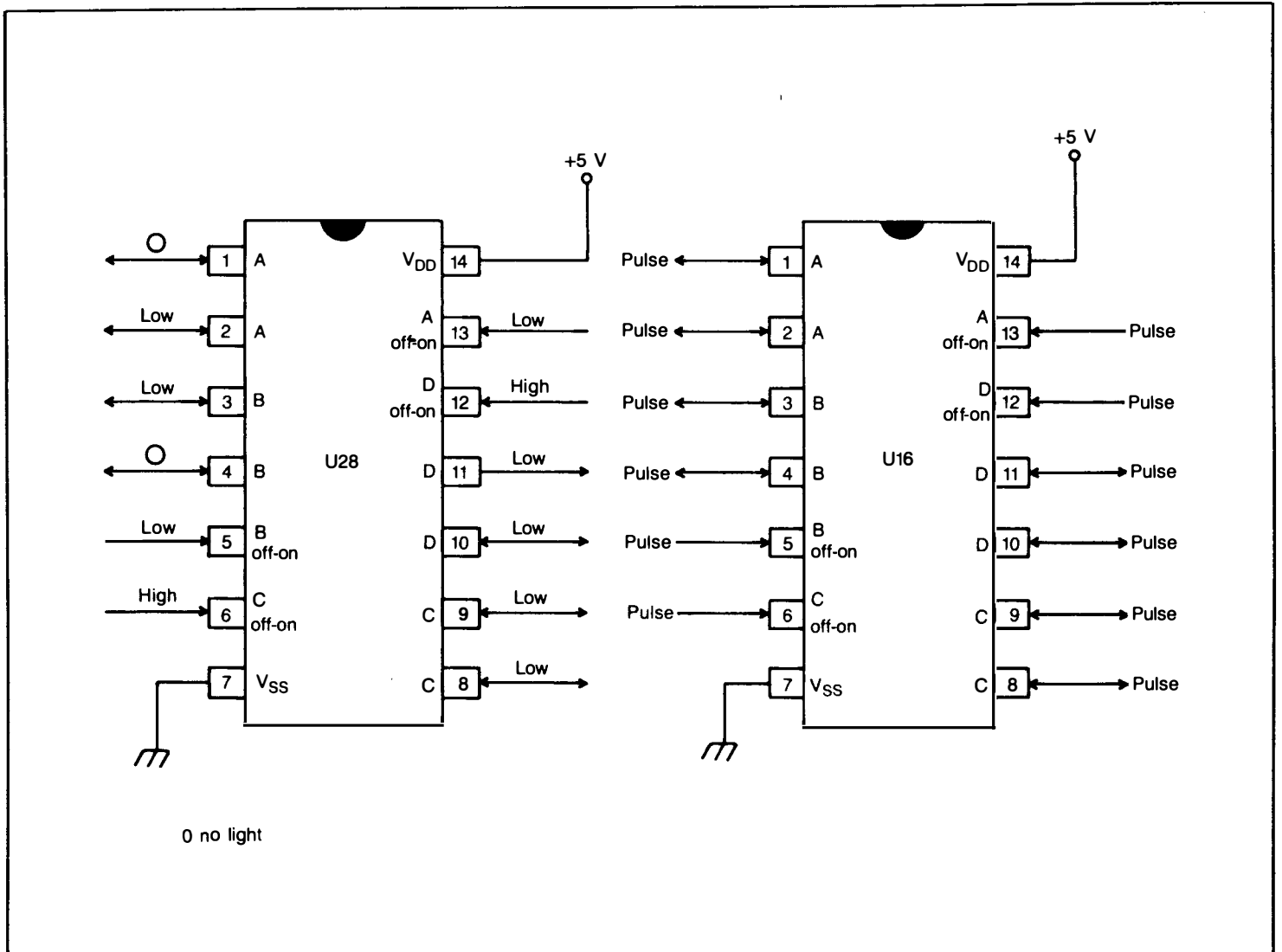


Fig. 8-25. The switches are conducting different type signals. U28 test points are an assortment of highs and lows. U16 is working with all sorts of pulses.

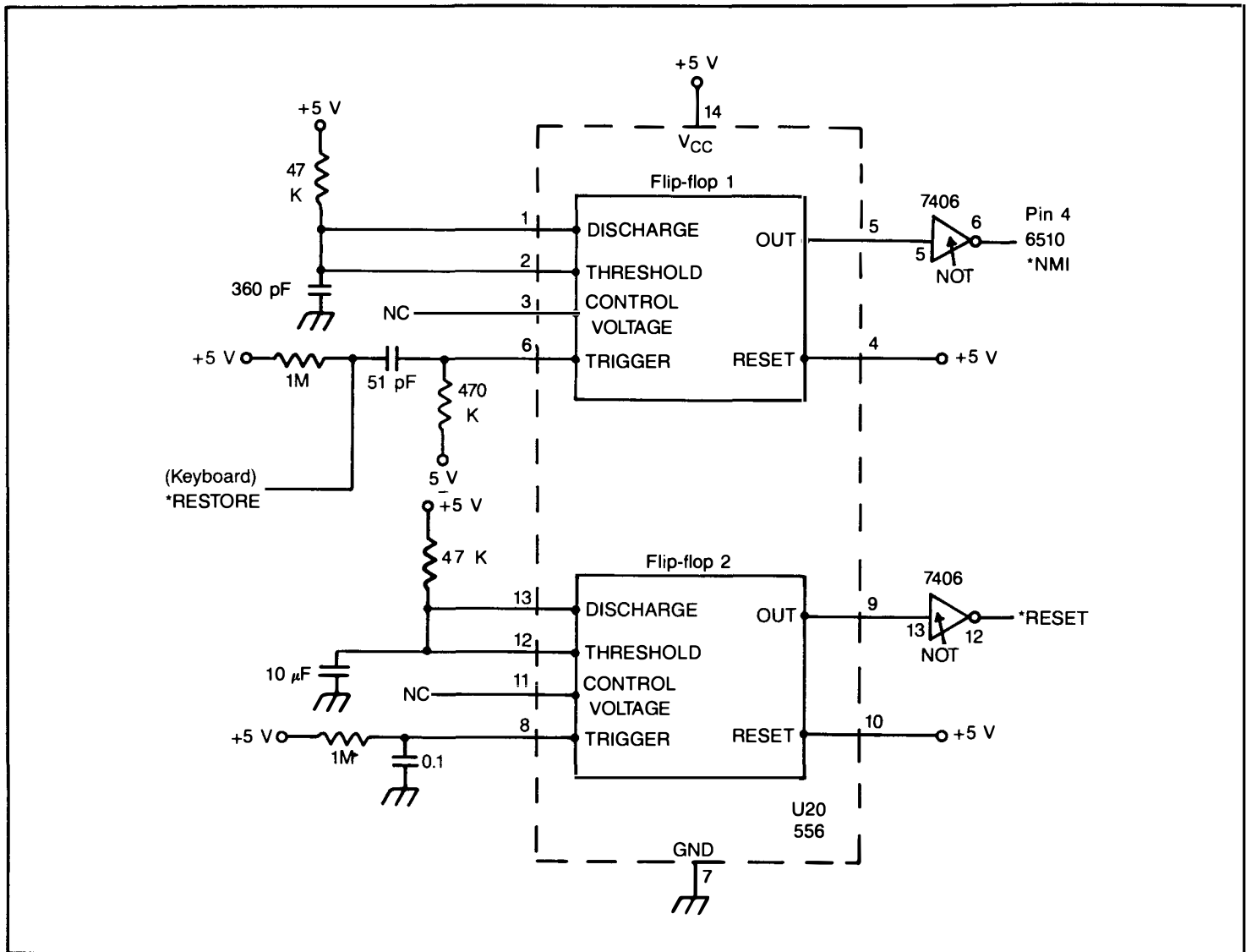


Fig. 8-26. There are two separate timers in the 556. One conveys the keyboard \*RESTORE signal to the MPU. The other handles the \*RESET signal.

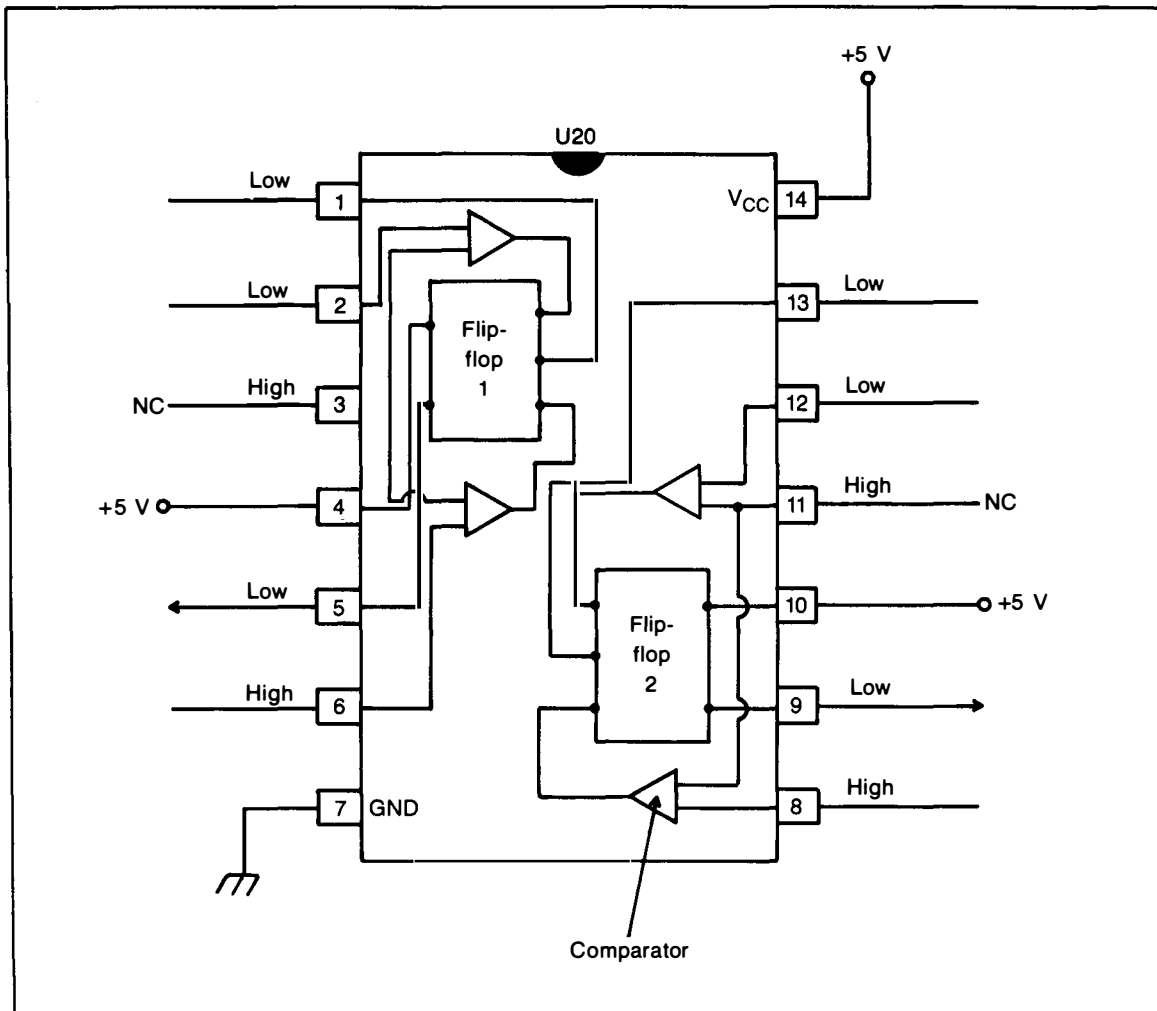


Fig. 8-27. The timers can be probed with this test point chart.

a section is made to keep on flip-flopping, it is called an astable oscillator.

The reset circuit uses one section of the 556 to time out the reset sequence. The reset circuit connects to the MPU, the two CIAs, the serial bus, the user port, the Cartridge Expansion plug and the SID chip. The reset 556 chip emits either a high or low according to the state of the flip-flop.

The reset circuit is used to start the MPU and the other circuits as power is applied to the computer. When the output after the NOT gate is low the circuits are shut off. If a high exits the reset

stage, the circuits are turned on the MPU goes into its power-on sequence. The details of the sequence are discussed in Chapter 12.

The other timer on the 556 is wired into the \*NMI circuit. This nonmaskable interrupt is a function of the RESTORE key on the keyboard. When you hit the RESTORE key, the impulse travels directly to the trigger input of the timer chip. This in turn, causes the chip to flip-flop once and output a signal to pin 21, \*IRQ, of one of the CIAs and also to pin 4, \*NMI of the 6510. The RESTORE effect then takes place.

## OTHER CHIP-LIKE COMPONENTS

In addition to the main LSI chips, the RAM and ROM chips, and all these support chips, there are a number of other components I have hardly touched on. First of all, I mentioned the MC4044 unit that is in the clock circuit. It is not really a

digital-type chip. It is used in frequency controlling applications. It is instrumental in producing phase changes and frequency variations of the computer's basic crystal controlled frequency. This chip will be covered in the clock chapter.

Another little IC that appears in the clock cir-

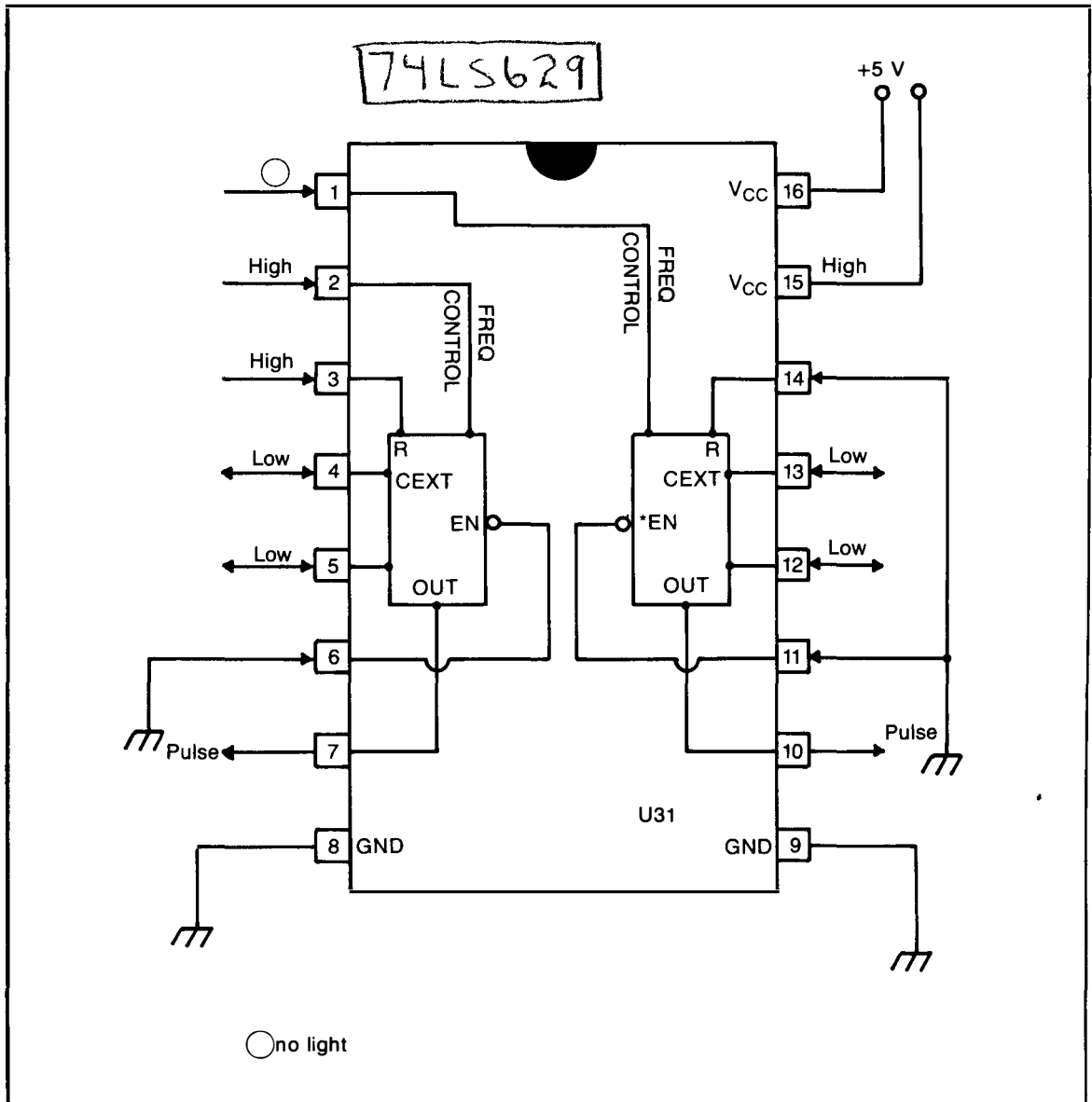


Fig. 8-28. The 74LS629 is the oscillator circuit of the computer. These test point results must be present or the machine will shut down.

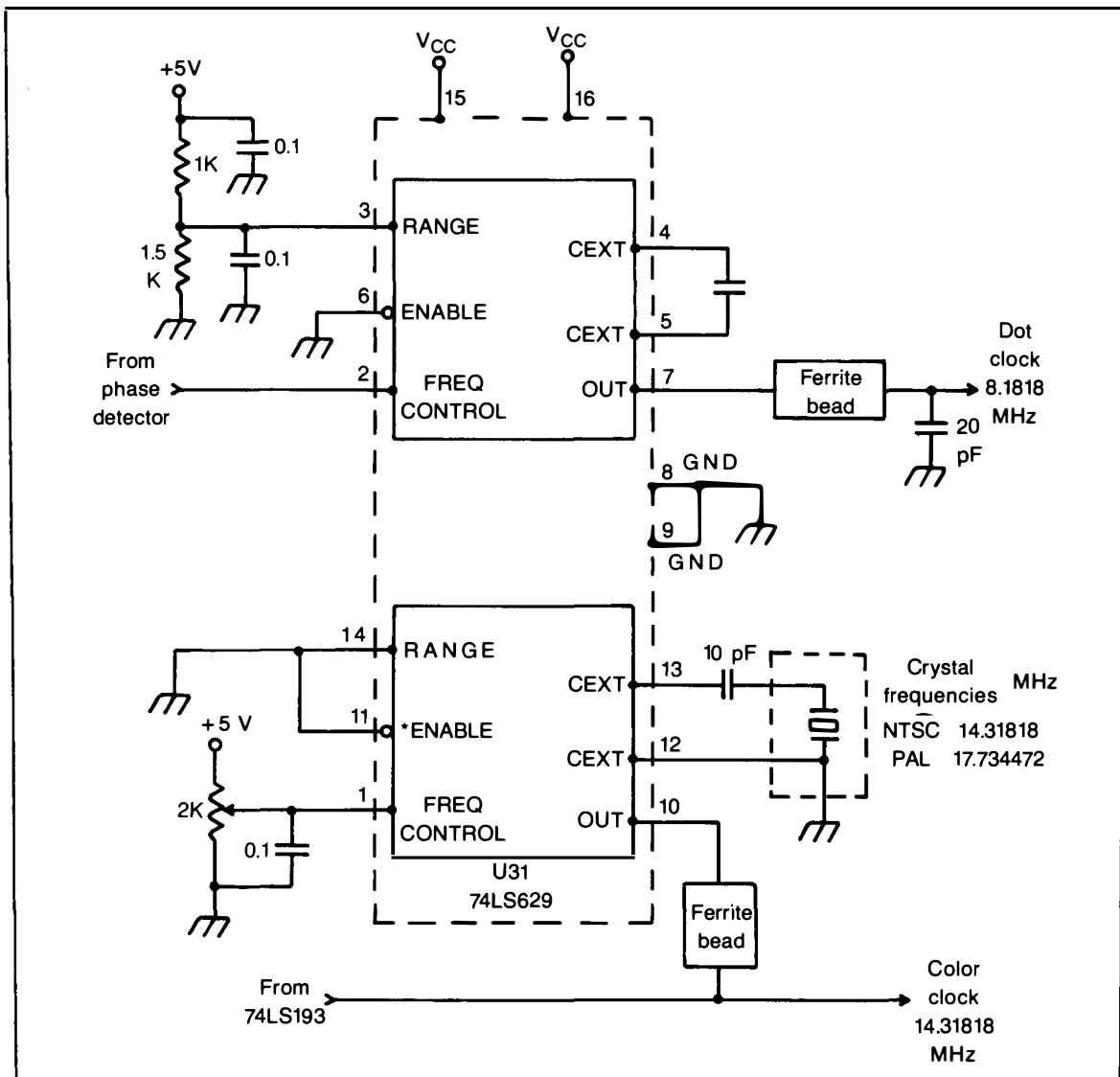


Fig. 8-29. The actual master oscillator frequency is produced in only one section of the chip. The crystal oscillator is installed at pins 13 and 12 of the bottom section.

cuit and is also covered in the clock chapter is the 74LS629N. It is called a Dual Voltage Controlled Oscillator with Enable. It is an oscillator circuit that produces the computer's basic frequency along with the 14.31818 MHz crystal. It is covered in more detail in the clock circuit. The pinout and test point data is given in Fig. 8-28. The schematic drawing is shown in Fig. 8-29.

In the power supply, there are two regulators, a 12 volt type called MD7812CT and a five volt type named 7805. These are mounted on the print board (Fig. 3-2). Inside the external power supply along with the power transformer, there is another regulator. These are discussed in detail in Chapter 20, Power Supply.

A black and white photograph showing a person's hand holding a logic probe. The probe is connected to a keyboard, likely a Commodore 64 keyboard, which is resting on a computer system. The probe has several buttons and a small display on its face. The background is a plain, light-colored surface.

## 9. System Block Diagram

**T**HE COMMODORE 64 LOCATION GUIDE POINTS out the 32 chips on the print board. The discussions of the chips gives you some idea on what types of jobs they perform in the computer. The symptoms of trouble can give you some indication of what chips or their companion circuitry can be causing a trouble that you are attempting to get fixed. The test point charts allow you to take vom or logic probe readings and compare your results with the states that should be found on the DIP pins. These measures should give you a leg up on most troubleshooting and repair jobs.

At this point, it is time to examine the computer in detail. I'll start with the block diagram of the computer in this chapter and then examine individual blocks in subsequent chapters.

### BLOCK DIAGRAM

From a block diagram point of view, computers today are not much different than the original electronic types used after World War II. The 18,000 vacuum tubes in the 1950 ENIAC have been

transformed to 18,000 microscopic transistors on one chip, but the block diagram view is about the same. Figure 9-1 presents the block diagram for the 64.

The central block in the diagram is based around the 6510 MPU. MPU stands for microprocessing unit. The core of the device is the ALU which is the arithmetic logic unit. That is the part that takes the input information and does arithmetic and logic manipulations upon the data. The finished data is then output. The ALU is surrounded inside the 6510 by numerous registers and gates that are covered in Chapter 12.

There is one important input-only block. Inside this block is the keyboard and one of the CIA chips. The keyboard practically uses up that CIA's capacity. When you hit a key, the impulse created by a keyboard row being shorted to a keyboard column is transferred to the CIA. The CIA in turn transfers the impulse to a register in the MPU. The keystrike is then processed and appears on the TV display.

Another large segment of the diagram consists

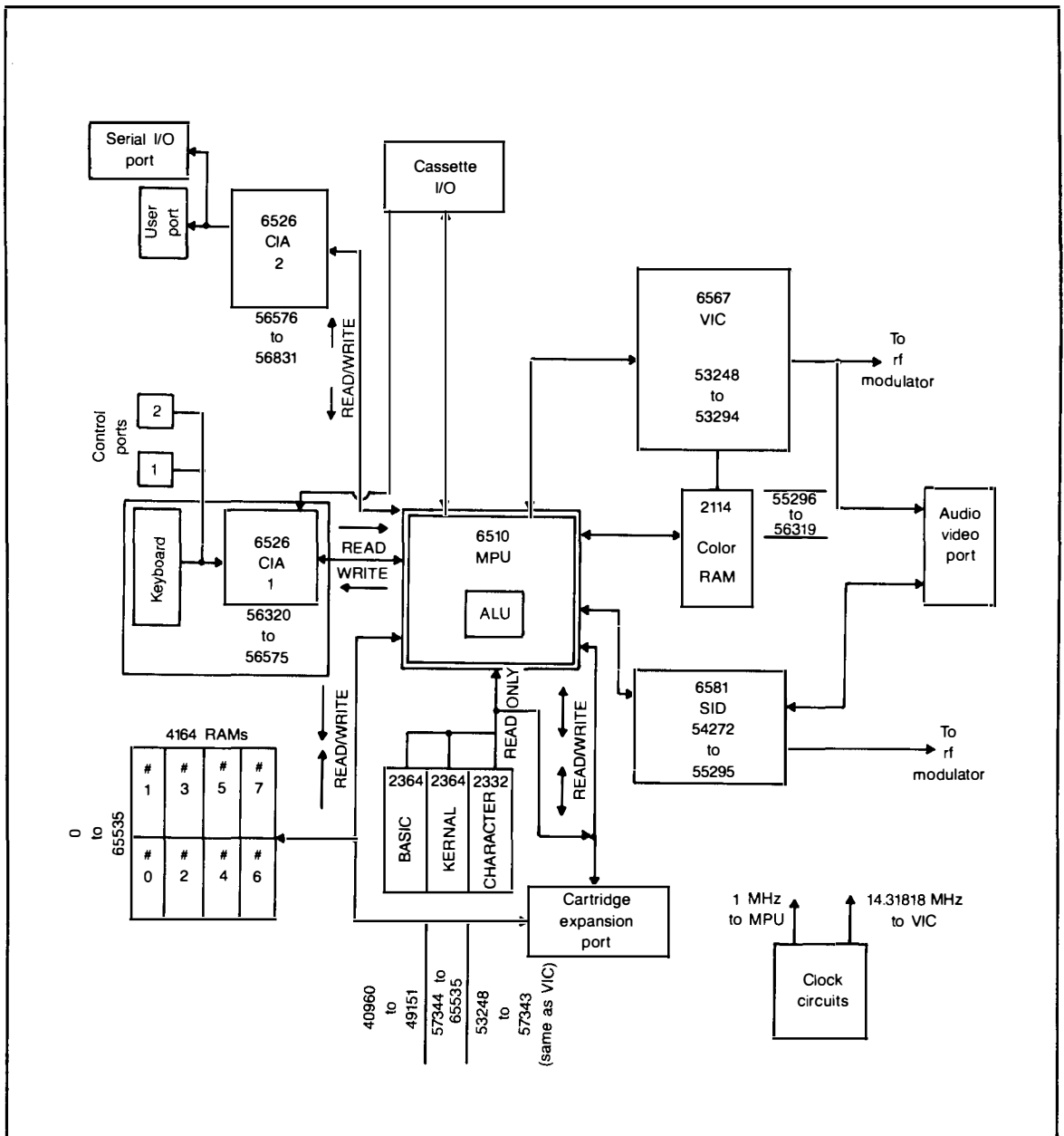


Fig. 9-1. The 6510 microprocessor with its ALU is the chip that makes the machine a computer.

of the residents of the 64's memory map. The main work that the MPU does is transfer binary bits to and from the residents of the memory map. The MPU can be the transmitter of the data, or the receiver. When it is transmitting data, it is writing

to the map. During the receiving, the MPU is reading the data from the map.

The chips that live in map addresses are most of the large ones that we have touched on so far. The support chips do not have addresses, although



they are in the data transfer pathways. They just help out and perform specific jobs to expedite the data transferring.

In the memory map area, the first obvious chips are the eight 4164 dynamic RAM chips. The MPU works with them constantly. The MPU can transmit or receive from these read/write chips. The MPU is able to store housekeeping instructions for operating the computer in some RAM locations. It is able to store the keyboard input information in a section reserved for screen memory work. The MPU can store addresses of sprite characters. The MPU can store programs you write and data that goes with the programs. The storage of information is easily accomplished with the eight RAM chips. The MPU can then retrieve any information by properly addressing the locations containing desired information.

The next group of memory map residents are the ROM chips: the BASIC ROM, the Kernal ROM and the Character Generator ROM. The PLA while it is technically in the ROM category, does not have any specific addresses. It does not contain any stored data. It has stored multiple pathways for directing the traffic signals that are passing through its insides.

The three addressed ROMs can transmit data to the MPU, but they cannot receive data from the MPU, since its bit holders are full of factory burnt in data. When one of the ROMs has a location addressed properly, the MPU can read the location's contents easily.

The two CIAs also have addresses, but they are not storage depots like the RAM and ROM chips. They are the ports of exit and entry for the 64. The MPU can transmit to the CIA addresses and then be able to send data out of the digital circuits to peripheral devices. Also the MPU can receive information from the CIA addresses and enjoy getting data from peripheral devices. The data that travels to and from the CIAs has no storage facilities inside the 64. The data must pass through the port to wherever it is coming from or going to. The CIAs only have a few addresses in comparison to the thousands of the RAM and ROM.

The other residents are the VIC, SID, and the

Color RAM chip. The VIC is addressed so that the MPU can contact it and conduct the video display business they are built to do. The SID is addressed to allow the MPU to use sound. The Color RAM has addresses to allow the MPU to place data into its bit holders.

The Color RAM has different size locations than all the rest of the residents. The Color RAM only has four bit holders in each location. All the rest of the residents have eight bits to a location. Eight bits are able to hold one of 256 combinations of highs and lows. The four bits in the Color RAM locations can only hold 16 different combinations of highs and lows.

The four bits per location is really all that is needed in the Color RAM. All it does is change colors in its respective TV display locations. One location controls the color of one of the 1000 blocks in the TV display. The 16 variations of bits in a location lets the addressed location change the lit block to one of 16 different colors. There will be more about this in the VIC chapter.

The overall block diagram shows that the 6510 MPU is Grand Central to the rest of the computer. Although the MPU doesn't have the intelligence (the RAM and ROM contains that), it does do the majority of the heavy work that goes on. The computer system consists of the MPU in the center, connected to the memory map residents around it.

## **MPU AND DYNAMIC RAM**

The 6510 MPU has four types of lines connected to its 40 pins. In Fig. 9-2, you can see 16 address lines. They are A15-A0, and they travel around the print board as a bus line. There are 8 data lines, D7-D0. They also go all over the board as a bus. Next are eight lines that act as a special input/output port right on the MPU. They do not run as a bus although they emerge from an 8-bit register in the MPU. They are designated the lettering P7-P0. The rest of the pins from the MPU are control lines. They are all individual and do a lot of different jobs.

One of the destinations of the MPU lines is the RAM array. The 16 address lines are connected into the 74LS257 multiplexers on their way to the RAM chips, as shown in Fig. 9-3. The two chips

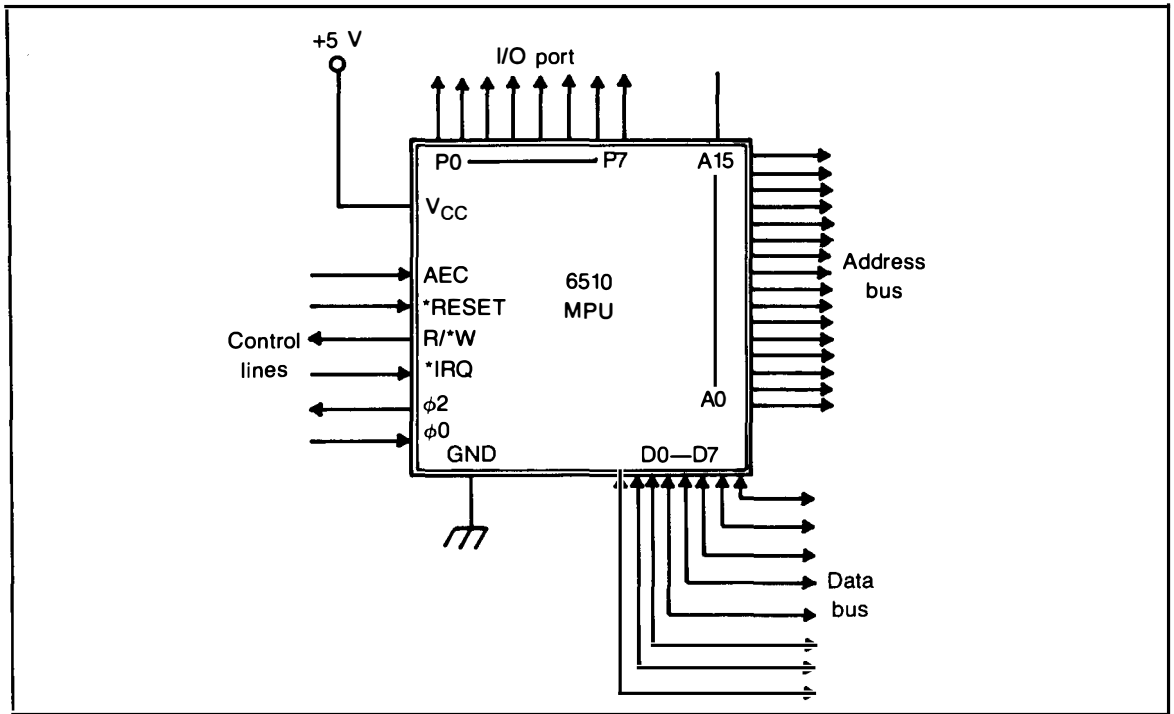


Fig. 9-2. The 6510 has four types of lines connected to it. They are the address, data, control, and one-byte I/O port.

receive eight lines each. Address lines are one way bus lines. The bits in an address can go out from the MPU, but they do not return. The address bits thus enter the two multiplex chips heading towards RAM. They will not be making a return trip.

The multiplexers output the row address and column address of a location. Notice that the address lines going into the multiplexers are staggered by fours. That is, the top chip is receiving lines, A15-A12 and A7-A4. The bottom chip is receiving A11-A8 and A3-A0.

The eight bits needed to address the rows of the RAM matrixes are A15-A8. The eight bits needed to address the columns of the matrixes are A7-A0. The chips with their AND and OR gate internal circuits are able to automatically output together, A15-A8. Then the strobe signal \*CAS comes along and turns off those input pins. \*CAS opens up the remaining input pins and they, A7-A0, enter the RAM chips.

Inside the RAM chips, there are two strobe signals going on and off. The two signals are

originating in VIC. One is \*RAS. It strobes the row address A15-A8 into the row address decoders internal to the chips. This addresses the rows. Next the other strobe signal \*CASRAM strobes the column address bits A7-A0 into the column address decoders inside the RAM. This addresses the columns.

Once the rows and columns are addressed, one bit in each of the chips is open for business. That business, of course is, being written to or being read. The addressing of a RAM bit is all the address lines and multiplex chips do between the MPU and the RAM array.

The next job is accomplished by the data bus lines. There aren't any data bus support chips between the MPU and the RAM. The lines D7-D0 are wired directly to the D0 and D1 connections on each chip. There are eight chips and eight data lines to the chips. The two pins on each chip are wired together but they act separately. One is an input line and the other is an output line. They are able transfer data in or out of the addressed bit. If the

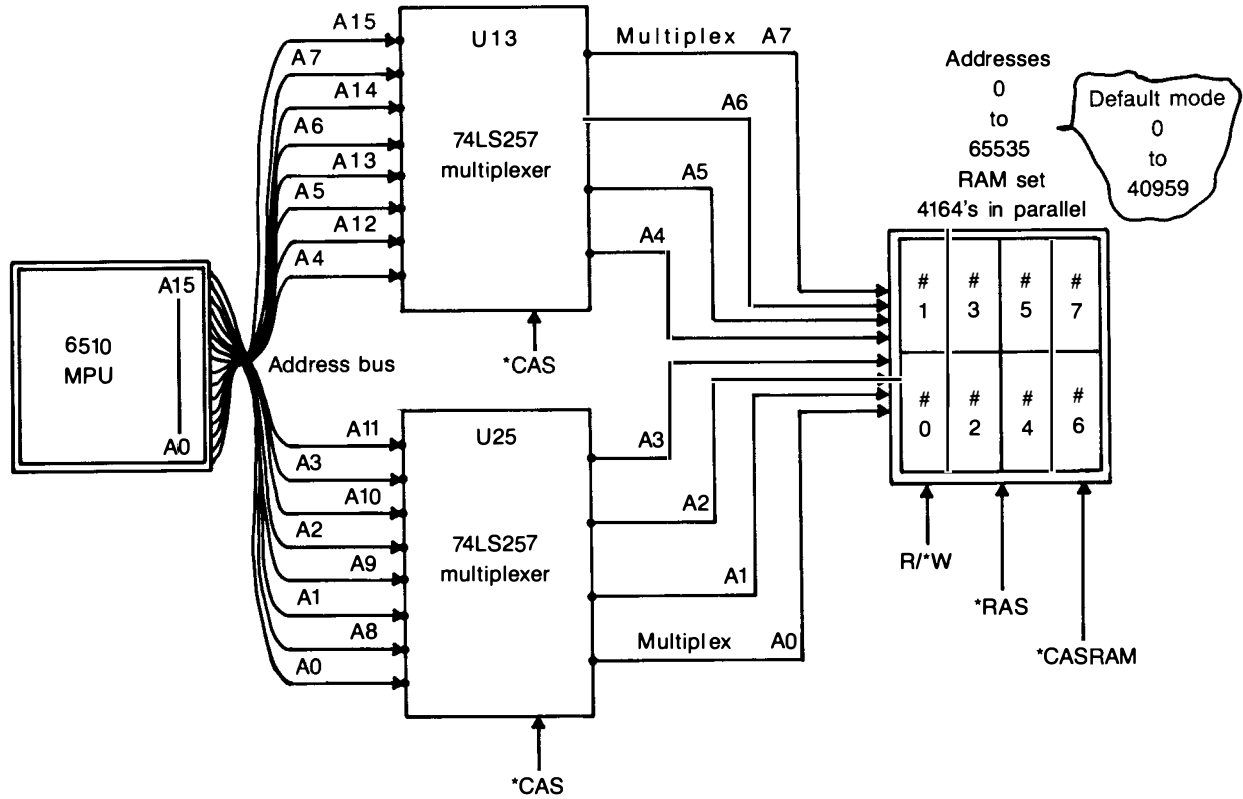


Fig. 9-3. The dynamic RAM array needs two multiplexer chips to help out with the addressing chore.

addressed bit is being written to, the data byte comes over the data bus and each RAM receives its respective bit of the byte. When the addressed bit is being read, each RAM gives up a copy of the high or low it is holding in storage.

The I/O register in the MPU works out of the P7-P0 pins. They operate into other circuits. Their connections to RAM are the assignment of addresses 0 and 1 to help these pins. The contents of these helpmate RAM locations are brought into the MPU in the regular way through the data bus. These P7-P0 pins have no direct connection to RAM. This unusual register in the 6510 will receive further discussion in the MPU chapter.

Only one of the control lines of the MPU goes to RAM. It is R/\*W, or read/write control line. It is normally held high in the read position. When it is forced low, it is in write position. The line leaves pin 38 of the MPU and goes to a lot of the chips. Eight destinations are the RAM array.

The R/\*W line is a one way control. It originates in the MPU and heads out. It does not return. R/\*W connects to \*WE, write enable, pin 3 of all the RAM chips. When it is high, the \*WE will permit the chips to be read. If the R/\*W goes low, the reading ability is cut off and the RAM can be written to.

The arrangement between the MPU and RAM deals with the smooth transfer of highs and lows between the two entities. It involves some other devices. The two multiplex chips are needed to work with the VIC produced \*RAS and \*CAS strobe signals to get the rows and columns addressed smoothly. The MPU-RAM connection is vital in the computer scheme of things.

## MPU and ROM

It is a fact that the Commodore 64 got its name from the size of its memory map. The 65,536 locations are loosely called 64K. The first location is numbered zero and the last 65,535.

When the designers assign location address numbers, they are very careful. Each address must open up its assigned location, but the address must not open up any other location at the same time.

If an address, which is nothing but a collection of 16 highs and lows, does open more than one address at a time, the program could crash.

One of the marvelous features about the 64 is the arrangement of its memory locations. There are a number of ways you can change the addressing around. This provides great flexibility during different applications. This feature is discussed in detail in the Memory Map Chapter. Figure 9-4 illustrates the default addresses, which means the addresses that are available when you first turn on the machine.

The 64 comes on normally with the BASIC language ready to use. In the mode, RAM is assigned the locations from 0 to 40,959. The addressing that we examined in the MPU-RAM discussion used these numbers. The ROMs have larger number addresses.

The BASIC ROM has 8K addresses from 40,960 to 49,951. The Kernal is assigned the addresses from 57,344 to the very top of memory, 65,535. The Character ROM can be addressed at 53,248 to 57,343. The Character ROM shares these addresses with other residents, but there is no conflict, as you'll learn in Chapter 13.

The three ROMs are fairly straightforward, because they are meant to be read and not to be written to. The MPU uses the same one way address lines, A15-A0, to address the ROMs. The BASIC and Kernal take lines A12-A0 and use the 13 bits to address one of the 8K locations inside the ROM. A15-A13 are sent to the PLA. The PLA decodes the bits and selects the ROM addressed with either the \*BASIC signal or \*KERNAL signal.

The Character ROM has its internal location addressed with A11-A0. It only needs 12 address lines because it only has 4K addresses in its matrix. A15-A12 are sent to the PLA. The PLA produces \*CHAROM to select the chip.

Once a ROM is selected and its matrix location addressed, it can do only one job. It outputs a copy of its burnt in eight bits to the eight outputs, D7-D0. The eight bits travel directly to the MPU.

## MPU AND CIA INTERFACE

Each CIA only has 16 internal locations assigned

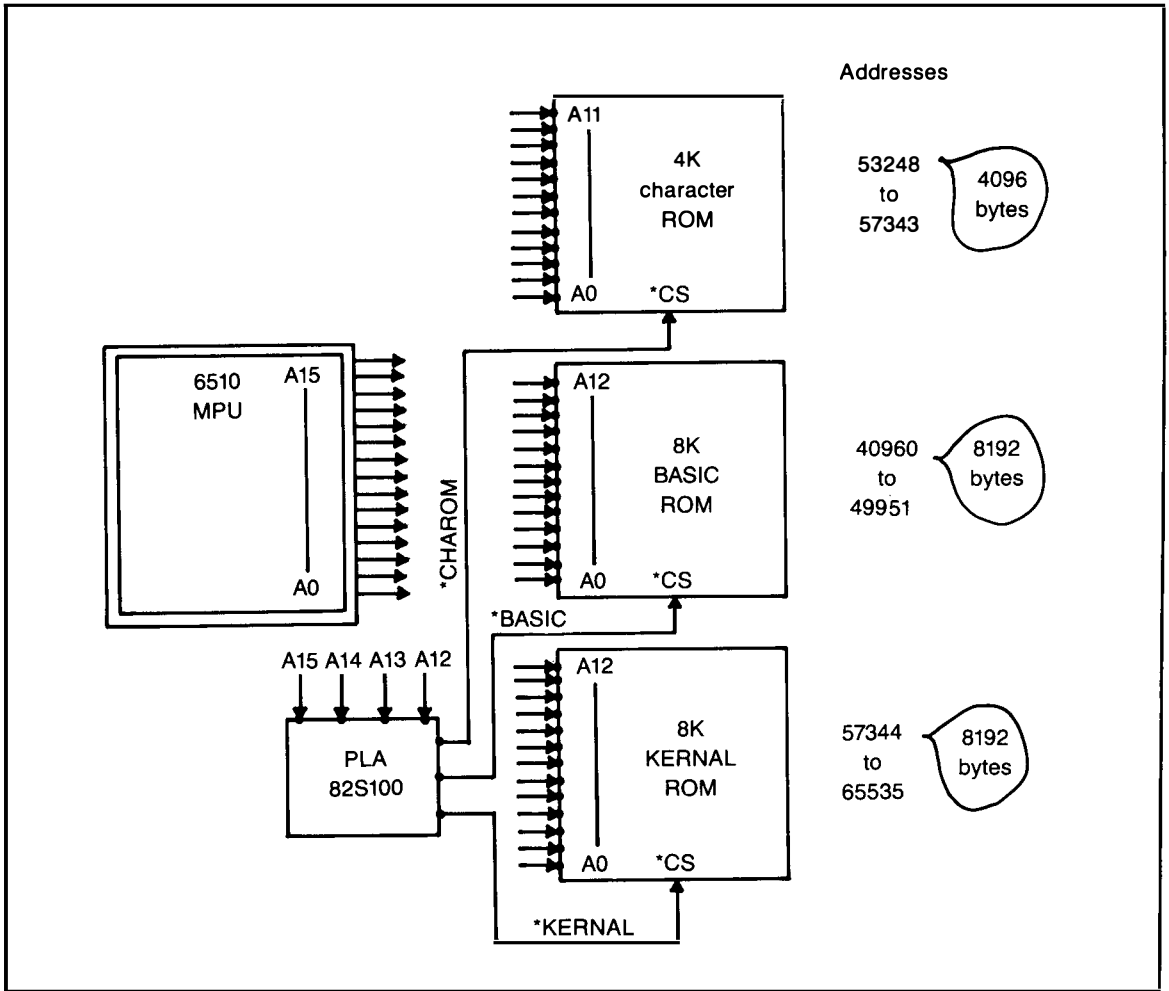


Fig. 9-4. When the 64 comes on normally, the three ROMs respond to these addresses.

on the memory map. They are contacted by address lines A3-A0. Figure 9-5 shows how the address lines connect to the CIAs. CIA means Complex Interface Adapter, and the device is a complex little machine. It is discussed in Chapter 16. The CIA operates with the MPU. The MPU sees it as simply some more addresses in the memory map. However, the CIAs are standing between the MPU and the peripheral devices as Fig. 9-6 shows. Through the CIAs the MPU has instant communication with the peripherals.

The CIAs have eight data lines that connect to the data bus D7-D0. The MPU is at the other end

of the data bus. the CIA can send data or receive data at will, just as RAM can. The data pins of the CIAs connects to the inside of the computer. The CIAs have 16 external data lines that connect to the peripherals. These lines are two sets of eight. They are designated with the lettering PA7-PA0 and PB7-PB0.

There are two sets of registers, almost identical, in the CIA. The PA lines come from one register set, and the PB lines come from the second set. The D7-D0 lines connect to both sets. Each set of registers has two addresses. The data bus services whichever registers are addressed.

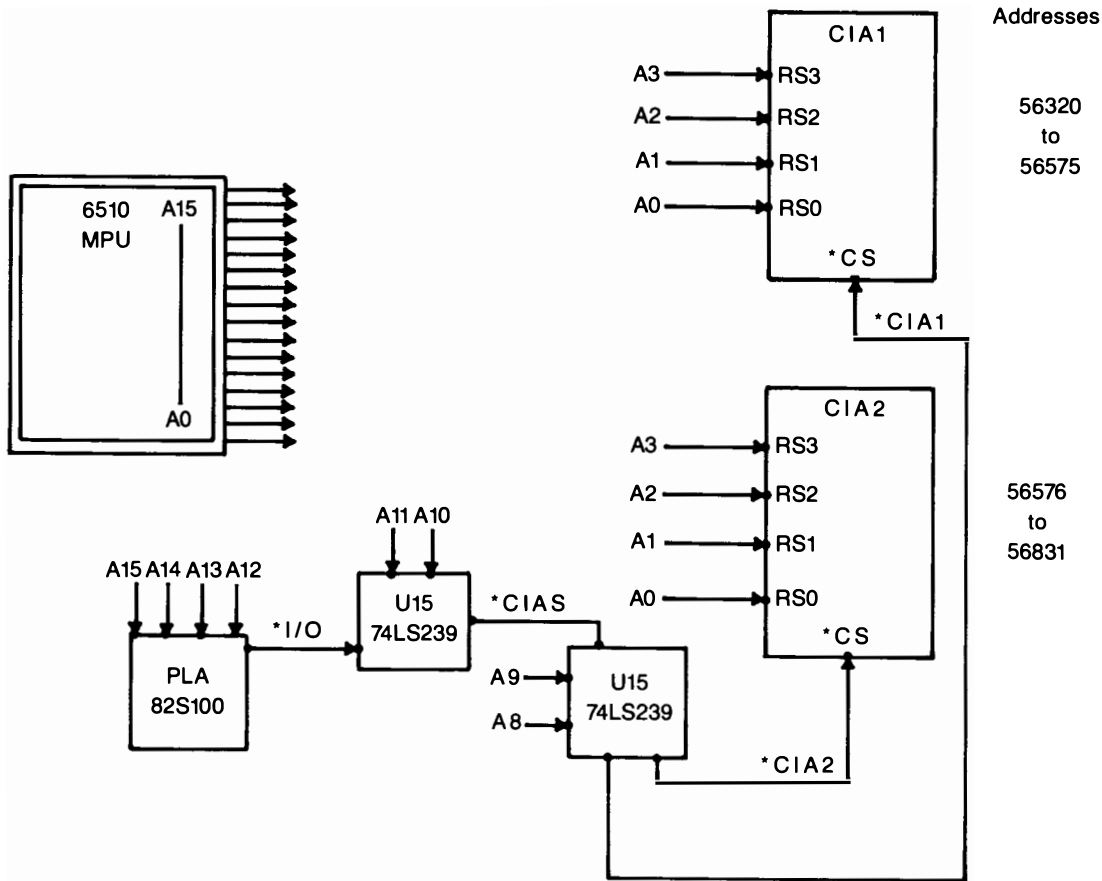


Fig. 9-5. The MPU addresses the two CIAs at these addresses. Each CIA has 256 bytes assigned in the memory map.

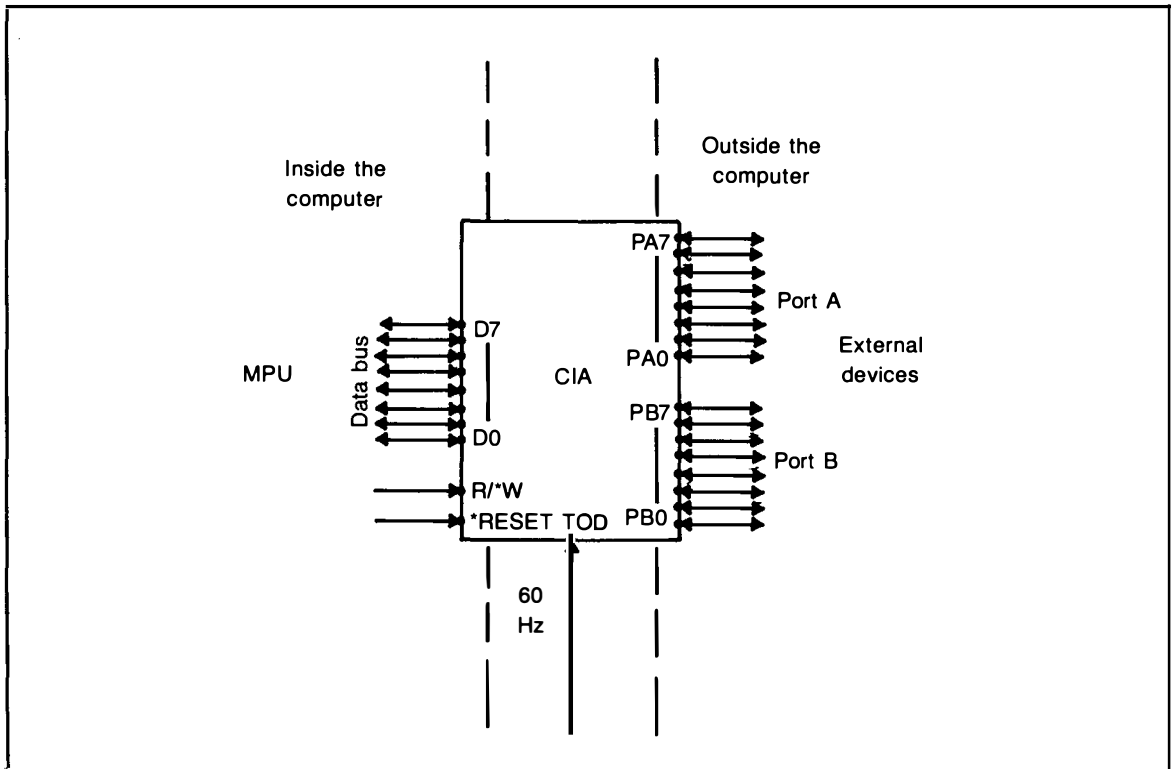


Fig. 9-6. The CIAs have the normal 8-bit connection to the internal data bus but possess 16 data bits for connection to external devices.

One of the CIAs is used to I/O the keyboard input and the control ports. The second CIA I/O's the user port, and the serial I/O port. The two chips are identical, but as you'll see in the test point charts in Chapter 16, the logic probe readings of each are not anywhere near the same. This is due to the different sort of peripherals the devices are acting as ports for.

Inside the CIA, the two ports, A and B, use six registers to do their job. Each register consists of one byte. Each register is contacted by its address. In addition to these six, there are ten more registers performing other jobs. These other duties have to do with timing, serial I/O and interrupts. These ten registers also have their own addresses giving the registers a total of 16 addresses.

The 16 addresses are opened up individually through four address pins. They are R3-R0. They connect to the address bus lines, A3-A0. The four

lines can produce 16 combinations of highs and lows, to unlock each of the 16 registers.

The CIA has one pin called \*CS to select the chip. These selection highs and lows are coming from the 74LS239 decoder chip as mentioned in the last chapter. The CIAs also are controlled by the MPU with an R/\*W connection. They receive data from the MPU when the line is low and send data to the MPU if the line is high.

The chip is initialized by \*RESET when the computer is first turned on. It is kept in step by means of a  $\phi 2$  clock signal from the MPU. The CIAs are connected at pin 19, TOD, to the electric company's accurate 60 Hz frequency. This makes it run like a clock radio through its Time of Day connection. This connection works with four internal CIA registers that constantly keep track of AM/PM hours, minutes, seconds, and 10ths of seconds.

## MPU AND VIC II

The 6567 VIC II in the Commodore 64 is an amazing chip. It does have addresses in the memory map and the MPU contacts it continually through the addresses. In addition, the VIC can act on its own. It can form addresses and make connections with other residents without the help of the MPU. Refer to Fig. 9-7.

The VIC II, with this addressing capability, is used in video game machines besides being a video controller in computers like the 64. It is a versatile chip. It gives the 64 powerful video game capabilities.

The chip has 47 registers that can be addressed by the MPU. It has 14 of its own address lines that gives it the power to contact a 16K group of locations in the memory map. The details of its activity are covered in Chapter 17.

The VIC is a 40-pin DIP, and its pins are an assortment of inputs, outputs, and input/outputs. As you can see in Fig. 9-8, inputs include the chip select, \*CS, the color clock input drive,  $\phi$ IN, the R/\*W line and the address lines to choose one of the 47 registers. Some of the outputs are the signals \*RAS, \*CAS, sync, luminance, color, and AEC, the address enable control. Other lines are the usual D7-D0 data bus connections and the address lines, again going the other way to access a chosen 16K portion of memory.

If you'll notice, the address lines share some of the pins. A0-A7 are on pins 24-31. A8-A13 also occupy pins 24-29. When the MPU is contacting the VIC chip, access is gained to one register on the chip, with a low on \*CS to select the chip, and address bits into pins A5-A0 to select the register. The six address bits can form 64 different binary

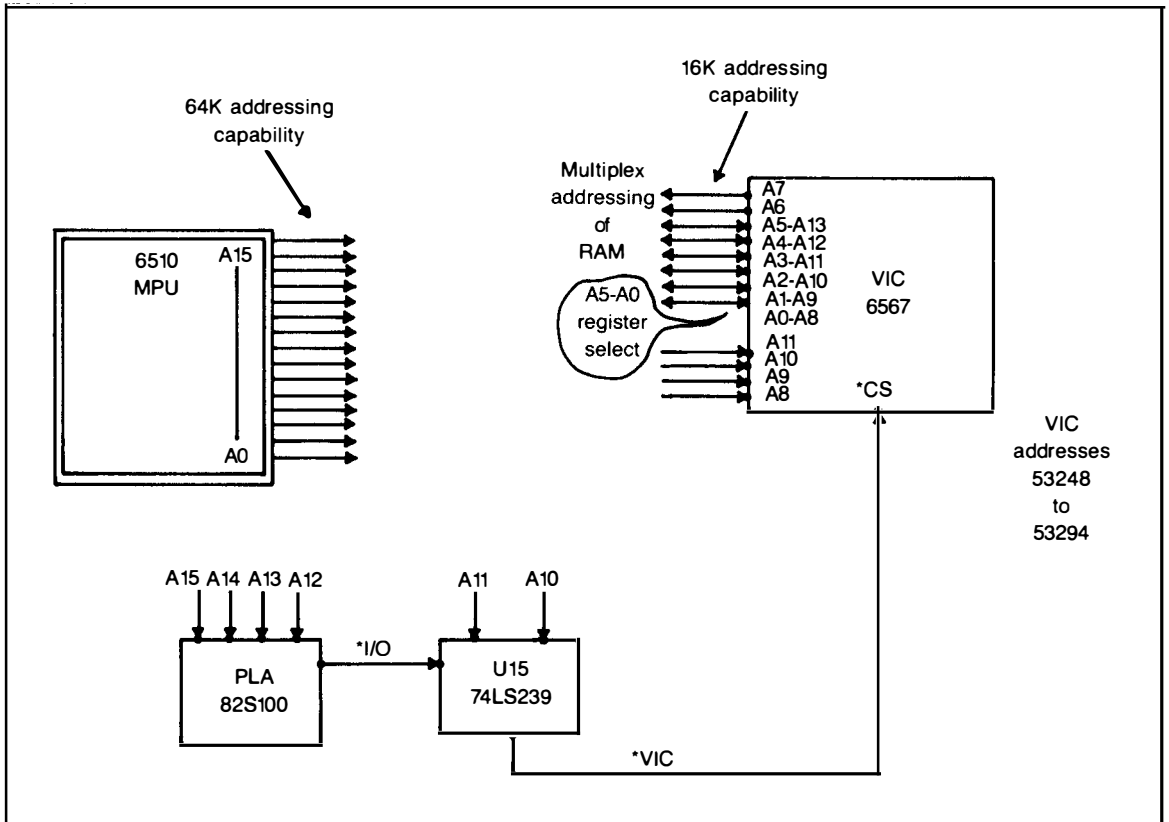


Fig. 9-7. The VIC not only takes up normal residency in the memory map, but it is also able to do some addressing on its own.



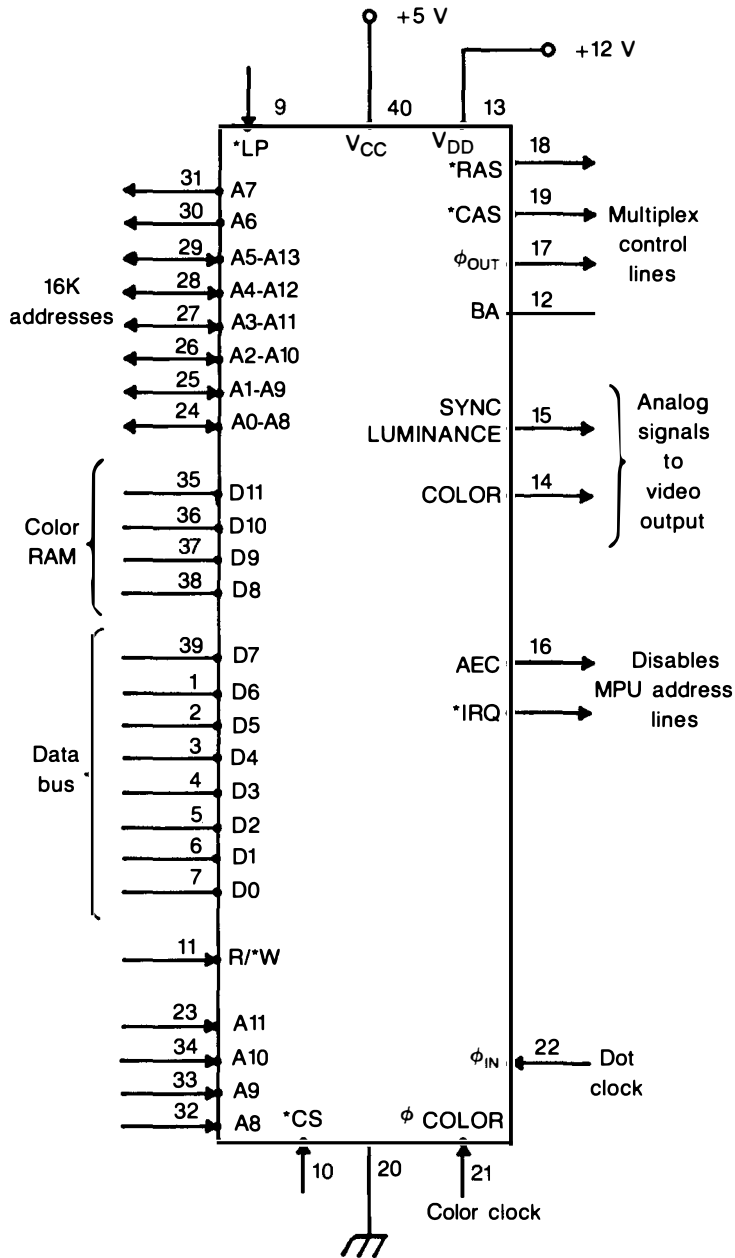


Fig. 9-8. The VIC is a large 40-pin chip that is able to perform on its own as a game controller.

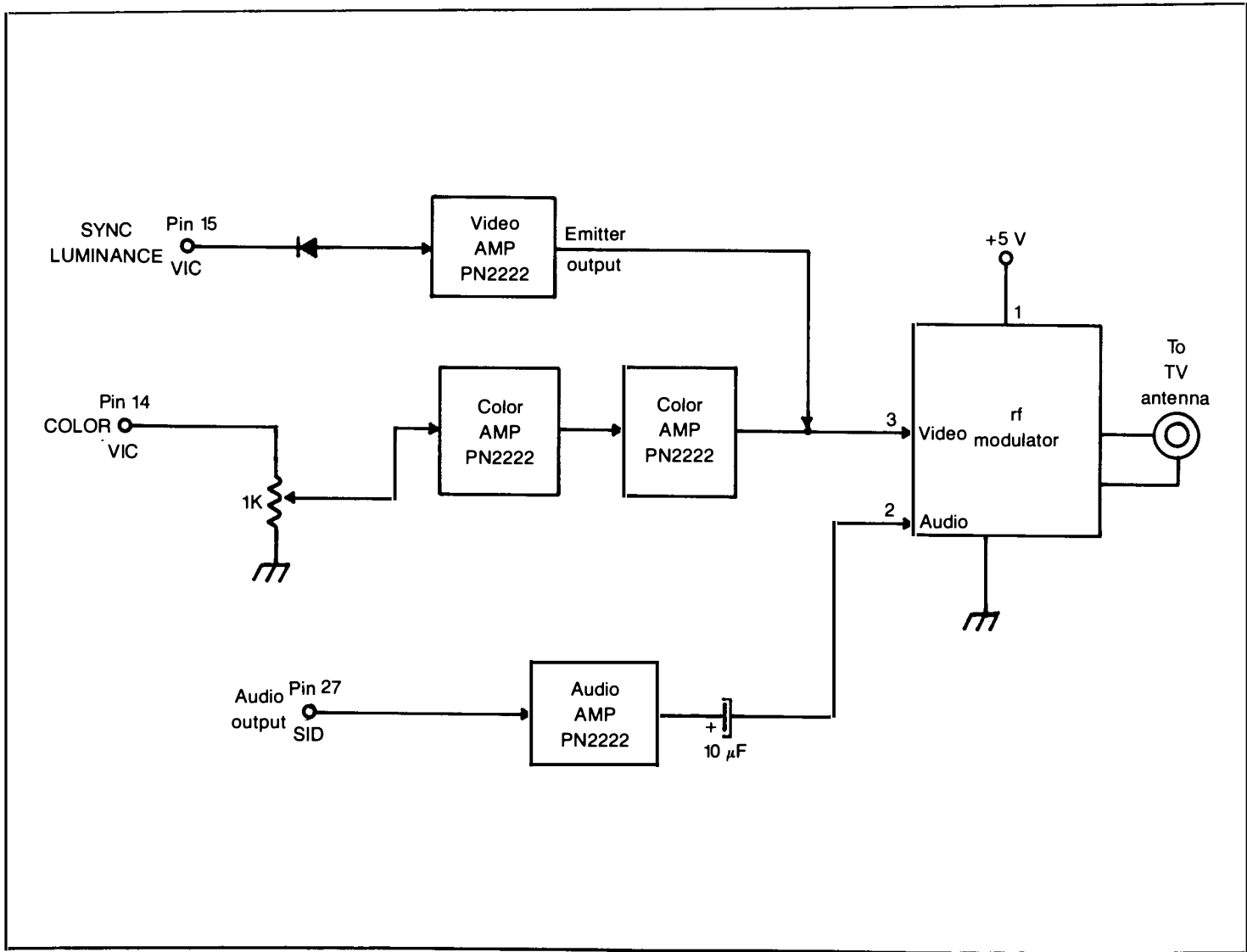


Fig. 9-9. The audio and video outputs are sent to the rf modulator where they are installed into a TV channel 3 or 4 carrier wave.

code addresses, more than enough to select one of the 47 registers. During the input accessing the rest of the pins lie dormant.

When the VIC wants to access memory, the address pins become outputs. The address bits originate inside the VIC and the pins send bits the other way. All 14 address bits are output through A13-A0. The 14 bits can form 16K addresses.

The 14 bits are on eight pins. A13-A8 are on the same pins as A5-A0. In order to output the address without conflict, the bits are multiplexed. During the time \*RAS is brought low and is active, bits A6-A0 output. A13-A7 are held back. Then when \*RAS goes high and \*CAS goes low, A13-A7 are output and A6-A0 are latched. The multiplexing permits the VIC to access 16K of memory somewhat in the same way that the MPU is able to perform.

The VIC is able to read the video RAM 60 times a second this way, among other things. The video RAM holds the codes of the characters that are displayed on the screen. This addressing capability is one of the reasons the VIC is considered such an excellent game device.

While the VIC is conducting its addressing jobs, it could develop a conflict with the MPU. The VIC sends out a couple of signals to avoid the conflict. It outputs the AEC as a low. This disables the MPU address bus lines. VIC also outputs a low from BA, bus available. This connects to the MPU RDY line and holds the MPU in check.

The inputs to the VIC are all digital signals. The outputs we just mentioned are also digital. Besides them there are three analog outputs. These are combined to produce a composite color TV signal. This signal is quite like the one that you receive from a commercial TV station.

Pin 15 outputs the sync and luminance parts of the VIC formed signal. They are fed through a diode into an npn transistor, a PN2222. Refer to Fig. 9-9. Pin 14 outputs the color signal. It is passed through a small potentiometer network into a pair of PN2222 transistors. The output of the color transistors is injected into the emitter circuit of the video amp. The color is mixed with the sync-luminance, and the final signal is output to both the rf modulator and the audio-video plug. This circuit is discussed in detail in Chapter 18.

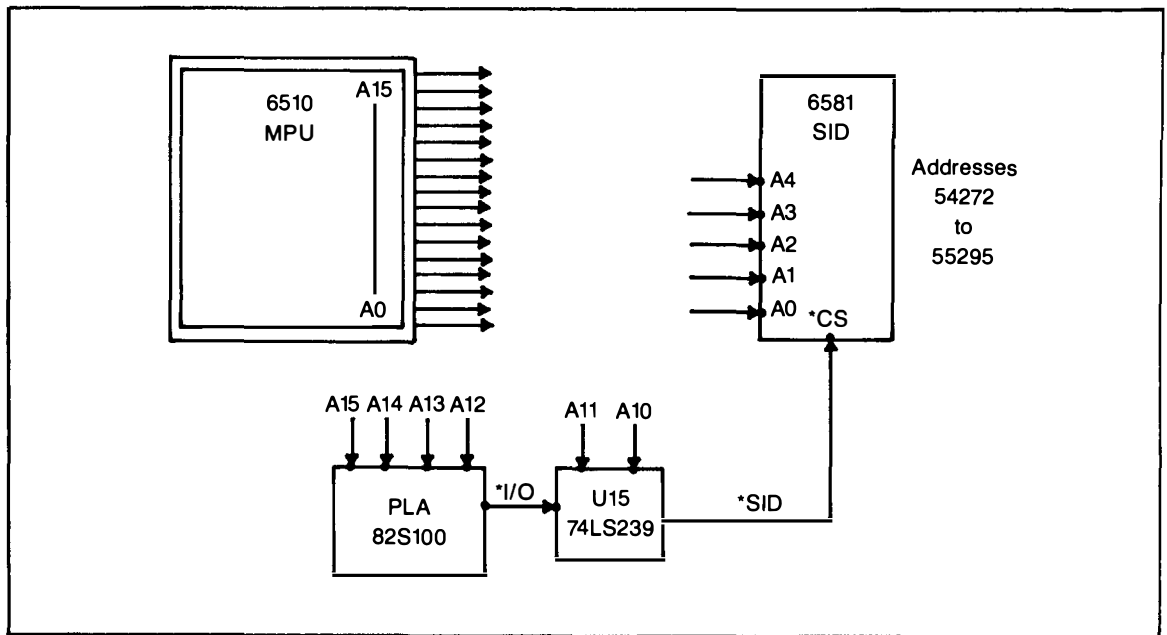


Fig. 9-10. The SID only needs five address lines besides the chip select because it only has 29 register locations.

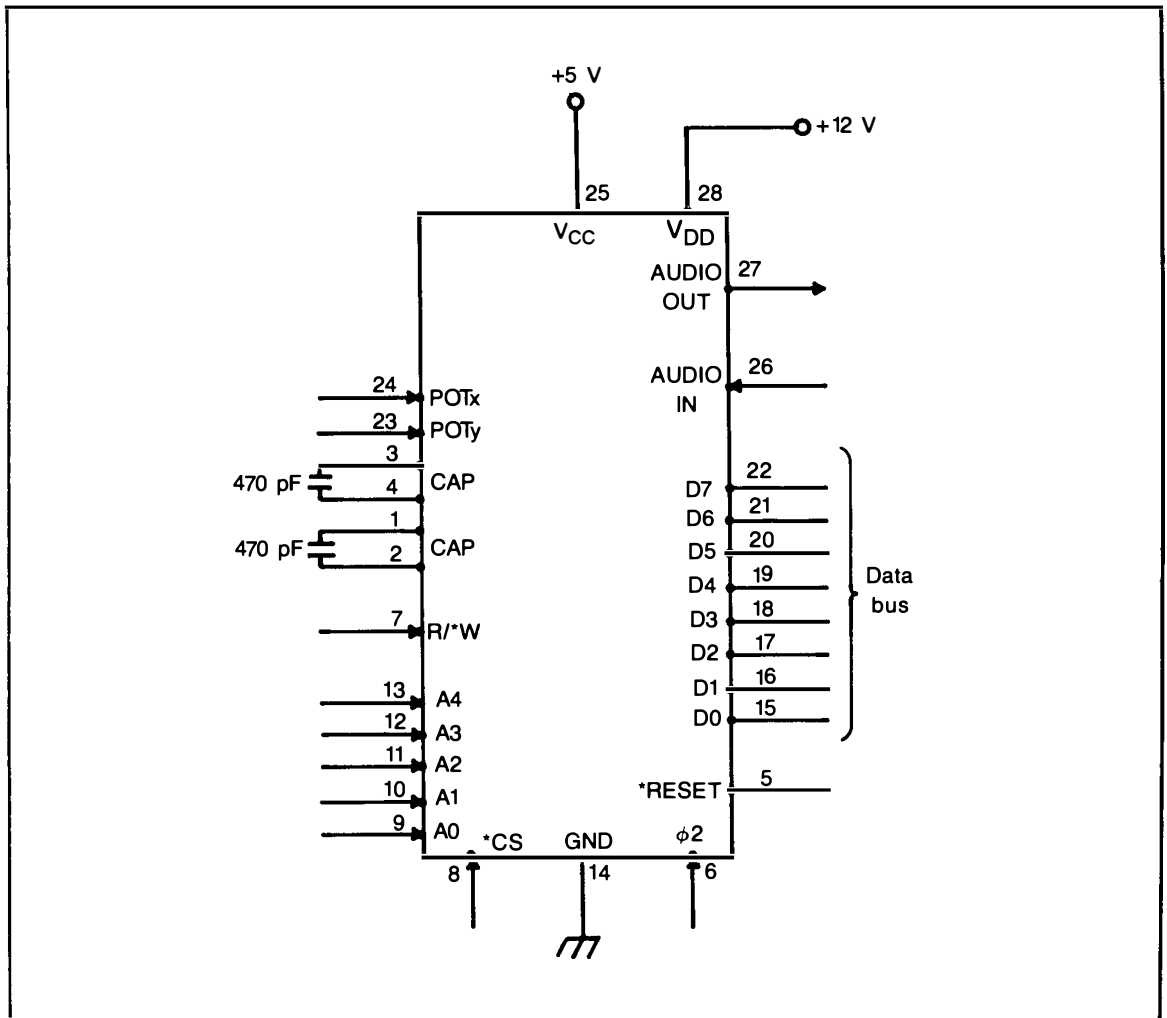


Fig. 9-11. The SID needs eight data pins since all its registers are 8-bits wide.

The circuitry leading up to the VIC is tested and probed with the vom and logic probe. There is no need for scope checks. You are only dealing with highs and lows. Once the signal leaves the environs of the VIC, it becomes an analog TV signal. These signals can be tested accurately with the ordinary TV service scope. The waveforms that should be present are found in Chapter 17.

### MPU AND SID

The Commodore 64 has an exciting music syn-

thesizer and sound effects system to go with the arcade type video it is capable of producing. The sound system is based around the 6581 SID chip. The Sound Interface Device is a 28-pin DIP. You can tell that it is part of the 6510 family by its generic number, 6581.

The SID can be addressed by the MPU through address bus lines A4-A0. Refer to Fig. 9-10. Five address lines are able to contact 32 locations. There are 29 registers in the SID. The address lines permit a read or write to these locations. Should a read or write be executed to one of the three remaining

addresses that do not exist, a read returns garbage and a write is simply ignored. The SID, from the MPU's viewpoint, is considered to contain only 29 memory addresses.

In Fig. 9-11 there are the usual eight data pins, D7-D0 at 22-15. The MPU supplies SID with data during a write operation. SID sends data to the MPU during a read. The R/\*W line from the MPU decides the direction data will flow. When the line is high, the SID sends data to the MPU. When R/\*W goes low, the SID is the recipient of the data. The details of the SID operation are covered in Chapter 19.

POTX and POTY at pins 24 and 23 are inputs from the control ports. The peripherals that are plugged into the control ports supply the input. The inputs are sent to a 4066 quad switch where it is routed to the SID Pins. The signals set the positions of potentiometers in the chip. The AUDIO IN and the CAPs mix and filter the audio. This is covered in the SID chapter.

SID, after all the waveform generating and modulating has one audio output at pin 27. The output can have a peak-to-peak maximum of two volts. There is a dc level of six volts and this can be coupled to any audio amplifier. In the 64, there is PN2222 transistor and a 10  $\mu$ F capacitor that does the coupling. These can be seen in Fig. 9-9. The audio output can then be taken off at the audio-video plug. Most of the time, though, it is sent to the rf modulator. There it joins the video output, and together, they go to the TV set that the computer uses as its display.

SID is a complex chip that contains both MOS and TTL components. At pin 28, VDD, the chip gets + 12 volts to power the MOS parts. At pin 25, VCC, the chip receives + 5 volts to energize the bipolar transistors.

SID is one of the chips with a \*RESET line. It also has a  $\phi$ 2 clock input to keep it in time with the pacing of the MPU.



## 10. Servicing Logic Gates

**I**N ORDER TO INTELLIGENTLY SERVICE THE various chips in a computer, you should have an understanding of how memory locations are numbered, what the highs and lows that pass through gates mean, and how they are stored in registers. The requirements of understanding are not that stringent. If you have ever tried to work with any type of code, even a Dick Tracy decoder pin, you can probably become familiar with location addresses and register contents quickly.

The Commodore 64 can be handled with two numbering systems. One is decimal, the same system you have been using all your life. The second system is binary, which has only two numbers, 1 and 0. The one is represented by a high voltage and the 0 with a low. There is another system that would become necessary if you did a lot of computer servicing. It is called hex, which is short for hexadecimal. For the most part though, you can get by nicely in routine servicing of the Commodore 64 with only decimal and binary. Binary is mostly used in servicing as highs and lows rather than 1s

and 0s. The 1s and 0s are more valuable during programming, but for repair jobs, highs and lows are the order of the day.

### DECIMAL AND BINARY

Decimal numbers are used in the Commodore 64's BASIC to list addresses. In the 6510, there is a 16-bit register called the program counter. It does most of the 6510's addressing chores. You have seen where a row of 16 highs and lows can form 64K combinations. Each combination of highs and lows in the program counter can form one of the 64K addresses on the machine's memory map. The program counter bits are connected to A15-A0 of the address bus.

The first combination of bits in the Program Counter is LLLL LLLL LLLL LLLL. The row of lows in Fig. 10-1 is called address number 0 in decimal. (There are no spaces in the actual register. I put them there for easier reading of the 16 bits.) The program counter automatically places these bits onto the address bus. As soon as these bits

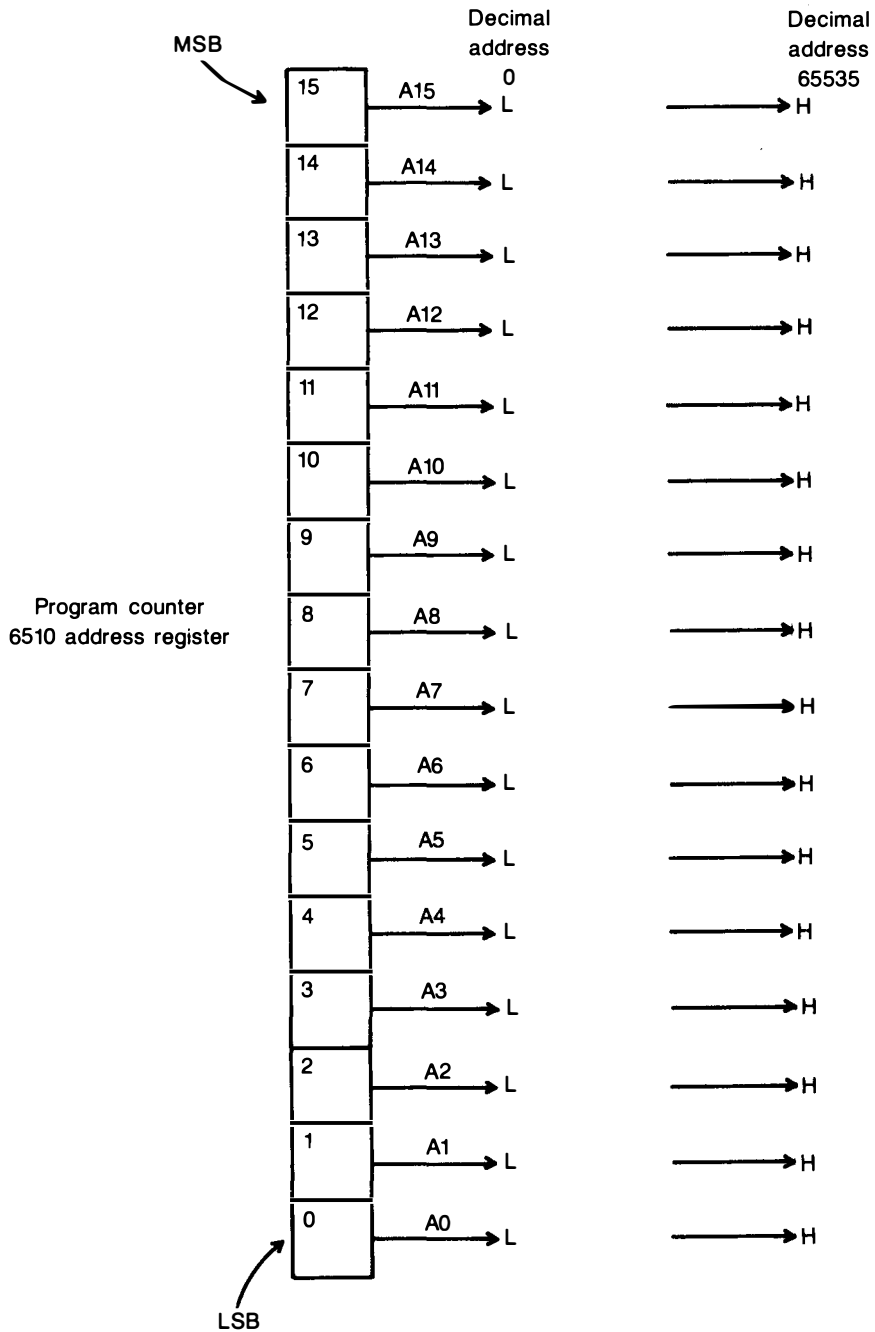


Fig. 10-1. The first possible set of bits in the PC codes decimal address 0. The last set of bits codes decimal 65535.

leave the MPU, the program counter is incremented by one automatically. The PC is built to do this.

The next address is LLLL LLLL LLLL LLLH. This group of bits will address location 1 in decimal. The next address is LLLL LLLL LLLL LLHL. This is address number 2. The program counter continues its incrementing. If it is not stopped, it will mindlessly keep counting up to HHHH HHHH HHHH HHHH. This is location 65,535. It is the 65, 536th physical location, since the first location has an address of 0.

Let's add up the Hs and Ls and see how they relate to the decimal numbers, or in other words, how can they be coded back and forth. It really is easy to crack the code. I'll do it with 16 bits, and then you can handle any size register. Other registers you might want to code either way could be the 8-bit ROM locations or the 4-bit Color RAM.

The bits in the address bus are names A15 through A0. A15 is called the MSB for the Most Significant Bit. A0 is called the LSB for the Least Significant Bit. The bits in between are either higher or lower according to what bits they are referenced from. The LSB A0 when related to decimal has a value of 0 or 1 according to whether it is storing a low or a high. This, of course, seems obvious. Not quite as obvious is the next higher bit A1. It has a decimal value of either 0 or 2. A low is 0 and a high is 2. The next higher bit A2 has a value of 0 or 4. The values continue to double on a high till bit A15 has a value of 0 or 32,768.

In order to convert a register to its decimal code, all you have to do is add up the values of all the bits. Any bit holder containing a low is counted as a 0. All bit holders storing a high is counted by its significant value. The place values are given in Table 10-1.

With the aid of Table 10-1, you can convert highs and lows to addresses and vice versa. For instance, suppose you are probing the address bus and you find a fixed set of bits on the bus. The bus is stuck on that number and will not respond to any attempt to change it. You want to know what address the bus is pointing to. The bits are HLLH HHHH LLLL LHLL. The MSB is the first H and

**Table 10-1. Binary to Decimal Conversion.**

Significance		Low	High
LSB	A0	0	1
	A1	0	2
	A2	0	4
	A3	0	8
	A4	0	16
	A5	0	32
	A6	0	64
	A7	0	128
	A8	0	256
	A9	0	512
	A10	0	1024
	A11	0	2048
	A12	0	4096
	A13	0	8192
A14	0	16384	
MSB	A15	0	32768

the LSB is the last L. The address can be converted to decimal by adding the values of the bits together. The conversion would look like this:

### Binary to Decimal

MSB	H	32768
	L	0
	L	0
	H	4096
	H	2048
	H	1024
	H	512
	H	256
	L	0
	L	0
	L	0
	L	0
	L	0
	H	4
	L	0
LSB	L	0
Address		<hr/> 40708



The address on the stuck bus is decimal 40708. On the Commodore 64 memory map, 40708 is an address that is used by a cartridge ROM. This bit of information is a clue to indicate where and what is causing the stuck bus.

Sometimes it is useful to code a decimal number to its binary equivalent. There could be a case where the 64 will respond to certain addresses but not to others. One of the address lines could be shorted to ground or to another line. Suppose one of the addresses that will not work is 1024. If you convert decimal 1024 into binary, you might get some idea of what line is inoperative.

To convert the decimal, you look for a combination of bits that will add up to 1024. That one is easy. A10 is exactly 1024. It's address is LLLL LLHL LLLL LLLL. It looks like address bus line A10 is out of commission. It can be tested for shorts or an open.

That was an easy conversion. Suppose you had to code 53281 to binary. That would take a bit of number juggling. The first thing to do is locate the closest bit value that is less than the desired number. Then keep adding values till you arrive at the correct binary code for the decimal. 53281 is converted in the following manner:

	0	L
	0	L
	0	L
LSB	1	H
<hr style="width: 10%; margin: 5px auto;"/>		
Total	53281	

The conversion back and forth is not difficult. Coding binary into decimal is simply a matter of adding the significant decimal values of each bit together. Getting the binary bit layout from a decimal number requires breaking down the number into a group of decimal numbers that consist of decimal significant values. Then the corresponding bits are arranged from MSB to LSB. Number conversion is a must during some repair work with the 64.

## HEXADECIMAL

Most of the time during troubleshooting and repair work on the 64, decimal and binary coding will suffice. The BASIC ROM comes on using decimal. The logic probe and vom reveal binary highs and lows on the various test points. If you can intelligently relate the two representations of voltages, you can do practically all the jobs where numbering comes into play.

There are some occasions though, where it would be handy to use hexadecimal too. Hex is just a third way to express the decimal or binary values. It is used extensively in machine language programming. Its use during repair jobs is to code binary in another way.

Hexadecimal means six plus ten. That is, it is a numbering system where you count ten numbers from 0 to 9 and then count on up to 15 with A, B, C, D, E and F. After F comes 10. Hex lends itself to computers because a nybble of binary numbers counts from 0 (LLLL) to 15 (HHHH). The hex numbers line up with binary exactly. Table 10-2 shows the binary equivalents for hex numbers I through F.

Hex is a convenient way for programmers to

### Decimal to Binary

MSB	32768	H
	16384	H
	0	L
	4096	H
	0	L
	0	L
	0	L
	0	L
	0	L
	0	L
	0	L
	32	H

**Table 10-2. Binary Equivalents of Hex Numbers.**

Hex	Binary
0	LLLL
1	LLLH
2	LLHL
3	LLHH
4	LHLL
5	LHLH
6	LHHL
7	LHHH
8	HLLL
9	HLLH
A	HLHL
B	HLHH
C	HHLL
D	HHLH
E	HHHL
F	HHHH

use binary. Table 10-2 shows that one hex number can represent four binary voltage states. That is why I put a space between every four voltage levels. Each space is between one hex number. The four voltage levels are one hex number. It is much easier for a programmer to code one hex number than four voltage states, four 1s and 0s, four trues and falses, four sets and resets, or whatever way some engineer or programmer decides to call the logic states.

To convert any binary number to hex, put a space between every group of four bits. Then find the hex equivalent for each group. The total bit size of the binary number is not important. As Fig. 10-2 shows, there are four bit holders in each location in the Color RAM. You can express the bits held in each location with one hex number. There are

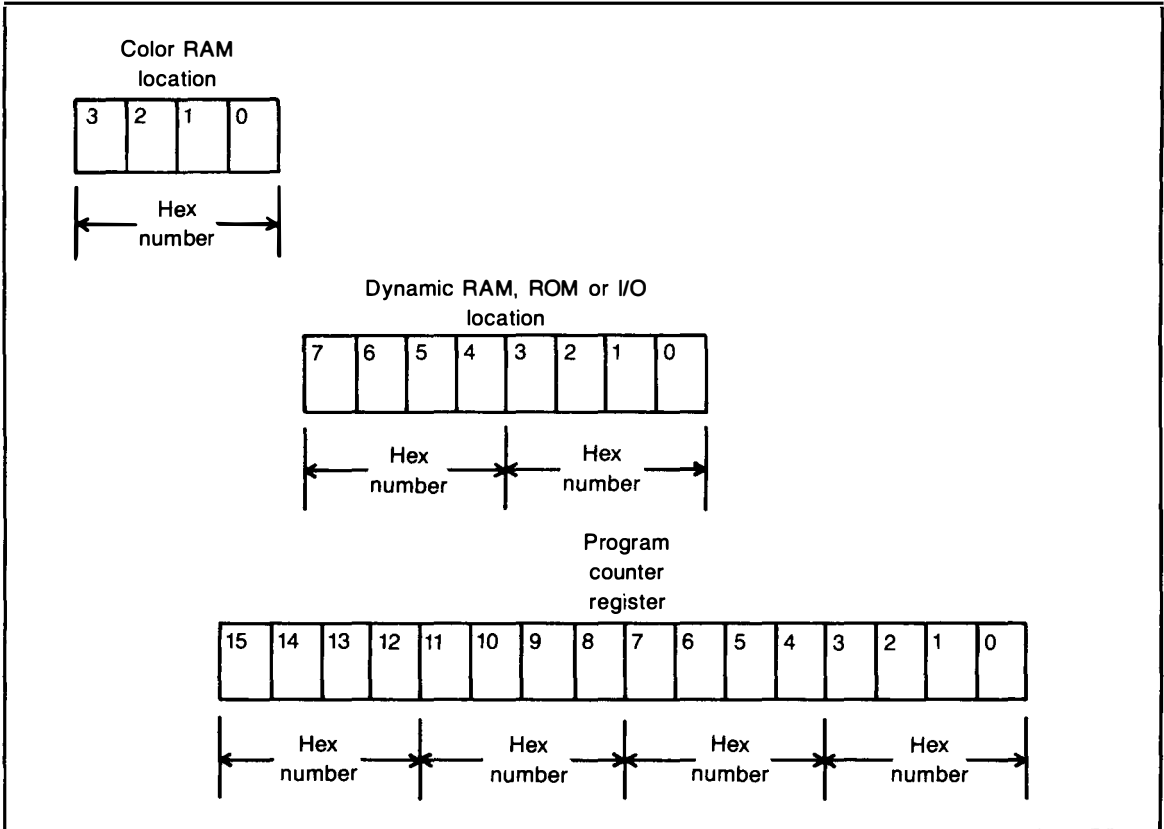


Fig. 10-2. A hex number is used to code four binary numbers. Two hex numbers can code a byte and four hex numbers the 16-bit PC.

eight bits in each ROM location. These location contents can be written in two hex numbers. The addresses in the 64 are 16 binary bits each. All the addresses in the 64 can be described with four hex numbers. Refer to Fig. 10-2.

## PEEK AND POKE

There are two routines in the BASIC ROM that you'll find are very useful during repairs. The routines can be used normally when the 64 has trouble but it is signing on with the READY and the blinking cursor. If the trouble is preventing the sign on message from being displayed, then the following service measures can't be used.

Assuming the 64 does sign on and there is trouble, you can signal trace the residents of the memory map with the BASIC commands PEEK and POKE. PEEK lets you read the contents of a location. POKE gives you the power to stick a byte into any location that normally accepts bytes. These two capabilities are excellent troubleshooting methods.

PEEK is a function. It is not a command. It needs a command to work. The usual command is PRINT. That way you can see on the screen what was in the location you peeked into.

To get a fast look at location 1024, you type PRINT PEEK(1024) and hit the Return key. Whatever bits are in 1024 will appear on the TV screen. However, the bits will be coded in decimal.

Location 1024 is the first place on the TV screen at the upper left hand corner. If the character P is being displayed, the code 16 will respond to the PEEK. This is the code the VIC picks up every 1/60th of a second to update the screen. Code 16 tells the VIC to display a P.

POKE is a command. If you want to POKE the number 7 into location 53280, you type POKE 53280, 7 into the machine. The BASIC routine runs the number 7 through the 6510 and onto the data bus. Meanwhile the MPU outputs the binary states for decimal 53280 and opens up that location. The binary state for decimal 7, LHHH, is installed into the four bit holders of the location. The POKE is complete.

Location 53280 just happens to be the place where the border color around the TV screen is stored. The LHHH forces the border to go yellow. This is a test. If the border follows instructions and goes yellow, then all the components that carried the command, from the keyboard to the TV screen are ok. If the border does not go yellow, this is a symptom and could lead to pinpointing the source of a trouble.

The PEEKs and POKEs are powerful servicing tools. They can signal trace a great deal of the circuitry in the 64. Of course you must understand what should be happening in the circuits when you try to PRINT a PEEK or force a POKE into a location. There will be more about PEEKs and POKEs throughout the rest of the book.

## GATES

Most of the circuits in the 64 are comprised of gates and registers. Registers are covered in detail in the next chapter. The logic states that travel through the digital parts of the computer are manipulated by the gates. Most of the manipulation consists of either maintaining the voltage in a bit or changing the state of the voltage. The states flash around the circuits in times that are measured in nanoseconds (billionths of a second) and are changed from +5 volts to 0 volts and back. Refer to Fig. 10-3.

The registers are storage areas for the bits. You can place a high or a low into a register bit and the register will hold the bit for as long as the correct power is applied. The registers are found in various sizes. There are *nybble registers* that hold four bits, *byte size registers* holding eight bits and *word registers* that are 16 bits across. The registers hold the bits through three general methods. In ROM chips, the bits are burnt in permanently. The latches and static RAM chips use flip-flops. Dynamic RAM stores voltage states in the capacitance formed in the insulated gate and ground of the MOS chip's insulated gates in the FETs.

With careful logic state changing in the gates and the reliable storage of the registers, computing takes place. When trouble strikes, it is often due to failure in a gate or register. The electronic cir-

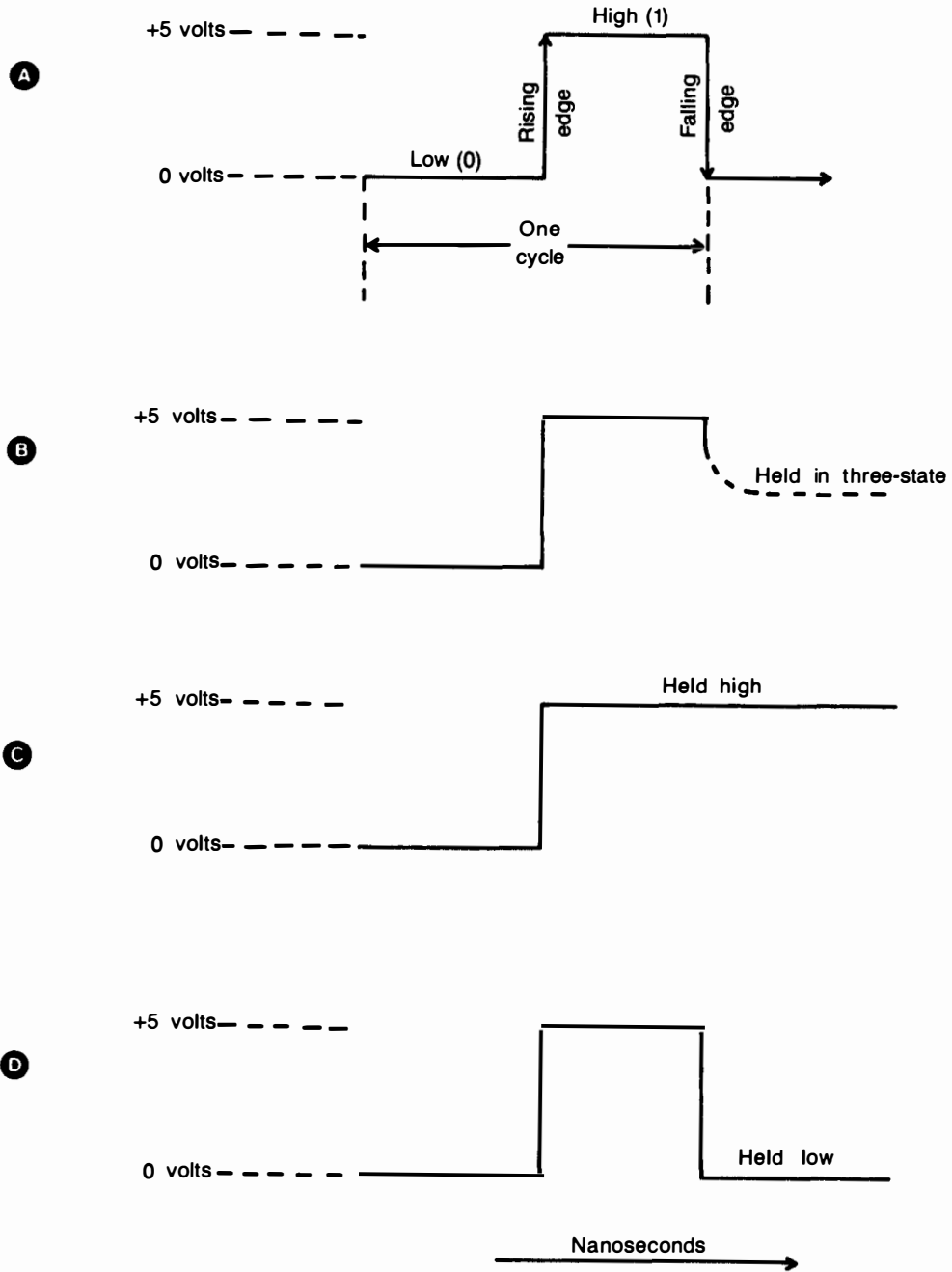


Fig. 10-3. The high and low states are graphed with voltage in the vertical plane and time in the horizontal. (A) changing state, (B) three-state, (C) low steady state (D) high steady state.

cuts on a chip can and do short, open, or spring a leak under voltage or temperature pressure. Figure 10-4 shows some of the breakdowns that can befall a chip.

To troubleshoot these types of failures, it is essential that you understand what is happening to the voltage states as they make their way through the gates and the registers. That way your vom and logic probe testing will make sense and point you to a trouble quickly.

There are a lot of gates in the Commodore 64. Most prominent are the 14 pin DIPs the 7406 and the 74LS08. The 7406 has six inverters and the 74LS08 contains four AND gates.

Not as well seen are the gates in the 74LS257 and 74LS258 chips. Each of these 16-pin DIPs have three inverters, eight AND gates and four OR gates. While the 74LS09 has its gates separated with a pin assigned to every input and output, the multiplex chips have the gates internally wired to-

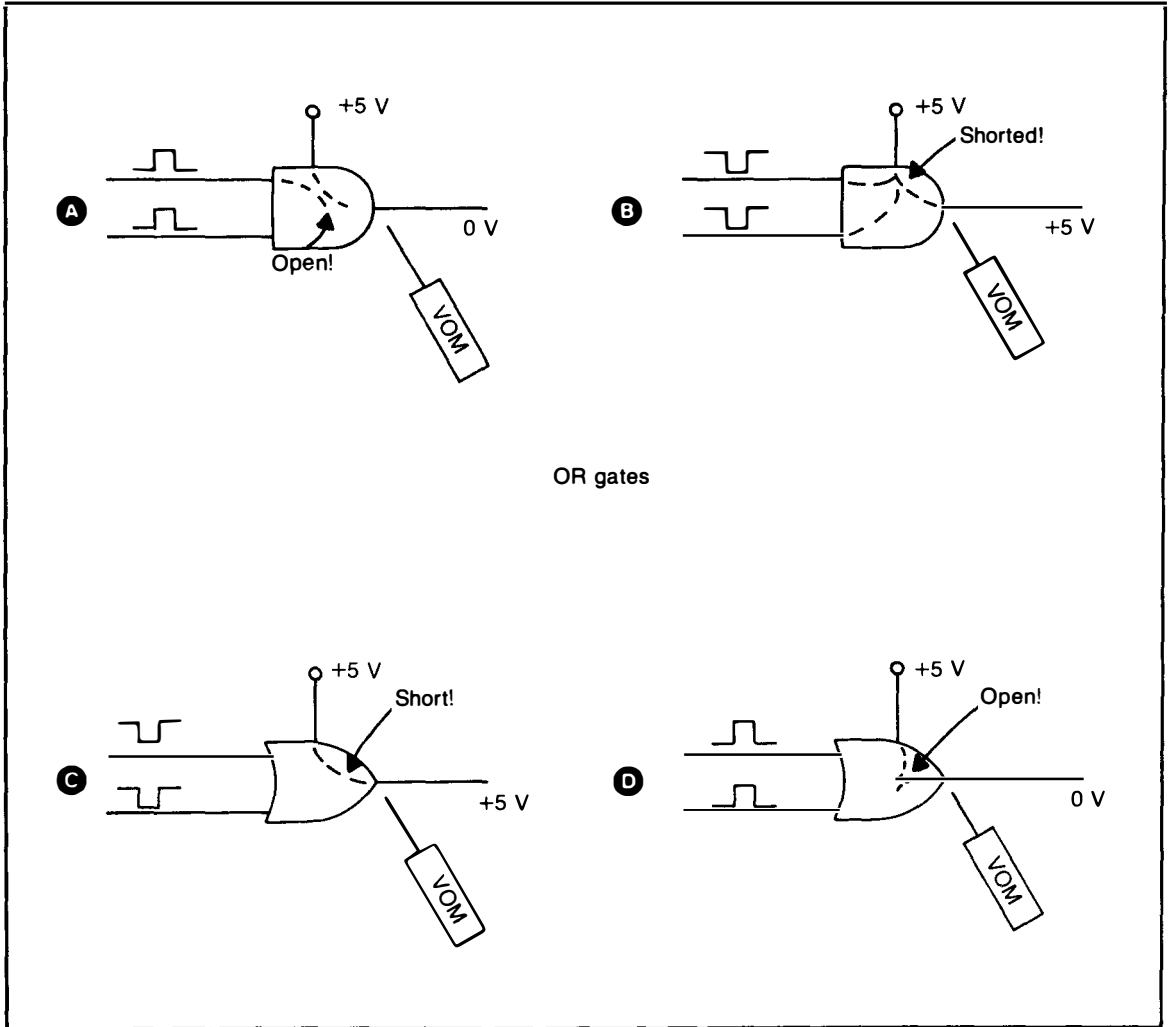


Fig. 10-4. If an AND gate should open, two highs could produce a low (A). If the same AND should short two lows could read output as a high (B). If an OR gate should short two lows could read output high (C). The same OR gate when open could output a low even though two highs are input (D).

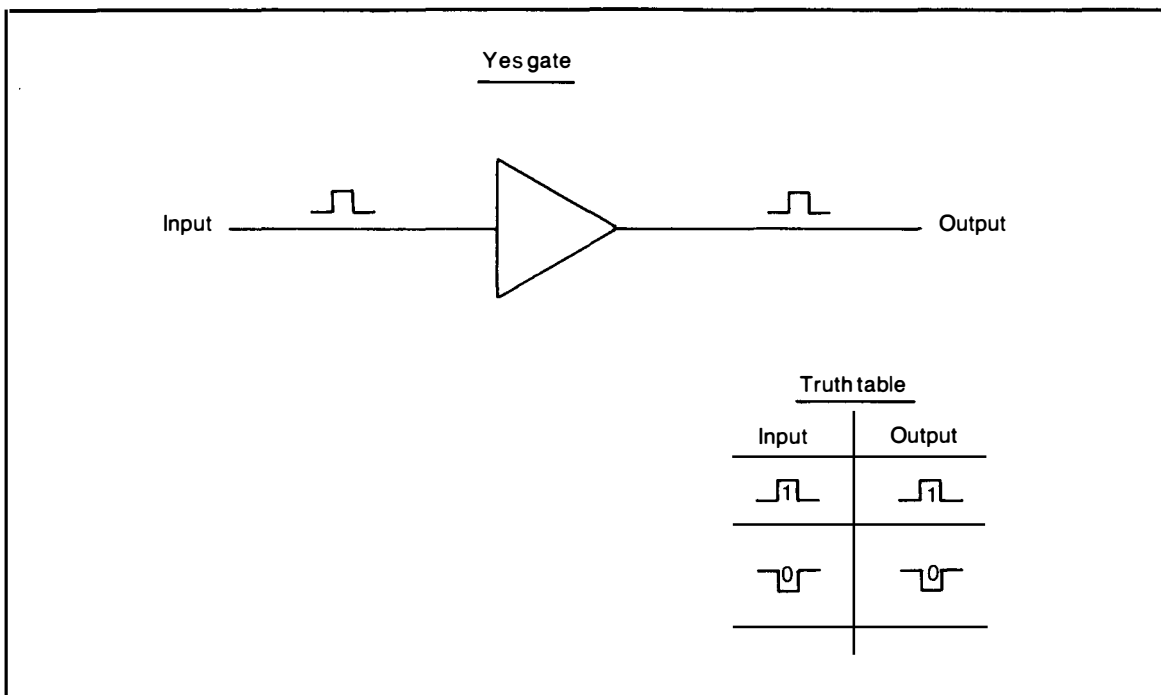


Fig. 10-5. The YES gate is so named because the output maintains the same state as the input.

gether. The AND outputs are wired to the OR inputs. You can't put a probe on the AND-OR connections.

The 74LS239 decoders also contain internal gates. The Character ROM has an AND gate in its chip select circuit. All of the large chips have various gates. The peripheral devices contain gates. Any and all of the digital circuits have dependence on gates. Gates, like any other electronic circuit, fail occasionally. You must know how to handle gates if you want to be able to repair your Commodore 64.

The gates of concern for the 64 are the YES, the NOT, AND, OR, and exclusive-OR, called XOR. The other gates you've heard of, such as NAND, NOR and exclusive-NOR are not being used in the 64 in an obvious way. During the repair of the 64 you will not need any troubleshooting information on these gates. Any NAND, NOR or XNOR function that might occur will be taken care of with a combination of NOT, AND, OR, and XOR.

## YES Gate

As the name implies, a gate is a barrier in a pathway that can open to let signal pass or close and block passage. The YES gate performs that task exactly. It has two connections, one input and one output. If a high or low arrives at the input and is permitted to pass, the same type of signal will leave at the output. That is, if a high arrives, a high will leave. Should a low arrive, then a low will leave. See Fig. 10-5. There is no change made to the logic state. You might ask, if there is no change in state, why bother?

The YES gate is an amplifier stage. It is needed to amplify current levels and to match impedances from one circuit to another. It gets its YES name because the signal does not make any logic state changes as it passes from one stage to another. YES gates are found in the large chips of the 64. They are the buffers that are mentioned on occasion.

Inside a YES gate is a transistorized amplifier. When the chip is using bipolar components the

amplifier could be based around an npn. The circuit is simple, as Fig. 10-6 shows. The npn is powered normally through the +5 volt supply. The high or low is input at the emitter. The output is at the collector. When a signal is input at an npn emitter, there is no phase reversal. The signal is amplified and the same logic state that was input is output. The only changes are the power output is increased and the impedance of the output is designed to match the next circuit.

The actual internal wiring of a YES gate is more extensive than this example circuit, but that is the way the gates work. On a normal schematic, since the internal wiring is not accessible, the entire YES gate is drawn as a triangle. The input lead is on a flat side and the output is at the pointed end opposite to the input.

All gates have a truth table. Truth tables got their name from the engineers that were using true and false to describe the high and the low states. If they had been using high and low, the tables

might have been called the high table. Technicians think of high (H) and low (L) when they see a truth table. The truth table might contain any of the logical state descriptions shown in Fig. 10-7.

The table is a handy test medium. The columns are labeled Input and Output. If the input is an H in a YES truth table, the output is an H. When the input is an L, the output is an L. Therefore, if you are testing a normal YES gate with the vom or logic probe, your test reading on the input should also be found on the output. The truth table is actually a voltage-state test table.

The three-state control opens or closes the YES gate. The three-state controls are used extensively in the Commodore 64. They are used to control YES gate buffers and other chip circuits. Figure 10-8 reviews the three logic states, high, low, and no state. The state of no state is vital to the operation of the computer.

For example, the address lines connect to every location on the memory map. The address that

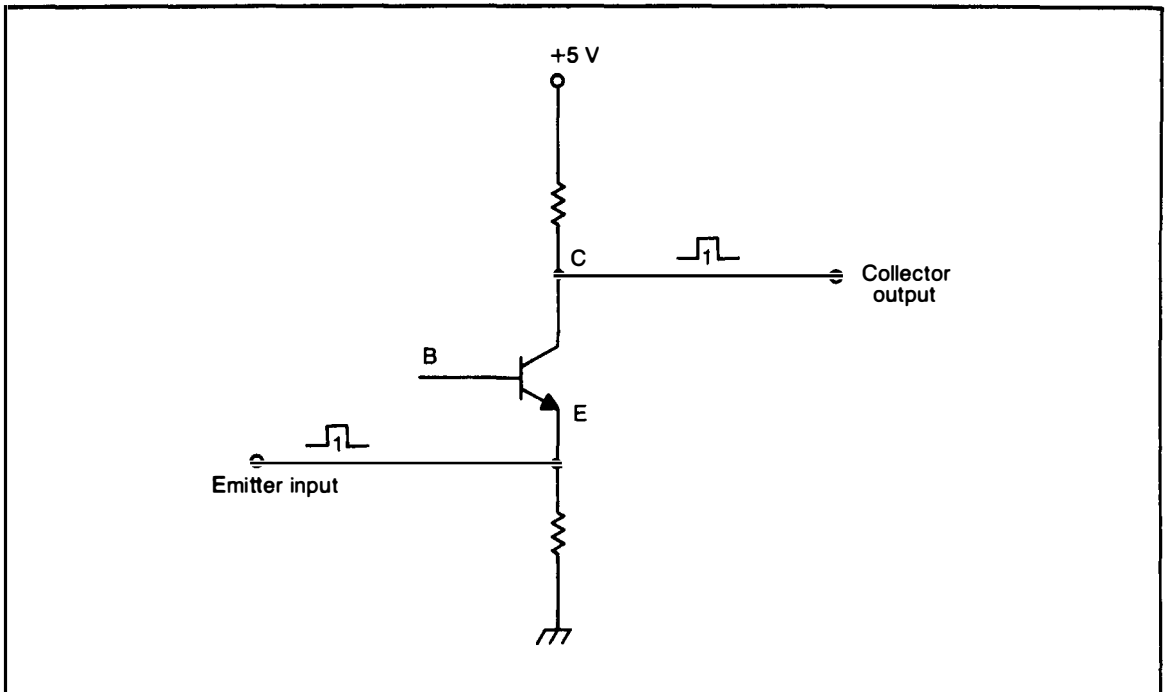


Fig. 10-6. An npn transistor with an emitter input and a collector output could be a Yes gate. Whatever state is input will appear in the output.



Logical 1	Logical 0
True	False
High	Low
H	L
+5 V	0V
Set	Clear
Set	Reset
Yes	No
	

Fig. 10-7. Here are nine variations on the descriptions of the two logic states. The logic states, regardless of the name, are still only voltages.

leaves the Program Counter and arrives on the 16 bus lines must open up only one address. If more than one of the addresses are read or written to, the computing will crash.

To avoid this, all the locations in the computer can be equipped with their own YES gate that also has a three-state control. While the computer is not addressing a location, all the three-state controls are off and the memory map cannot be accessed. As soon as an address goes out over A15-A0 the one location that is addressed has the three-state control in its YES gate go on. That way only the one location is contacted and all the rest of them stay in an inaccessible third state. There are a lot of three-state devices in the 64. The 6510, the RAM, the ROM, the gates in the address and data bus lines, and others have three-state capabilities.

### NOT Gate

The NOT gate in Fig. 10-9 looks a lot like the

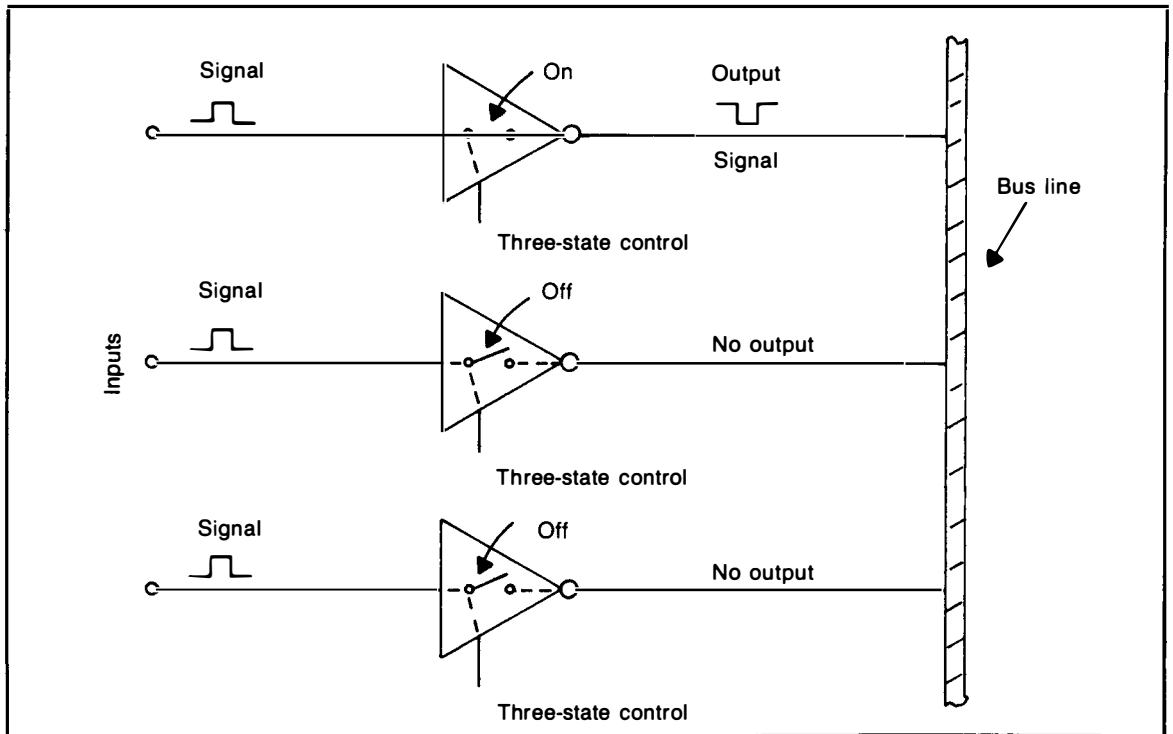


Fig. 10-8. The third logical state, when the chip is purposely turned off, is called three-state. It is vital to keep all circuits off except the one in use during an operation so they do not interfere with each other.



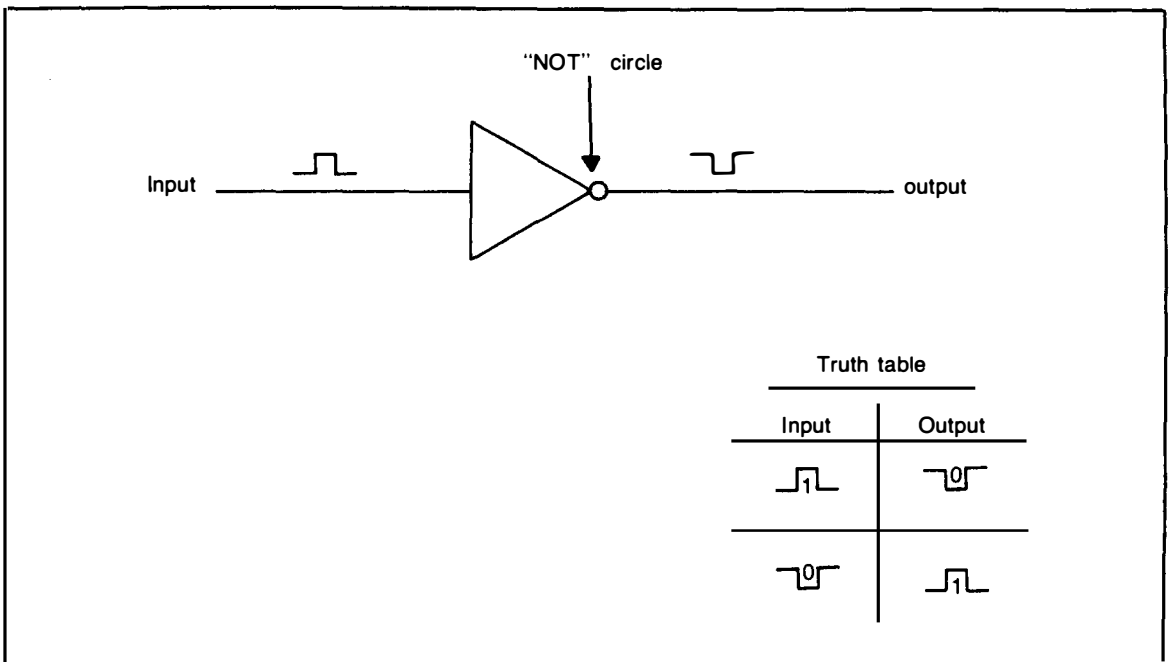


Fig. 10-9. The schematic symbol of the NOT gate looks like the YES gate except for the little NOT circle on the output point.

YES gate on a schematic drawing. It has the same triangle shape and the same input and output leads. The only difference is a tiny circle between the pointed end and the output lead. This is called the *NOT circle*. The NOT circle is what makes the device a NOT gate. Whatever state enters the gate, the opposite state leaves the gate. If a high is input, a low is output. Should a low go into the gate, the low is changed into a high and the high comes out. The truth table for the NOT gate is shown in the illustration.

Figure 10-10 shows the bipolar circuit in a NOT gate. It is something like the YES gate. An npn transistor could act out the part of the gate. The VCC is still +5 volts. The obvious difference between the two circuits is in the input. The NOT gate input is connected to the base of the npn. The YES gate had the input tied to the emitter. There was no reversal of logic with the emitter input. When the logic state enters the base, the state is reversed. The chip is also called an inverter.

The NOT gate or NOT circle is one of the most used abilities in a computer. The inversion of the

logical state is the opposite or the complement. NOT gates always output the complement of the input. During testing, the fact that the output logic level is always the opposite of its input provides a quick check to test an inverter.

The 7406N chip in the 64 has six inverters. They are all used in output lines that need the state of the line complemented. This chip was discussed in Chapter 8. There are many other NOT gates and circles in a lot of the other chips in the 64. They are vital to the operation of the computer.

The NOT function is found in all sorts of places in the Commodore 64. As you peruse the schematic during troubleshooting, there are many terminals described with a straight line drawn across the top of the name. This overscore means NOT. This is the NOT line. It designates an inverted quantity. The asterisk I've been using is a substitute for the line. I use the asterisk because it is simpler to use in writing. When you see the asterisk instead of the line, it means the same thing as the overscore. It's a practical measure with no other meaning. The asterisk is also recognized as a NOT.

When a NOT sign is used on a terminal, it usually signals the type of logic state that will enable the terminal. For instance, if you see a terminal called \*RESET, it means the pin is usually held high while the reset function is not being used. If and when the time arrives for the reset to be enabled, the terminal is injected with a low, and the reset goes into its routine.

On the other hand, if the terminal is called RESET without a line or asterisk, the opposite effects can be expected. The terminal will be held low when inactive. To enable the reset function a high is sent to the pin. The high turns on the reset routine.

When you are reading a schematic and you see a terminal name to describe its function, try to form the name in your mind. For instance, the read/write line in the 64 is described as R/\*W. This would be pronounced as "Read-NOT-Write". Actually the read/write line is usually held high in a read position. To produce a write the line must be forced low. Another example is the \*IRQ line. It is an interrupt request. It is held high till the interrupt is

required. Then the line is made to go low. A third example is RDY, a ready line. It is held low when it is on standby. To activate the RDY, the line is put into a high state. This form of thinking as you read the schematic should become second nature as you check out the test points on all the chips and connections.

## AND Gate

The AND schematic symbol is shown in Fig. 10-11. It looks like a short fat bullet lying on its side. In contrast to the total of two leads on the YES and NOT gates, the AND gates in the 74LS08 chip of the 64 have three leads. It has two inputs and one output. Actually, an AND gate can have three or more inputs, but it will only have one output.

When the AND gate is also equipped with a NOT circle between the blunt rounded end and the output lead, as shown in Fig. 10-12, it becomes a NOT AND gate. You probably have seen these gates. The NOT AND is shortened to NAND. The NAND gate outputs are the complements of the AND gate. We will discuss for the most part the

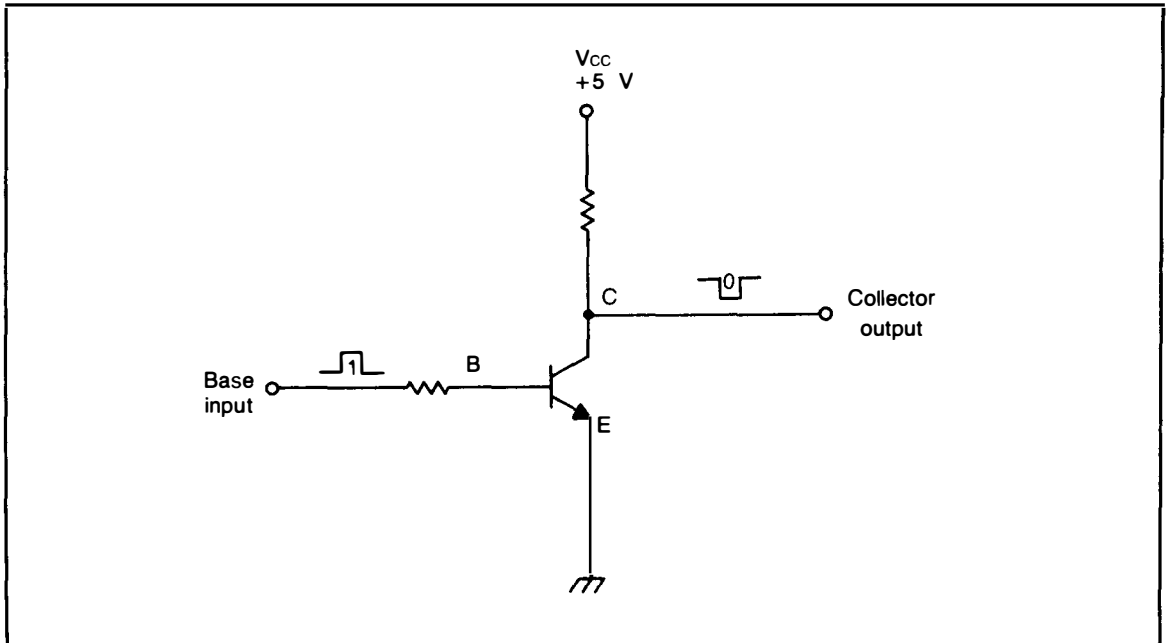


Fig. 10-10. The NOT gate can be constructed with the same npn transistor that the YES gate was made with. The input enters the base instead of the emitter and reverses the logic state at the output.

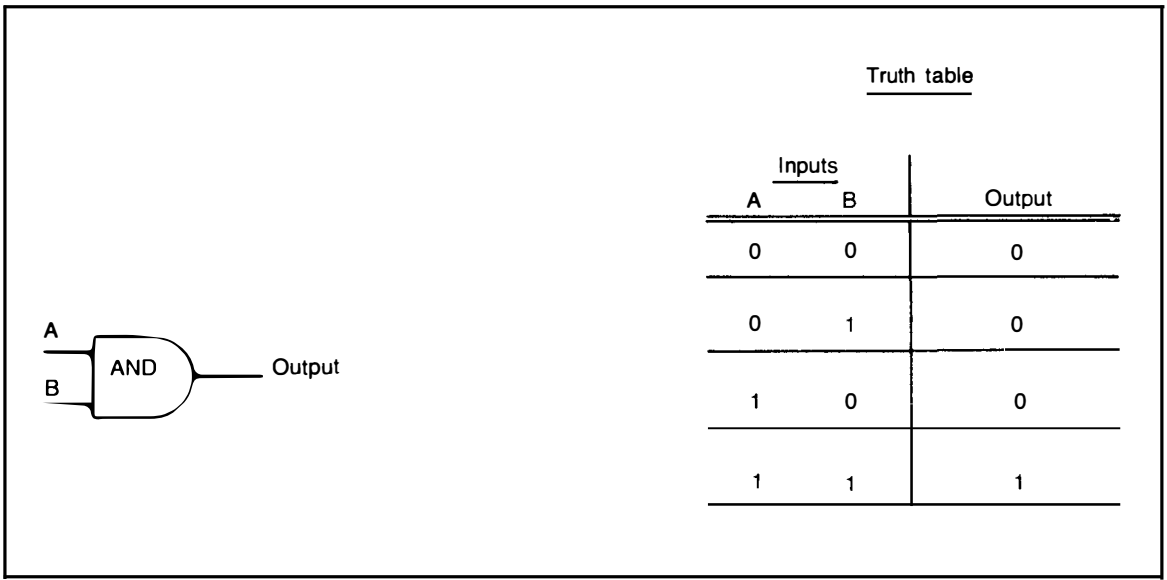


Fig. 10-11. The AND gate's output is low on all occasions except if both inputs are high.

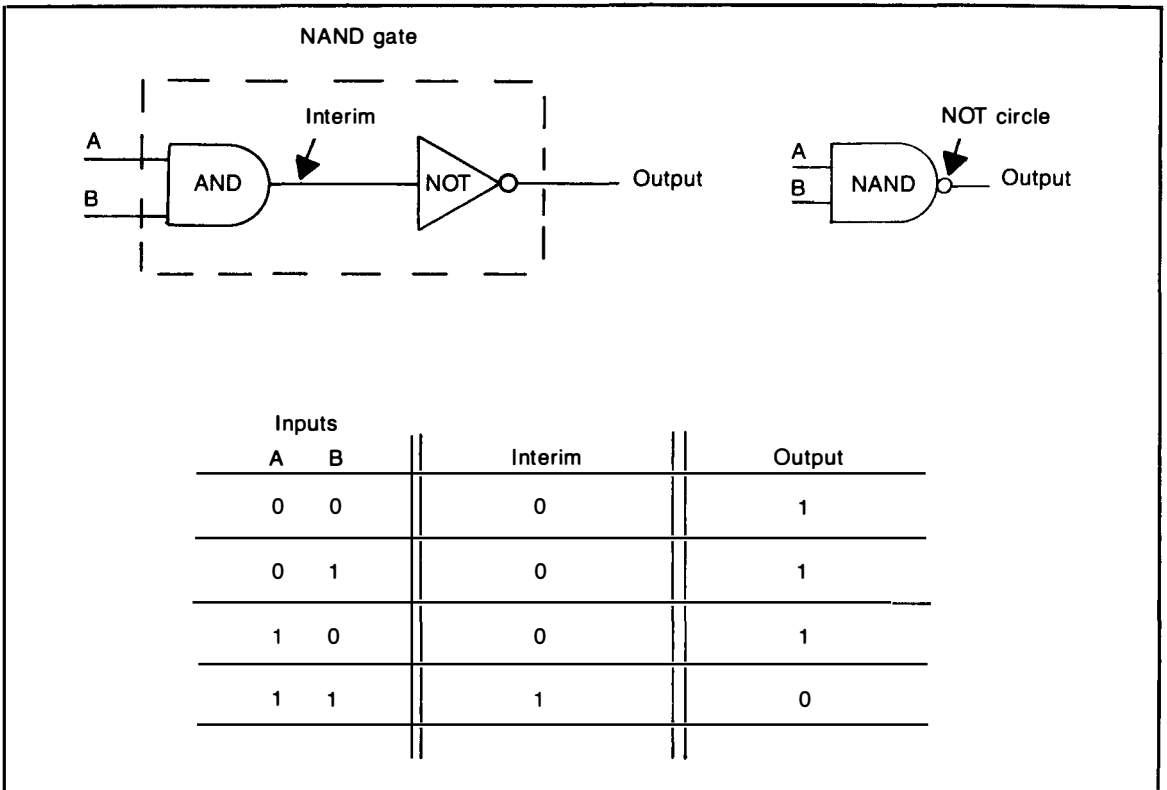


Fig. 10-12. The NAND gate is really a NOT AND gate. The interim output is that of an AND gate.

AND gate and not the NAND. There is an AND gate chip in the 64, the 74LS08. This chip was discussed in Chapter 8. There are no special NAND gates in the machine.

Inside the AND gate, as well as the other gates, there are a lot of microscopic components. There could be all sorts of transistors, resistors, diodes, and so on. In fact, any gate can be formed by configuring a lot of other type gates on a chip and wiring them to tailor a particular gate. The little fat bullet on its side could be designating dozens of separate components.

From the servicing point of view, all you need to do is think of the gate in the simplest terms so that the testing and repair proceeds as rapidly as possible. That way you can trace a signal and pinpoint the source of a trouble. Except for your natural curiosity, it really does not matter which minute section of a miniscule internal transistor has given up the ghost. The important thing is to realize the total gate is dead and needs replacement. You can test the inputs and, from your understanding of the logic, be able to predict what the output should be. If the correct output state is present then

the gate is ok. If the output logic is incorrect, then the gate becomes a suspect.

The classical description of an AND gate in electrical terms is a circuit consisting of an output load such as a light bulb with two switches in series. The only way that the bulb in Fig. 10-13 will light is if both switches are closed. If one or both switches are open, the circuit is open and the lamp won't glow. There are four possible positions the two switches can assume. Assume that the energy is supplied by a 5 volt battery. The bulb has a high of +5 volts applied when both switches are closed and a low of 0 volts when a switch or both switches are open. We can call an open switch L and a closed switch H. The possible inputs are the following:

Inputs	Results
L-L	No light
L-H	No light
H-L	No light
H-H	Light

The same type of events take place when there

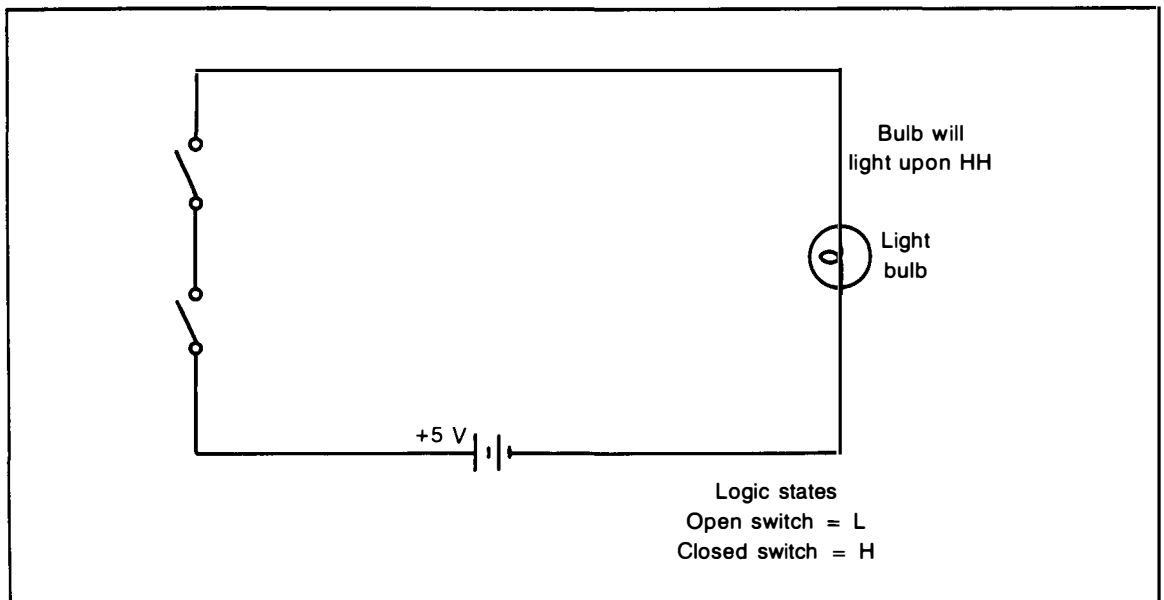


Fig. 10-13. The classical description of an AND circuit is made up of a battery, a light bulb, and two switches in series. An open switch is thought of as an L and a closed switch as an H. Only the application of HH will light the bulb.

are three AND inputs instead of two. The only difference is, three inputs create eight possible combinations that can be applied to the gate. Out of the eight inputs only one input group will light the bulb. That is when all three switches are closed. The inputs and results look like the following:

Inputs	Results
L-L-L	No light
L-L-H	No light
L-H-L	No light
L-H-H	No light
H-L-L	No light
H-L-H	No light
H-H-L	No light
H-H-H	Light

When the AND gate has four inputs, there are 16 combinations. Five inputs makes 32 combinations of possible logic states, and so on. No matter

how many inputs there are though, the only way the AND gate will output a high is if all inputs are Hs. When you are taking the state of the AND output, a high means all inputs are high. If one of the inputs are low and the output is a high, that could be a symptom of failure. The gate could have shorted or opened in a way that is causing the unexpected high reading.

The actual AND circuit could be based around a pair of pnp transistors wired in parallel. Refer to Fig. 10-14. The emitters of both the pnps connect to +5 volts through a resistor. The AND output is from the emitters. The two AND inputs are entering through the two bases.

When either or both inputs receive a low, one of the transistors or both will conduct and the output will be low. The only way a high can emerge from the output is if both inputs are high. When that happens neither pnp can conduct and the output emitters rise to the supply of +5 volts.

You could think of an AND gate as an elec-

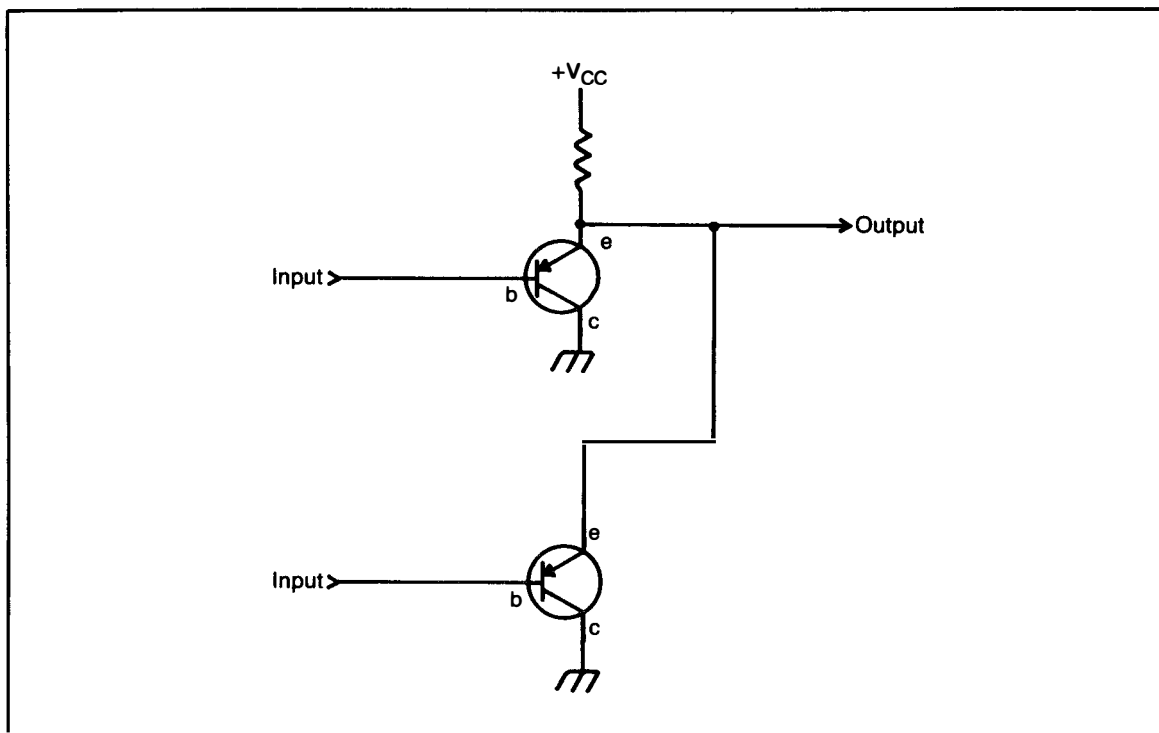


Fig. 10-14. Two pnp transistors wired in parallel with a base input and an emitter output can be an AND gate.

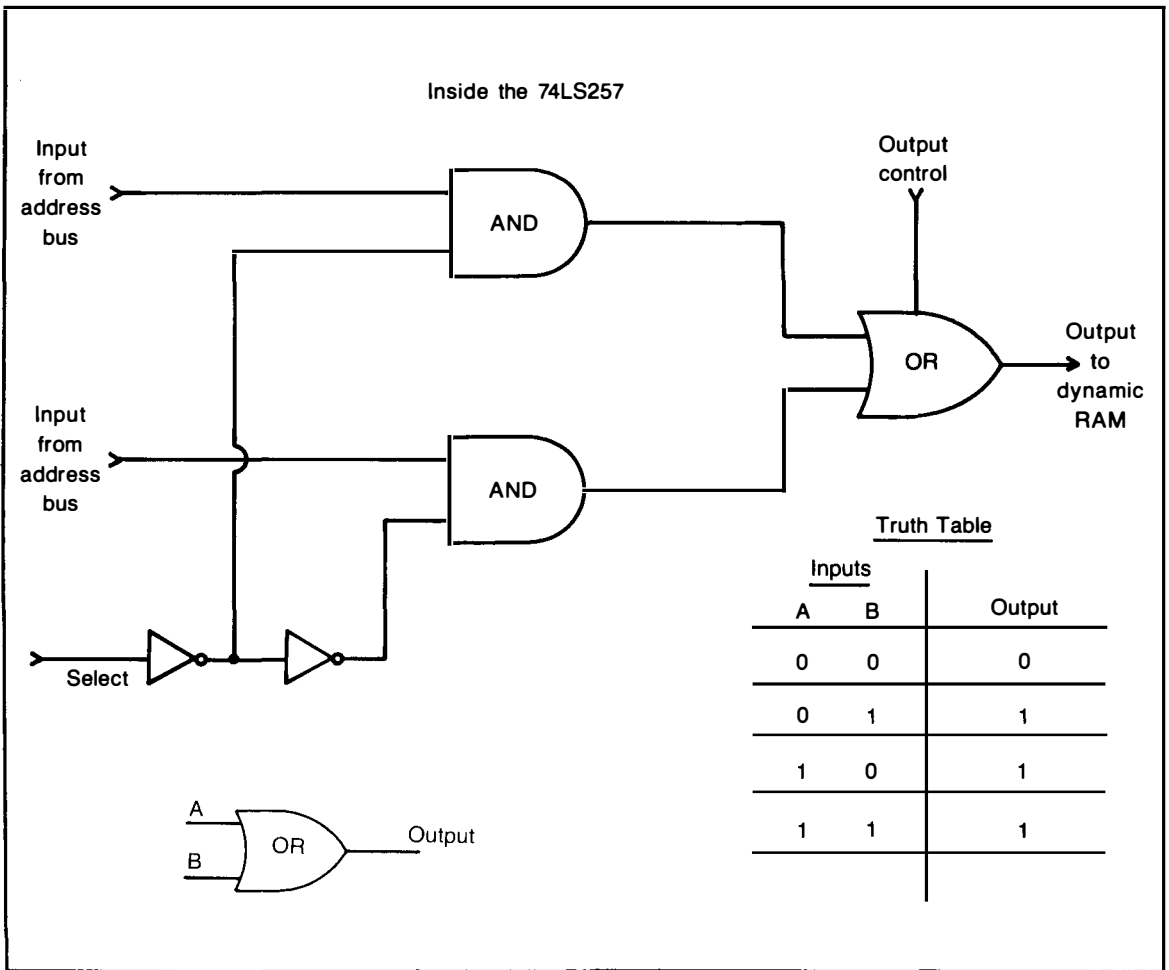


Fig. 10-15. The OR gate on one output pin in the 74LS257 receives signal from two AND gates. Each AND gate is operated by one address bit and a select signal. The OR gate symbol looks like an artillery shell lying on its side. The OR gate's output is high unless both inputs are low.

tronic combination lock of sorts. A low is output while the lock is shut tight. The only way the lock will open is when the correct combination is input. The correct combination is the input of all Hs.

## OR Gate

There are no logical OR chips in the Commodore 64. The OR gates that are in the 64, and there are plenty of them, are found in the internal wiring of other chips. Figure 10-15 shows that the 74LS257 chips use OR gates as part of the internal wiring. The OR gate outputs are connected to

pins but the inputs are attached to the outputs of the AND gates that they are doing the multiplexing with. The inputs from the address bus are the inputs to the AND gates. This chip was also discussed in Chapter 8.

The OR gate is drawn schematically like an artillery shell on its side, instead of a fat bullet like the AND gate. Refer to Fig. 10-16. The classical electrical representation of the OR gate is also shown as switches that operate a light bulb. The wiring in Fig. 10-17 is different than the AND circuit. Whereas the AND circuit had switches in se-

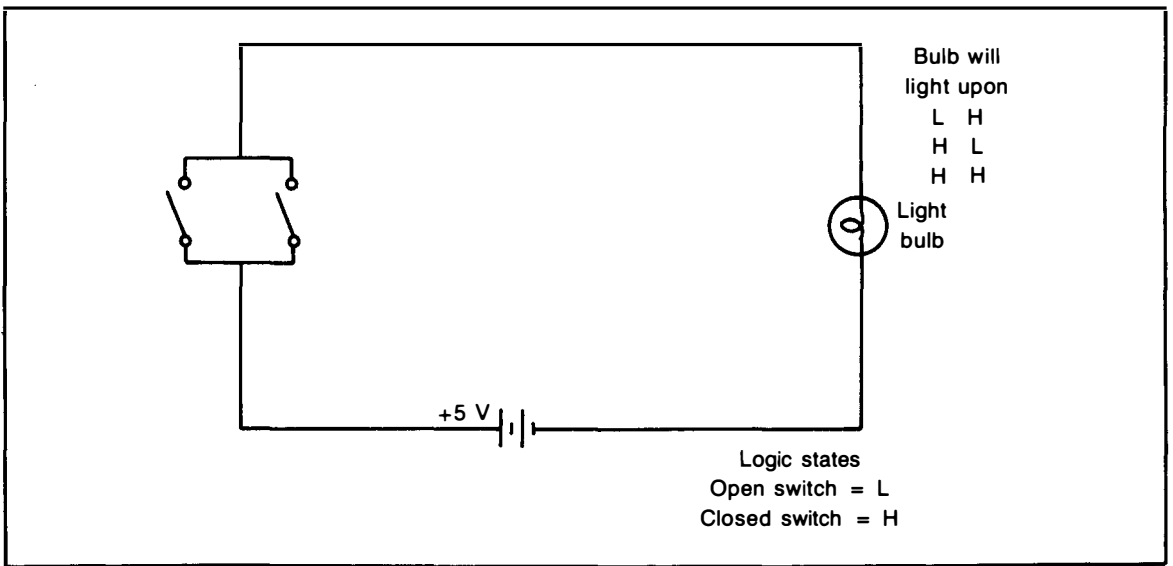


Fig. 10-16. The OR gate can be thought of as two switches in parallel. The bulb will light if either one or both of the switches is closed.

ries, the OR circuit has the same switches in parallel. The AND circuit also needed all the switches closed to light the bulb, but the OR circuit only needs a single switch closure to illuminate the bulb. The possible inputs are the following:

Inputs	Results
L-L	No light
L-H	Light
H-L	Light
H-H	Light

The OR gate also can have more than two inputs. There can be three, four, or more. Like the AND gate, two inputs produce four input possibilities, three inputs can have eight combinations of Hs and Ls, and four inputs make 16 possible ways that the inputs can be arranged. However, no matter how many inputs, the OR gate will output a low only when all inputs are low. Otherwise the OR gate will output a high. If just one input out of many is high and all the rest are low, the OR gate will put out a high.

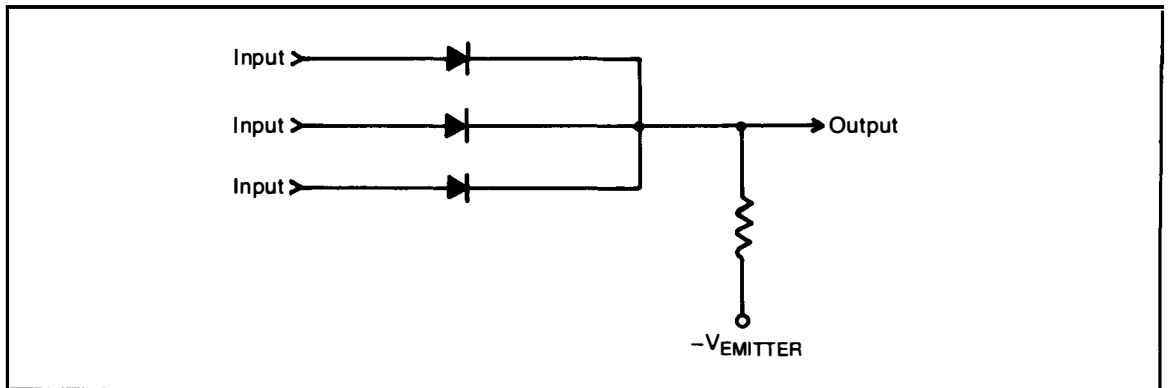


Fig. 10-17. The OR gate can be built with diode inputs in parallel.

An OR gate can be built with a number of diodes and a resistor as shown in Fig. 10-17. If you want a two input gate, two diodes are used. For a three input gate, three diodes are needed. One diode is needed for each input.

The diodes are wired in parallel. The anodes of the diodes are the input pins and the cathodes are tied together. A resistor from -5 volts is connected to the cathode junction. The junction is also the OR output pin.

While the inputs are all near 0 volts, the diodes do not conduct and the output is held at -5 volts which is a low. If a high should appear at one of the anode inputs, that diode conducts. This makes the cathode junction rise up 5 volts to a relative high. A high on any of the anodes will also produce a high at the output. This is logical OR activity.

### Exclusive-OR Gate

The 64 does not use any exclusive-OR abbreviated to XOR, chips. The XOR function though is very important to the machine. One of the instructions that the 6510 will respond to is EOR, which is also a way to describe the action. EOR is the acronym for exclusive-OR. When you order the 6510 to EOR, it will exclusive-OR all the bits in the accumulator register with any memory location you provide. There will be more about this in the next

chapter on registers. Meanwhile, we will review the XOR gate so you will understand the EOR when you get to it.

The XOR gate symbol in Fig. 10-18 looks almost exactly like the OR symbol. The only difference is a space between the input leads and the bottom of the body of the symbol. A closer look reveals that at the input end, the leads are attached to a curve and there is a space between the curve and the rest of the symbol. If you think of the symbol as an artillery shell, then you can think of the input leads as the ramrod used to push the shell into the big gun. The output lead is on the pointed end of the shell appearing symbol.

Let's compare the OR and XOR inputs and outputs to see the difference the exclusiveness makes. If you are signal tracing a two-input OR or an XOR gate there are four input combinations possible for both gates. They are the same inputs whether the gate is an OR or XOR. The difference is in the outputs.

On the OR output, the state will always be a high except if both inputs are lows. If you logically think about these results, they do not really follow the definition of OR. The OR definition is, if one "OR" the other input is high, then the output is high. This is true for L-H and H-L. For L-L, neither input is high so the definition indicates a low, which will be present as long as the gate is ok.

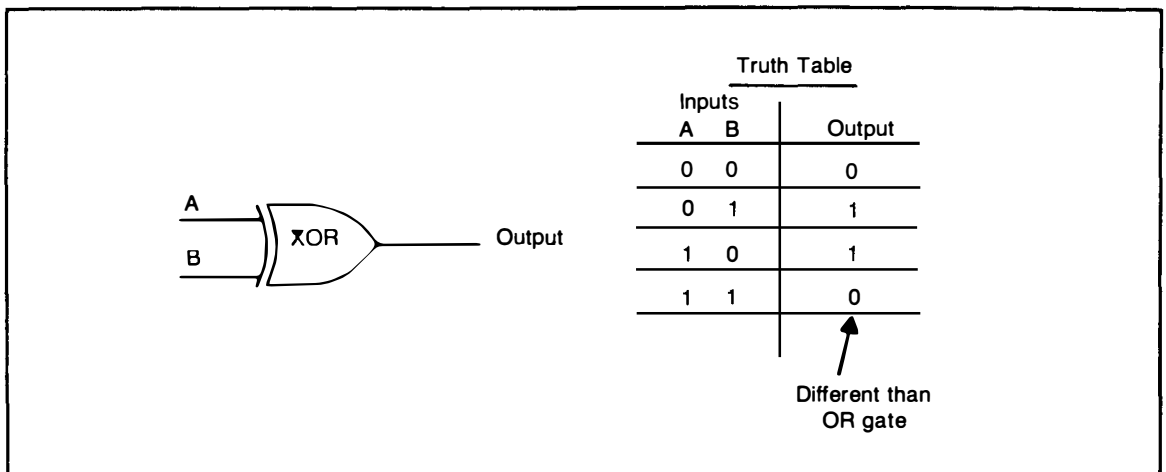


Fig. 10-18. THE XOR gate is similar to the OR gate, but when two highs are applied to the inputs, the XOR outputs a low.



For the last possibility though, both outputs are high, H-H. Yet the output is high.

The XOR gate is a variation of the OR gate that does follow the logic. When the four input possibilities are considered, they are identical to the OR gate except for the last possibility, H-H. When H-H are the inputs of the XOR gate, the output is a low. The XOR gate only outputs a high when one or the other of the inputs are high. If both inputs are the same, both low, L-L, or both high, H-H, the output is a low.

### NOR and XNOR Gates

You will not have to contend with NOR, XNOR, and other gate variations. The 64 deals mostly with AND, OR, NOT, and some XOR logic abilities. If you learn these four functions and understand what the outputs of the gates do with various inputs, troubleshooting the 64 will become appreciably easier. I include a brief description of these other gates simply for completeness.

NOR is NOT OR and XNOR is exclusive NOT OR. The NOT simply changes the outputs to a complement. The output of the NOR gate is the complement of the OR output. The same thing goes for the XNOR. Figure 10-19 shows the complementary

relationships. Compare these with Fig. 10-15 and Fig. 10-18.

### GATE TESTING TECHNIQUES

When a gate is to be tested the best equipment is the vom and the logic probe. These instruments however, can only give you the surface indications. As we have seen, the chips are circuits within circuits within circuits. Deep down at inaccessible levels are the bipolar transistors, the FETs and other components that actually do the computing. They are forever sealed at manufacturing in their microscopic space. You can think about them, but under normal servicing circumstances you'll not encounter them directly.

A gate is at the next level of size and they do have test nodes that you can measure and obtain readings. The gate circuit level is the level where most of your voltage and logic state testing will take place. There are inputs, outputs, VCC (collector) and VDD (drain) pins readily available. The more you know about the way the Hs and Ls progress through the gate level circuitry, the quicker you will spot incorrect logical states and pinpoint troubles.

The top level of circuitry is the chip itself. When the chip is considered as a single component

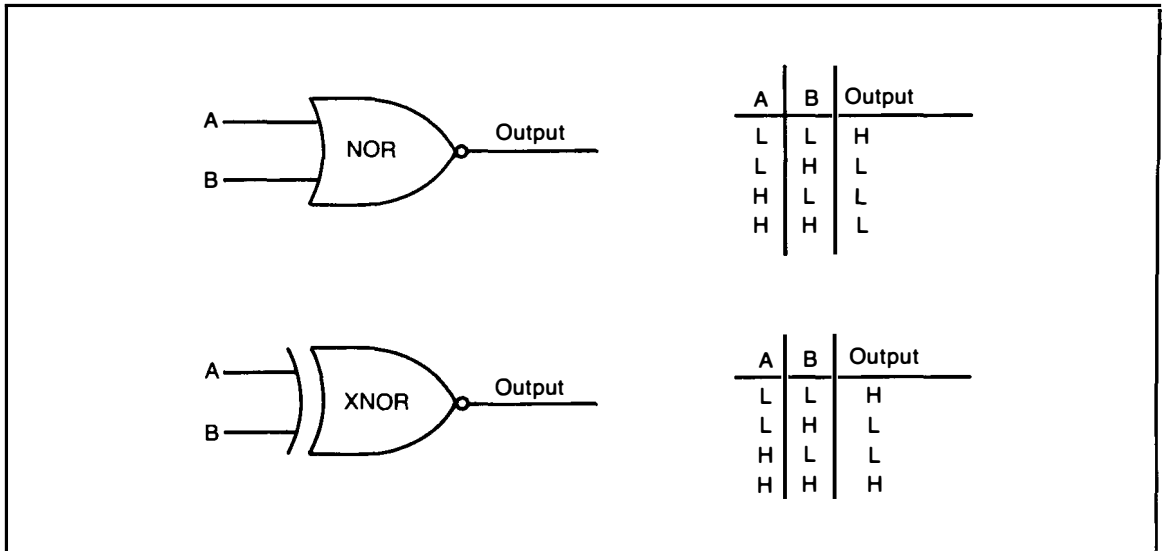


Fig. 10-19. The NOR and XNOR symbols and truth tables.

and is replaced as a test, you are in the overview mode. This is the first approach to a repair and is the gist of the chapters up to this one. More than 50% of actual troubleshooting that takes place in the field is in this overview mode. As a matter of fact, manufacturers have taken this mode of servicing a couple of steps further. They often will replace the entire board or even the entire computer to get you back in operation. The troubled computer can then be placed back on the factory production line and be recycled back to new.

Independent servicers, without the benefit of a large factory, follow the same procedure they have been using for years on TV repairs. Com-

puters are easily carried. The home computer or small business computer owner disconnects a broken unit and carries it into a repair shop. The servicer approaches the repair by changing the socketed chips that are indicated by the symptoms of trouble. Once the socketed chips are deemed ok, then the servicer will pull out service notes on the machine and attack test nodes with the vom or logic probe. He looks for incorrect voltages and wrong or missing logic states: missing or wrong supply voltages, highs where there should be lows, lows where highs should be present, three-stating in the wrong places, and so on. As you can see, knowing the way gates process highs and lows is a must.

A black and white photograph showing a person's hand holding a digital multimeter. The multimeter is connected to a circuit board, likely a computer component, with various components and a keyboard visible in the background. The scene is dimly lit, focusing on the testing process.

# 11. Servicing Digital Registers

**B**Y THIS TIME, YOU SHOULD KNOW THAT, besides the logic gates, digital electronics requires another entire category of TTL and MOS devices. They are called registers. Registers are found throughout the digital circuits. There are many in all the main LSI chips. There are thousands upon thousands of them in RAM and ROM. Every resident of the memory map is a register of one sort or another. A number of the rest of the smaller chips are forms of registers.

The important difference between a gate and a register is illustrated in Fig. 11-1. A register can store a high or a low while a gate cannot. As soon as a logic state arrives at a gate, if it is not three-stating, the high or low gets a quick passage through to the output. It can't linger in the internal circuit. It is forced on through. The only time required is the propagation delay of the electron movement, which is typically about 15 nanoseconds. Besides this 15 billionths of a second pause, the state passes through the gate.

A register on the other hand has bit holders. It is a place of storage. Once a high or low enters

a register bit holder it can stay as long as you want it to, providing the electricity stays on.

A register is composed of one or more single bit containers. In the 4164 dynamic RAM chips, there are 65,536 single bit registers on each chip organized as  $65,536 \times 1$ . The 2114 Color RAM chip is organized as  $1024 \times 4$ . This chip has 1024 4-bit registers in its memory matrix. The 74LS373 D Latch connected to the Color RAM has one 8-bit register. The Program Counter in the 6510 is a 16-bit register. All the various sized registers do the same type of job. They are all able to store bits according to their organization.

Each form of register except the dynamic RAM uses flip-flop circuits as bit holders. The dynamic RAM uses the voltage that is contained in a capacitor charge. Chapter 8 contains additional information on each of the following chips.

## FLIP-FLOPS

The flip-flop circuit is as old as electronics. It is a means of storing a charge. In the digital circuit, the presence of a charge is a high, and the lack of

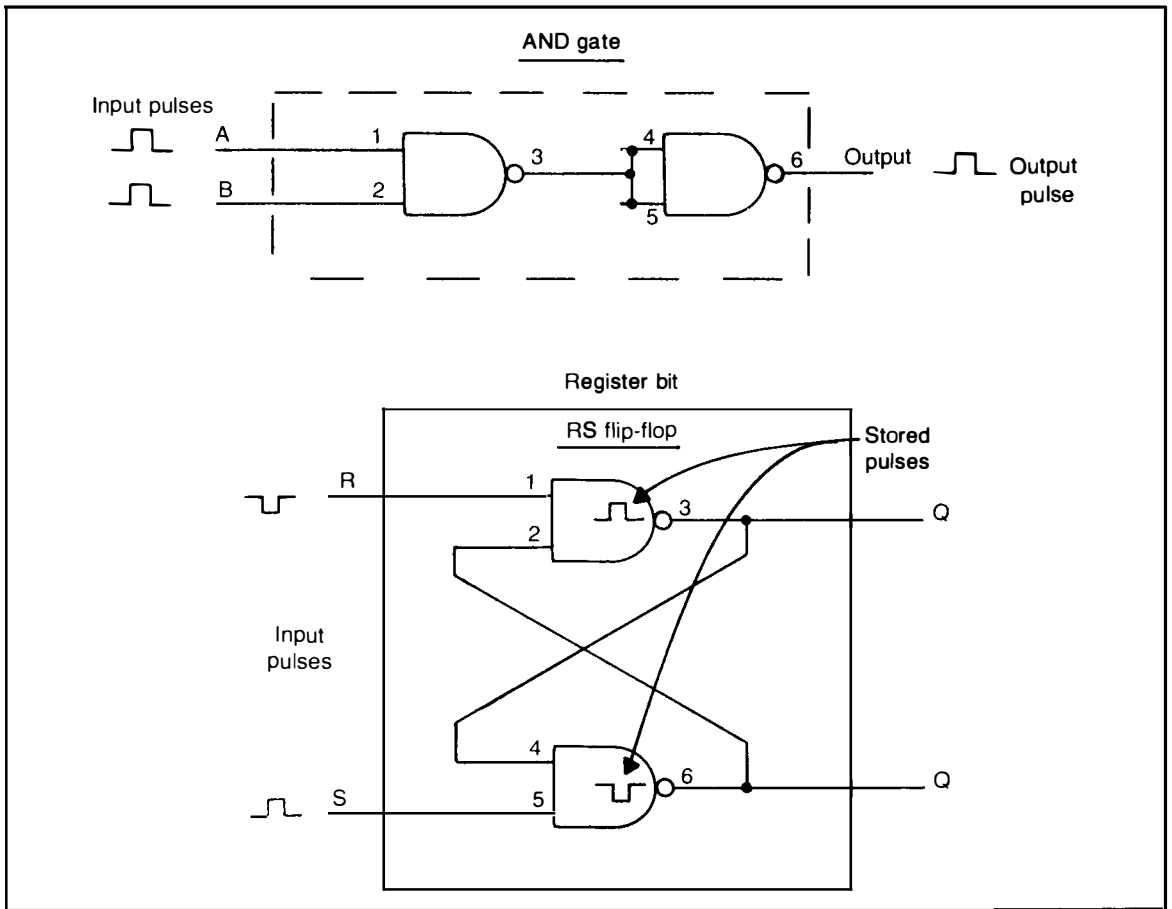


Fig. 11-1. The main difference between a gate and a register is that a register can store a logic state and a gate cannot. In the top circuit, two NAND gates are wired together to form an AND gate. Inputs are quickly ushered from the inputs to the output. The bottom circuit shows the same two AND gates wired as a register bit. Inputs can be stored in the internal circuitry.

a charge is a low. With a flip-flop, you can not only store the charge, but you can manipulate it. You can change a high to a low or a low to a high. This is a very valuable ability. When you couple the storage and the state changing with the basic gate logic of NOT, AND, OR, and XOR, you are computing.

A flip-flop can be built by crisscrossing two triode vacuum tubes, or two npns, or two pnp's, or two FETs, or even two gates. They are all able to characterize a flip-flop. Let's examine the step-by-step flip-flop action of the circuit in Fig. 11-2.

The circuit shows the transistors with inputs

into their bases. Since the inputs are into the bases, the individual pnp's are wired as inverters. The bases are then cross-coupled into the other pnp's collector. When you power the two pnp's, they both try to conduct. However, one is always slightly quicker than the other. There is feedback from the collectors to the other base. The quicker pnp will go into saturation. The slower one will thus be forced into cutoff. They will hold this state of conduction.

The one in saturation outputs a high as the pull up resistor has a large voltage drop across it. The one cutoff outputs a low since its resistor has no

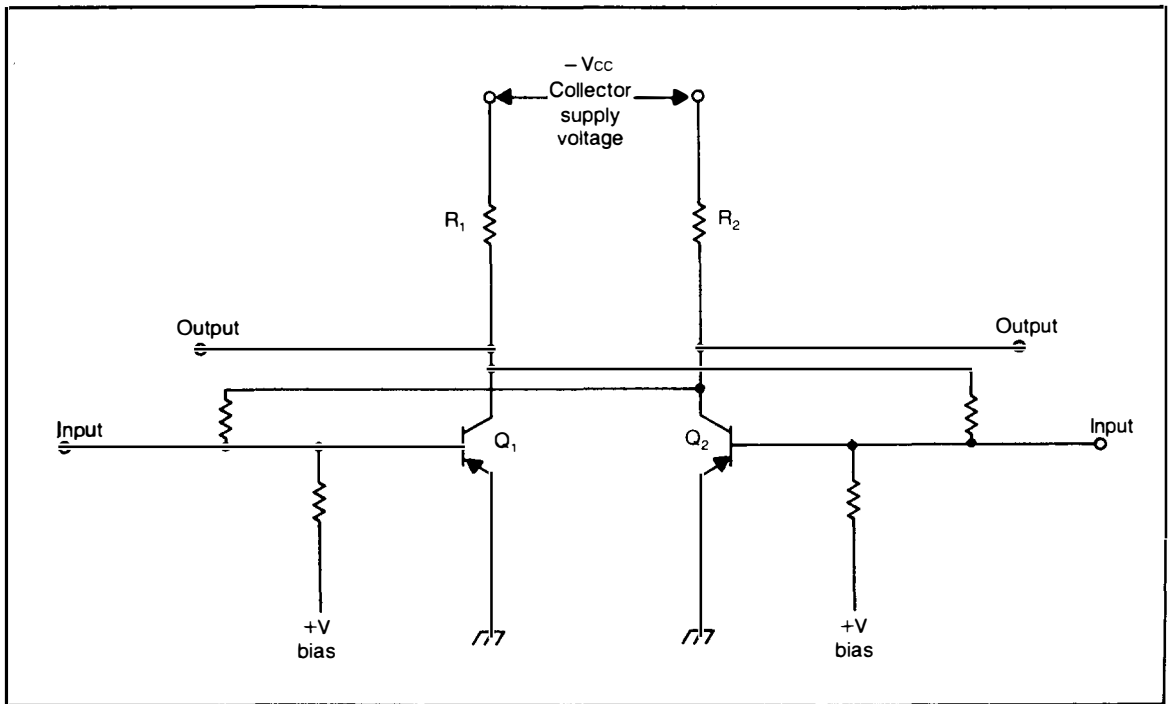


Fig. 11-2. The basic flip-flop can be built around two pnp transistors.

voltage drop and is at the  $-V_{CC}$  supply voltage. The two transistors will maintain this state. The only way it can be forced to change is if a high pulse is applied to the base of the cutoff pnp or a low pulse is sent to the base of the saturating pnp. If either event takes place the two transistors will flip-flop their logic states. The pnp that was cutoff will then saturate. The pnp that was saturating will cutoff. The flip-flop will then hold that state till another pulse comes along to change it again.

While you can't get into a chip to test flip-flops, it is useful to understand the way it stores a voltage state. There are a number of tests you can make on flip-flop circuits to see if they are able to store voltages. For instance, if you know the address of a suspect register or bit holder, you can use the BASIC PEEK function to see if it is actually holding the highs and lows that it is supposed to. The POKE command can be used to install test bits into the register and then PEEK can look to see if the test bits ever arrived. Refer to Fig. 11-3. There will be more about this in later chapters.

There are a number of chips in the 64 that are flip-flop types. The 74LS74 in the clock circuit contains two D flip-flops. The 74LS373 is called a D latch, and the 74LS193 in the clock configuration is known as a binary counter.

The 556 chip has two separate flip-flops inside. The flip-flops are completely independent of each other except for the fact that they share VCC and ground. The timing ability comes from some comparator circuits that are also built into the silicon. Let's examine the Commodore's chip flip-flops.

### 74LS74 D Flip-Flop

The 74LS74 chip contains two flip-flop circuits. The drawing of the pinout was shown in Fig. 8-6. There are two pins for VCC and ground and six other connections.

The D FF has one Data input. In fact, that is why it is called a D. Other type FFs are called D Latches and RS. The D has one Data (D) input. The input is a single high or low at a time. When the

logical level enters the D pin, it is stored in the flip-flop circuit.

When a low enters D, a low will appear at the Q output. Q is the important output. The other output, \*Q, will become a high. This is not an important output. It can be useful sometimes, but the Q output is the one that usually passes the logic state.

The flip-flop, though, will not relinquish the state it is storing that easily. It must be triggered by a clock pulse that goes high. The FF will only give up its stored state when a high clock arrives. The 74LS74 requires a high clock pulse to pass the state from D to Q. Other FFs can use a low clock pulse for the data transfer.

The 74LS74 has two other input capabilities. They are Preset and Clear. They are used sometimes, but not all the time. When it is necessary to preset the Q output to a high, the input pin preset receives a low. When Q is required to be a low, then the low is inserted at the Clear pin.

Figure 8-7 shows how the two 74LS74's FFs are installed in the computer clock circuit. One FF connects to the Jumper Select at its preset pin 4. The output Q is attached to a binary counter pin. The Clear, D, and Clock pins, 1, 2 and 3, are all grounded. The FF therefore has a preset input and

a Q output. The Jumper Select is low for the American NTSC TV arrangement. In the U.S. the preset receives a low. With preset low, Q is high. The FF, therefore, is setting the standard for U.S. operation by inserting the high into the binary counter. That is all that FF is doing. \*Q had no connection and the rest of the pins are grounded.

The other FF is receiving a Clock signal from the counter at pin 11. The preset, pin 10, and the Clear at pin 13 are both held high at VCC. The highs keep them inert since they only affect Q when they go low. D and \*Q are shorted together. This renders them inactive. The Clock signal then turns Q off and on. Q is output to the phase detector part of the MC4044 chip. Qs frequency then drives the chip.

That is the way the two FFs on the 74LS74 chip are used in the 64. You can test them with the logic probe. The test point information in Fig. 8-6 gives the results that the test equipment should reveal. Any deviation from those readings could be a symptom of a defect.

### 74LS373 D Latch

The 74LS373 is a D latch. The D latch is a close

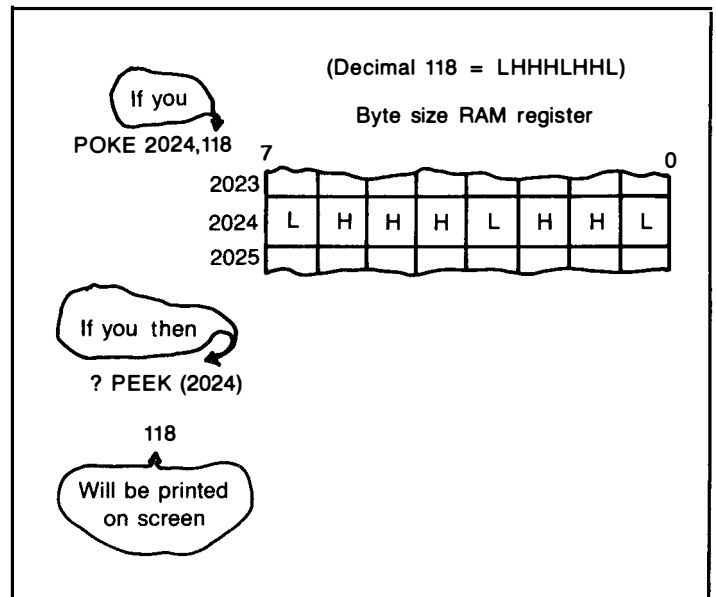


Fig. 11-3. Any register in the memory map and every bit in the register can be tested with POKE and PEEK tests.

relative of the D flip-flop. This latch has eight FFs in it. Latches can come with four, six, or eight to a package. They are usually used in bus lines and different bus lines in different applications use different packages. The Commodore 64 uses this eight pack in the address bus. The eight data inputs, D7-D0, connect to the VIC bidirectional address lines and the output of 74LS258 multiplex chip. The outputs Q7-Q0 attach onto A7-A0 of the address line. Refer to Fig. 8-19 and Fig. 8-20.

A latch is a parallel output device. With this eight pack, you can send eight bits over eight address lines and have them stored or latched till you want them to continue their journey. The latch is different than the D in that the D FF uses a clocked input to release the stored contents. A latch uses a latch enable pin. Pin 11 is the latch enable.

When data arrives at the eight FFs, it is stored. The latch will hold the data while the latch enable

pin is low. As soon as a high arrives at the enable pin the data is freed and allowed to pass onto the Q output. In this 64 circuit, the data comes from two places. The data entering D5-D0 is an address from VIC. The data at D7 and D6 arrives from the multiplex chip. The output pins Q7-Q0 connect to A7-A0 of the address bus. The chip is helping out in the complex addressing chore. Both the MPU and the VIC, at different times must address RAM. The octal latch 74LS373 works with the multiplex chip and the VIC to supply the correct address bits at the right time.

Each of the eight flip-flops have a D input and a Q output. The D and Q numbers show which is which. For instance, D0 is the input to FF 0 and Q0 is the output for FF 0. All of the flip-flops are tied together on one clock line and one three-state line. Pin 11 is the clock line input. It updates the latch with the strobe signal from the VIC (\*RAS).

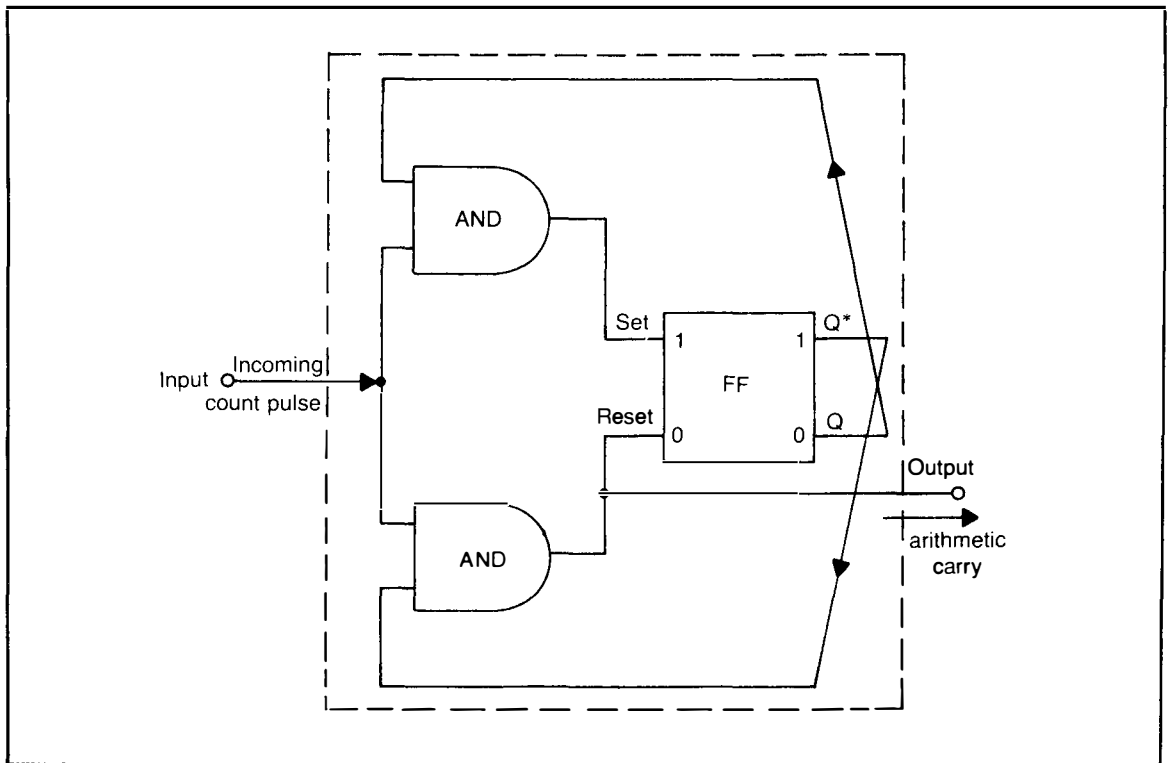


Fig. 11-4. The typical binary counter can count 0, 1. The output pulse can then be applied to the next stage as an arithmetic carry.

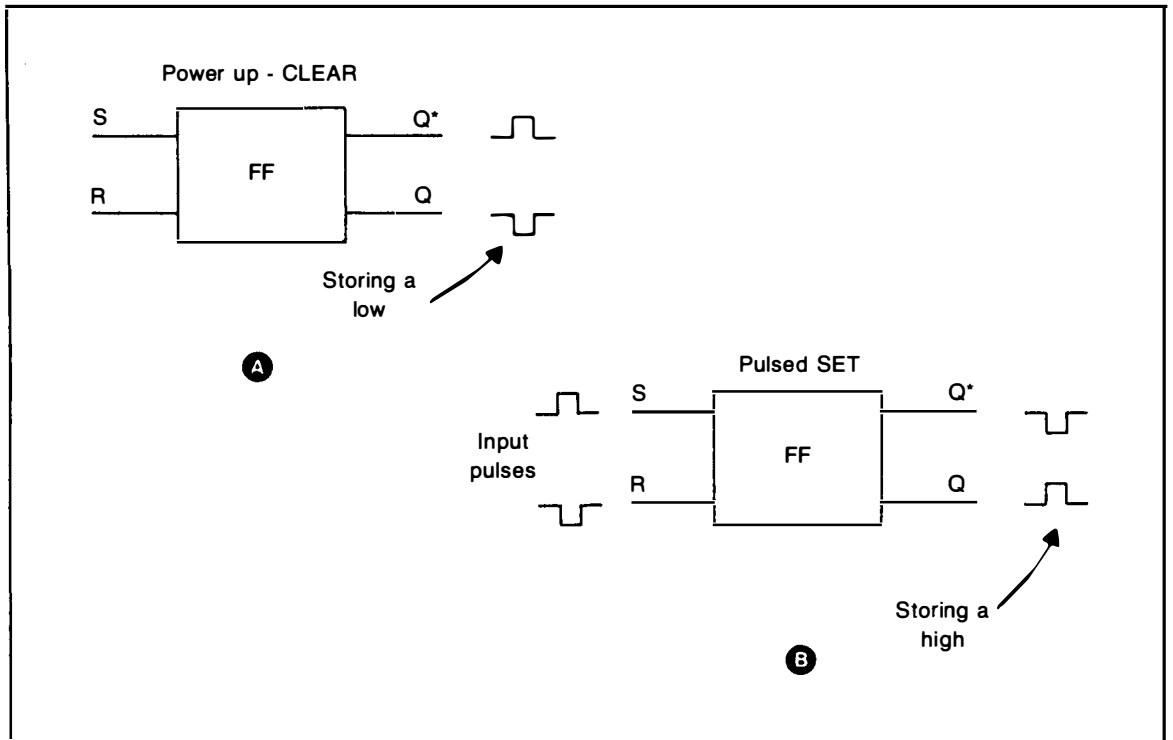


Fig. 11-5. In the RS Flip-Flop, the Q output exhibits the logic state.  $\bar{Q}$  always has the opposite state of Q.

It will open the latch and let the eight bits pass to the Q outputs. Almost at the same time, a new set of eight bits moves in and takes the place of the bits that just left.

Pin 1,  $\bar{OE}$  is the three-state control for the entire chip. The signal AEC, also from the VIC, will turn the chip on and off by exercising the three-state capability.

The octal latch is an 8-bit register. It is a limited type of register. All it can do is receive the eight bits and hold them. Then, when given an enable signal it outputs them. The latch is a handy device for temporary storage of bits while they are being flashed around the bus lines during the movement of address or data bits. Flip-flop circuits are a lot more versatile than the latch shows them to be. Let's see what else they do in the 64.

### 74LS193 Up/Down Counter

A computer must be a good counter and the 64

is just that. Counting can be done with the circuit in Fig. 11-4. It has two AND gates and a flip-flop. One flip-flop can't count very high. It can only count from 0 to 1, but it can count. It can count from 0 to 1 because it can attain two states, a low and a high. The low is 0 and the high is 1. Besides registering the 0 and 1, the FF with the aid of the AND gates develops an arithmetical carry.

The binary counter, as it is powered up, could start up with an output state of low or numerically, a 0. The 0 state comes about as the two cross-coupled circuits develop the following states. There are two inputs and two outputs to a flip-flop. Let's call the inputs S and R, the outputs Q and  $\bar{Q}$ . These are the pin names for a popular FF called the RS-flip-flop. If you think of the two cross-coupled circuits in Fig. 11-5 as being two pnp bipolar transistors, R and S are the base inputs.  $\bar{Q}$  is the collector of the S pnp and Q is the collector of the R pnp. At power up, the  $\bar{Q}$  output assumes



a high state of 1 and the Q output a low state of 0. With Q being held low, the entire FF is thought of as being in a 0 state. \*Q is automatically held high when Q is low. The two sides are the complement of each other.

At that time, if a high pulse is applied to the S - \*Q input, the pulse causes the two sides to flip-flop. The outputs at \*Q and Q reverse their positions. \*Q goes low and Q goes high. With Q now high, the entire FF is thought of as being in a state of 1. Therefore the binary counter has done the following. It receives pulses at its input. Each incoming pulse changed the state of the counter. If the counter is in a 0 state it sets to a 1. Should the counter be in a 1 state it resets to a 0. Incidentally, the S stands for set and the R means reset.

Since a single flip-flop can only count 0, 1, as the counter resets, or is thrown back into a 0 state, a pulse leaves Q. This is an arithmetic carry and can be applied to the next FF in a register.

The AND gates in the circuit are used to direct the incoming pulses to either the S or R input terminals. The top AND gate has two of its own inputs. One is coming from Q and the other from the incoming count pulse. The bottom AND gate inputs are arriving from \*Q and the same incoming pulse. The top AND gate's output goes to Set and the bottom output to Reset. The output containing the arithmetic carry leaves via Reset.

The AND gate that receives two highs during the state changing will be enabled and output a high to its FF connection. The other AND gate will receive a high and low. It is disabled and outputs a low to its connection. Both AND gates receive highs together from their common count pulse input. Only one of them receives a high from the Q and \*Q outputs since they are always the complement of each other.

Suppose \*Q is high and Q is low. The high from \*Q goes to the bottom AND gate, joins up with the

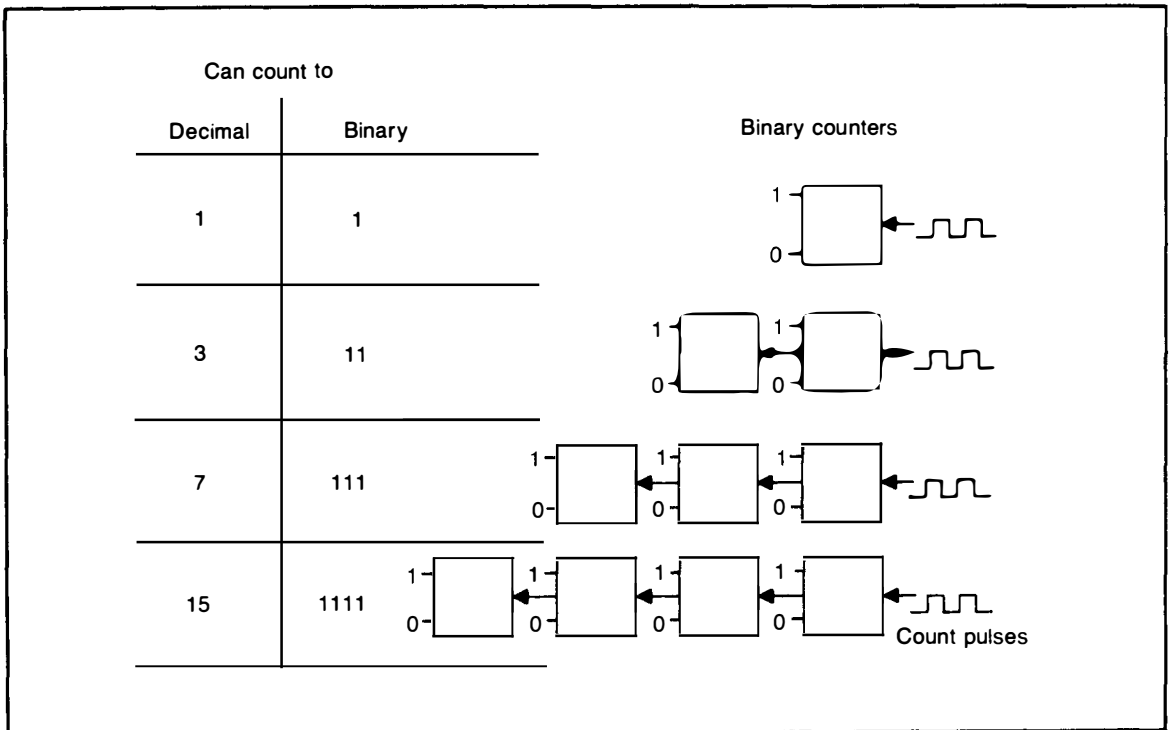


Fig. 11-6. Two counters can count 0, 1, 10, 11. Three counters can count 0, 1, 10, 11, 100, 101, 110, 111. Four counters can count 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.

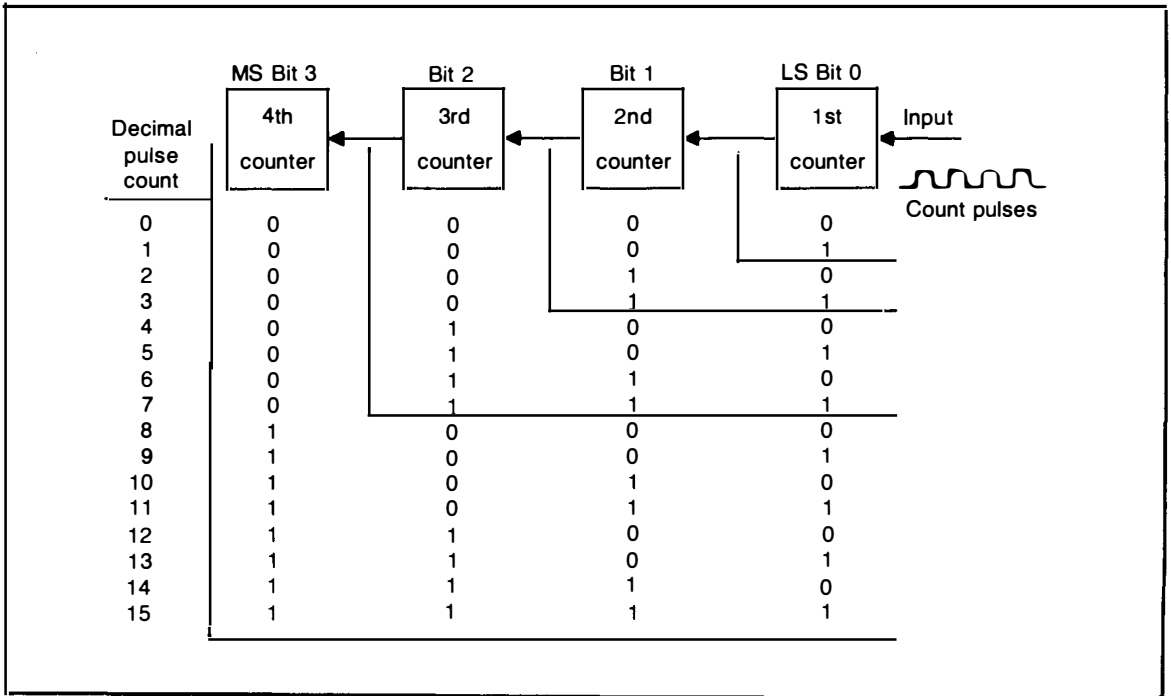


Fig. 11-7. Four binary digits are called a nybble. It takes the four bits to code one decimal digit.

incoming high pulse and the bottom gate outputs a high. A 1 is applied to Reset. The circuit, upon receipt of the high, flip-flops.

As the next pulse arrives, \*Q is now low and Q is high. This places two highs on the top gate. The gate is enabled and a 1 is applied to Set. The circuit again flip-flops.

To reach a higher count you must attach more FFs as in Fig. 11-6. If you connect a second FF to the first, you'll be able to count up to binary 11, which is equal to decimal 3. A third counter lets you take the count up to 111 which is decimal 7. The fourth counter raises the count to 1111, decimal 15. Note that each additional FF permits another 1 to be added to the total binary number. In the Commodore 64, the counter is a 4-bit type which counts from 0 to 15. The 74LS193 is an up/down counter which means you can count from 0 to 15 or back down from 15 to 0.

The way the counting proceeds from FF to FF is through the arithmetic carry output. The carry pulse from the first FF to the second bit holder

causes the second one to change states. If the second counter held a low, the flip-flop will turn it into a high. If a high had been there, the circuit will go low and a carry pushed forward to the third FF.

With three counters, a carry pulse from the second counter will cause the third one to flip-flop. When there are four counters the third one has the ability to send a carry pulse and make the fourth counter change states. This could go on and on.

Figure 11-7 shows a 4-bit counter and how it counts sixteen times, from 0000 to 1111. Four bits are a nybble and one nybble is equal to one decimal number up to 15. Decimal 0 is 0000 in binary. Decimal 15 is 1111, decimal 9 is 1001 and so on. The ratio of four binary digits to one decimal, is the code. When you count in the computer, each number is made up of 4-bit nybbles.

The 8-bit register has two nybbles. The 16-bit register has four nybbles. The nybble to number relationship is the basic connection between the way the computer counts and humans count.

In the 74LS193 4-bit binary counter, the

counting operation is straightforward. The four bits are initialized at 0000. As the highs in the pulses enter the input of the first counter, the circuit flip-flops to a 1. The count becomes 0001. The next pulse reverses the first counter to a 0. However as the reset takes place, a carry of 1 is injected into the second counter. Now the nybble reads 0010. The pulses keep coming. Each pulse either sets or resets the first counter. On each reset a 1 is carried to the second counter. The first and second counters can take the count from 0000 to 0011. In decimal that is 0, 1, 2, and 3.

As the fifth pulse enters the nybble counter, the second counter outputs a 1 to the third counter. The count becomes 0100. The sixth pulse makes it 0101, the seventh pulse 0110, and the eighth pulse 0111. The count continues pulse after pulse, 1000, 1001, 1010, 1011, 1100, 1101, and finally 1111. The following pulse resets all the bits to 0000.

In the 64, the 74LS193 sits in the center of the clock circuit. It receives the master clock frequency of 14.31818 MHz. It counts the pulses one by one in the four bits. Every 16 pulses, the counter starts over at 0000. Just before it starts over it outputs a pulse from pin 6, QC, to the input of the 74LS74 CK pin 11. Refer to Fig. 8-7. Since it is outputting one out of 16 pulses it is in effect dividing the master frequency by 16. This injects a signal near 1 MHz into the flip-flop coupling to the phase detector. This is the frequency that the 6510 runs at. There will be more detail about these circuits in Chapter 14.

Pins 4 and 5 are the inputs for the down count or the up count. Refer to Fig. 8-10. Pin 4 the down, is tied to a supply voltage high, which disables it. Pin 5 receives the 14.31818 MHz signal from the crystal circuit. It becomes the input count.

Pin 16 is VCC and pin 15 is tied to it. Pin 15 is a data-in terminal that goes to an AND gate. Pin 15 is thus held high and not otherwise used. Pins 12 and 11 are connected to a capacitor to ground. This keeps them out of the action. Pin 14, Clear is grounded and plays no part in the chip's countings. Pins 10 and 1 are tied together and both go to the Jumper Select. They are held high for the PAL frequencies and low for the NTSC standards.

Pin 9 is the data input from the flip-flop working with the Jumper Select. Pin 6 is the counter's final output after the master frequency has been divided by 16. That output enters a coupling flip-flop and then goes to the phase/frequency chip, the MC4044.

This division by 16 is just one of the many jobs a binary counter chip can do in computers. More details of these circuits are covered in Chapter 14.

## 556 Dual Timer

The 556 chip in the Commodore 64 has two identical sections. There is a flip-flop circuit in each section that acts as a 1-bit register. The register stores a control bit. One of the control bits turns the reset circuit on and off. The other control bit enables the \*NMI pin of the 6510. Both flip-flops are very stable, and once set or reset they will maintain that state unless forced to change. Refer to Fig. 8-25 and Fig. 8-26.

Control of the control bit is accomplished with special circuits called comparators. A *comparator* circuit acts something like a flip-flop, but it has some differences that make it difficult to use as a register. The main difference being a reference voltage that is applied. A flip-flop does not need this special voltage to store a state.

The comparator circuit in Fig. 11-8 is based around two pnp transistors wired with a common emitter resistor and two collector resistors in parallel to  $-V_{CC}$ . The input signal enters through the base of one pnp. The output signal exits at the collector of the other pnp. The base of the second pnp receives the reference voltage.

The reference signal, as it is applied, forces the second pnp to either cut off or saturate. When the pnp is cutoff, the collector output is the same as  $-V_{CC}$ . If the pnp is saturating, the collector voltage rises. Therefore, the circuit can attain two stable states.

The input voltage can change the total comparator voltage state if it is changed. If a low is at the input pin, the comparator will exhibit one state. A high at the input changes the comparator state to the other stable state. In effect, the comparator

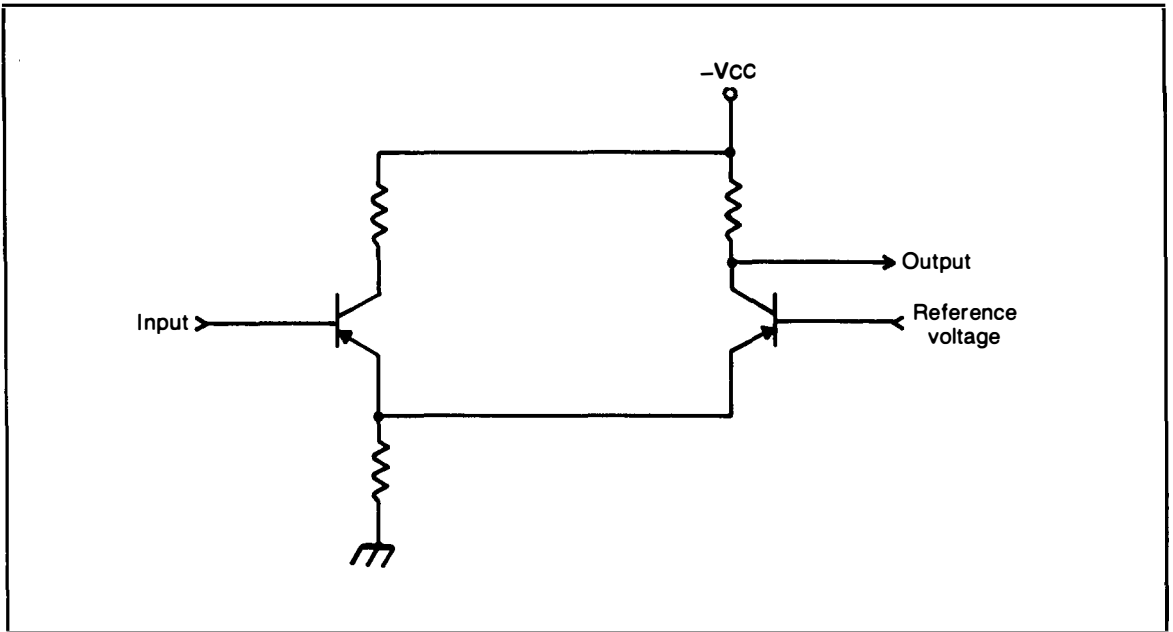


Fig. 11-8. The voltage comparator circuit is often found in computers. It acts somewhat like a flip-flop except that it changes states according to the control of a reference voltage.

can output a high or a low in response to a high or low at the input.

The output of the comparator controls the state of the flip-flop. For example, in the 556 chip that controls reset, the trigger is tied to +5 volts. When the computer is first turned on, the supply voltage goes from zero volts to +5 volts. This rise to +5 volts triggers the comparator which in turn sets the flip-flop. The output of the flip-flop is a high. The high is changed to a low as it is passed through the NOT gate, a section of the 7406. This low pulse is transferred to the \*RES pin of the 6510. This action outputs the starting address of the reset routine. That is how the computer powers up and eventually gets the READY sign and the cursor on the screen.

The other part of the 556 performs a similar job with the \*RESTORE signal. When you press Restore on the keyboard, the resultant pulse is sent to the trigger of the 556 section. This produces the output signal \*NMI which goes to pin 4 of the 6510 to produce the \*NMI effect. Both of these circuits are operating in a one-shot mode.

## COMPUTING REGISTERS

When you get right down to it, the digital insides of one computer is quite like any other. The Commodore 64 is different in certain ways, but it computes just like any other brand. Computing is performed in registers. These are the registers in the MPU, memory, and I/O chips. The support chips that we have just discussed to do jobs like latching, buffering, decoding, multiplexing, and other duties to help move the binary highs and lows from place to place in the proper form at the right times. These same types of support circuits are also found on the main LSI chips to help move the highs and lows from place to place on the chips. The computing itself, though, takes place in the registers of the large, important chips.

What computing boils down to is the manipulation of numbers. All the calculating, all the communications work, the composition of graphics, and the other tasks that the computer is able to do is nothing more than the manipulation of numbers. The numbers are clever codes of the letters and characters but they are still numbers. The numbers

are represented in the registers as highs and lows, but the logic states are numbers nevertheless.

The registers can perform a limited group of manipulations on the highs and lows. You have already seen how a register of four bits can add bit after bit, from a start of L-L-L-L to a total of H-H-H-H. This is one of the jobs a register can do. The other jobs are equally easy to comprehend. The registers are not capable of doing any truly complex things. To itemize, registers can add or subtract, shift or rotate, increment or decrement, clear, complement, AND, OR, or XOR.

Besides being able to perform these jobs in the registers, they are built to be able to move the contents of the registers from one register to another. The contents of MPU registers can be stored in RAM registers, the contents of RAM or ROM registers can be loaded back to the MPU and data contents of the 6510 registers can be stored, loaded or swapped among each other. In addition, 6510 register contents can be sent to I/O chips for transfer to peripherals, or peripheral contents can be sent to the 6510 registers via the I/O chips. During servicing it is best to keep in mind that the numbers being manipulated or moved are voltage highs and lows.

During the manipulating and moving of the numbers, the question comes up, where should this data be moved to? The register that answers this question is the program counter. It is a special reg-

ister that does the addressing. It is examined in detail in the next Chapter 12.

## Shifting

An important ability of a register is shifting. What is shifting? It is the talent a register has of being able to shift bits from side to side. For example, suppose an eight bit register you are working with contains LLHL LHLH. If you instruct the register to "shift the contents one bit to the left," the register will go to a state of LHLL HLHL. The contents of the register moved one bit to the left. If you now told the register to move one bit to the right, the register would then change back to its original state, LLHL LHLH. That is shifting. Another form of shifting is called rotating. Shift and Rotate are important instructions that the MPU and other chips with shift registers can respond to.

Shift registers are not new. A shift register is formed by adding some storage flip-flops to an ordinary register as shown in Fig. 11-9. If an 8-bit register is to be able to shift, it needs seven additional flip-flops. The seven FFs are installed between the eight individual register bits. They will latch the individual bits as they are transferred during the shift. For example, the register contents from above LLHL LHLH, could be shifted one bit to the left in the following way.

First a copy of each H bit is placed into the latch

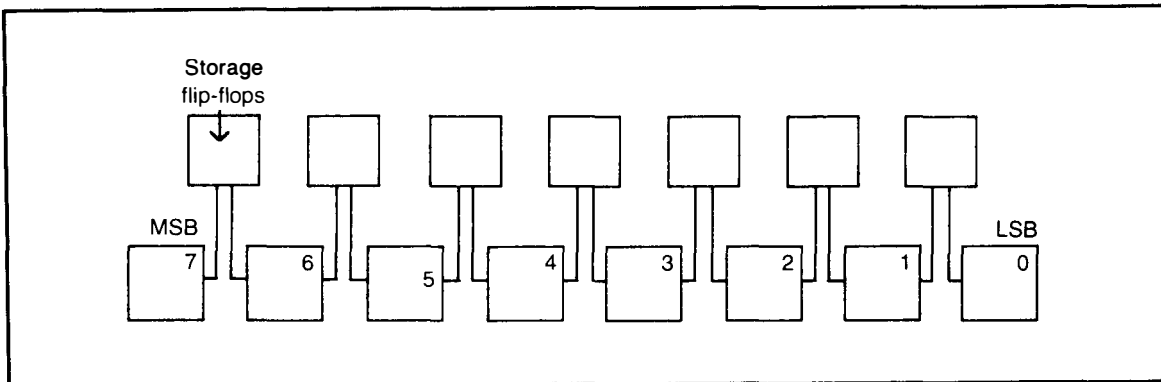


Fig. 11-9. The shift register needs additional storage flip-flops between register bits. First, the states that are in the bit holders are transferred to the storage flip-flops. Next the main register bits are all cleared. Lastly, the stored bits are returned to the main register but shifted one bit to the right or left, whichever is desired.

to the left of the register bit. Next, all the register bits are replaced with Ls. The last step is to move the Hs out of the latching FFs into the next bit to the left. The one bit shift left is complete.

The shifting and rotating of bits in a register is a valuable talent. The 6510 responds to machine-language instructions such as Shift Left, Rotate Right, etc. Upon receipt of such an instruction the register performs the shift. Between the register bits and the latches installed between bits, the highs and lows are shifted. What does the shift do numberwise?

Let's think of the Ls and Hs as binary 0s and 1s. Suppose the shift register is filled with 0000 0001. In decimal, this is the number 1. The register is then given the instruction "shift one bit to the left." It does so and the register then reads 0000 0010. The 1 moved to the left and the value of the register in decimal is now 2. You are probably thinking, why this is exactly like the binary adder. No its not. Even though the adder, after an addition of 1, totaled 2, this shift register has not added but doubled its decimal value.

This becomes clearer as we instruct the register to shift another bit to the left. The register then shows 0000 0100. This is the decimal value of 4. The register has doubled again. The adder register, after another add would be three. Should we shift left once more the register goes to 0000 1000, which is decimal 8. The shift doubles the value of the register when it goes one bit to the left. This fact becomes very valuable when you are manipulating numbers as you compute.

If the shift left produces a doubling of numbers, what happens to the numbers during a shift right? The value of the register is divided in half as the register is made to shift right. One move to a higher significant bit multiplies the register value by two. One move to a lower significant value divides the register value by 2. This is one important function of a register during the task of number crunching.

The shift register has other jobs it can do. It aids during the transferring of numbers from place to place. As data is transferred from register to register it is coded, decoded, latched, buffered, multiplexed, and changed in numerous other ways

to get the data to the right place, in the right form, at precisely the correct time. Sometimes a byte of data is transferred in parallel fashion, all bits moving over a byte-sized bus, eight abreast. Other times the byte is transferred in a serial way, over one wire, one bit at a time, in a long single file column. There also comes a time when the parallel group of bits must leave the eight lines in the bus and enter one wire in single file.

Sometimes a serial signal must be converted to enter a parallel bus line. The shift register can do the honors. This use of the shift register is strictly a transfer medium and has no bearing on the numerical value of the bits. There are a number of these types of shift registers in the I/O circuits of the 64. The shift register can receive a serial data input at its lowest significant bit. The bits enter one at a time and are immediately shifted to higher bits. As a full byte of bits enters the byte sized register, the bits fill the register. The serial bits are thus latched into the register. Each bit holder has an output terminal. The outputs are all connected to another latch. This latch is connected to the parallel lines of the data bus. The latch is opened and the bits flow onto the data bus as a parallel signal. Figure 11-10 illustrates this process.

The reverse is also part of the shift registers duties. The parallel data bus is able to latch its contents. The latch is then connected to the shift register. The latch empties into the shift register in a parallel fashion. The shift register can then shift the bits to the right and output them one at a time out of the least significant bit.

## Clearing

An important job that registers can do is CLEAR. The word clear means to reset a register or bit to a low, or numerically a 0. Figure 11-11 illustrates this idea. A register usually must start operating in a clear state of 0.

While the various names for this process all mean the same thing, there are some fine distinctions. The word reset seems awkward when referring to the start of a register. It seems to make more sense if a register is clear when it starts off and at-

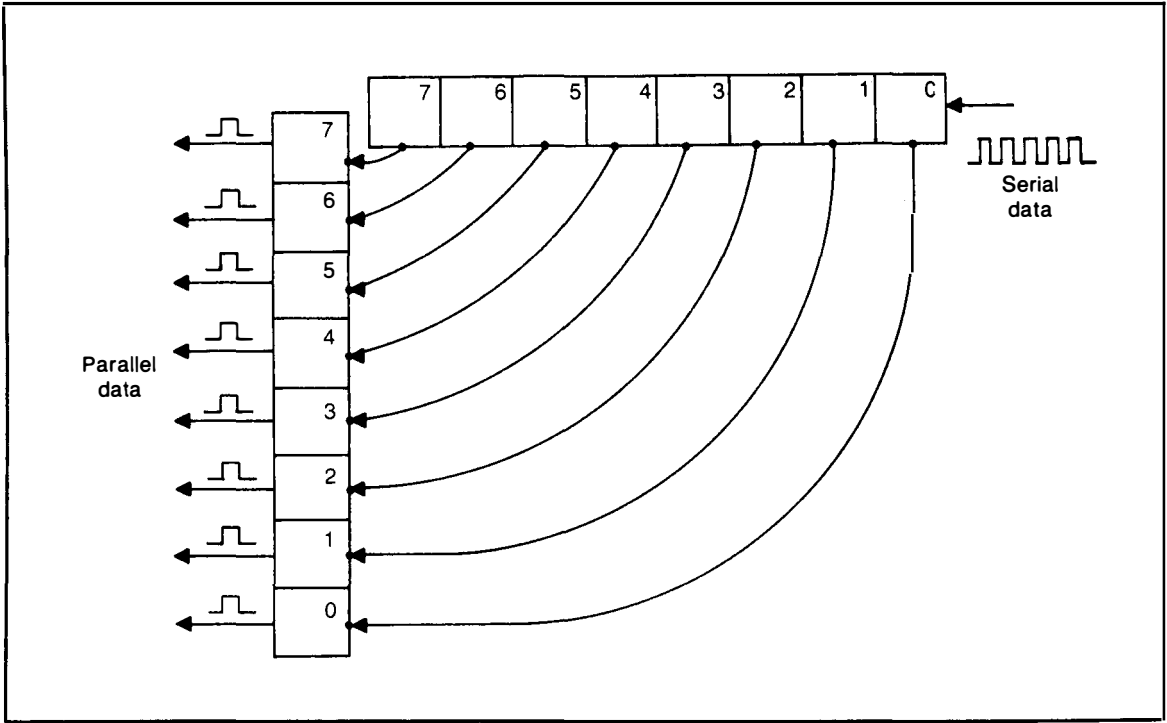


Fig. 11-10. The shift register is handy to convert serial data to parallel, or vice versa.

tains a state of reset the second time is goes to a 0. A register should become set to a 1 before it can then be reset to a 0.

Whatever the variations in definitions, you can

think of the register being clear, as long as it is in a state of 0. When a register is cleared all the bits are reset to 0. There are many electronic ways a register can be zeroed. You can add the correct

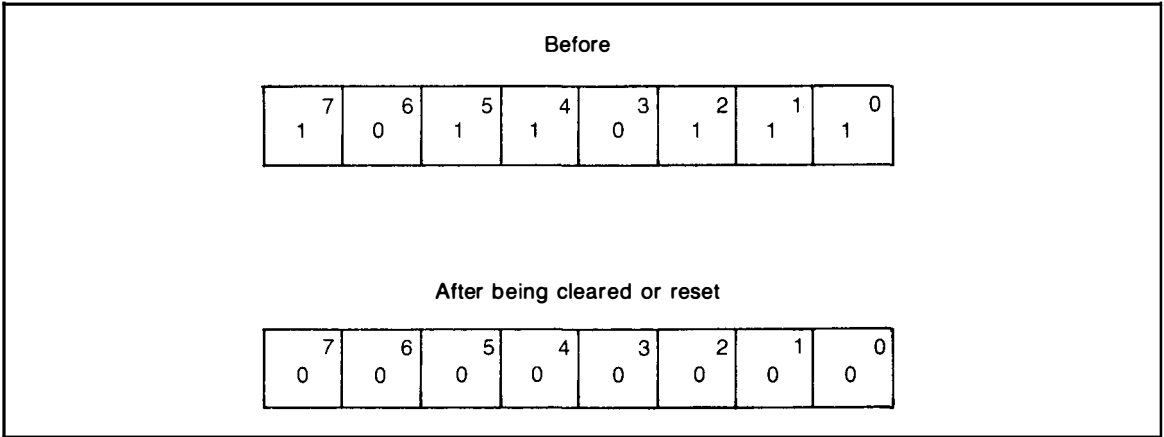


Fig. 11-11. When a register is instructed to clear or reset, all the bits are replaced with 0s, no matter what was there beforehand.

states to a register and end up with all 0s. The register can be shifted into a complete 0 state. The register can have 0s POKEd into it. The register can have AND or OR logic applied to it to produce 0s. The clearing of registers is an important ability and besides programming is used extensively in the factory during quality control testing.

## Complementing

When a register is complemented, all the highs are changed to lows and the lows are changed to highs. This process is shown in Fig. 11-12. This is easy to do electronically. If you transfer a byte of bits from one register to another you can change the state of the individual bits. All you need do is pass each bit through a NOT gate as it moves from register to register. The 1s will change to 0 and the 0s will become 1s.

At first glance, this looks like an interesting but not very useful data manipulation. What value can the complement maneuver be? As it works out, in binary arithmetic, complementing is an important function.

The circuits needed to add binary numbers are relatively simple. We've explored the binary counter circuit somewhat. With that circuit it is easy to add numbers. It is also easy and straightforward to add one register to another. Simple multiplica-

tion can also be performed since multiplication is only the addition of a lot of the same numbers. The lightning fast registers perform multiplication by simply adding the group of same numbers together. It doesn't need a multiplication table like a human does. To get  $4 \times 6$ , the registers just add  $4 + 4 + 4 + 4 + 4 + 4$ .

While addition and multiplication lend themselves to FF registers, subtraction does not. In order to design circuits that will subtract quickly, expensive and difficult designs have to be manufactured. There is an easier way to get the computer to subtract. It is called *subtraction by addition of the complements*. This is tricky, but once understood, simple.

It works out, and you'll have to take my word for it, that if you take the complement of a binary number, and add it in a certain way to another binary number, the result will be the same as if you had subtracted the original number. That is how the 6510 in the 64 does subtraction.

I'm not going into a proof; there are plenty of books around on the subject if you are curious. The knowledge of the subtraction method is not vital to the repair of the 64. For your information though, the 64 performs the subtraction with the two's complement.

When you change the 1s to 0s and the 0s to 1s

Before

7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	1

After being complemented

7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0

Fig. 11-12. If a register is instructed to complement, all 0s are changed to 1s and all 1s to 0s.



in a register, it is called one's complement. In order to obtain the two's complement, you add a 1 to the register contents. That makes the register contain the two's complement. In order to subtract the original register number, the two's complement number is added. The result of this addition is the same as subtracting the original numbers.

In case you are curious about complex multiplication and division of numbers, microcomputers usually have special subroutines installed on their ROM chips. These routines are called everytime a long multiplication or division problem has to be solved. This has nothing to do with complementing. The important use of complementing is subtraction. There are other things a programmer might do. These are of little interest during troubleshooting or repair.

### **Increment, Decrement, and Jump**

The Commodore 64 has a number of registers that are continually counting up, counting down, or just jumping from one binary number to another. For example, the 16-bit Program Counter is constantly doing this. The PC always contains a binary number. The number is the next address in the memory map that will be contacted. When the number advances by one, it is said to be *incremented* by one. If the number is reduced by one, it is *decremented* by one. Should the number just change to a number that is far from the previous number, it is making a *jump*.

These three register abilities are vital to the operation of the PC. The PC is attached to the 16 address bus lines. The 6510 gets connected to whatever number the PC puts on the address bus.

The PC is built to automatically increment by one after every address cycle while running a program. It does the incrementing continually unless it is instructed by the program to do otherwise. The incrementing is simply adding a 1 to the least significant bit of the register. If there is a low in the LSB it is replaced by a high. Should there be a high in the LSB, the high is transferred to the next higher bit and the LSB ends up with a low. Refer to Fig. 11-13. As each new high is applied to the LSB the total binary number is incremented by 1.

Note that this process is addition in the register and not shifting, which continually doubles the total with each shift. The numbers increase one at a time as the addition carries from bit to higher bit.

Decrementing is the reverse of incrementing. During the decrement job, the total binary number in a register is reduced by 1. For instance, in the 6510 there are two registers called the X and Y index registers. One of the techniques of indexing has to do with decrementing a register after each addressing cycle. As a 1 is removed the entire register binary number is reduced by 1.

Incrementing and decrementing registers by 1 is a common practice during computing. In addition, registers can be changed drastically. For example, the program counter can receive a JUMP instruction followed by a large number. When the JUMP arrives, the number could be added to the present address on the PC. A new larger number is then produced on the PC. The new number is placed on the address bus. The addressing then jumps over a group of numbers and lands on the new number placed on the bus. This is a very important function of the program counter register.

### **ANDing and ORing**

AND and OR gates have been discussed. With the aid of AND or OR gates, registers can also perform the AND and OR jobs. As Fig. 11-14 shows, if you connect the outputs of a byte sized memory location and a 6510 register to eight AND gates, the outputs of the two registers can be ANDed together. The outputs of the AND gates could then be placed into a third register for storage. The third register would contain the results of the ANDed registers.

The ORing of two registers can be accomplished in the same way. The same position bits from two registers are injected into OR gates and the outputs of the gates placed into a third register. XORing two registers can also be easily performed with this technique. ANDing, ORing, and XORing are all easily done between memory locations and the accumulator register in the 6510. The XORing is called EOR by Commodore.

Program counter  
incrementing by 1

First address  
↓ LSB

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2nd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3rd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4th	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
5th	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6th	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
7th	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
8th	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Next addresses ↑

Fig. 11-3. A register like the program counter is able to increment automatically. Other registers need increment or decrement instructions to effect the change.

### ANDing two registers

CPU  
register

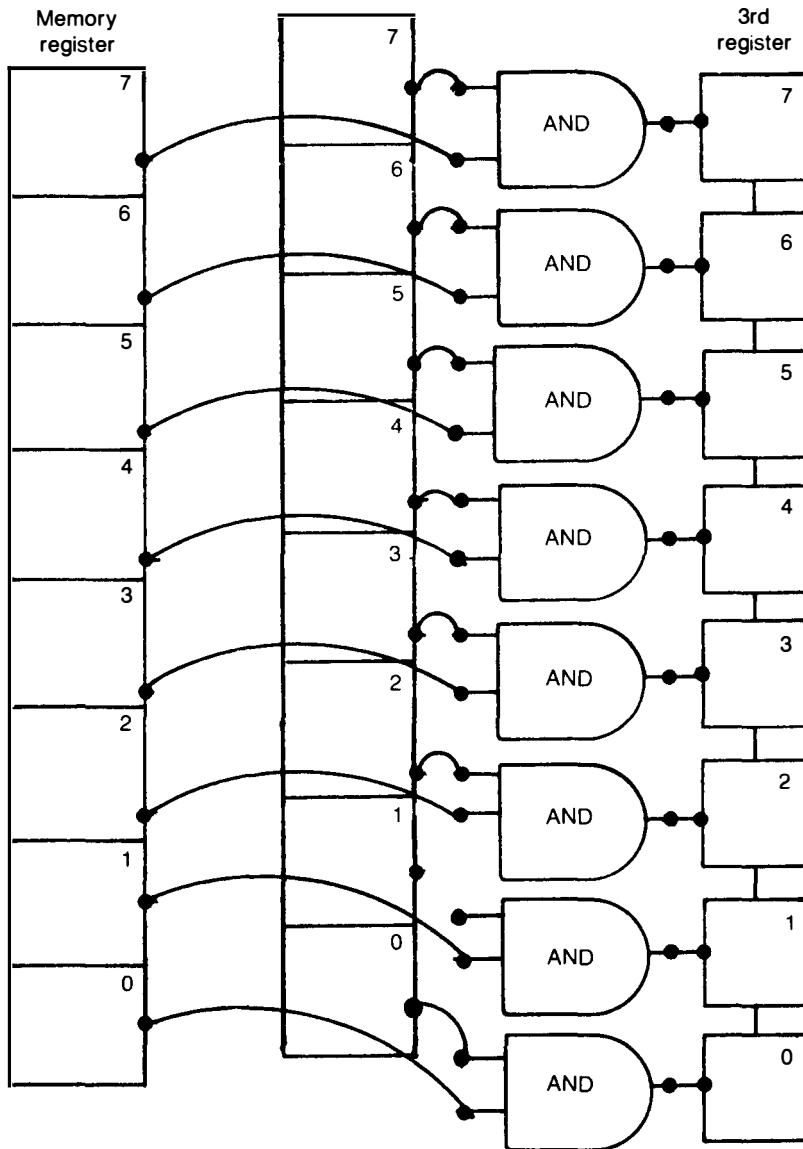


Fig. 11-14. When a complete register is ANDed with a second register bit by bit, the output results are stored in a third register.

There will be a lot more about registers in the following discussions of the details on the main LSI chips. However, you'll find that there isn't really too much variety in what FF registers are doing. They only do a few jobs. They all can store, latch, and pass on binary data. Then some of them, but

not all are able to increment, decrement, clear, complement, shift, and rotate. Lastly, a chosen few are able to add, subtract, multiply, divide, AND, OR, and XOR. That's about all registers can do. That's all they have to do to get the computing done.



## 12. 6510 Microprocessor

**T**HE 6510 IS A DESCENDENT OF THE 6502 MPU that is found in the VIC 20, also by Commodore. The 6510 has one major difference. Otherwise it is almost electrically identical to the 6502. The 6510 has a special 8-bit bidirectional I/O port in its system. This port has bits that are used to manage the memory in the machine. These bits will be discussed in detail in Chapter 13, Memory Maps.

The first thing that strikes your eye when you look at the inside of the 6510 is the 16 address lines emerging from 16 three-state address buffers. Refer to Fig. 12-1. If you are testing the machine with a logic probe, you'd touch down on these terminals first to see if the 6510 is outputting a stream of data that is doing the addressing. If the 6510 is addressing, the logic probe will read PULSE on all 16 pins. If there is no pulsing the 6510 is not addressing.

Figure 12-1 shows that the three-state buffers are controlled by a pulse from the VIC II chip (AEC). The address enable control line is able to thus turn the addressing of the 6510 off when VIC wants to use the address bus. AEC is held high when the buffers are working ok. If AEC goes low

then the 6510 three-state buffers will get turned off. If you find that the address lines are not pulsing, make sure they are not turned off by AEC.

### ADDRESSING

The block diagram shows the 16 buffers are fed from two eight bit internal bus lines: address bus high and address bus low. The high and low lines correspond with the output connections. A15-A8 are the high lines and A7-A0 the low. These designations follow true to form back to the innards of the chip. Deep inside are the two sections of the program counter: program counter high and program counter low.

Also deep in the chip are the other registers. Figure 12-2 shows how these registers interact. The stack pointer register connects to the L part of the addressing system. The ALU connects to both the L and the H pins. The input data latch also connects to both L and H.

When you touch down on the addressing pins, the address bits are coming from one of these four registers. The most prominent register is the 16 bits

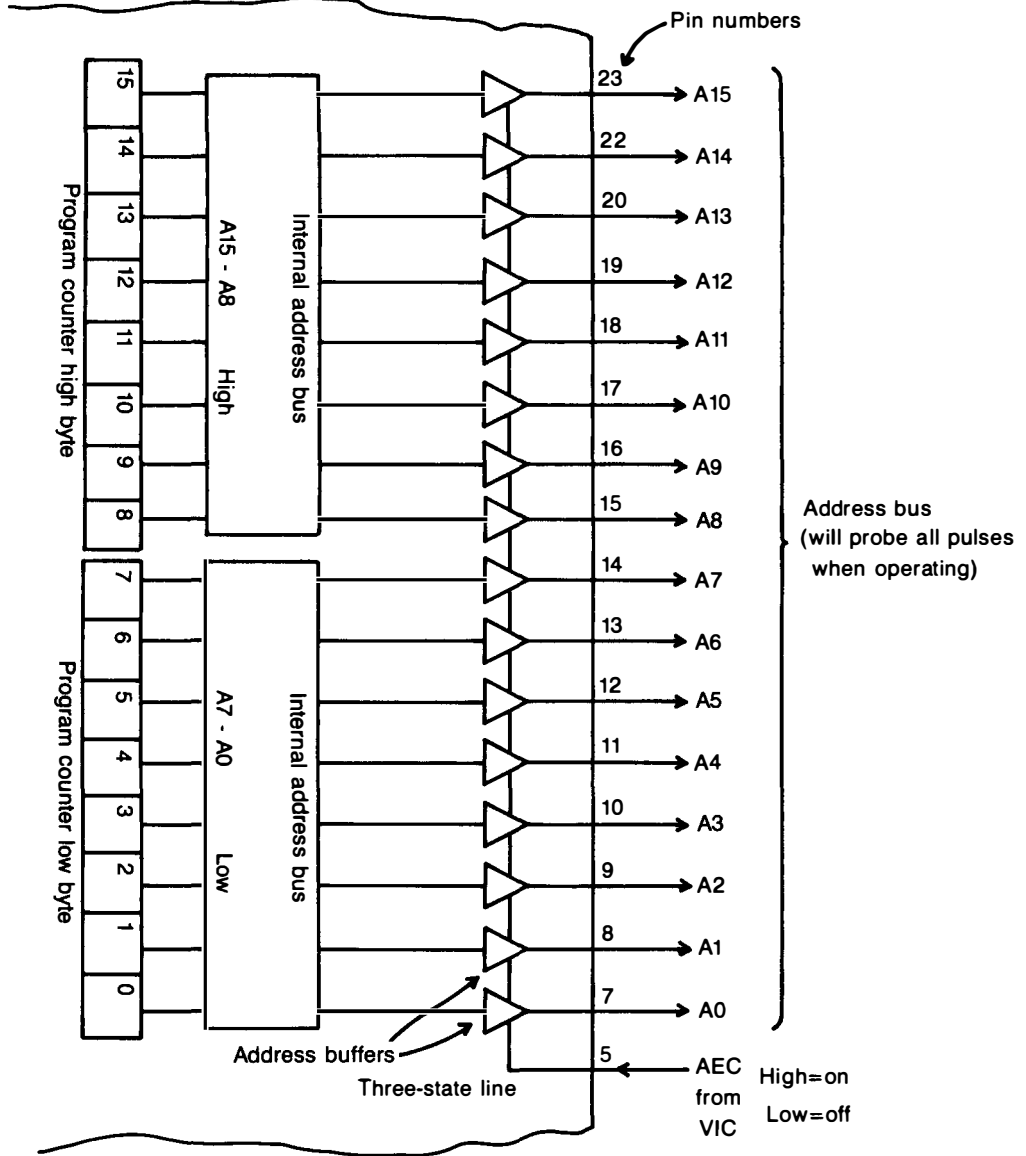


Fig. 12-1. The 6510 has a set of three-state buffers in its output that can be turned off and on with an AEC from the VIC.

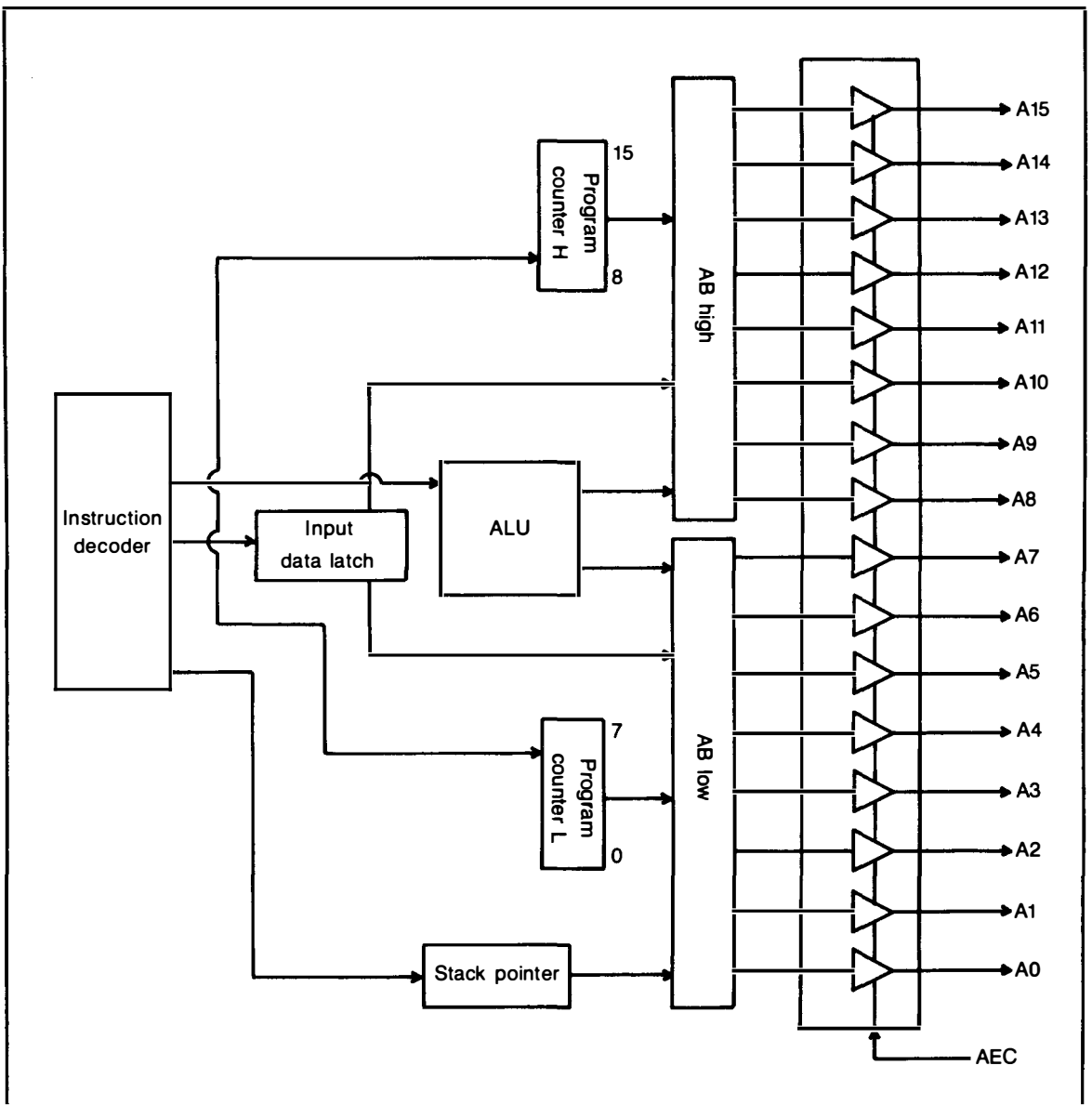


Fig. 12-2. The stack pointer is able to form addresses from 256 to 511 only. It is attached to the lower address bits and not to the upper.

of the program counter. It can address the entire 64K memory map because it has 16 pins to form bit combinations.

The ALU and also the data latch, DL, can address the entire memory map with their outputs. The stack pointer though is only attached to the

lower address lines, A7-A0. It can only point to 256 memory locations. It addresses with the lower address lines. What does all of this mean?

## STACK

Computers have what is known as a stack. The

6510 is no exception. In the 6510, there is a register that has one job. It is charged with the responsibility of always containing the first address of the stack. In this computer, the stack is 256 bytes long. The stack is installed at decimal locations 256 through 511. This is the 16 bit addresses, 0000 0001 0000 0000 through 0000 0001 1111 1111. Note that the higher eight bits are 0000 0001. They are always the same throughout the 256 addresses of the stack. The lower eight bits range from 0000 0000 to 1111 1111. Figure 12-3 illustrates the memory organization of the stack.

The stack never changes its location. 0000 0001 is the binary address of Page 1 of the memory map. There are 256 pages in the 64K memory map. On each page there are 256 locations, like byte sized words on a page. There will be more about the pages in the next chapter.

The stack pointer always holds one of the 256 addresses of the stack. If the stack is empty the register points at address 1111 1111. If the 6510 pushes a byte into the stack, the byte enters location 1111 1111 and the pointer decrements to the next empty lower number, 1111 1110. The pointer keeps decrementing as the stack fills up. The stack can store a page of bytes.

If the stack is near full, the pointer could read 0000 0100. Should the 6510 pull a byte off the stack, the pointer would then increment and point to the next higher empty byte holder, 0000 0101. What sort of data does the stack contain?

The stack is mostly used by the programmer and the program to hold the states of registers in the MPU. For example, a program might be running and suddenly an interrupt occurs. The interrupt could be a peripheral device that needs immediate attention from the 6510. It has priority over the program being run.

The MPU is forced to stop at the next convenient moment. Meanwhile, all the registers in the 6510 are in the middle of a lot of data manipulation. If the MPU just stops to service the peripheral interrupt, what happens to the data in the registers? It could all be lost as the 6510 processes the peripheral data.

As you guessed, all the data in the pertinent

registers are pushed onto the stack before the MPU deals with the interrupt. The register bytes are stored in the stack temporarily. First the contents of the 16-bit program counter are pushed onto two stack locations. Then the contents of the X and Y Index registers are pushed on two more locations. Following that, the one byte in the accumulator is pushed on. Lastly the eight bits in the condition code register are placed into one byte. When the registers are stacked, the 6510 then services the interrupt.

At the end of the interrupt routine, the MPU goes back to the program that was in progress when the interrupt arrived. The MPU now must restore the registers to the states they had been in. The 6510 now starts pulling the bytes out of the stack. The pointer is addressing the last byte to go in, the Condition Code register. The CC state, which was the last byte in becomes the first byte out. Then the rest of the bytes are pulled in the reverse order that they entered. The pointer ends up at the original location it had been addressing.

The bits in the stack pointer were never pushed onto the stack because they aren't affected by the interrupt. They remain in the MPU to remember where the bytes were stacked during the interrupt.

## ARITHMETIC LOGIC UNIT

The 6510 is the result of at least 30 years of evolution. There is nothing dramatically new about the MPU circuits, except for the fact they are now microscopic. Essentially the same electronic manipulations have been going on for years in vacuum tubes, transistors, and then integrated circuits. The ALU, the arithmetic logic unit, is not a new innovation. It is as old as computers. It is also the most vital section of a computing system. It does the data manipulation.

A typical ALU has two byte sized inputs and one 8-bit output, as Fig. 12-4 shows. The ALU is able to do the following things as it crunches the two inputs into a single output: it adds the two inputs to result in one output; it uses its ability to add to produce subtraction, multiplication, and division; it can complement its internal registers and make



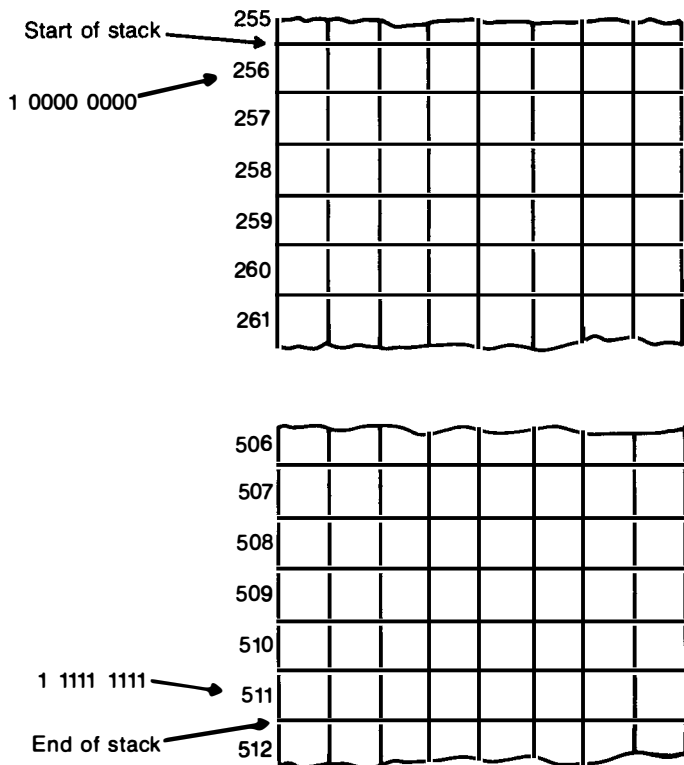
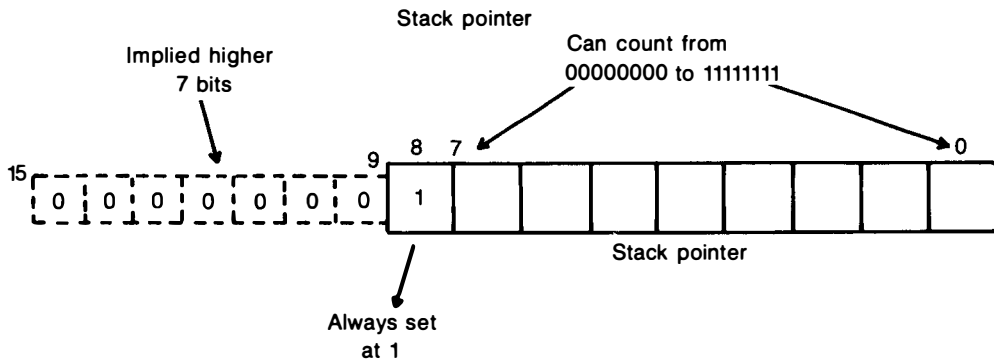


Fig. 12-3. The stack pointer register is nine bits wide. The upper seven bits of an address are implied to be 0000000 and the eighth is fixed at a 1. The stack is assigned to reside in RAM at addresses 256 to 511.

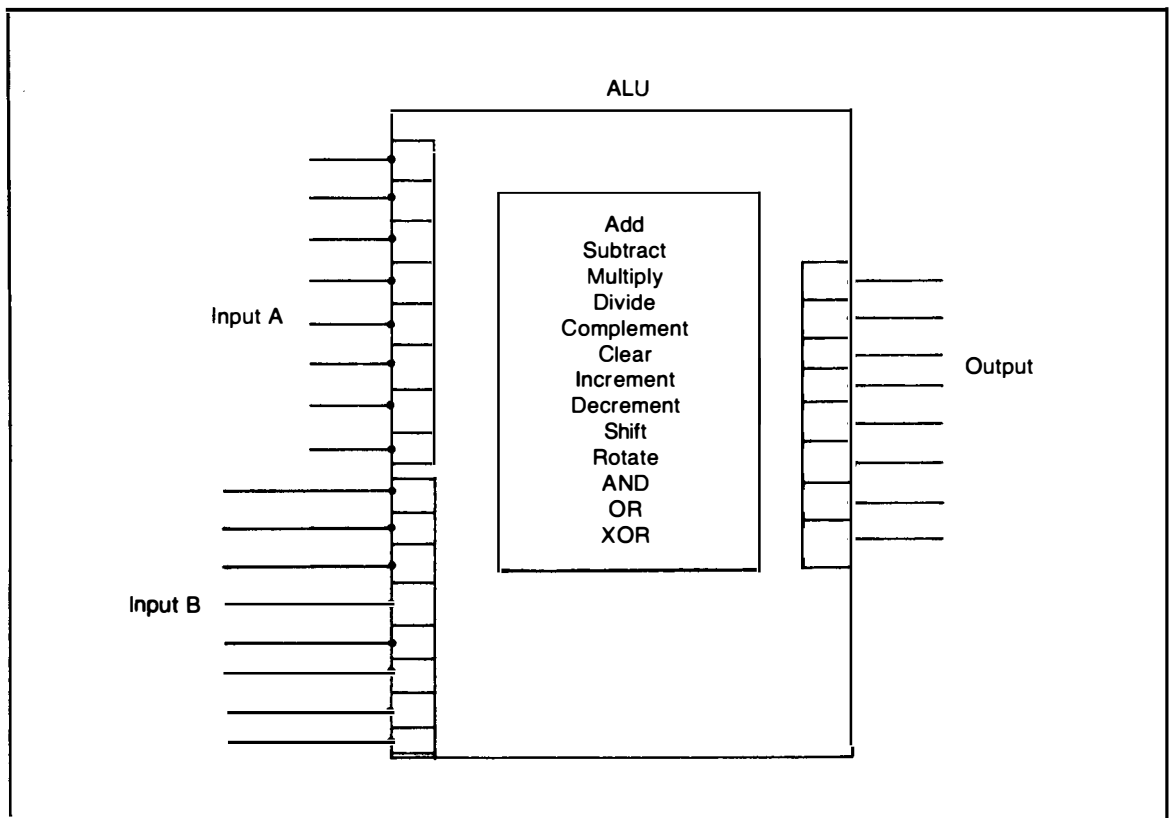


Fig. 12-4. The typical ALU has two byte-sized inputs and one 8-bit output. It is able to manipulate the inputs in the 13 listed ways.

use of math programs contained in ROM; it is able to clear one of the inputs; it is able to increment or decrement one of the input bytes by 1; it can become a shift register and shift all the bits in one of the inputs either to the left or right; it can perform logic manipulations.

## Shifting

When a shift left is made the value of the register is doubled. If a shift right is made the value in the register is divided by two. When the shift right is made the LS bit falls out of the register into a special bit holder and a 0 is forced into the MS bit. During a shift left the MS bit falls into the bit holder, and a 0 is placed into the LS bit. Refer to Fig. 12-5.

The shift ability has a variation called rotate.

Rotating works in the same fashion as shifting except for one major difference. As Fig. 12-6 shows, when you shift a byte, one of the end bits falls out into the bit bucket c. A shift left loses the MS bit. A shift right caused the LS bit to fall out of the register. With the rotate function, the lost bit does not leave the area. It is caught in a bit holder. The contents of the bit bucket is brought around to the other end of the register and installed there. If you rotate right, the lost LS bit is also caught and the contents of c is brought around to become the new MS bit.

## Logic Manipulations

The ALU is naturally logical. It contains, besides registers, AND, OR, and XOR gates. It can perform these logic manipulations along with its mathematical jobs. The complementing men-

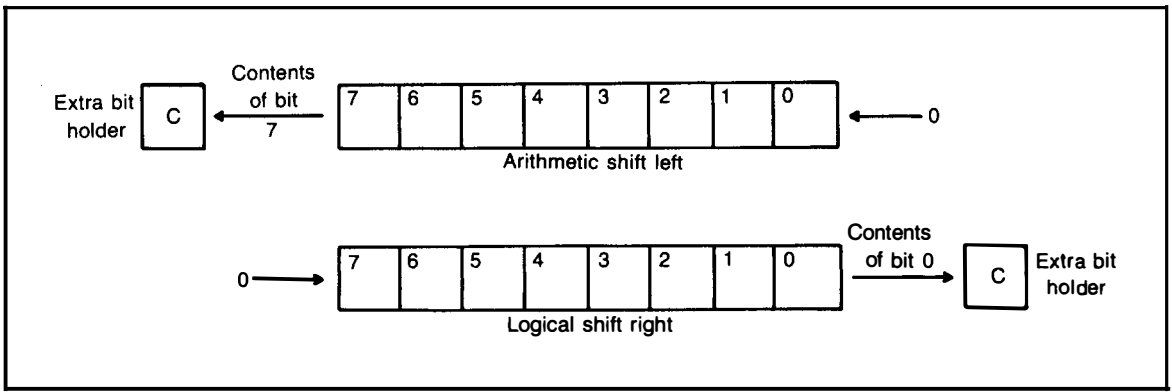


Fig. 12-5. During an Arithmetic Shift Right, a 0 is pushed into bit 0, and the contents of bit 7 fall into the C flag holder. During a Logical Shift Left, a 0 is pushed into bit 7, and the contents of bit 0 fall into the C flag holder.

tioned earlier is actually a logic type job. When a register is complemented it has all its bits run through NOT gates.

The two inputs of the ALU can be ANDed together. There is one AND gate for every bit position in the two inputs. For example, the two MS bit positions, bits 7, can be the inputs to one AND gate. The resultant AND output becomes the output bit 7. Each set of bits can be ANDed in the same way. The AND operation proceeds simultaneous-

ly in parallel. They form a total AND byte.

When you AND a byte with another byte, all the bit sets that had 0s will output a 0. All the bit sets that had two 1s will output a 1. ANDing a byte allows you to *mask* a byte. If you want to make sure that certain bit positions will end up with 0's, you AND those bits with 0's. This procedure, shown in Fig. 12-7, is known as masking off certain bit positions.

On the other hand if you want bit positions to

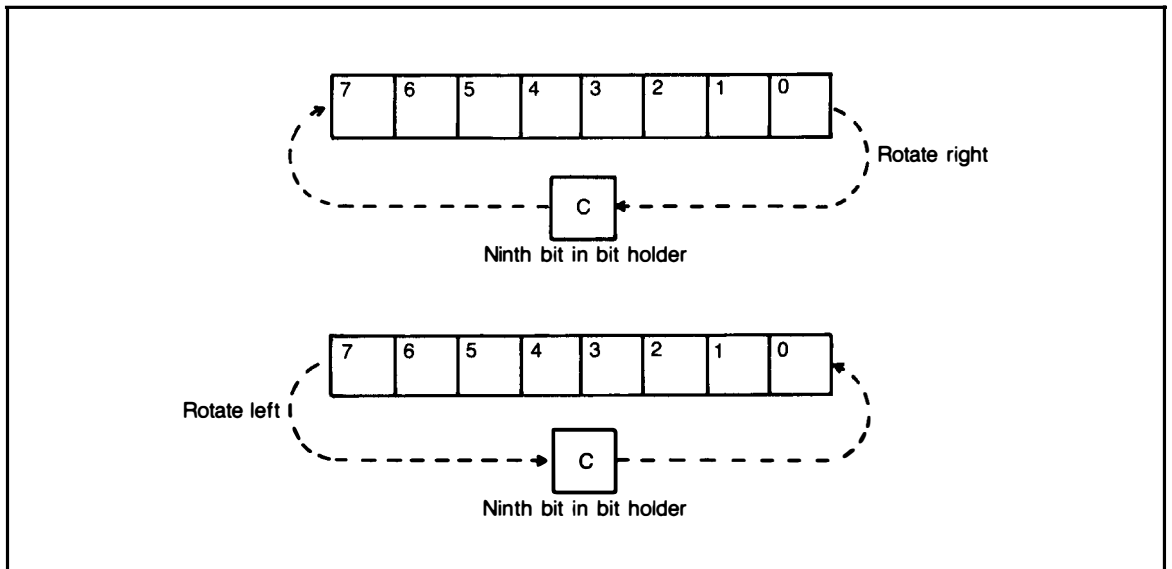


Fig. 12-6. During Rotate instructions, the loose bit falls into the C flag, but the contents of the C flag are pushed out and into the bit that was left without contents. The C flag connects the register into a closed ring and acts as the ninth bit.

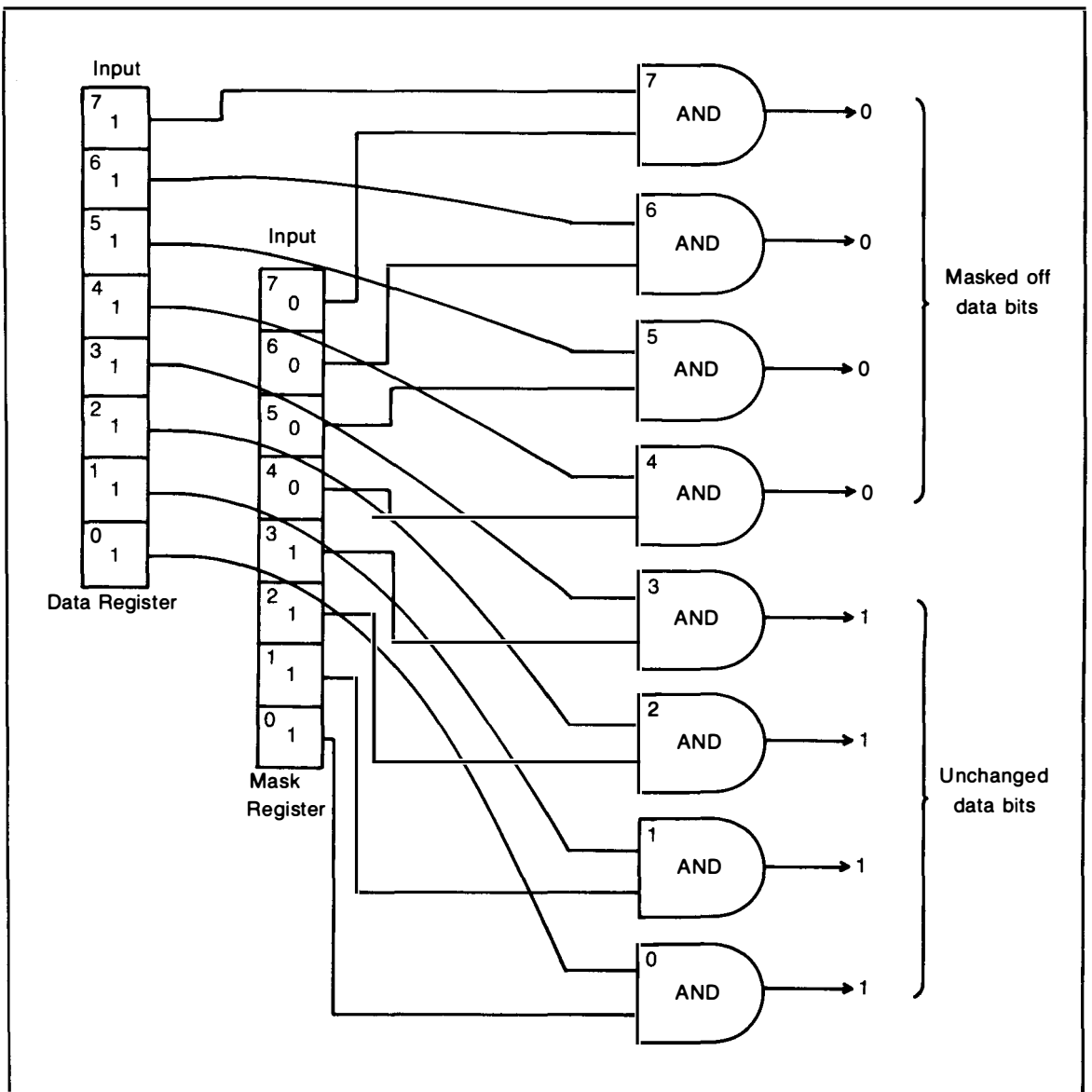


Fig. 12-7. Register bits can be masked off to 0s by ANDing them with 0s. The bits ANDed with 1s will remain unchanged.

remain unchanged, they are to be ANDed with 1s. These bits are thus not masked and will be the same before and after ANDing.

The ALU ORs two input bytes in the same way. Whenever one of two inputs into an OR gate is a 1, the output is a 1. The only way two ORed bits can produce a 0 is if both inputs are 0s. The

total result of one byte being ORed with another byte is an ORed output byte.

When it is necessary to change certain bits in one of the ALU inputs to 1s, all you have to do is OR the chosen bits with 1s. Fig. 12-8 demonstrates the process. Whether a bit is a 0 or a 1, when it is ORed with a 1 it can be outputted as a 1. The

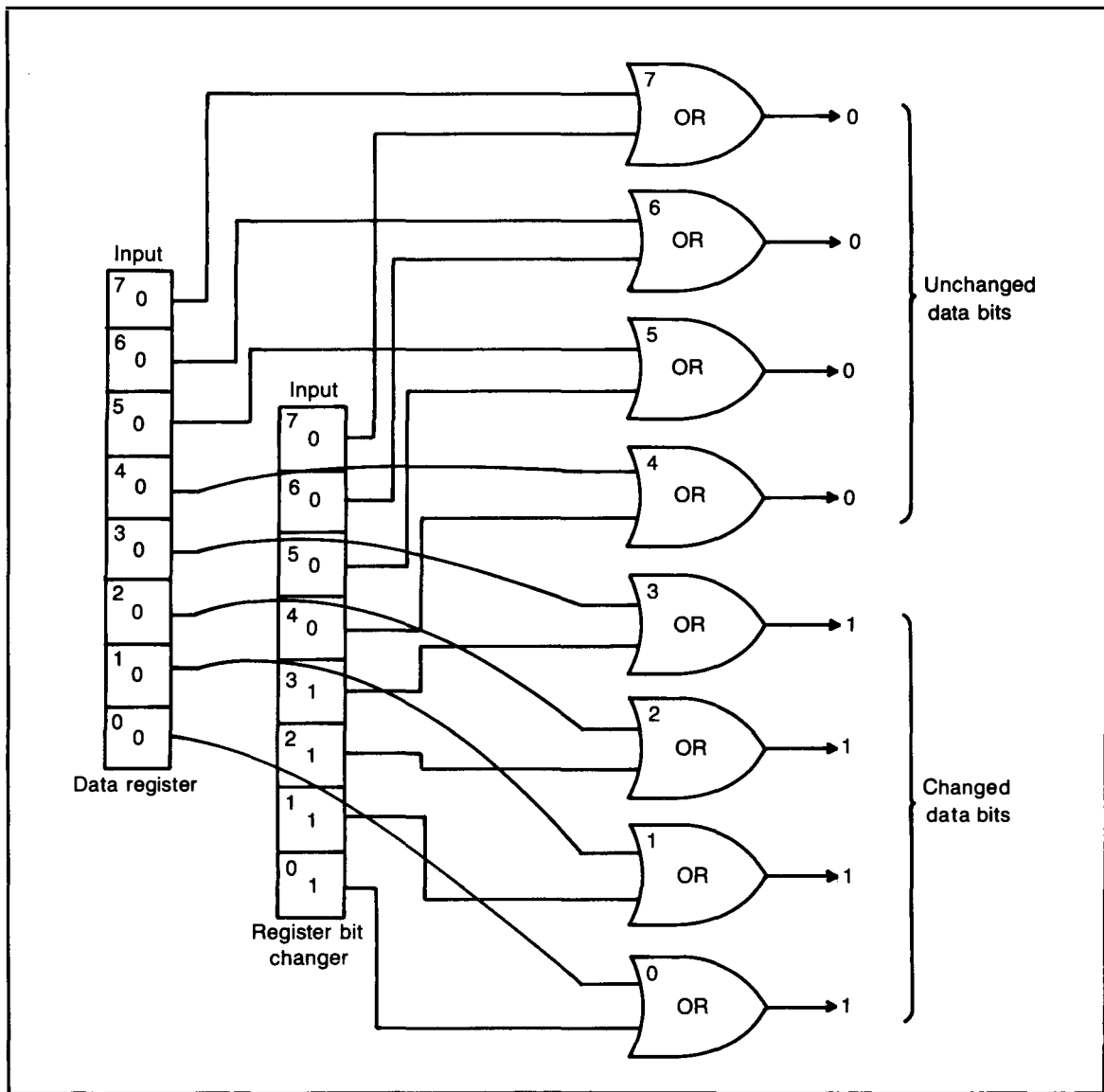


Fig. 12-8. Register bits can be changed to 1s by ORing them with 1s. Those ORed with 0s will remain unchanged.

bits that are ORed with a 0 will not have any changes. The 1s will remain 1s and the 0s will remain 0s.

The ALU is also able to XOR, or as Commodore calls it, EOR two input bytes. When two input bits are the same, either a pair of 1s or 0s, the output bit will be 0. If the two bits are different, a 1 and a 0, in either way, the output bit will be

1. The total result of a byte being EORED with another byte is a total EORED byte. The EOR operation is used to detect errors and correct code. This is mostly the programmers province although it could be used in diagnostic programs.

## ACCUMULATOR

The accumulator is called a scratch pad. This name

came about because it is a holding place for the numbers that go in and out of the computer. The accumulator is an 8-bit register that is wired to the ALU. The programmer doesn't think very much

about the ALU. He is concerned with the accumulator. To the programmer, the ALU is just a calculator the accumulator uses. The connections are both eight bit bus connections.

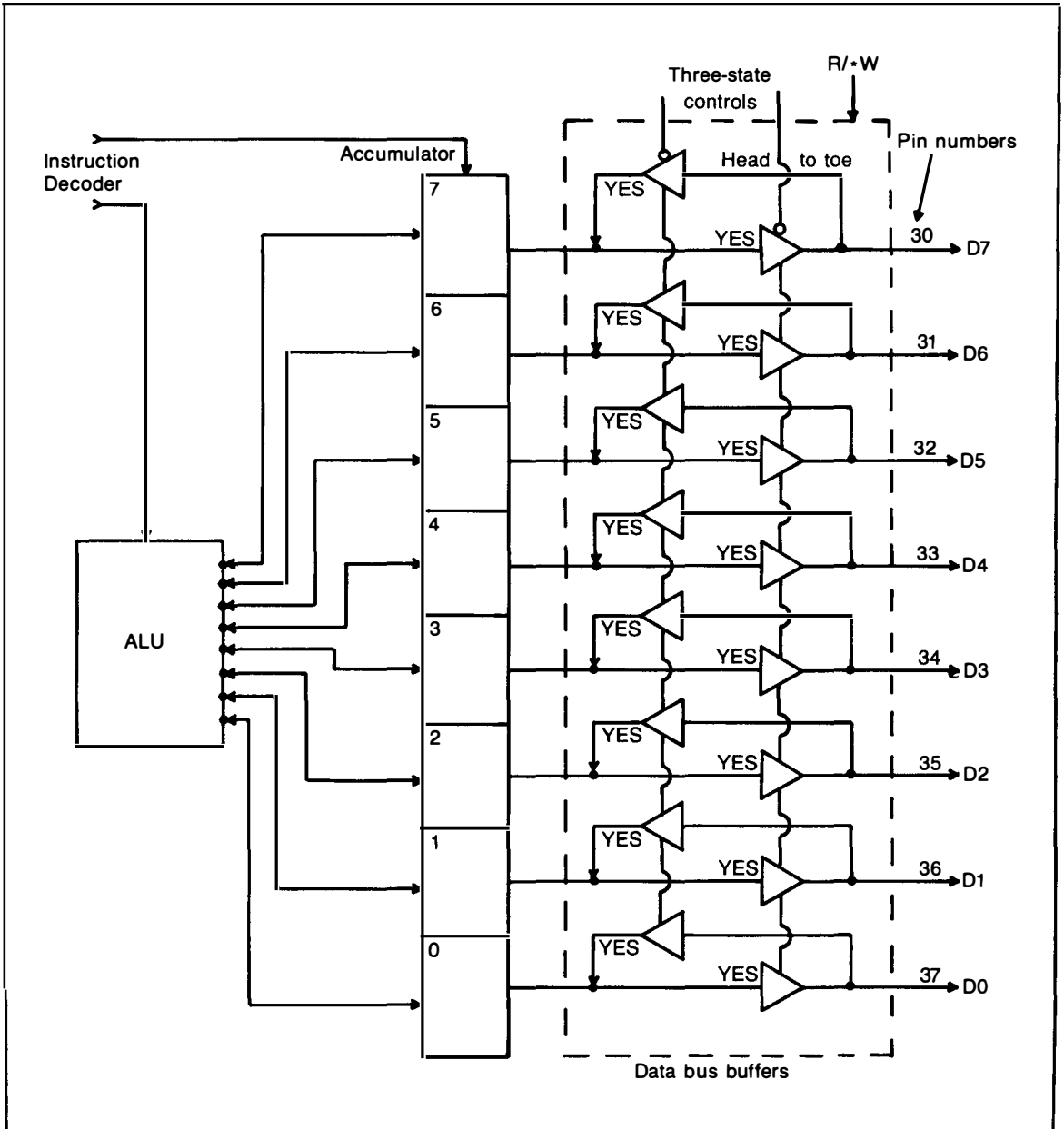


Fig. 12-9. The accumulator outputs to the data bus. Inbetween are a set of buffers that decides whether traffic should flow and also which way the traffic should go.

The accumulator is thought of as a versatile register. This is because it can use the ALU as a calculator. The accumulator with the help of the ALU is able to do all the jobs we've listed.

The accumulator is wired, through eight lines, to the internal data bus of the 6510. The internal data bus leads to the data bus buffers and then out of pins D7-D0. The connection scheme is shown in Fig. 12-9. When you are probing the data pins of the 6510 you are in effect examining the accumulator. The logic probe should show pulses on all eight pins just like the address bus readings.

The accumulator can either receive data from the data bus or output data to the bus lines. The data bus buffers decide which way traffic should flow. Inside the microscopic data bus buffer stage are 16 YES gates. The gates are three-state types. There are eight gates that are wired to send bits out of the 6510 and eight gates wired to receive bits from the memory map. The wiring is "head-to-toe". The 16 gates are all in series with the eight data lines. Two gates to a line, each gate pointing in the opposite direction from the other.

The three-state YES gates are controlled by the R/\*W line. When the line is high, the eight gates that process the read operation are on and the eight gates that allow a write are off. If the line goes low,

the reverse takes place. The eight read gates go off and the write gates turn on. The accumulator accepts data during a read and outputs data if a write is ordered.

## INSTRUCTION SET

All of these wirings, registers, and bit manipulations are all well and good, but what are they doing? The answer is running programs. Computers are there to run programs. It is not the intent of this book to cover programming, but in order to be able to troubleshoot a 64 you have to know the way the bits are moving as a program is run.

In the 64, programs are run in bytes. When you write a program in BASIC, the BASIC ROM changes the program lines you write into bytes of highs and lows. These highs and lows are stored in RAM in sequential addresses. There are other programs stored in ROM. They are also in sequential addresses. The ROMs output highs and lows to the 6510 just like the RAM locations.

### Instruction Byte

The stored bytes of highs and lows form instructions and data. Refer to Fig. 12-10. The instructions dictate what type of manipulation is to

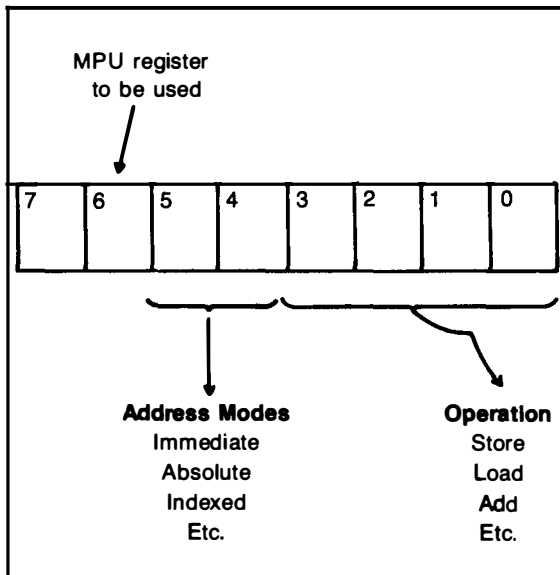


Fig. 12-10. The 6510 uses 8-bit instructions. In a typical instruction, different bits are coded to result in specific jobs to take place. For example, bit 6 could choose the MPU register to be activated. bits 5 and 4 might determine what addressing mode is required. Bits 3, 2, 1, and 0 could specify what operation is to be performed.

be performed on the data. Typical instructions are store, load, add, subtract, complement, clear, increment, shift, AND, and OR. Then there are also instructions that will change addresses such as jump, branch, call, and return. The instructions are all byte sized. Since a byte has 256 possible bit combinations, there are 256 possible instructions that can be written. The 6510 is designed to respond to 56 of the possible bytes.

Each bit in an instruction byte has a job to do. Some bits cause the ALU to perform a job, and other bits give the addressing mode. Still other bits can specify the registers that are to perform the instruction.

## Fetch and Execute

When a program is run, the instruction execution cycle, in general, proceeds in the following way. It conducts three cycles. One is the *fetch*. Two is the *decoding*. Three is the actual *execution*. The operation is loosely named fetch and execute. Figure 12-11 illustrates one cycle.

As the fetch begins, the program counter outputs the 16 address bits onto the address bus. The higher bits select the chip that is addressed and the lower bits address the byte location on the chip. At the same time, the R/\*W line is activated: a high for a read, or a low for a write. The R/\*W line sets up the data bus buffer in the 6510 and the buffers in the memory chips that are selected. Let's use a read signal in this example. We'll fetch an instruction.

Upon receiving the address from the 6510, the memory chip decodes the address it received in its internal decoder stage. The decoder then sends a pulse to the location that the address bits identify. The location is enabled while all the other locations on the chip remain turned off.

The accessing takes a few hundred nanoseconds, according to the access speed of the chip. For example, the 4164 RAM chips are rated as 200 nanosecond types. The eight bits in the location are then placed on the eight lines of the data bus. For instance, the eight 4164 chips are all selected together and each one outputs one bit to its line of the data bus.

The eight highs and lows flash through the data lines to the 6510. They arrive at the instruction register, the IR. The IR is eight bits wide and latches the 8-bit instruction. Once the instruction is latched in the IR, the fetch cycle is complete. A byte of data, rather than an instruction, could have been fetched in the same way. The only difference is that the data would have been sent through the data bus buffer instead of being latched in the IR. The 6510 would have known to do that since an instruction would have preceded the data and arranged things. Figure 12-12 illustrates a data read.

The decoding cycle begins as the IR releases the instruction and feeds the eight bits into the instruction decoder. The decoder is a microscopic PLA something like the 82S110 chip discussed earlier. It takes the bits and produces a set of output bits for use in the other registers of the 6510. The decoder can send one bit to the Y index register, one bit to the X index register, one bit to the stack pointer, one bit to the ALU, one bit to the accumulator, and one bit to the low bits, A7-A0 of the program counter. It also has a one bit connection to the input data latch. These decoding bits are able to turn these registers on or off according to the instruction being processed.

The execute takes place as the decodes are driven by the clock. Each clock cycle makes the decoder perform an instruction step. The various instructions of the 6510 need different numbers of cycles to complete the instruction. Some instructions like "decrement the Y register" can be executed in two cycles. Other instructions like "arithmetic shift left" might need six or seven cycles to complete. The 6510 is using a clock that is running at about 1 MHz. This means a complete cycle takes about 1000 nanoseconds or 1 microsecond. When you think of the timing, it is usually in terms of cycles.

Once the instruction is decoded, the various registers that are taking part in the instruction are turned on and the instruction is able to take place. During a read, the data (or an address where the data is to be found) follows the instruction bits on the data bus. This byte after the instruction, enters the 6510's internal data bus through the data bus



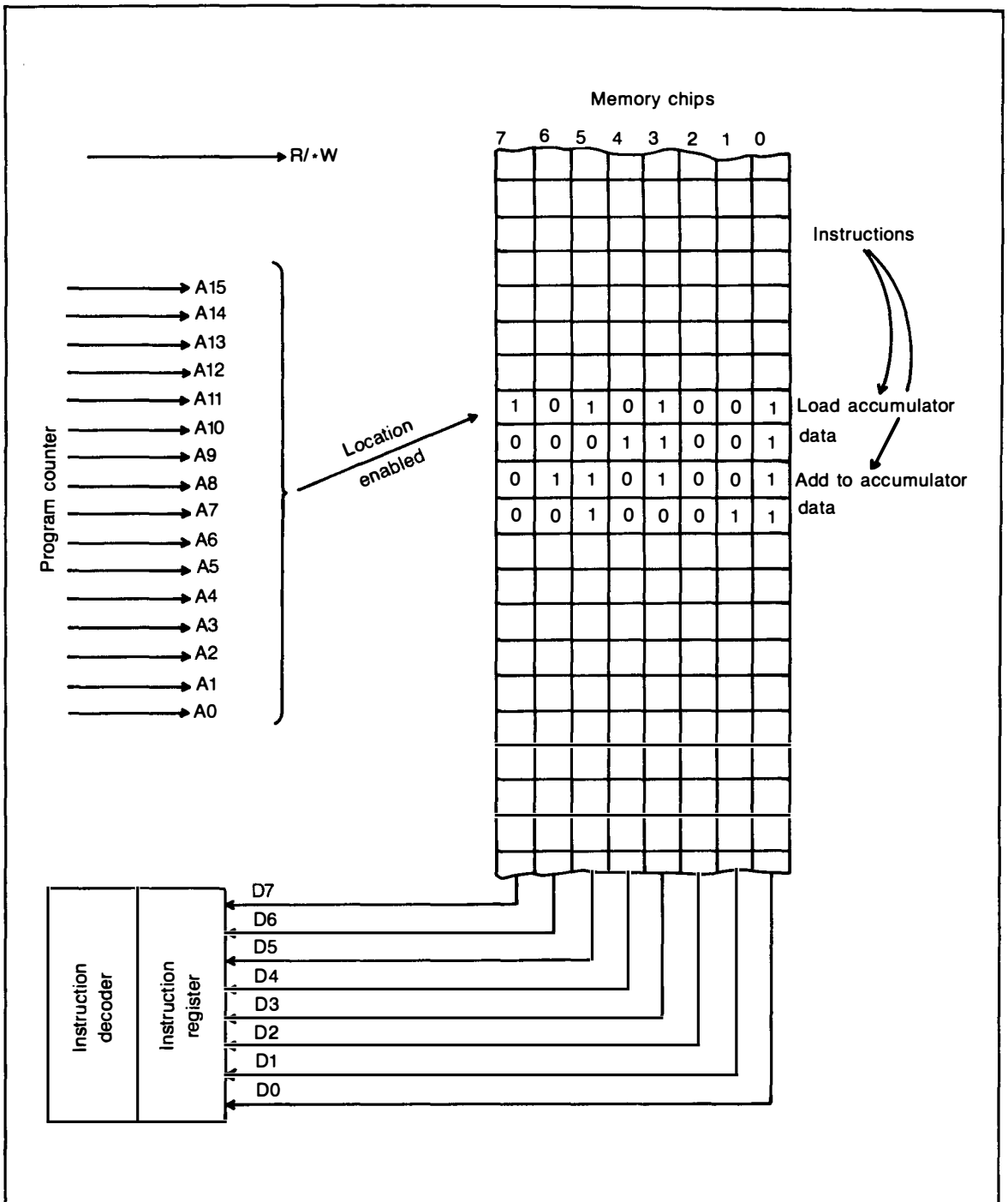


Fig. 12-11. The "fetch and execute" performance of the MPU requires addressing, data movement, decoding, and then executing the dictates of the instruction.

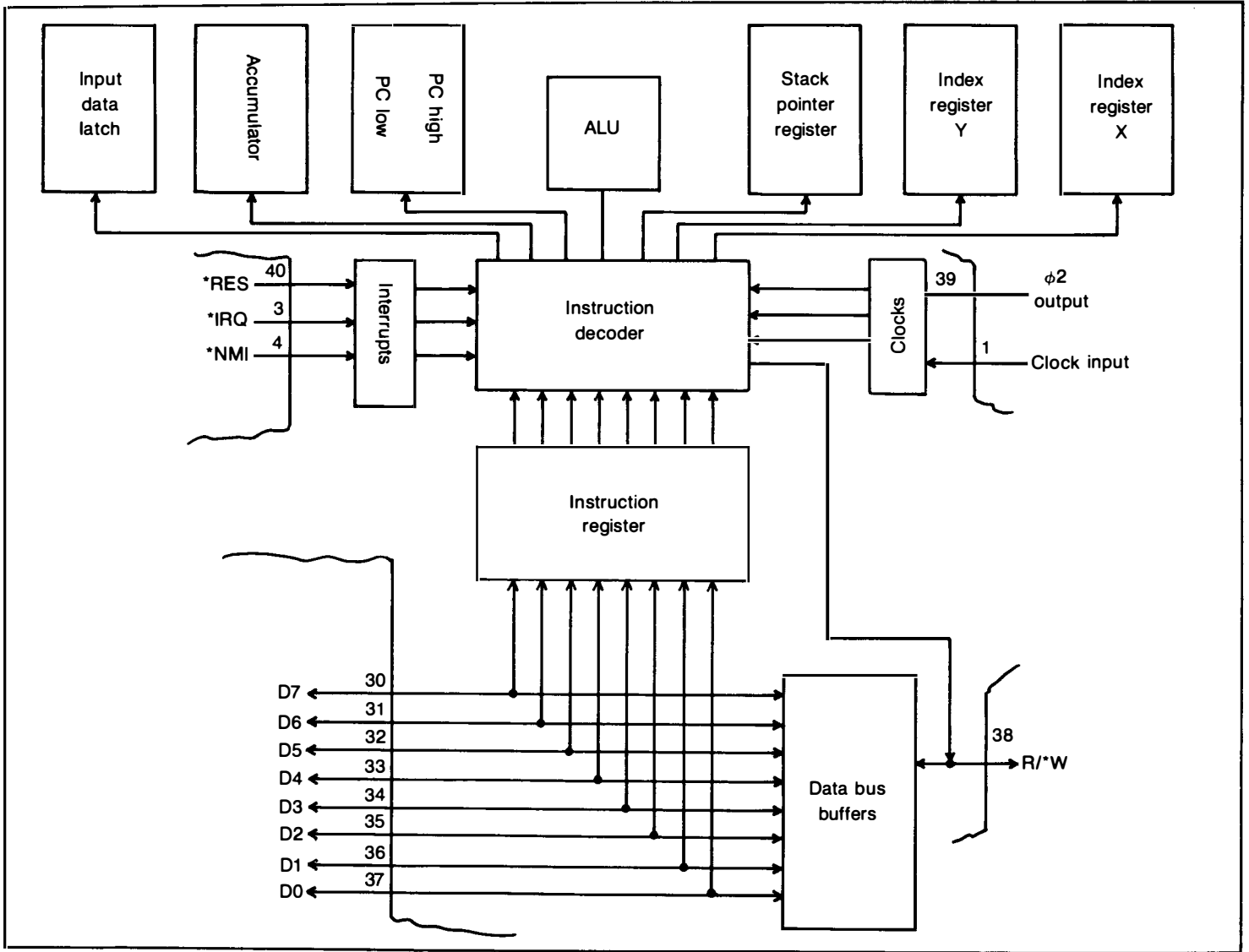


Fig. 12-12. Once the instruction is decoded, it sends specific bits to the registers it is connected to.

buffers. The data then enters the enabled registers, connected to the data bus, and is processed. The next instruction is ready to be fetched and executed.

There is an automatic incremter built into the program counter of the 6510. As soon as an address is output by the PC, the binary value in the PC is incremented by 1. That way as soon as the 6510 finishes the fetch and execute cycle, the next address in numerical order is addressed. The PC will increment unless an instruction like jump, branch, call or return is encountered. Then the PC will hold off on the incrementing and follow the addressing dictates of the instruction.

Once the special addressing instruction, such as jump or branch is dispatched, the PC returns to the automatic incrementing it is built to do. The PC can start at the beginning of a long program and will increment continually unless it is stopped by one of the special addressing instructions. There are only a few addressing type instructions that the

6510 will respond to. There are 10 branch instructions, two jumps, and two returns.

## INDEX REGISTERS

The programmer uses the X and Y index registers in the 6510 continually. Indexing is a programming technique that is invaluable. The index registers are used to look up data in tables that are installed in the memory map. For instance, a multiplication table can be placed into a program. The address of the first number in the table can be installed in the index register. Then, when a lookup is needed, an offset number is added to the table start number in the index register. The offset plus the table start number forms an address. This address contains the desired lookup. The table location is addressed and the lookup is retrieved.

This facility and some other programming tricks makes the index register valuable. During repair work the index registers do not play too much of a role. They are part of the hardware block

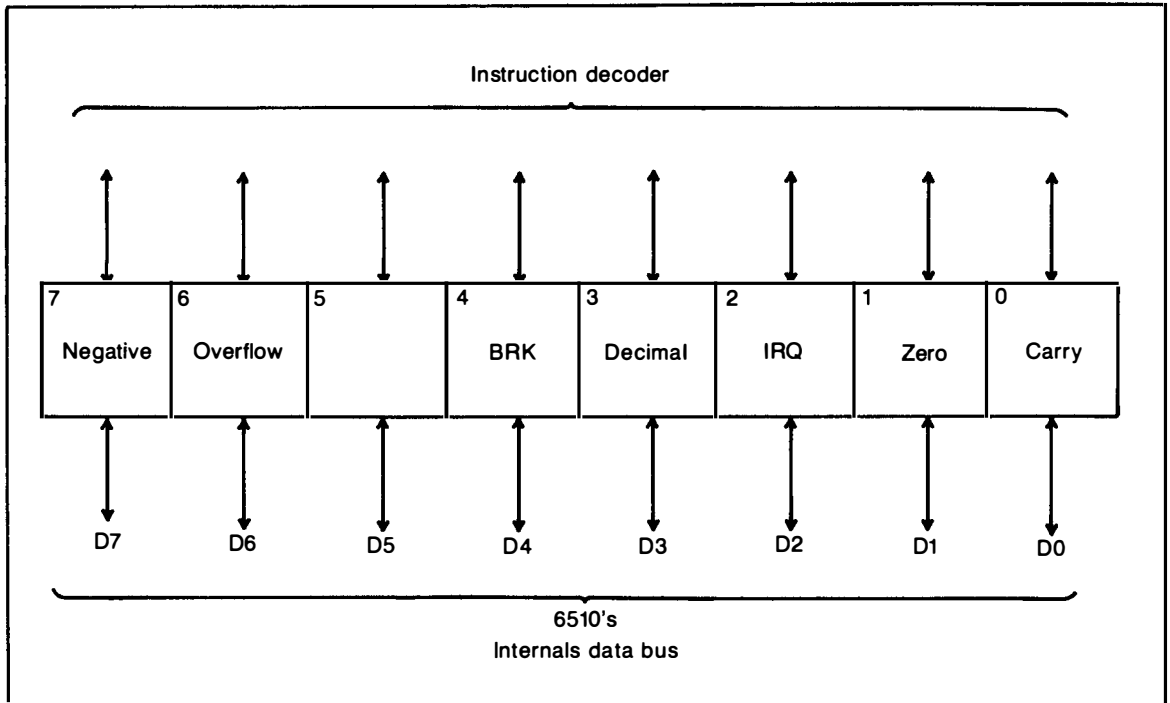


Fig. 12-13. The flag register, also known as the status register, is connected between the data bus and the instruction decoder. The flags monitor every decoding operation and are set or cleared accordingly.

diagram though and a servicer should be familiar with their operation.

## FLAG REGISTER

The flag register is shown in Fig. 12-13. It is also called the status register and is a special eight bit type. The 6510's status register has seven of the bits in use. The bit position 5 is unused. All the rest of the bits are designated particular individual jobs. The individual bits are called flags. The flag can be either in use or not being used. When the bit is set to a 1, the flag is in use. A 0 bit means the flag is lying still.

The flag action is vital during programming. As a program is run, there is rarely an instruction that is run that a flag is not set or reset. If the flags were represented by lights, the lights would be flashing continually as a program ran. The machine-language programmer must be completely aware of every flag that is set or cleared. Each flag movement performs some sort of duty and has important meaning.

The flag register is attached to the instruction decoder through one 8-bit bus and to the 6510's internal data bus through another eight parallel lines. Refer to Fig. 12-14. The flags do most of their business with the ALU through the internal data bus. The main job that the flags do is get set when the accumulator register attains a particular state. Let's examine some of the states the accumulator can get itself into.

The flag register in the 6510 has its eight bits arranged as shown in Fig. 12-15. Note that the programmer's diagram does not resemble the hardware block diagrams in any way. The programmers model concerns itself mainly with the bit sizes of the registers. The hardware diagram is mostly worried about the way the individual lines and the bus lines conduct the data, addresses, and instructions. The different approaches to the same Commodore 64 exemplifies the difference in the way the programmer and the technician view the computer. To repair the machine requires an acquaintance with both points of view.

Bit 7 of the flag register is the N flag. N stands

for negative. The negative refers to the numeric value of the contents of the accumulator register. The accumulator is also an 8-bit register. During calculations that use negative (-) numbers, the accumulator must denote the fact that its contents are a negative number. If the most significant bit is a 1, the accumulator is holding a negative number. As long as the MS bit is a 0, the number is positive (+). Therefore as soon as the MS bit of the accumulator becomes a 1, bit 7 of the flag register gets set. The flag becomes a 1 instead of its standby 0.

There is a complication with this arrangement. The flag gets set whenever the accumulator MS bit becomes a 1, whether the computer is doing calculations, or word processing. This is a problem for the programmer. All the tech need know is the flag will be set when the MS bit of the accumulator goes to a high.

The N flag being set can cause a change of address. The address can branch to an out of sequence address during the time that the N flag is set.

Bit 6 is the Overflow flag. It is used in two's complement calculations, but the repair job is not interested in those calculations. For repair purposes, the Overflow flag gets set when there is an arithmetical carry from bit 6 to bit 7 of the accumulator.

When there is an overflow, a special correction routine must be used to straighten out the overflow. The overflow can change the sign of accumulator bit 7 and must not be permitted to remain uncorrected.

Bit 5 of the flag register is unused in the present 6510. That does not mean it will always be unused. Future editions of the 6510 could include use of bit 5.

Bit 4 is a Break flag. There is one programming instruction called Break. The instruction will set the flag. It is useful during program writing and running.

Bit 3 is an arithmetic helper. When the flag is not set, the 6510 runs its mathematics normally in binary mode. During some operations, the binary mode of counting from 0 to 15 ( $10 = 16$ ) in decimal is clumsy. It is easier in these cases to be able to count from 0 to 9 ( $10 = 10$ ) in decimal. This is

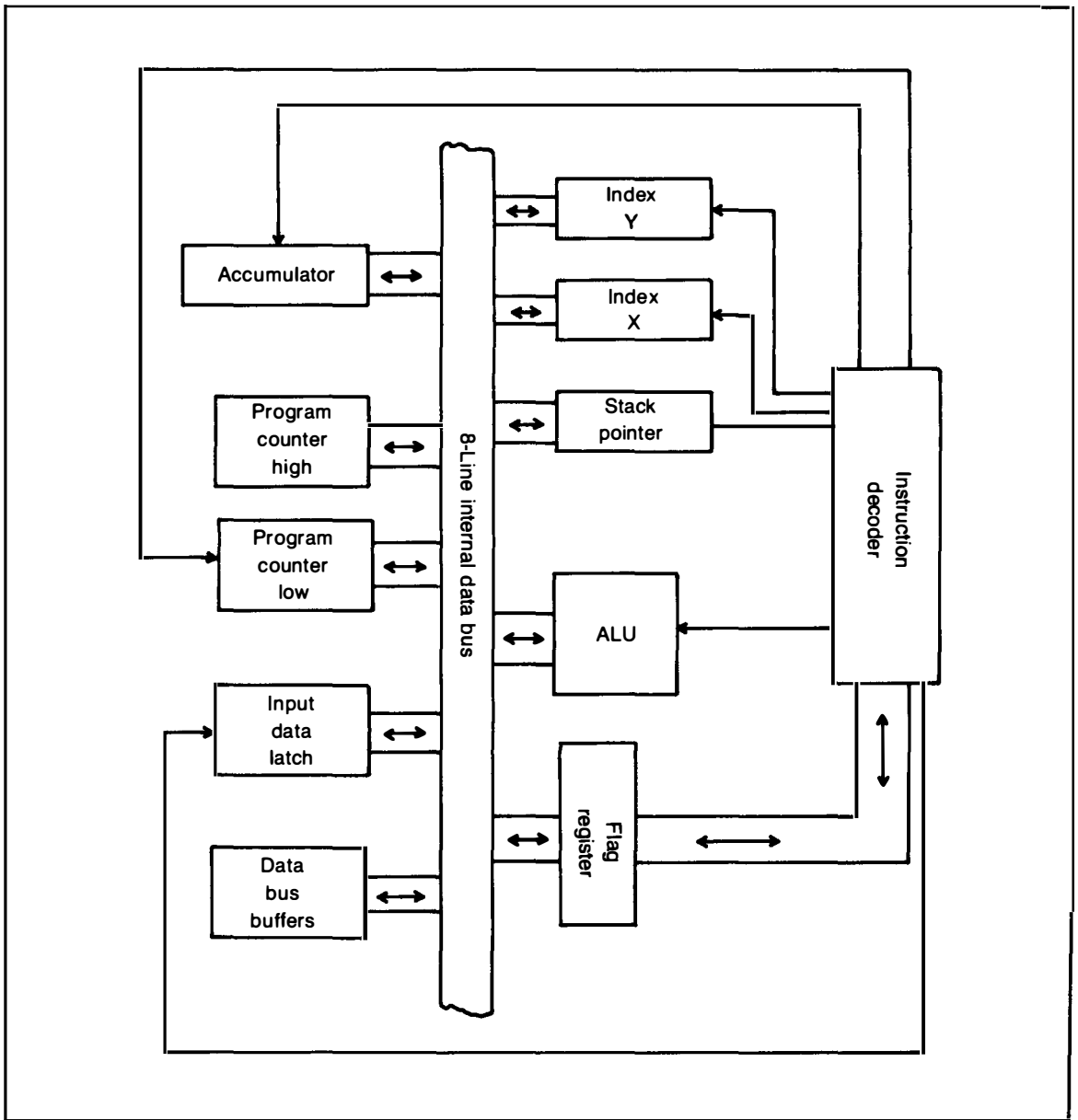


Fig. 12-14. The flag register does most of its business with the ALU. The flags get set or cleared as the accumulator register goes through specific gyrations.

called BCD for binary coded decimal. When bit 3 is set by certain instructions, the 6510 reverts to a BCD operating mode.

Bit 2 is the IRQ flag. IRQ stands for interrupt request. It is referred to as the interrupt mask bit.

When this bit is set, the interrupt is masked and no other interrupt can interrupt. The bit can be set with some instructions by the programmer, or the 6510 itself will set it during a reset situation. An interrupt will also set the bit.

Bit 1 is the Z flag. The Z flag like the N flag can get set under many conditions. It gets set whenever the accumulator goes to a state of all zeroes. The Z bit is built to get set whenever the accumulator 0 state occurs, intentional or not. That is the programmer's problem.

Bit 0 is the C flag. It acts like the ninth bit of the accumulator. It will receive the bit that gets carried out of bit 7 of the accumulator during arithmetic. It also will catch the bit that falls out of the accumulator during shift or rotate operations. For more details on flags, refer to a book on 6502 (the 6510's predecessor) machine language. For our

repair purposes, the above knowledge is sufficient to run tests on the hardware.

## INTERRUPTS

The instruction decoder has three inputs that are called interrupts. They can be seen in Fig. 12-12. These are software devices that make the computer appear automatic. They all do similar jobs. They are called Reset, IRQ, and NMI.

The Reset mechanism in the Commodore 64 is used to get the machine going. I discussed the 535 timer chip earlier. It is part of the Reset system. When you first turn on the 64, the electricity

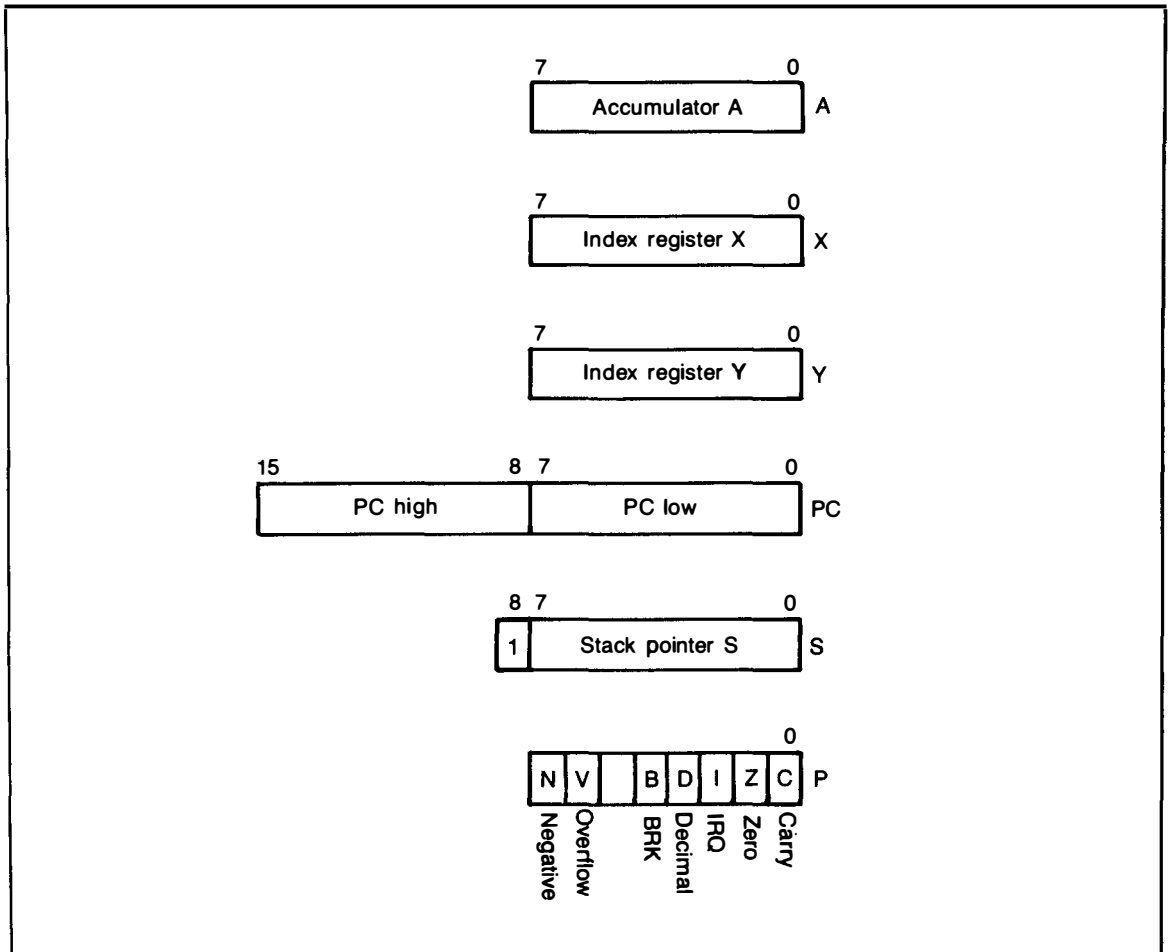


Fig. 12-15. This is the machine language programmers block diagram of the 6510. It concerns itself mostly with what registers are present and their various bit sizes.

courses in. At pin 8 of the 556 is the timer's trigger. As the voltage goes from 0 to +5 volts, the trigger is pulled. After two clock cycles a \*RESET low signal is output from the timer. It goes to pin 40 of the 6510 (\*RES). This sets off the reset routine.

The instruction decoder receives the \*RES low and turns on the R/\*W line to read. Six more clock cycles take place, and the IRQ flag gets set so no interruptions can stop the operation. The 6510 is built to load the program counter with two addresses at that time. The addresses are at the top of the memory map near the top location in the Kernal ROM. That address is read by the 6510.

Those byte locations hold a 16-bit address. The 6510 receives the address and promptly opens up that location. That location is the first address of the operating system's initialization program. This operation was covered previously in the discussion of the Kernal. As soon as the housekeeping routine is run, the computer is then READY with its cursor blinking. That is what the Reset input into the instruction decoder does.

The second interrupt line into the decoder is called \*IRQ, interruption request. It is normally held high and becomes active as it goes low. When the \*IRQ signal arrives, the 6510 will go into an interrupt sequence. The interrupts typically are originated by peripheral devices that want to send or receive data to the 6510 through one of the I/O chips. Note the \*IRQ line goes to the CIA at pin 21. It also connects to the cartridge expansion plug and a pin on the VIC.

When an interrupt is detected, the first thing the 6510 does is complete the instruction it is working on before it will acknowledge the \*IRQ signal. Once it completes the current instruction, it checks flag bit 2, the IRQ bit. As long as the IRQ bit is not set, the 6510 will proceed to service the interrupt. Should the bit be set though, the 6510 will ignore the new interrupt request.

Once in action on the interrupt the 6510 stores some of its registers in the stack. The program counter's current address goes into the stack. The states of all the flags are stored in the stack. The 6510 then goes to the flag register and sets the IRQ

flag so no other interrupt can interrupt the sequence it is about to undertake.

Next the 6510 places the last two addresses of the memory map on the PC one at a time. The 6510 does this automatically when an \*IRQ is recognized. In the Commodore 64, those addresses are also part of the Kernal. The contents of those two addresses are two bytes that form still another address. This formed address is the beginning location of a routine that services the \*IRQ.

As we humans follow these interrupt procedures, they seem complicated and difficult to understand,. The computer though, performs this game of using address after address to finally locate the service routine as second nature and has no difficulty. Once the interrupt has been serviced and the peripheral that caused the interrupt has been satisfied, the 6510 goes back to the processing it had been doing before the \*IRQ signal had arrived. The 6510 goes to the stack and places the program counter and flag values back into the PC and flag register. A special interrupt return instruction is executed and the 6510 resumes its program running by addressing location after location.

The third interrupt is called \*NMI for non-maskable interrupt. It can be activated by a low into pin 4 of the 6510. It works quite like the \*IRQ interrupt, except for one hitch in the procedure. Both interrupts pins, upon receiving a low, turn on. Both preserve the contents of the PC and the flag Register in the stack. Both automatically address two locations near the top of memory. The difference is in the I flag mask bit. The \*IRQ will not recognize an interrupt if the I mask is set. It will ignore all interrupts under that condition. \*IRQ will only acknowledge an interrupt if the I bit is clear.

\*NMI acts differently. The \*NMI interrupt will always recognize a low whether the I mask bit is set or clear. NMI stands for nonmaskable interrupt, it cannot be masked out like the \*IRQ. In lots of computers, the \*NMI is used in case there is an emergency interrupt like a power failure. In the Commodore 64, the \*NMI is used for a different purpose. It is used in the Restore key circuit.

When the Restore key is pressed, it triggers off part of the 556 timer chip. This outputs a signal

through the 7406N NOT gate. This signal is connected to \*NMI of the 6510 and can cause the \*NMI interrupt to take place. Note that pin 4 the \*NMI pin is held high by a pullup resistor to +5 volts till a \*NMI low will appear to turn on the interrupt sequence. The \*NMI signal is also connected to one of the 6526 CIAs. This action will be covered in Chapter 16, The CIAs.

## VECTOR ADDRESSES

It was discussed earlier that the three interrupts each have a program routine that they execute as they are enabled. The 6510 is built to go through a part of the routine, but it is the responsibility of the manufacturer to provide the rest of the program for the particular computer he is producing.

The 6510 is built to place the following addresses onto the program counter when interrupts are enabled. If the \*IRQ is turned on, the 6510 addresses two locations at the very top of memory, 65535 and 65534. When \*RES is activated, the next two lower locations are addressed, 65533 and 65532. Should \*NMI receive a low, locations 65531 and 65530 are addressed.

In the Commodore 64, the Kernal ROM occupies these automatically addressed locations. In the Kernal, these locations have been burnt in with bytes of data. Each location has one byte so that the two together form one 16-bit binary number. This number is fetched by the 6510 and it is placed into the program counter. The 16-bit address is the first location of the program that will service the interrupt.

A confusing word that describes the locations at the top of memory in the Kernal is *vector*. All that means is that there are two locations whose combined contents can form one address. For instance, 65535 and 65534 together contain a vectoring address. The whole operation is not unlike a game where you are given an address. At this address you will find another address that will tell you where to find a prize.

## OTHER PINS

At pin 2 of the 6510, there is a terminal called RDY,

for ready. Since there is no asterisk on the name, the experienced technician would know the RDY pin would be enabled if a high was injected. Since the majority of normal testing takes place while the computer is blinking the cursor, a high is its usual state to keep the 6510 ready.

The RDY pin was designed to allow the 6510 to get into sync with a memory device that is running at a much lower pace than the 1 MHz clock cycles. The RDY pin will stop the action of the 6510 when a low is applied. As soon as the low is removed by a high, the MPU will turn back on and resume its computing. That is the RDY signal does.

RDY goes to the 6510 from the output of one of the 74LS08 AND gates. The gate outputs a standby high most of the time. One of the two inputs of the gate comes from the cartridge expansion plug pin (\*DMA). \*DMA is held high because of a 3.3K pullup resistor. The other gate input is a signal BA, bus available, from the 82S100. As the name implies, BA is a high. The two gate inputs, two highs, output the AND gate high. The gate will continue to output a high as long as \*DMA stays high. If \*DMA goes low, then the AND gate will go low, and RDY becomes low and stops the 6510 in its tracks. The \*DMA low will be coming from a peripheral through the cartridge plug.

There are six pins named P5 through P0. Actually, \*NMI is really P6 and RDY is P7 to round out a byte. They all belong to that special I/O register in the 6510 that I alluded to before. This register will start off the next chapter.

P3 is a cassette write control line. It outputs to the cassette and tells it when a write operation is to take place. P4 is a cassette sensing output. P5 is a cassette motor off-on switch.

The final pins on the 6510 are VCC and ground. VCC comes from +5 Vdc. It has a filter and bypass capacitor in the line to the MPU. The filter is a 10  $\mu$ F at 25 WV and the bypass is a .47  $\mu$ F. The filter smooths out any 60 Hz hum, and the bypass gets rid of higher frequency pulses.

## TESTING

As the main actor in the drama of computing, when



6510 / MPU

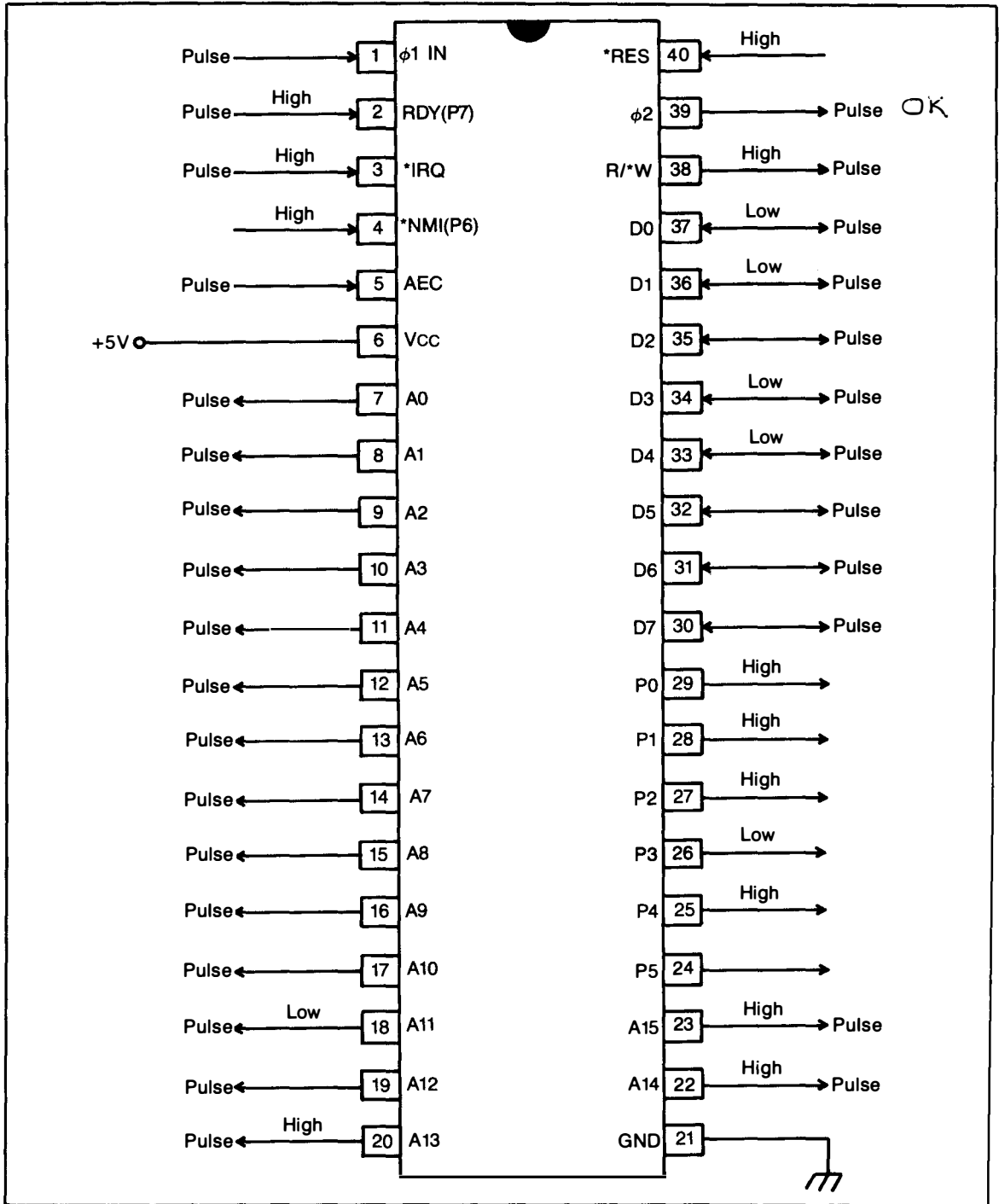


Fig. 12-16. This test point chart of the physical layout of the 6510 shows the inputs, outputs, and the logic probe readings that should be present when the 64 has been turned on and is awaiting keyboard strikes.

the 6510 fails the computer goes down. Most of the time failure results in no computing power at all. In some cases, a defect could allow the computer to do some work but will selectively stop other operations. If you can figure out what selective operation is gone, you could puzzle out the actual defect.

Fortunately, the failure of MPUs in general and the 6510 in particular is rare. Once a 6510 is installed on the print board and passes factory quality control, it has proven to be remarkably reliable. This doesn't mean one will never conk out, but look for other chip failures first.

If you suspect a bad 6510, the best test is to try a known good replacement. This might not be that convenient to do unless you have a replacement or own a second 64 for swapping chips purposes. As far as I can tell, the only source for the 6510 is the Commodore service department. It is their chip, and they own all rights. I have not seen the 6510 available in any other parts companies. Fortunately the chip is very reliable.

If you want to test the chip, the 6510 test point chart, Fig. 12-16, shows what should be present at each pin. The tests were made on my 64. The machine was energized, had the READY sign on, and the cursor blinking. This is a good position for the computer during the pin testing of chips. All the test point charts of the various chips were made under the same conditions.

All is well with the chip under test as long as the pins read the prescribed voltages or states. If a discrepancy is found, that constitutes a clue and

bears further investigation.

To quick check the 6510, the schematic and test point chart should be handy. Note the service chart works around the pinout of the chip. You can test quickly with the service chart. If you locate a problem pin, then you can check the schematic to see where voltages are originating from.

As a rule of thumb, the 6510, has pulses on all the output address and the two way data pins. Pulses are also to be found at the clock input  $\phi 0$  and the clock output  $\phi 2$ .

The three interrupt inputs should all read highs. They are held high until they must be enabled. Then they receive a low, but on normal standby, they are high. The asterisks indicate the state of high. You can often expect to find a high on standby when you see the asterisks and a low when there is no asterisk.

Don't take that as a hard and fast rule though. For example, RDY does not have an asterisk, yet the pin reads high. This is because it is enabled while the computer is using the 6510, as during standby. To turn off the 6510, you must disable the RDY pin with a low from a peripheral through the \*DMA line. Be on the alert for these deviations from the rule of thumb on highs and lows. They could lead you on a wild goose chase.

The R/\*W line is held high in a read mode. The line only goes low when a write operation is to take place. VCC is obviously to be a high and ground is always a low. If they are not, you have uncovered the clue that is causing any trouble.



## 13. Memory Maps

**T**HE 6510 PROCESSOR HAS 16 BITS IN ITS PROGRAM counter. 16 bits can address 64K of memory. Yet there is a lot more than 64K locations on the chips in the machine. There are 64K locations in the eight 4164 RAM chips alone. Besides that there is 8K in the Basic ROM, 8K in the Kernel and 4K in the Character ROM. The machine is able to contact another 4K of I/O devices and 1K of Color RAM. That is a grand total of 85K that the processor is able to make connections with. Clearly the 6510 must have some addressing tricks to contact all these memory locations.

The trick is that the 6510 sets up a number of different memory maps. Refer to Fig. 13-1. Each map variation is given 64K from the pool of 85K. The processor uses different map arrangements for a variety of jobs. In addition, during a program run the 6510 is able to bank in and out of the different maps. There are eight easily configured map layouts.

### I/O PORT

The key to this handy map switching is the extra

I/O port that the 6510 comes equipped with. The port, shown in Fig. 13-2, is an 8-bit register. Two of the bits have been covered in the last chapter. They are pins 4 and 2, the \*NMI and RDY connections. These two functions do not have much to do with the addressing. They were placed on bits 7 and 6 because otherwise those bits would have been unused. This way the pins are utilized in a valuable way.

Bits 5, 4, and 3 also have little to do with addressing. These three bits are connected to pins 24, 25, and 26 of the 6510. All three are cassette control lines.

It is bits 2, 1, and 0 that are used in the memory management scheme. They are shown in Fig. 13-2 at pins 27, 28, and 29. They put out the signals \*CHAREN, \*HIRAM, and \*LORAM. All three pins are connected to +5 volts through 3.3K pullup resistors. The pins are held high and only go low when they output their respective signals.

The three signals all go to pins 6, 7, and 8 of the 82S100 PLA chip. In the PLA chip, they will combine with some other signals to configure the

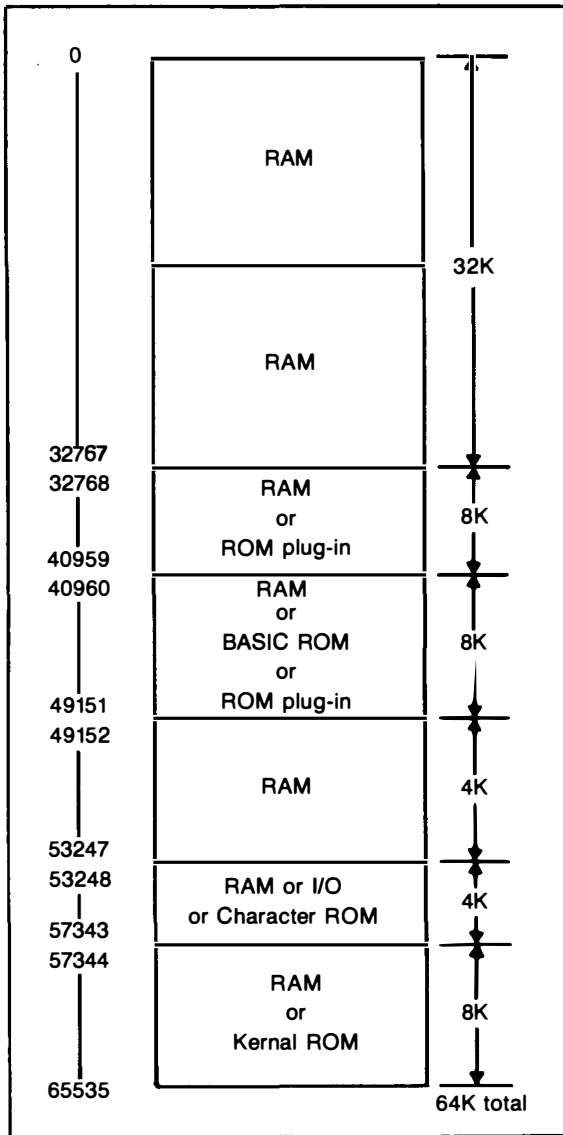


Fig. 13-1. The 64K memory locations the 6510 is able to address contain 85K of RAM, ROM, and I/O. Some of the locations are doing double and triple duty.

different versions of the memory map. Two such signals are coming from the cartridge expansion plug. They are \*EXROM and \*GAME. There will be more about this later.

In Fig. 13-2 there are two registers shown: the Data Direction Register and the Output Register. They are both connected to the 8-bit data bus of

the system.

Neither the DDR nor the OR registers exist in the 6510. These two registers are on the RAM chips as decimal locations 0 and 1. Address 0 is the Data Direction Register and location 1 is the Output Register. Since the internal data bus and the system data bus on the print board are connected together, the registers act as if they are inside the 6510. The only restriction is that these locations, 0 and 1, cannot be used for any other purpose. They belong to the 6510 and are in constant use. You can consider the registers as part of the 6510. There are circuits in the 6510 that control the two locations as if they were in the processor. The locations are not independent of the processor like the rest of RAM.

The 6510 I/O port acts just like other I/O ports in the machine. Details of port action are discussed in Chapter 16. As you'll read there, the Data Direction Registers control the direction the signal travels as it passes through the Output Registers. This port acts in the same way.

DDRs can be programmed bit by bit. The DDR bits are coupled to the respective bit in the OR it controls. That is, bit 0 of the DDR connects to bit 0 of the OR. Bit 1 of the DDR connects to bit 1 of the OR, and so on through bits 7. Refer to Fig. 13-3.

If you program a bit in the DDR with a 1, the corresponding bit in the OR becomes a single bit output port. Should you program a DDR bit with a 0, the same bit in the OR becomes a single bit input port. The perspective is from the processor. When a bit is an output, it is an output from the processor to the rest of the computer. An input bit is an input to the processor from the rest of the computer.

A one bit port passes a single line. This I/O register is able to pass eight lines of signal. When the 64 is first turned on, it sets up the system to run BASIC. This is the default memory map. *Default* in computerese is the state the computer adopts automatically. To switch the system to another memory map configuration, the I/O port must be programmed accordingly. For the default map, the operating system programs the I/O port routinely as it powers up.

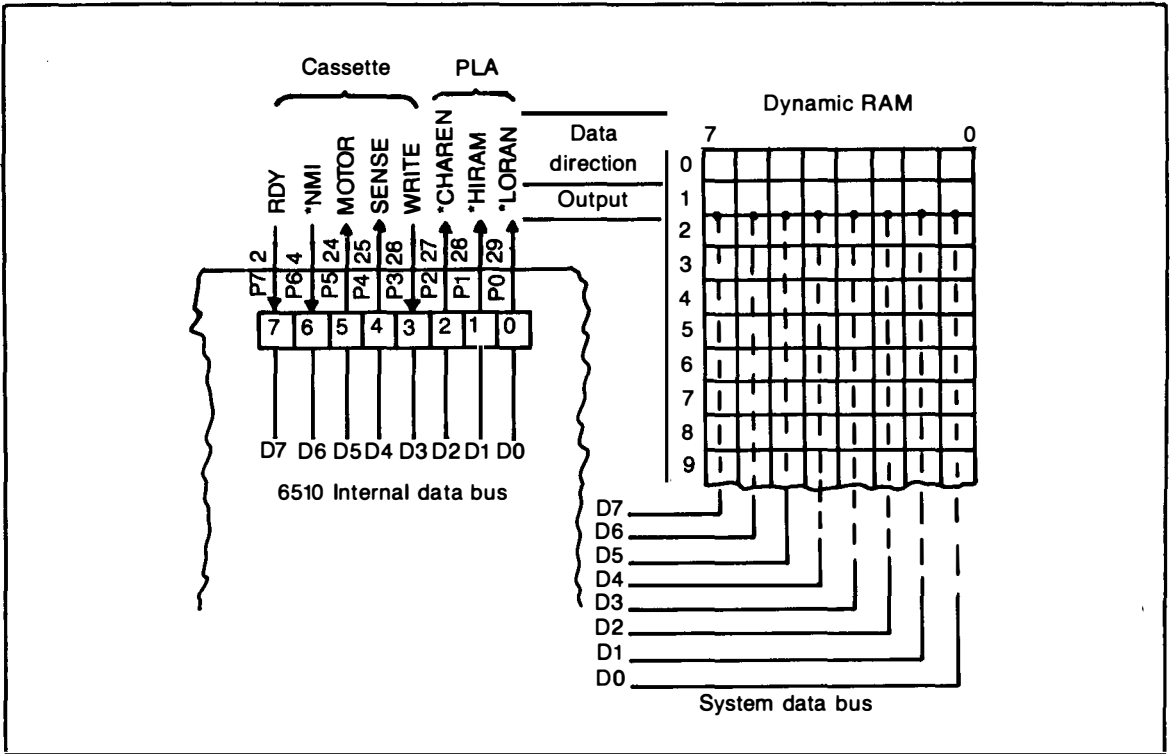


Fig. 13-2. The I/O port inside the 6510 works directly with RAM locations 0 and 1, just as if they were inside the 6510. RAM location 0 is the data direction register, RAM address 1 is the peripheral output register and P7-P0 inside the 6510 is the peripheral output buffer register. The peripherals being contacted are the cassette and the PLA.

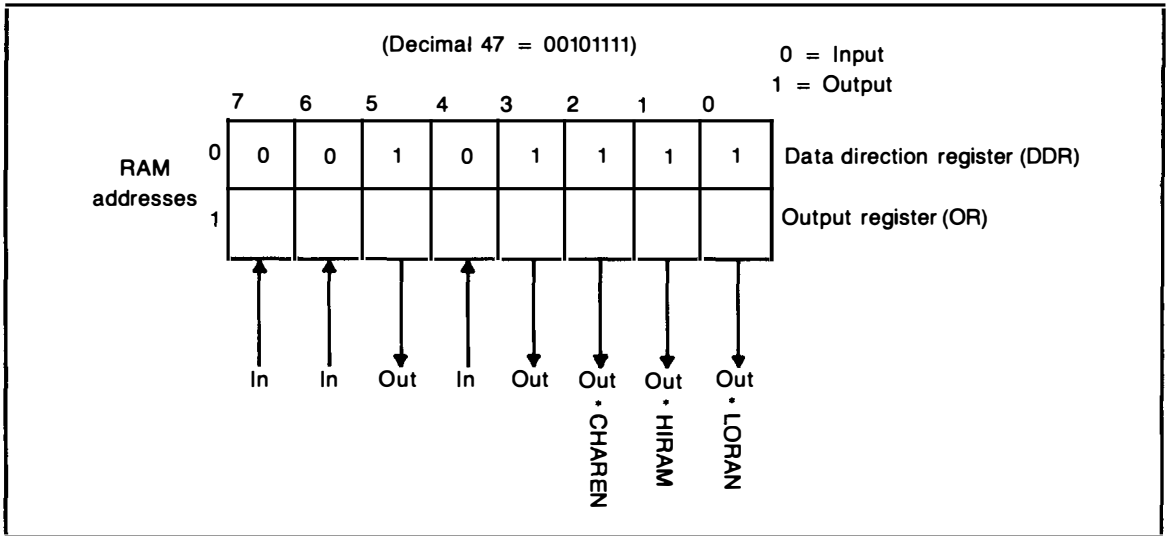


Fig. 13-3. If you program a bit in the DDR with a 1, the corresponding bit in the OR becomes a single bit output port. Should you program a bit in the DDR with a 0 the corresponding bit in the OR becomes a single bit input port. The perspective is from the 6510. The rest of the computer acts as the peripheral.

## DEFAULT MAP

When you turn on the 64 the operating system sends bits to decimal address 0, the DDR. It sends eight bits through data lines D7-D0. It sends 0s to bits 7 and 6. This makes the \*NMI and RDY lines inputs. This has little to do with the memory map, but these bits are in the OR register. That way if a \*NMI or RDY signal does occur the input port will accept the signals and send them on into the 6510 for processing.

At the same time, the Kernal sends signals 1, 0, 1 to the bits 5, 4, and 3 of the DDR. This configures the corresponding OR bit as output, input, output. These three lines are the cassette motor switch, the cassette sensing circuit, and the cassette write line. The sensing circuit awaits signals to react. It is the input line. The other two lines are outputs to turn the cassette off and on, and to conduct any writing to the cassette.

Those are handy port jobs but not closely related to the memory management operation. The last three bits, 0, 1 and 2 are in the memory scheme of things. All three of the DDR bits, 0, 1, and 2 are sent 1s from the Kernal. This makes all three OR bits into outputs. The 6510 is able to control which memory map it wants to use by outputting either 1s or 0s through these three port outputs.

## I/O SIGNALS

The line \*LORAM, which loosely refers to low RAM, exits bit 0 of the OR. Since the POR is wired effectively inside the 6510, the \*LORAM signal comes out of pin 29 of the processor. Refer to Master Schematic 11 in the Appendix.

For BASIC, the line is a high. It enters the 82S100 chip at pin 8, I1. After some processing, it leaves the PLA at pin 17, F1, as a low signal called \*BASIC. The low is connected to \*CS of the 8K BASIC ROM. This low enables the chip, and it is on all of the time that the BASIC default mode is in operation.

Should \*BASIC happen to go high, it will disable the BASIC ROM and the BASIC operation will cease as the 8K of the chip drops out of the memory map. When the BASIC ROM is not

available, the memory map becomes something else besides the default map.

The line \*HIRAM, is loosely referred to as high RAM. The signal leaves bit 1 of the OR through pin 28 of the processor. It goes directly to pin 7 of the PLA, I2. It leaves the processor as a high and is changed to a low in the PLA. It leaves the PLA as \*KERNAL from pin 16, F2. From there, it is wired into \*CS, the chip select of the 8K Kernal ROM.

If the \*HIRAM signal goes low, then a high will be applied from the PLA to the Kernal's \*CS input. When that happens, the 8K of the Kernal will drop out of the memory map and be out of the circuit. The 6510 will not be able to address it. A map change has taken place and the Kernal ROM is not one of the residents.

The line \*CHAREN originates in the 6510 and exits through bit 2 of the OR. It loosely refers to the Character Generator ROM. It leaves at pin 27 of the processor, P2. It goes directly to pin 6, 13, of the PLA. Note that all three of these control lines are attached to +5 volts through three 3.3K pullup resistors. All three lines are normally set to highs. \*CHAREN is high as it enters the PLA.

The signal \*CHAROM exits the PLA out of pin 15, Fe. The character generator acts in an opposite manner compared to the other ROMs. The character generator, which is not used too often once a program is underway, is not included in the default memory map. When \*CHAREN is high, the character ROM is left out of the map. It is only when the CHAREN bit is cleared to a low that the ROM is given space in the memory map.

There are two other lines that play a part in arranging different memory maps. Both of these lines emerge from the cartridge expansion plug. They are named \*GAME and \*EXROM. The word GAME is obviously a part of some cartridge that will be plugged into the socket. The name EXROM comes from external ROM. Most cartridges are forms of ROM chips.

The two lines are held high when not in use. They are connected to a pair of 3.3K pullup resistors to +5 volts.

These five control lines make up a data bus of

sorts. The three lines from the 6510, \*LORAM, \*HIRAM and \*CHAREN are output as individual lines. They do not arrive as members of the normal D7-D0 data bus. That is why the special I/O register is needed in the 6510.

The two lines from the cartridge plug are companion signals also connected to the PLA decoder. Between the five lines the 85K locations can be managed by the 6510 which is able to only address 64K.

## MEMORY DECKS

The memory map in the Commodore 64 is a doubledeck affair. The bottom deck is 64K bytes long and is made up of the eight 4164 dynamic RAM chips. The addresses to the RAM, in decimal is 0 to 65535. The 6510, with 16 address lines, is able to address each location easily.

This decking is not one long unbroken stretch. The deck is sectioned off in pieces. Each piece is called a *page*. Each page is 256 bytes in length. This puts 256 pages in the total of 65536 bytes. ( $256 \times 256 = 65536$ ) This paging setup of the memory is important to programmers. The programmer must not run across page lines under certain programming situations. Figure 13-4 illustrates this page arrangement.

The top deck is where the rest of the memory chips are installed. The top deck contains the other 21K of the total 85K memory. The addresses of the top deck match up with the bottom deck. In decimal they range from 32768 to 65535. It is a duplex arrangement with top deck chips sharing addresses with the bottom deck chips. However, the top deck chips do not fill up all the addresses. A lot of the possible locations are uninhabited.

As you check the occupancy of the top deck in Fig. 13-5, there are no filled locations till address 40960. At that address is the first byte of the BASIC ROM. The chip then has occupancy to location 49151 which is a total of 8K locations. As you go on along the top deck the next chip area is I/O that begins at 53248. A closeup of the I/O area in Fig. 13-6 shows that it belongs with VIC II. Its addresses are 53248 to 53294. This comprises the 47

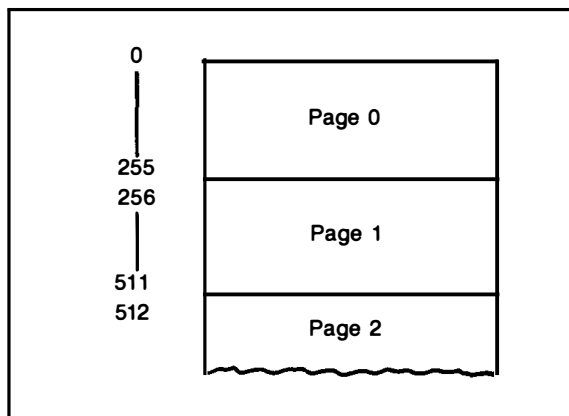


Fig. 13-4. The memory map arrangement consists of 256 pages each 256 locations long.

registers in the VIC.

Continuing along the top finds the next addresses are on the SID chip. They are 54272 through 55295. This totals 1023 locations. The Color RAM chip is next in line at locations 55296 to 56319, another 1K of addresses.

The next chip area is for CIA1 and CIA2 addresses. THE address space is 56320 to 56831. Mounted piggy back over the entire 4K and forming a sort of third deck is the Character ROM. The two levels both use the same address space. As mentioned earlier the Character ROM is not contacted often. It is put up here out of the way. The rest of the I/O's are in constant use and make the most out of these locations.

The final chip on the top deck (Fig. 13-5) is the Kernal. It occupies locations 57344 through 65535. This is an 8K stretch.

The paging applies to the top deck too. By manipulation, the addresses of the chips just described, can be switched around for various programming purposes. Each 64 pages comprise a section of memory. When the addresses are switched, attention must be paid. Things get unstable if a chip gets into a wrong 64 page section or happens to straddle two sections. This is more of a programming worry than a repair concern.

Besides the double decking, the memory map can have some other residents. These are external

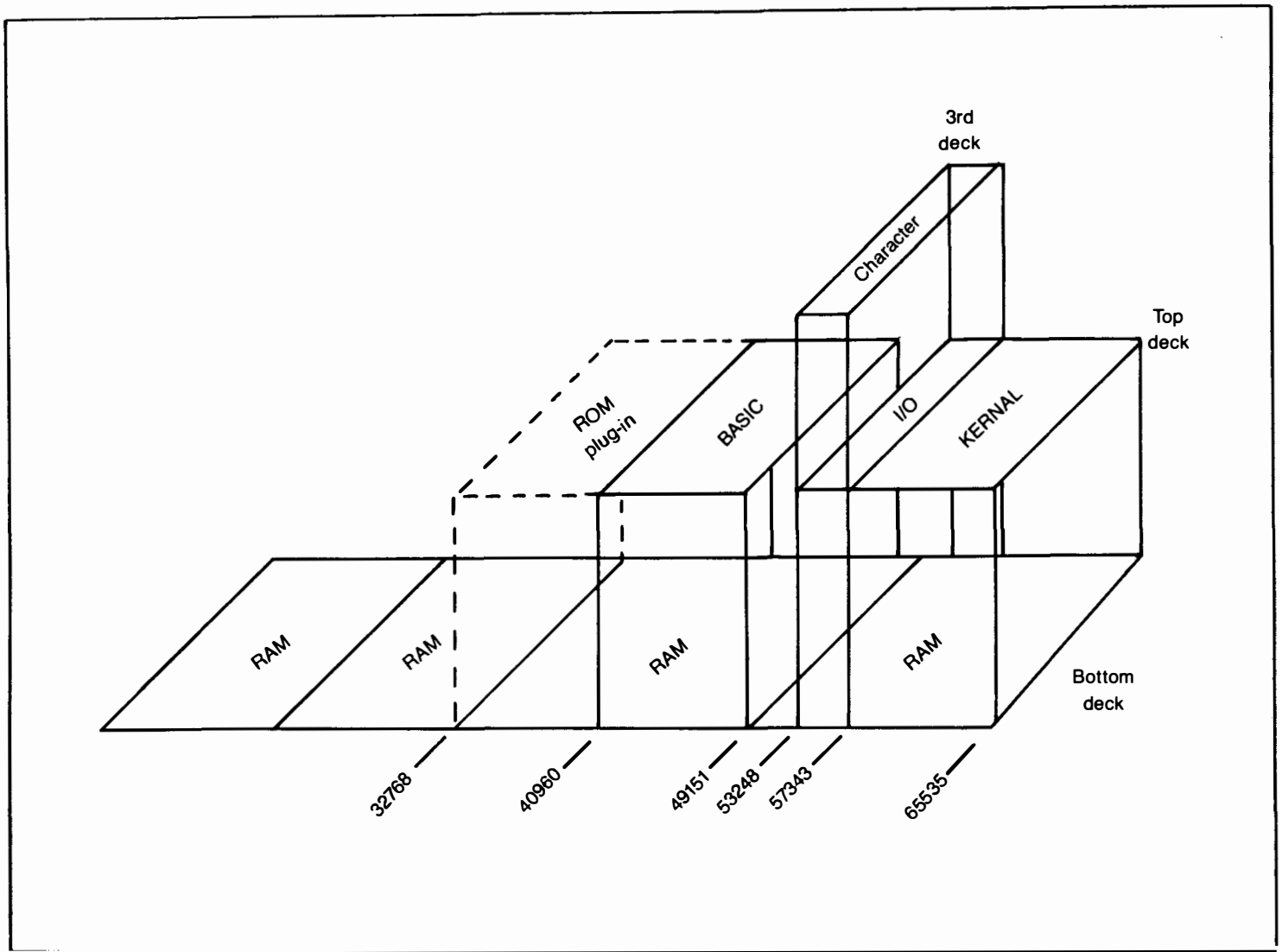


Fig. 13-5. The 85K is installed in the memory map in a bottom deck and top decks. The bottom deck contains the 64K RAM. The top deck holds the BASIC ROM, Kernal ROM and I/O addresses. The Character ROM resides on the third deck above the I/O locations.



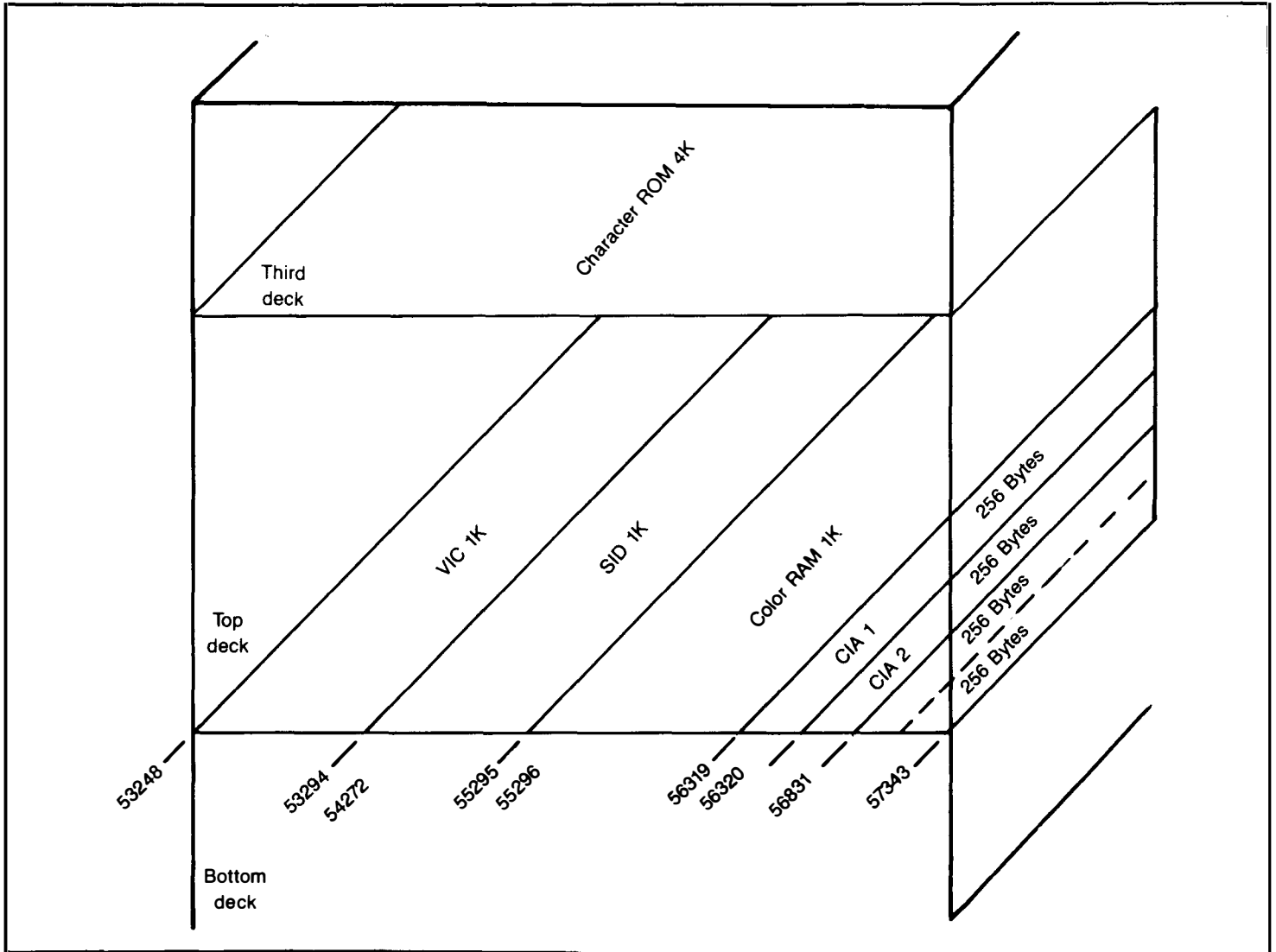


Fig. 13-6. A closeup of the I/O addresses 53248-57343 shows the VIC, the SID, the Color RAM, and CIA 1 and 2. At the end of the I/O neighborhood are two unused 256 byte areas.

devices such as cartridge ROMs. They gain entrance through the ROM plugs to the print board. An 8K cartridge can be plugged into an empty top deck area from locations 32767 to 40959. Another ROM cartridge can be plugged into the same locations as the BASIC ROM, 40960 to 49151. When it is plugged in, the BASIC is disabled. The BASIC isn't needed in these circumstances.

## READING AND WRITING TO THE DECKS

Since the two decks share the same addresses, there is bound to be some confusion. The top deck could be read when the bottom deck data is needed or the bottom deck could be written to when the top deck was really desired. There are some rules to solve the problem.

First of all, there are eight ways to layout the memory map. Some arrangements are convenient for certain types of computing and other arrangements for other things. Be aware of just which chips are active.

For example, the so called default memory map, is shown in Fig. 13-7. Starting at location decimal 0 the RAM locations extend to 40K. This is composed of two 16K sections and an 8K section. All of these locations are on the bottom RAM deck, with no chips above them on the top deck.

The next 8K of the map includes the BASIC ROM. The ROM is enabled by the signal \*BASIC out of the PLA. The BASIC ROM is mounted on the top deck above 8K of RAM locations. Since the ROM is enabled, it will respond if it is read. The ROM has priority over the RAM while it is enabled.

An interesting phenomenon takes place if you should write to the ROM. Naturally when you write to a ROM you are wasting your time. A ROM cannot receive data and store it. It is a read only device. However, the ROMs in the 64 are on the second deck above RAM locations with the same addresses. If you write to the ROMs, any data sent slips under the ROM and gets stored in the RAM locations that are simultaneously addressed. The data that gets stored this way though, can't be read as long as the ROM is enabled. It can be read if the ROM is disabled. This storage trickery can be

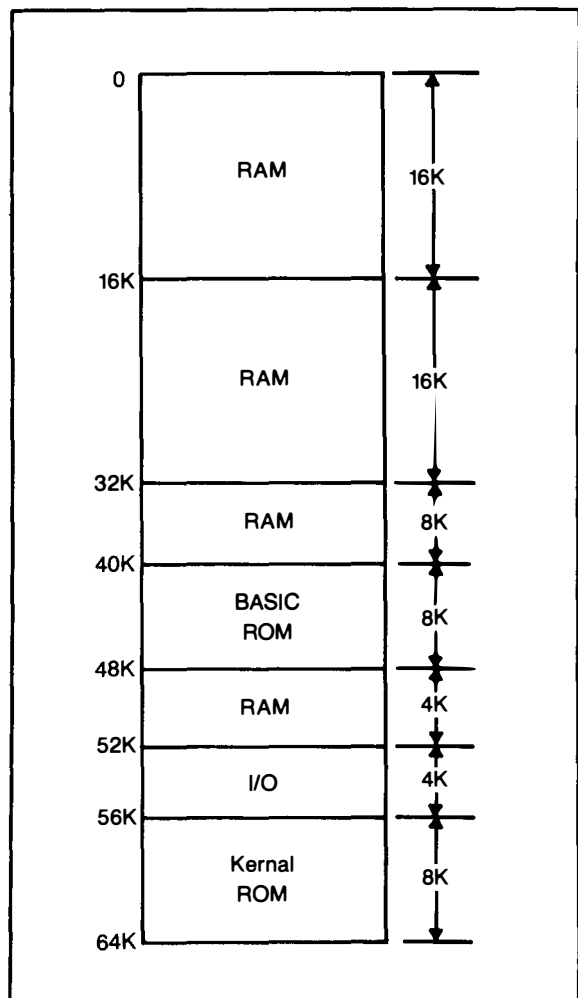


Fig. 13-7. The default memory map has RAM up to 40K. Next comes the BASIC ROM followed by some more RAM. After that, the I/O assortment has addresses. The final chip on the map is the Kernal ROM.

handy during some programming.

The next 4K of the map has a RAM buffer. This buffer is located on the bottom deck. The next 4K is set aside for the I/O chips that are sitting beneath the Character ROM. The last 8K of the 64K is reserved for the Kernal ROM.

The default map is arranged by the operating system of the 64 when you first turn it on. The top deck ROM and I/O chips are designed into the system to have priority over the bottom deck 64K

location length of RAM. To set up the default map, all that is needed is to enable the ROM or I/O chips that are to be residents. The RAM will take care of itself. The map will set up automatically if the BASIC ROM, the 4K I/O, and the Kernal are enabled.

The PLA does the enabling and disabling of the three ROMs with signals \*BASIC, \*KERNAL and \*CHAROM. \*CHAROM simply banks the Character ROM in and out of the memory map when it needs to send some characters to be stored in RAM for programming reasons. \*BASIC and \*KERNAL either puts the two top deck ROMs into the memory map or removes them so the RAM on the bottom deck at the same number locations can perform reading and writing.

To get the default map installed, the operating system must enable the BASIC and Kernal ROMs. It must also be sure that the cartridge expansion plug enabling signals, \*GAME and \*EXROM are off. Those two signals are deemed off as long as they are held high by their pullup resistors to +5 volts. As long as they are high, they are sending a code of 1 to the PLA.

There are four signals needed to arrange a memory map. They are inputs to the PLA: \*LORAM, \*HIRAM, \*GAME, and EXROM. When these four signals are all highs, the default map sets up automatically. The BASIC and Kernal are enabled.

These four highs are decoded in the PLA to make \*BASIC and \*KERNAL high for enabling the ROMs. Also, the four highs make the PLA output signal, I/O, high. This enables the 74LS139 decoder chip. That outputs signals \*CIA1 and \*CIA2. They enable the CIAs in the 4K I/O space.

To sum up, the operating system automatically prepares the default memory map by causing the four map enabling signals to all be highs. This turns on the BASIC and Kernal ROMs. It also keeps the two CIAs operating in the 4K I/O space.

## OTHER POSSIBLE MAPS

The default map is the one used most of the time. It comes up automatically and requires no special programming. There are some other map ar-

rangements that also come up automatically. This comes about when you plug in a cartridge or interface other devices. Then there are maps that machine-language programmers will arrange to produce special programming. For repairing the 64, all that is needed is to know what maps might be used on the 64 for whatever purpose.

The various maps are produced by changing the states of the four input signals to the PLA. These four signals are the map makers. As the highs and lows are changed, different map layouts are produced.

How are the inputs to the PLA changed around to create the different memory maps? There are no data, or address bus lines to the PLA to carry the high or low inputs. A glance at the schematic shows that \*LORAM is coming from pin 29, P0 of the processor. \*HIRAM is arriving at the PLA via pin 28, P1. These are pin output lines from bits 0 and 1 of the I/O register in the 6510. They are at location 1 of RAM. Refer again to Fig. 3-3.

Location 0 of RAM is the data direction register that configures location 1. There are highs in the DDR's bits 0 and 1. Therefore the OR bits 0 and 1 are configured as output lines. They connect through some buffers to the PLA as \*LORAM and \*HIRAM.

RAM location 1, therefore, becomes an output port to the PLA. Any highs or lows in bits 0 and 1 of the accumulator can be written to RAM location 1. They will transfer out of the location and become signals \*LORAM and \*HIRAM. They travel the wires to the PLA and act out a code in the PLA. Any one out of the seven possible maps can be formed this way.

For instance, suppose you make \*LORAM a high and \*HIRAM a low. Both \*GAME and \*EXROM remain high. The only change from the default map is \*HIRAM is made low. The effect though is dramatic. A low \*HIRAM disables both the BASIC and Kernal ROMs. The I/O chips remain enabled and active.

With BASIC and Kernal out of action the bottom deck RAM locations take their addresses. The map is now all RAM except for the 4K I/O reserved for the CIAs.

In the same way, other map layouts are produced. There is one that just disables the BASIC ROM but leaves the Kernal and the I/O section intact. This is done by making \*LORAM a low and \*HIRAM high. The cartridge signals stay high. Still another layout disables all of the top deck leaving nothing but 64K of RAM on the bottom deck. All four signals are made low for this arrangement.

The remaining four maps occur when cartridges are plugged in. The BASIC Expansion cartridge ROM forces \*EXROM low but the other three signals remain high.

## PEEKING AND POKING THE MAP

When the computer is using the default map, it will obey BASIC commands or respond to function requests. During a large percentage of troubleshooting you are able to use BASIC as an aid. Of course, if the machine is out of commission altogether, BASIC is of no avail.

When you can use them, PEEK and POKE can act as a probe. With PEEK, you can read the contents of any location on the map. With POKE, you can write to any RAM location or any I/O address. PEEK is a function that will return an eight bit binary number that is stored in RAM or ROM. The number will be coded into decimal and printed on the screen. POKE allows you to write an 8-bit binary number to any RAM location. You enter the decimal code of the binary bits and the command will decode the decimal and install the bits.

As mentioned earlier, should you write to a ROM the bits will be forced in the RAM location

beneath the same number ROM location. If you then try to read that location, the ROM bits will be returned not the RAM. All these facts are the way the 64 is supposed to work. If it acts wrongly, that could be a sign of trouble.

You cannot PEEK or POKE a chip that is not a resident of the memory map. You can only PEEK or POKE an address. All addresses are 16 bits and range from decimal 0 to 65535. BASIC uses decimal. To get a number out of a location and have it printed on the screen, all you have to do is type, PRINT PEEK (the address in decimal), and then press Return. To install a number into a RAM location, you must type POKE (the address in decimal); (the number in decimal) and then press Return. These are direct moves and do not require any programming line numbers.

The PEEK and POKE powers are very handy for quick checking the chips in the memory map. You can read the values of ROM locations to see if the chip is operating or the location is holding the correct number. You could write a number to a RAM or I/O location. Then you could read that location to see if the number ever arrived safely. If it didn't, that could indicate a defective chip or a bad I/O interface circuit.

There are all sorts of tests like those that PEEK and POKE permit. In these cases, the BASIC ROM becomes a valuable piece of test equipment. The ROM acts as a signal injection device. If you are a programmer you can use PEEK and POKE in your own programs to quickly test all of RAM, the ROMs, and the I/O circuits. Chapters 14 through 19 provide some examples of these test techniques.

A black and white photograph of a Commodore 64 computer keyboard and mouse. The keyboard is in the foreground, and the mouse is to its right. The background is a plain, light-colored surface.

## 14. Clock

**T**HE COMMODORE 64 RUNS IN TIME TO A 14.31818 MHz clock. That is the master frequency of the machine. Just as your body functions are driven by your 72 beat per minute heart, the computer functions are driven with every cycle of its clock. The clock is based around a crystal that is cut to oscillate normally at exactly 14.31818 MHz. The crystal is connected into pins 12 and 13 of the 74LS629N dual oscillator chip. A tiny 10 pF capacitor connects between the crystal and pin 13. Refer to Master Schematic 3 in the Appendix.

The pins attach the crystal to a resonant circuit inside the chip. A 2K potentiometer and a 0.1 $\mu$ F capacitor set the exact frequency of the oscillation. They are connected to Pin 1 of the chip. There are two oscillator circuits in the chip. The crystal circuit uses up one of the circuits. The other one is found later in the clock stage.

The frequency of the oscillator can be checked with a frequency counter. Connect the counter to pin 10, which is the output of the chip circuit. The counter will read the frequency directly. If the fre-

quency is slightly off, it can be adjusted with the 2K pot.

### SINE WAVE TO SQUARE WAVE

The crystal rings and produces a sine wave at the desired frequency. The sine wave is an analog waveshape like that in Fig. 14-1. Digital computer circuits cannot use sine waves. They can only respond to square waves. A sine wave is continuous, and it has an infinite number of highs and lows in one cycle. A square wave cycle consists of one high and one low. The digital circuit is built to handle square waves only.

The section of the 74LS629N that receives the sine wave input, converts the sine wave to a square wave. The output of the chip at pin 10 is square. The frequency remains the same; only the waveshape is changed.

One cycle of the perfect square wave is thought of as having the following timing. The wave could start with a duration of half the cycle at the low voltage. At the end of the low duration, the voltage goes

to the high voltage in no time at all. Then for the duration of the rest of the cycle, the voltage remains high. As the cycle ends, the voltage falls to low, again in no time at all.

While this is theoretically a square wave, it is not possible for the voltage to go to high in zero time or drop to a low in zero time. There must be some time elapsing during the transitions, no matter how fast the wave changes. The actual waveshape slopes to the right and upward as it goes from low to high and to the right and downward as it continues from high to low. The low to high is called a *rising edge*, and the high to low transition is called a *falling edge*. The edges are vital in digital circuits to trigger off a lot of the activity.

The 14.31818 MHz square wave that exits pin 10 has two destinations. The signal at this frequency is given a name. It is called  $\phi$ COLOR (Phase COLOR).  $\phi$ COLOR is first of all sent to pin 21 of the VIC. It provides frequency control there. This is covered in Chapter 17. In addition, the signal is coupled to pin 5 of the next chip in the clock area, the 74LS193, the up-down binary counter.

## WORKING FREQUENCY

The crystal's frequency is needed as an input to the VIC, but that is the only need for the 14.31818

MHz in the machine. The other uses of the square wave require lower frequencies. The binary counter along with the 74LS74 D flip-flops have to divide the frequency so it may be sent to other large chips, like the processor. The 6510 needs a driving frequency around 1 MHz.

The binary counter receives the master frequency out of pin 10 of the oscillator through a ferrite bead. The signal enters the counter at Pin 5, the count up input. Count down at pin 4 is disabled by being held high with +5 volts. Pins 4 and 5 are coupled into the four flip-flops inside the chip.

As each square wave enters the counter, it starts the flip-flops changing states. The flip-flop state changing is further affected by states injected at pin 9. The original frequency is divided at the output pin 6. For those of you who are interested, the output waveshape can be viewed on an expensive scope. The output of the binary counter, pin 6, will show six square waves if the scope is reading at a 0.2 microsecond rate. A sample waveform is shown in Fig. 14-2. The logic probe will read a pulse. This is a good test point to determine if the clock is running. No pulse, of course, indicates that the clock is off.

Pin 6 is connected to 1/2 of the 74LS74 D FF. The pulse enters pin 11, the clock input. It causes

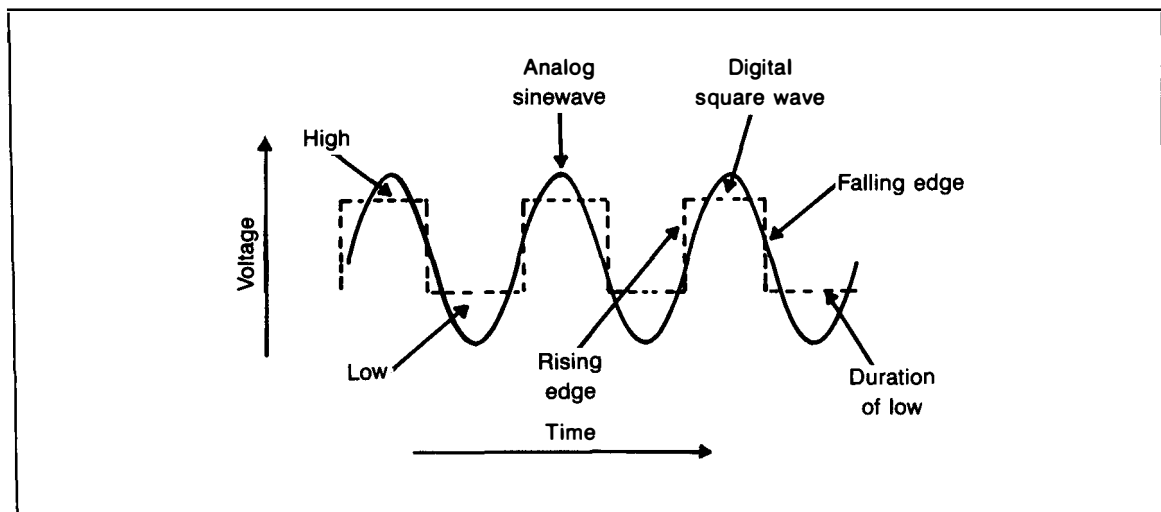


Fig. 14-1. The oscillator produces an analog sine wave at the crystal frequency. The sine wave must then be converted to digital square waves before the computing circuits can use it.

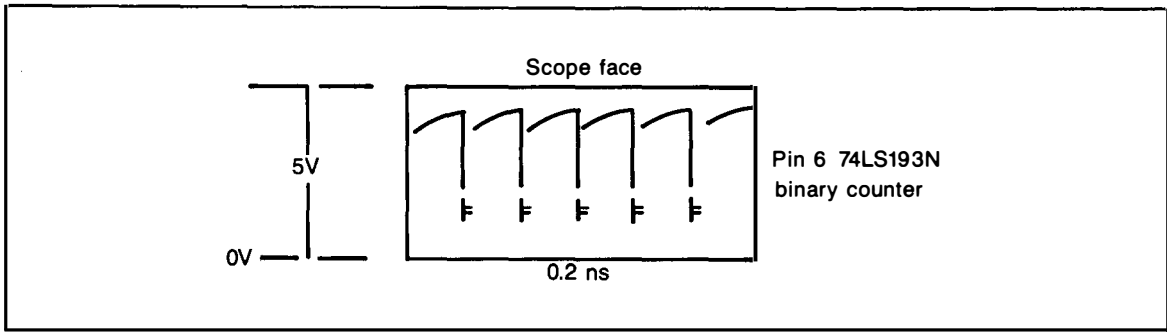


Fig. 14-2. If you take a scope picture at pin 6 of the 74LS193N binary counter at a 0.2 microsecond rate, there should be six square waves.

the FF to count once and then output a clean pulse to the next chip, an MC4044, from Q. Refer to Fig. 14-3. The MC4044 is a Phase/Frequency Detector.

The MC4044 is a tricky little chip that has the job of changing the incoming frequency to an 8.1818MHz signal called DOT CLOCK. The DOT CLOCK is then input to VIC as one of its frequencies.

As the oscillator enters pin 1, a second signal from VIC enters pin 3. The two signals are compared in the frequency phase detector. Refer to Fig. 14-4. The signal from the VIC has a frequency near 1 MHz and is called  $\phi$  zero (phase zero).

The phase detector output is then sent to another section of the chip, called the charge pump. The charge pump is a circuit that builds a staircase waveform at the output. Each incoming pulse charges a capacitor. As the pulses enter, one by one, the capacitor develops a larger and larger

charge. The waveshape forms one step at a time as the capacitor charges.

The staircase builds till a certain threshold is reached. At that point, the capacitor discharges and begins charging anew. This staircase waveshape represents a desired frequency. It is to be fed to the other half of the 74LS629N chip, which is wired as a counter, rather than an oscillator.

Before the staircase is allowed to get to the counter, it is passed through the PN2222A transistor. (Refer to Master Schematic 3.) It exits the transistor out of the emitter and gets amplified slightly and matched to the counter in a YES gate. The transistor-gate circuit acts as a filter to eliminate any ripple that might be in the frequency.

The signal enters the 74LS629N at a pin called Frequency Control. The counter produces the DOT CLOCK frequency. In North America that frequency is 8.1818 MHz. In Europe it is 7.88 MHz. From

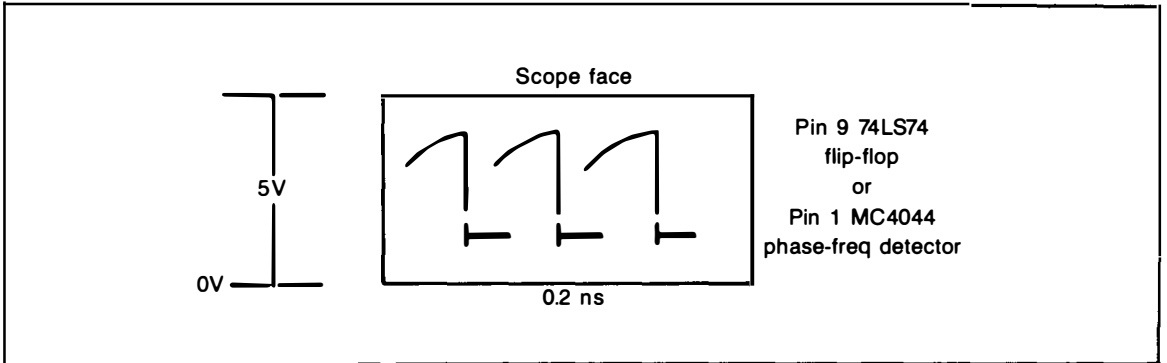


Fig. 14-3. The scope at pin 9 of the 74LS74 at a 0.2 microsecond rate reveals three square waves.

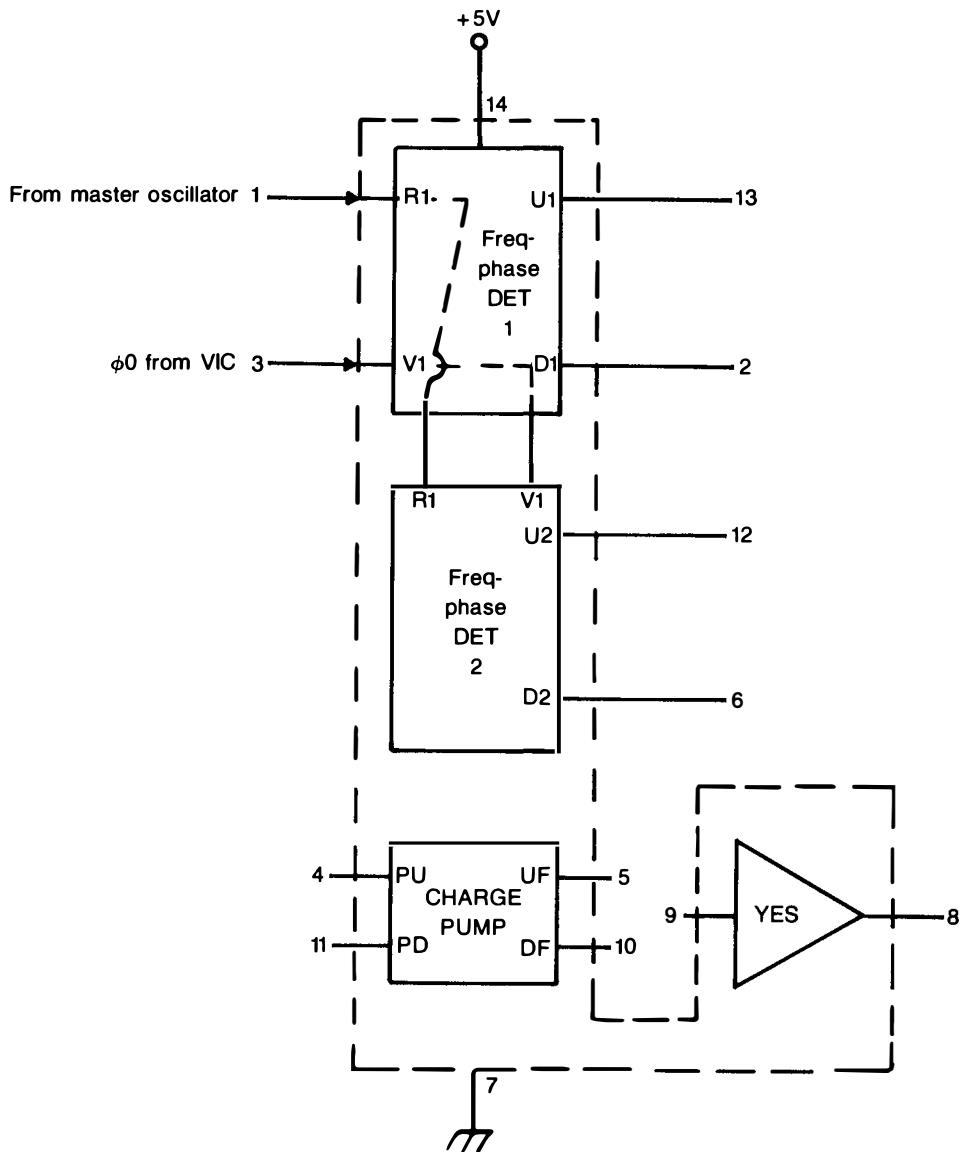


Fig. 14-4. The MC 4044 chip is a Frequency Phase Detector system.



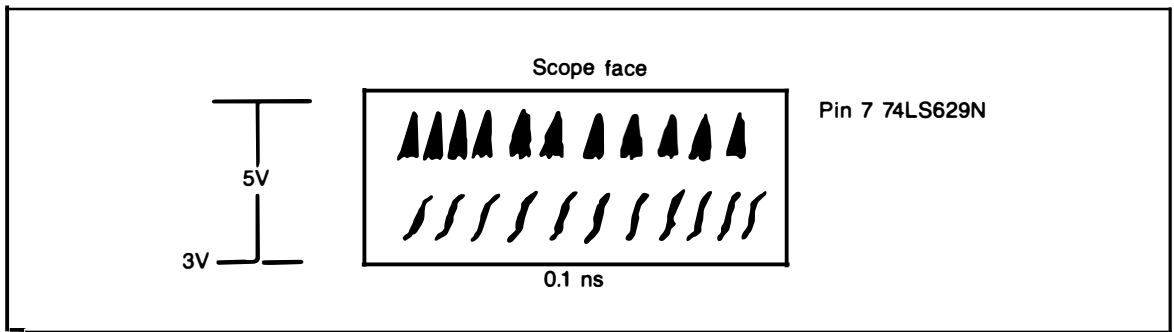


Fig. 14-5. The Dot Clock frequency is 8.18 MHz for the NTSC American market. In Europe, 7.88 MHz is used to service PAL. A scope set at 0.1 microsecond will display this Dot Clock pattern.

there the DOT CLOCK goes to the VIC. At pin 7 of the counter, on the far side of the ferrite bead, the logic probe should show a pulse. If you take a scope reading, at 0.1 microseconds you'll see a dozen or so pulses like those in Fig. 14-5. This is a good output test point to see if the clock is performing properly.

### 6510 TIMING CONTROL

The  $\phi$  zero signal from the VIC is input to the processor at pin 1, called  $\phi$  zero IN. The signal, shown in Fig. 4-6, goes directly into a processor chip area called timing control, shown in Fig. 4-7.  $\phi$  zero is the only input to timing control, but there are four outputs. One output,  $\phi$ 2 OUT, leaves the processor at pin 39. The other three inputs are connected to the instruction decoder.

The timing control circuit provides the pro-

cessor with two signals that are 90 degrees out of phase with each other. The 6510 requires these two out of phase signals in order to run. They each pulse at about 1 MHz.

The decoder matches the two clock signals with the action of the address and the data bus. The two internal signals are called  $\phi$ 1 and  $\phi$ 2. These internal processor signals should not be confused with the other  $\phi$  signals running around the digital circuits.  $\phi$ 1 is the square wave that drives the bits on the address bus.  $\phi$ 2 is the other square wave that moves the data between the processor and the memory map.

The timing control area of the 6510 takes the  $\phi$  zero signal and splits it into two signals, the  $\phi$ 1 and  $\phi$ 2 that were just mentioned. The signals are then injected into the instruction decoder.  $\phi$ 1 and  $\phi$ 2 are the working frequencies of the 6510. Fig-

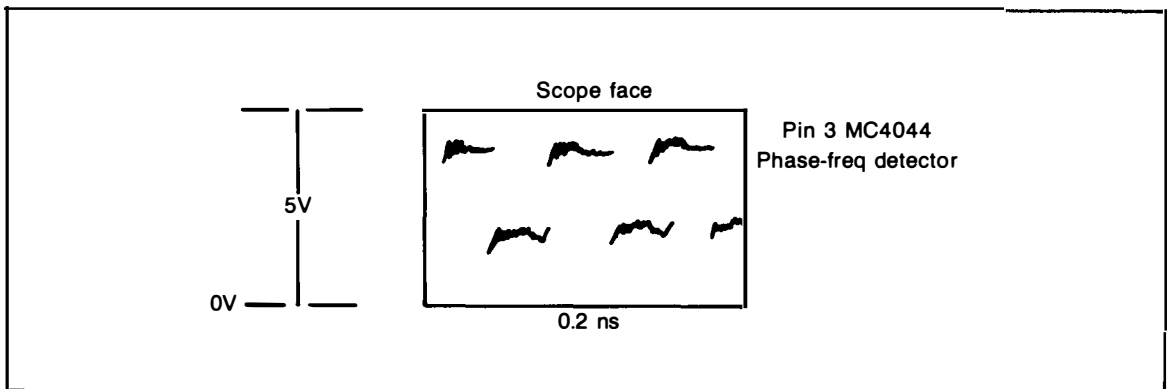


Fig. 14-6. At pin 3 of the MC4044 chip,  $\phi$  zero from the VIC is found when the scope is set at 0.2 microseconds.

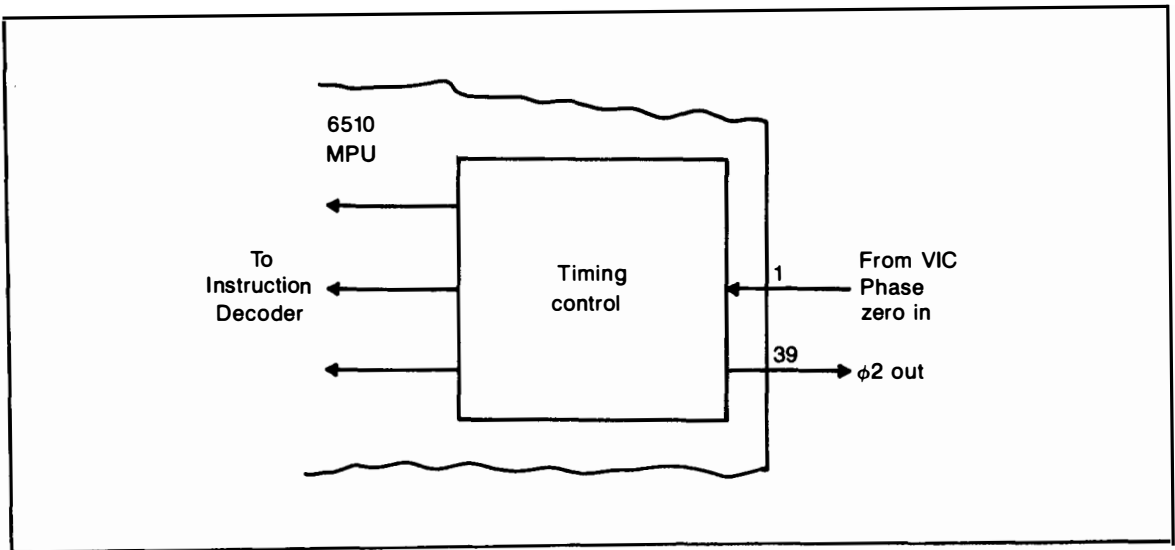


Fig. 14-7. The  $\phi$  zero frequency is input to a phase detector circuit in the 6510. The circuit in turn produces three different forms of the 1 MHz signal that the processor runs at.

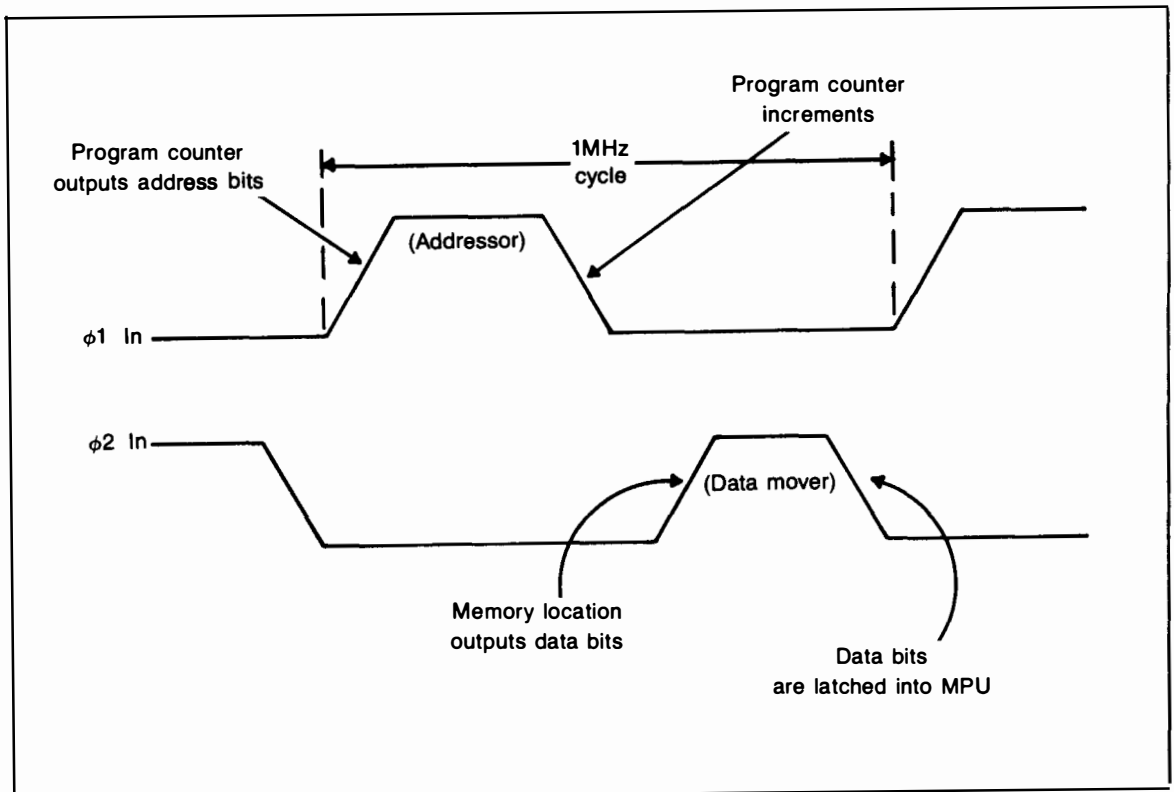


Fig. 14-8. Two of the 1 MHz signals produced are  $\phi 1$  and  $\phi 2$ .  $\phi 1$  drives the program counter while  $\phi 2$  operates the data bus.

ure 14-8 shows the timing of the signals. The two signals are quite alike except for the phasing. Both signals are operating at a 1 MHz frequency but the highs and lows are staggered. The sketch shows that  $\phi 1$  has a high at the same time that  $\phi 2$  is low. Then when  $\phi 2$  goes high  $\phi 1$  goes low. Note the slanted edges that represent the time elapsed between the highs and lows.

Figure 14-9 shows one complete cycle at 1 MHz. That means the cycle takes place in one millionth of a second. Since there are 1000 billionths of a second in one millionth of a second and billionths are called nanoseconds, they cycle takes 1000 ns to occur. The ns is the conventional measurement of the clock cycles. The high and low represent the voltage involved, as usual.

The important part of the cycles, to the computer, is the duration of the high in the cycle and the voltage changes of the edges. The low really doesn't do much but mark time while the highs and edges are occurring. The high in  $\phi 1$  lasts for a duration of 430 ns. The high in  $\phi 2$  lasts for a duration of 470 ns. The slanty rise and fall edges take place in 25 ns.

What are the highs and edges doing? Let's go through one cycle of  $\phi 1$  and  $\phi 2$ . The cycle begins at the bottom of the rising edge of  $\phi 1$ . Remember that the instruction decoder, where the signals are working, is connected through circuits to all the registers and internal bus lines of the 6510. The rising edge of  $\phi 1$  is changing from a low to a high in 25 ns. The fast changing voltage has a trigger effect.

As mentioned,  $\phi 1$  is concerned with addressing. Therefore the rising edge triggers the program counter. It forces the PC to place the address in the counter onto the address bus.  $\phi 2$  meanwhile, is low and not doing much. Note the falling edge of the previous  $\phi 2$  cycle has hit bottom right before the rising edge of  $\phi 1$  took off.

Once  $\phi 1$  puts the address bits onto the address bus, the duration of the  $\phi 1$  high takes place. It lasts for 430 ns. This is plenty of time for the address bits to travel the length of the address bus and open up a location on the map.

During the 25 ns that the edge takes to fall, the

PC is triggered again through another circuit. This trigger makes the PC increment by one. The next sequential address is thus up and waiting in the PC to begin the next cycle with.

As  $\phi 1$  begins its low, the low at  $\phi 2$  comes to an end. The rising edge of  $\phi 2$  begins. The 25 ns trigger of  $\phi 2$  places data on the data bus. If the operation is a read the data in the addressed location is placed on the bus. Should the operation be a write, the data in 6510 is placed on the data bus. Either way the rising edge of  $\phi 2$  places data on the bus.

Next, the duration of  $\phi 2$  high takes place for a period of 470 ns. During that time, the data is able to travel from the memory to the 6510 or from the data bus buffer in the 6510 to memory. At the end of the  $\phi 2$  high, the falling edge either latches the data from memory into the instruction register of the 6510, or latches the data from the 6510 into the map chip location.

The 6510 knows to ship data to the memory map when it receives a STORE instruction. It will direct the data to its instruction register or data bus buffer when it is following a LOAD instruction. These two instructions are the most used members of the instruction set.

## OTHER TIMING SIGNALS

The 6510 has a need for some other timing signals. They are produced in the computer as a byproduct of the  $\phi 1$  and  $\phi 2$  signals. There are five of them. Four are generated in the 6510 and one in the VIC. The first signal is an edge to trigger the data bus direction. It is the  $R/\ast W$  signal.

The  $R/\ast W$  signal is a steady state that the instruction decoder outputs. The  $R/\ast W$  line is connected to the data bus buffer and also to the output pin 38 of the 6510. When the line is high, the buffers are enabled to receive data from the data bus. If the line is low the buffers are enabled to transmit data to the data bus.

When you look at the  $R/\ast W$  condition with reference to  $\phi 1$  and  $\phi 2$ , certain coincidences become apparent. The state remains steady as  $\phi 1$  and  $\phi 2$  begin a cycle. Both of their edges take place and the  $R/\ast W$  state remains steady. The state stays

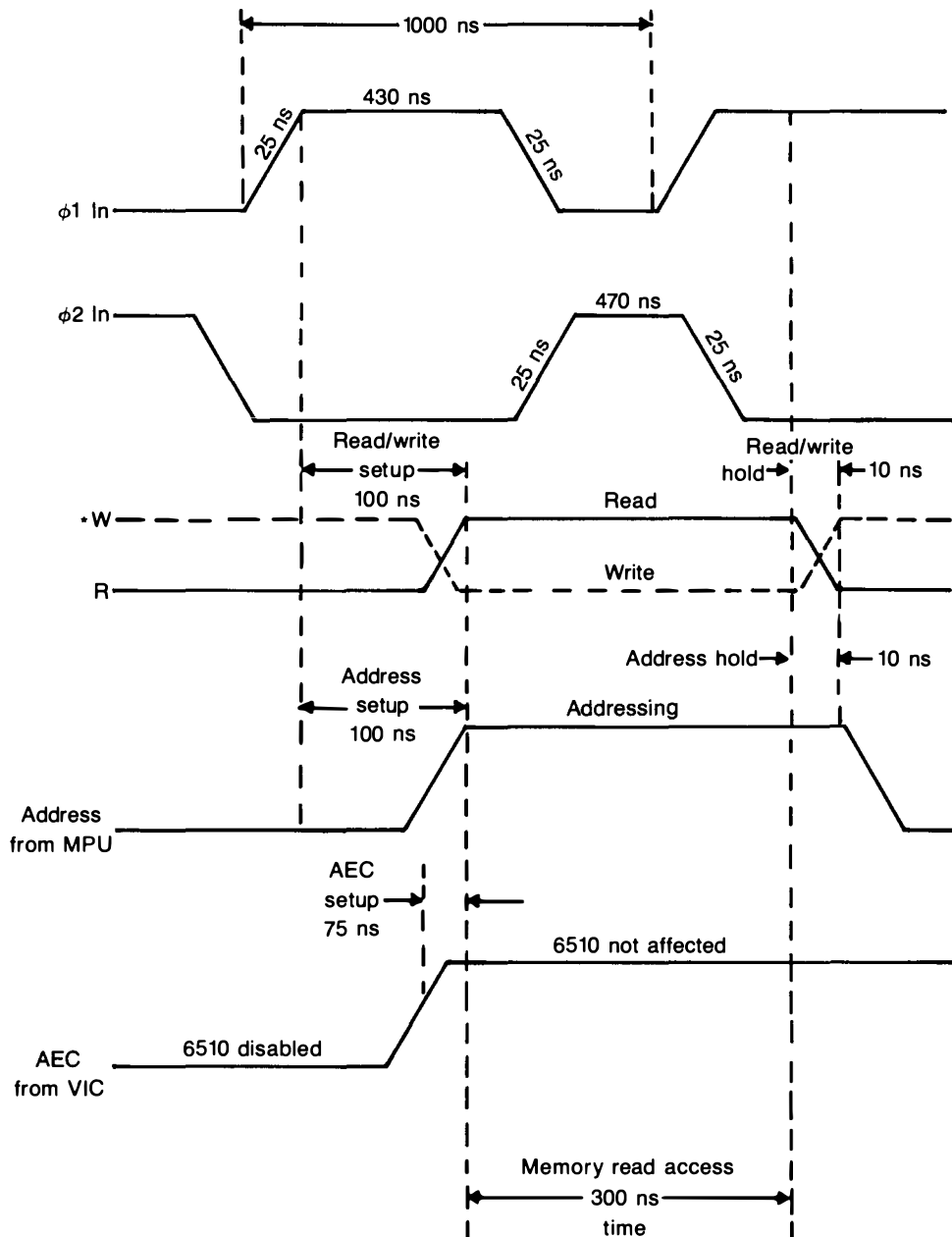


Fig. 14-9.  $\phi 1$  IN is the signal that gets the addressing done. Note it is high while the Read goes high or the \*Write signal goes low.  $\phi 1$  is also high when the address bits from the MPU enter the address bus.  $\phi 1$  is also high when AEC goes high and frees the buffers in the address line. When  $\phi 1$  goes low,  $\phi 2$  goes high. During the  $\phi 2$  high, note that the memory access time takes place.

that way for a period of time called *read/write setup time*. This is the time required for the R/\*W state to setup and be very stable. That way when the state changes it won't shake anything up.

The read/write setup time is defined as about 100 ns. At the end of that setup time, the R/\*W is able to safely change states. The setup time begins as  $\phi 1$ 's high begins. The setup time finishes 100 ns into the  $\phi 1$  high duration. If the 6510 is reading, R/\*W, at that time, goes high. Should the 6510 be writing, R/\*W will go low.

Once the line goes high for a read or low for a write, it stays that way for awhile. It remains in that state for the rest of the  $\phi 1$  high and during the  $\phi 1$  falling edge. It continues to hold for the rest of the cycle as  $\phi 1$  goes low. This is called *read/write hold time*. The 6510 requires the R/\*W to hold after the line stabilizes its state at least 10 ns, but it usually will hold for about 30 ns right at the end of the cycle. After the hold time, the line can change, and it won't disturb the sensitive lightning fast signals.

## Address Signals

The  $\phi 1$  clock triggers off the address out of the PC. An address signal from the instruction decoder does the job. The address signal gets its start as the rising edge of  $\phi 1$  goes from low to high at the beginning of the cycle. At that instant, the address Setup Time is begun. The signal travels to the PC. The address Setup Time is typically around 100 ns. At the end of the setup time, the address signal changes states. The setup time is needed with the address line too in order not to destabilize the activity.

After the address signal changes states, the PC can safely put the address bits on the address bus. The address signal then holds the state till the cycle is over. The address must be stable on the address bus at the end of the cycle for an address hold time of at least 10 ns. Typically the address bus will be designed to hold for about 30 ns.

Another address signal in the Commodore 64 comes from the VIC. The VIC in the 64 is almost like a second processor. During game time, it does take the play away from the 6510 and operates the

computer as co-pilot. The VIC does this by using one of its own generated signals that is called address enable control, or AEC. The AEC line runs all around the 64. One of its destinations is pin 5 of the 6510. Pin 5 connects to the three-state buffers that can control the address bits, leaving the 6510 off and on. In this way the AEC line can shut down the 6510 while the VIC is running the operation, or let the 6510 be.

The 6510 must contend with the AEC line even when it is running the computer. It must make sure the AEC is disabled while it is operating. The AEC will disable the 6510 if it is low. It will let the 6510 run when it is high. The 6510 must make sure it is high. It must also observe an *AEC setup time* if the AEC has to be changed from low to high. The AEC setup time is a maximum of 75 ns. The setup time must have concluded by the time the R/\*W and address signals have finished their setup times. There is no need for any AEC hold time. While the 6510 is operating, the AEC line just stays high and does not interfere.

## Data Timing

The R/\*W, address signal, and AEC all were timed with the high of the  $\phi 1$  signal. Data timing works with the high of the  $\phi 2$  signal that occurs about 90 degrees after the  $\phi 1$  in the same cycle. The first thing to observe in a processor is the *memory read access time*. The 6510 has an access time of 300 ns. What does that mean? It refers to the amount of time it is available during a cycle, to receive data from the memory map.

The access time of the 6510 begins during the high of  $\phi 1$ . As the read/write Setup Time finishes, both R/\*W and the address signal have just experienced their rising edge. The memory read access time, shown in Fig. 14-9, begins at that instant. The 6510 from that point in time has 300 ns to either read data from the memory or write data to the memory.

The 4164-2 RAM chips in the 64 have 200 ns access timing. The -2 on the name denotes that time. The 200 ns of the RAM is the speed the RAM is able to send data or receive data. Since the RAM

is so fast, it can easily transfer data with the 6510. The RAM transfers data in 200 ns after it is addressed. The 6510 has 300 ns to complete the transfer. That is plenty of time to do the job with 100 nanoseconds left over.

Once the R/\*W and address signals are active, the access time of the 6510 commences.  $\phi 1$  completes its high and goes through its falling edge.  $\phi 2$  then begins its rising edge. Refer to Fig. 14-10. The trigger effect of the edge enables the addressed chip, and the data concerned is placed on the data bus. During a read, the data heads for the 6510.

It takes the data about 100 ns to travel the data bus from RAM to the 6510. This is sort of a setup time period. Once the data arrives at the 6510, it can enter one of two places. If the data is an instruction code the 6510 prepares to accept it in the instruction register. Should the data be an operand the 6510 will take it in the data bus buffer. The program dictates which goes where.

The data pauses there at the 6510 entranceway while the data stabilizes. The traveling could have upset the decorum of the data. The data stability time period is about 60 ns. During stabilizing time the falling edge of  $\phi 2$  occurs. This opens up the IR or the data buffer. The cleaned up data enters the 6510.

Once the data is strobed into its register, a *hold time* occurs. The hold time is brief, about 10 seconds. The data is then ready for processing in the 6510.

During a write operation the routine is very similar to the read. The access time is the same 300 ns and begins in the same way; when R/\*W and the address signal finish their beginning edge. The  $\phi 1$  high falls and the  $\phi 2$  rising edge takes place. The memory chip is enabled and the 6510 places data on the data bus. The data speeds towards the addressed location. A data setup time for the write operation takes place. It is about a 100 ns time period. The data from the 6510 then approaches the memory location.

Again there is a data stability time segment of about 60 ns. The falling edge of  $\phi 2$  occurs. The stable data is stored into the RAM location. A 10 ns hold time takes place. The transfer is complete.

## Reading and Writing to Peripherals

When the 6510 wants to read data from a peripheral, all it has to worry about is the setup time. The R/\*W and address signals arrange the direction of the data bus and open up the register in the peripheral. This is all done during the rising edge and high of  $\phi 1$ .

A setup time of about 150 ns is then needed to stabilize the data bus to receive the data. Again, refer to Fig. 14-10. Once the setup time has elapsed, the rising edge of  $\phi 2$  occurs and the data is placed on the bus. The data travels to the 6510 as the high of  $\phi 2$  takes place. Then the falling edge of  $\phi 2$  comes along. The data is given entry to the 6510's instruction register or data bus buffer for processing.

The writing to peripherals is the reverse operation. The 6510 addresses the peripheral and makes the R/\*W line a low for the write. As the  $\phi 2$  rising edge continues, the 6510 places the data for the peripheral on the data bus. The data travels to the peripheral. The falling edge of  $\phi 2$  occurs and the data is strobed into the peripheral register. A setup time was not needed, but a considerable hold is required. It is called *delay time*. The data needs a 300 ns delay in order to insure that the data to the peripheral arrives with validity. This works fine, except that it slows down the transfer of data to the peripheral.

The way the clock drives the 6510 to address locations and transfer data has a lot of intricate steps. Each step is simple and not confusing. There are just a lot of steps.

## TESTING THE CLOCK

When the computer is receiving power and is not producing any output at all, the clock could have stopped. It is useful then to be able to quick check the clock to see if it is running or not. The best place to check it is at a stage where the sine wave has been converted to a square wave. If you test near the crystal, the test probe could load down the oscillator circuit and kill the frequency even if it is indeed running.

In the 64, a convenient place to test is the in-

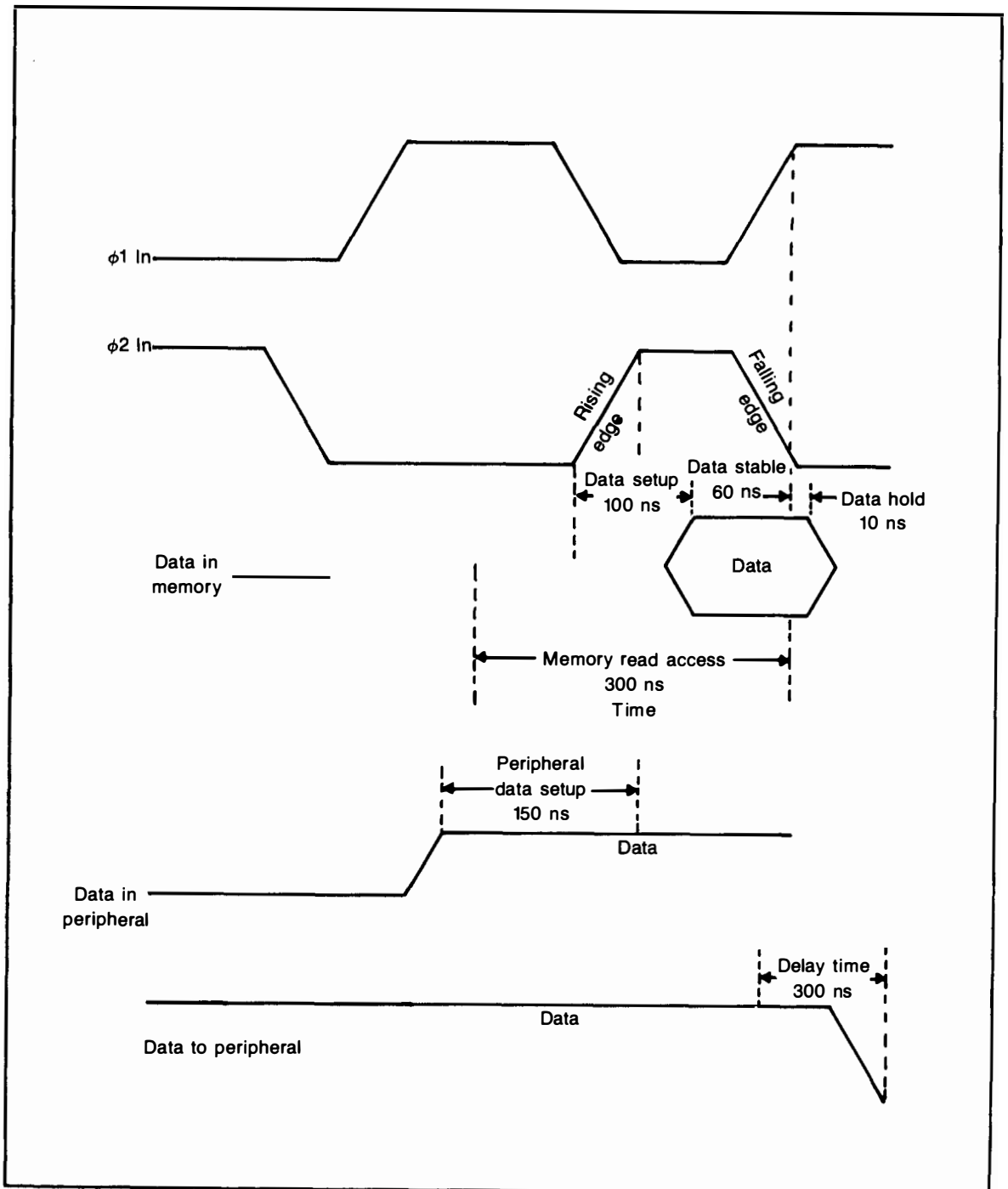


Fig. 14-10. Data from memory must be stable and valid when the falling edge of  $\phi 2$  attempts to strobe it into the 6510. The data needs a 100 ns setup time and at the end of the cycle a 10 ns hold time.

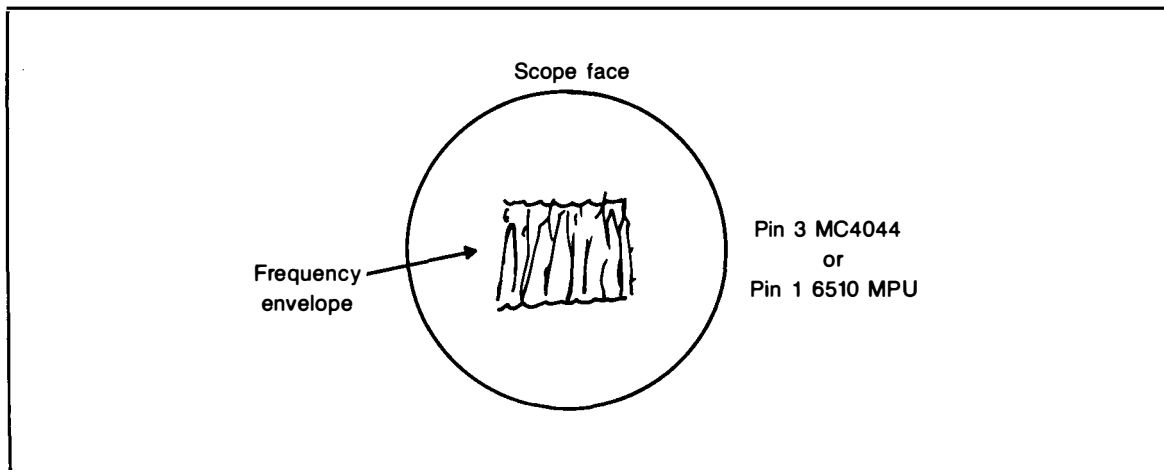


Fig. 14-11. A quick check of the clock can be made at pin 3 of the MC4044 chip even with an ordinary cheap scope. The presence of a frequency envelope usually means the clock is running ok.

put  $\phi$  zero signal to the 6510. The signal there is nowhere near the crystal oscillator circuit. It is a square wave, and it is at its lowest frequency, about 1 MHz. You can use any inexpensive scope with a high impedance probe. An ordinary TV repair scope can't view the 1 MHz signal, but it can produce an envelope containing the frequency. If the envelope, like the one in Fig. 14-11 appears, the clock is running. The clock frequency is also probably ok because the crystal is cut at 14.31818 MHz and won't ring unless the master frequency is near that value.

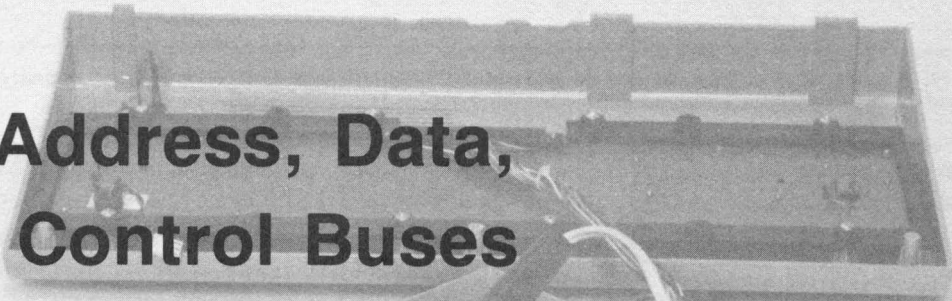
A second quick check can be made with the logic probe. There are a lot of test points that verify that the clock is on. Any of the address or data bus connections should show a pulse on the probe. A pulse means the clock is running. In fact, any of the pins on the test point charts that indicate the presence of a pulse are clock test points. If a pulse is on any of these modes, the clock is oscillating. All of these pulses are originated in the clock circuit. Of course, if there are no pulses anywhere in the machine, the clock has stopped.

The vom will also provide a valid go-no go test of the clock. The needle will read sort of a three-state value around 2 volts on the meter face. The needle will appear a bit unstable and might show an appreciable wobble. This means a running clock.

It was mentioned in the clock circuit discussion that a frequency counter and an expensive scope could be useful. The frequency counter will provide the actual frequency that the clock is running at. You can set the 2K pot mentioned with the counter. If you should have occasion to use one, try to use a pickup loop rather than a probe connection. The loop will pick up the frequency by induction and not make a physical connection to the clock circuit. This is advisable. A probe connection will work fine, but it is more risky.

An expensive scope will be able to actually display the various clock frequencies. Figures 14-2, 14-3, 14-5, and 14-6 show that the waveforms will actually look like. However it is rare that you'll need these scope details for repair work. The waveforms are a lot more valuable to the design engineer than to the servicer.





## 15. Address, Data, and Control Buses

SOMEWHAT LIKE THE STATEMENT, "ALL roads lead to Rome," this computer's roads originate in the 6510. There are three types of roadways that go to and from the 6510. The eight lines of the data bus go to and from the processor. The address bus though, only conducts the signals out of the 6510. The individual control lines vary. Some are inputs and others are outputs.

Keep in mind that these signals are nothing but highs, lows, and pulses. The lines can be in only four states. They can be high, they can be low, they can be pulsing, or they can be turned off. There is no other working condition that they can get themselves into. If you are testing the lines, those are the states you are looking for. These copper traces that travel around the print board and connect to all the major chips are very vulnerable when the board is exposed. They could be a source of problems during repair jobs. I cover these troubles later in this chapter.

### ADDRESS BUS

The address lines emerge from the 6510 at pins 7

through 23, with the exception of 21, which is GND, ground. Just inside the 6510 are 16 three-state buffers. The lines must pass through the buffers before they can get out of the processor. The buffers can be turned on or off as needed by an input line from the VIC. The control line, called AEC (address enable control), lets the VIC shut down the 6510 when it takes over the computer.

Once the lines become the address bus outside the processor, they are named A15-A0. Some or all of the lines connect to every chip in the memory map. Each chip and every register in each chip is given its very own address. Each address can be opened up while all the rest of the addresses remain closed. The 16 lines can carry 65,536 individual sets of highs and lows. These sets of highs and lows are each a combination to open up one and only one address. If more than one address opens during an addressing operation, the program could crash.

In this machine though, there are two decks to the memory map as seen in Chapter 13. There can be two different locations at one address. It is a duplex arrangement, with each deck equipped with

a possible 64K addresses. The entire bottom deck contains the 64K dynamic RAM. The rest of the chips on the map are on the top deck.

During troubleshooting, PEEK and POKE can be used to good advantage. They address and respond in the following way. If you POKE a byte of data into an address and the address is a ROM location, the data will be pushed into the RAM location beneath the ROM. Should you then try to PEEK that ROM address, you will read the permanent contents of the ROM and not the RAM contents you just POKEd in. That is the way it works. This can be confusing because there is a byte of RAM beneath every byte of ROM. With this in mind you can test every memory byte and every address and data line with the proper PEEKs and POKEs. There will be more about this at the end of the chapter.

## Address Assignments

The design engineer has assigned all the addresses in the 64 and worked out the combinations of highs and lows to unlock the individual locations. The bottom deck assignments were easy for him. All 65,536 locations were assigned to the eight 4164 RAM chips as shown in Chapter 6.

Each RAM chip is organized in a  $65536 \times 1$  format. This means that each RAM chip has 64K addresses, and each bit on a chip is given an address. There are eight chips with each chip contributing one bit to form 8-bit location. In order to contact a RAM address you must contact all eight chips. Their assigned addresses in decimal are 0 to 65536. In highs and lows that becomes LLLLLLLLLLLLLLLL to HHHHHHHHHH HHHHHH.

Recall from Chapter 13 that the top deck is not as uniform. It is assigned the same addresses as the bottom deck or the address is left blank. When the address is missing on the top deck, the bottom address becomes the place addressed.

There is no top deck address until the BASIC ROM is encountered at decimal 40960. When HLHLLLLLLLLLLLLL is placed on the address bus, the first location in the BASIC ROM is opened up and can be read. The assigned addresses

are shown in Table 15-1. After the BASIC ROM, the VIC II, SID, Color RAM, CIAs, and the Kernal are found. The Character ROM is a third deck perched upon the I/O area with the same addresses as the I/O. The last address in the Kernal is the same as the last address in the dynamic RAM, HHHHHHHHHHHHHHHH.

## Address Bus Connections

As address bus lines A15-A0 leave the area of the 6510, they split off into many directions (Master Schematic 2). One of the most direct connections for the bus is to the dynamic RAM. The lines break into two groups and connect to the 74LS257 multiplexer chips (Master Schematic 6). One multiplexer needs lines A15, A14, A13, A12, and A7, A6, A5, A4. The other multiplexer requires A11, A10, A9, A8, and A3, A2, A1, A0. The 16 bits enter the multiplexers and are coded to open the desired one bit locations on all eight chips.

The address lines are also connected to the PLA chip (Master Schematic 10). The PLA is only the recipient of four address lines. It receives A15, A14, A13, and A12. These are all chip select lines. The four lines are able to form 16 combinations of bits. This lets the PLA choose between 16 chip select options. The PLA is able to select from among the BASIC, Kernal, Character ROMs, the Color RAM, the CIAs and certain cartridge selects.

The other address bits A11-A0 are register selects and are connected directly to the above mentioned ROMs (Master Schematic 7). Note A12 is also connected to the BASIC and Kernal ROMs. When they are selected, in order to choose among 8K registers, 13 address bits are needed. There is no conflict due to the fact that the PLA uses A12 in its chip selection too.

The two 74LS239 decoder chips also receive some address lines. The top one has lines A11 and A10 while the bottom one gets A9 and A8. These two lines to each chip give each chip four options that it can combine with a third input to produce eight select combinations. These chips select the CIAs, VIC, SID, the Color RAM, and the cartridge expansion.

Table 15-1. Residents of the Double-Decked Memory Map.

Address Bits																
Decimal chip Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>64K Bytes of RAM</b>																
0 to 65535	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
<b>Default Map</b>																
0 To 40959	L H	L L	L L	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
40960 to 49151	H H	L L	H H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
49152 to 53247	H H	H H	L L	L L	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
53248 to 57343	H H	H H	L L	H H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
57344 to 65535	H H	H H	H H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H	L H
	64K	32K	16K	8K	4K	2K	1K	512	256	128	64	32	16	8	4	2

While the decoders select the VIC and SID chip, the registers in the chips are chosen directly with address lines. The VIC has 47 addressable registers that the 6510 is able to access. Address lines A5-A0 are able to form 64 address combinations (Master Schematic 4). Therefore address lines A5-A0 are connected to the VIC as inputs. They can choose among the 47 VIC registers and still have addressing capacity left over.

Actually, there are a lot more address lines connected to the VIC. They are output lines from the VIC that are used when it takes control of the computer and accesses memory on its own. This is discussed in Chapter 17.

A5-A0 lines are, therefore, bidirectional. When the VIC A5-A0 lines are inputs, the 6510 is accessing the VIC registers in normal fashion. When the A5-A0 lines are outputs along with the rest of its address lines, the VIC is in charge and doing its own addressing.

The SID is also selected via the 74SL239 decoder chip. The SID has 29 byte-sized registers for sound effects. It uses address lines A4-A0 to contact the registers (Master Schematic 8). The five lines are able to form 32 address combinations. They address the 29 SID registers.

The address bus has another branch to the cartridge expansion plug (Master Schematic 10). All 16 lines, A15-A0, are connected to 16 terminals of the plug. This gives the 6510 the ability to address 64K addresses in any peripheral that might be used in that socket or lets a peripheral connect to every address location in the computer.

## DATA BUS

In comparison to the address bus, the data bus is a simple affair. It has the eight lines, D7-D0. Every register in the memory map, except for the Color RAM, has eight bits. The Color RAM registers are only four bits wide. When the Color RAM is addressed, the lower four bits of the data bus handles the four highs and lows. The higher four bits of the data bus simply respond with lows that are ignored in the scheme of things.

The rest of the map, all with byte-sized

registers, is connected to the D7-D0 of the data bus. The bus is bidirectional. The 6510 is able to read and write on the bus. Every location is connected to the same data lines. This means all the D7s are wired together, all the D6s are connected, etc. Most of the time all the locations are dormant. When a location is addressed, that location and only that one, out of all the 64K in the memory, is accessed. It is read or written to. The rest of the locations are turned off and do nothing.

The data bus originates in the data bus buffers of the 6510. The buffers can receive data from the bus or transmit data to the bus. The instruction register is also connected inside the 6510 but it is an input only device. It can only receive data; it cannot transmit data to the bus.

The bus lines emerge from the 6510 at pins 30 through 37 (Master Schematic 10). D7 is on pin 30 and D0 on pin 37. This gives you a quick way to relate the D line with the pin number during testing. If you are servicing a lot of 6510 chips, it's a good idea to make an effort to remember the pin numbers and the jobs they are doing.

The data bus splits off into many branches. An important branch goes to the eight 4164 RAM chips. The eight chips are numbered 0-7 (Master Schematic 6). One data line is connected to each chip. D0 goes to RAM chip 0, D1 to RAM 1, D2 to RAM 2, and so on. That way a byte is stored with one bit in each chip. Pins 14 and 2 on each RAM chip are tied together and connect to one data bus line. One pin is for inputting a bit, while the other pin outputs a bit.

Other branches of the data bus connect eight lines to the ROMs, the CIAs, VIC, SID, and the cartridge expansion plug. There is one branch that only sends four of the data lines to a chip. The lines are D3-D0 and go to the 4066 Quad Bilateral Switch (Master Schematic 4). The 6510 can send the four lower bits to the Color RAM through these lines. The switch is needed to cut out the data bus here while the VIC is operating the computer. When the 6510 is on, the switch puts the bits right through to the Color RAM. When the VIC is in control it sends an AEC signal and causes the switch to cut off.

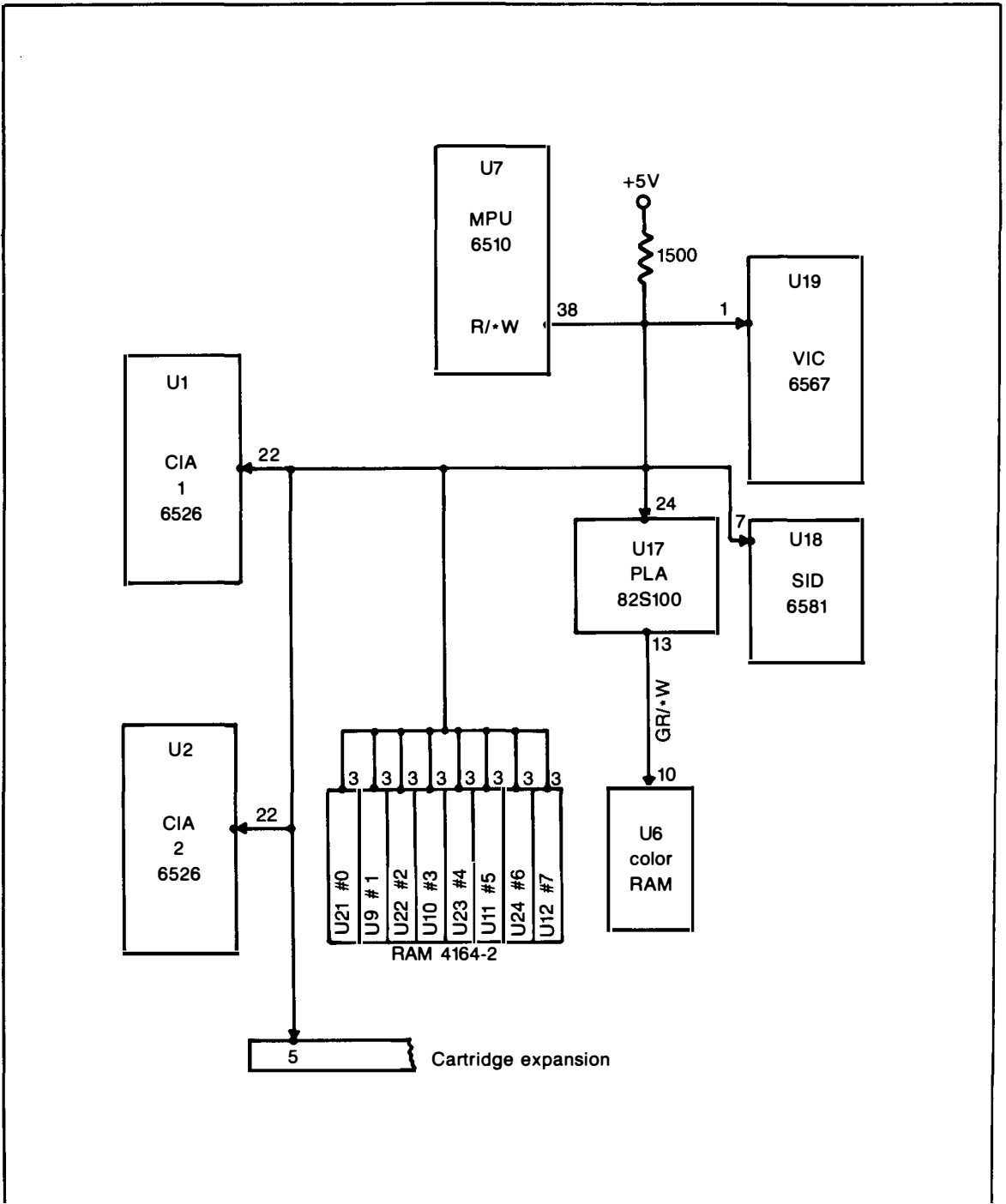


Fig. 15-1. The R/\*W line originates in the 6510's instruction decoder and is sent to RAM, VIC, SID, the CIAs and the cartridge expansion plug.

## CONTROL LINES

The control lines have been loosely referred to as a control bus. Actually the lines are all individual with each one having to do a particular job.

The lines are all inputs, outputs, or both for the 6510. The VIC also has control lines as well as address and data bus lines. These are covered in Chapter 17.

### R/\*W Line

The best known control line is the R/\*W. It is a 6510 output that travels to all the major chips. It connects all the R/\*W pins on the concerned chips together. The entire line is held high through a 1500 ohm resistor to +5 volts. When high, it is in a read mode.

The line, as shown in Fig. 15-1, goes to RAM chips, the CIAs, VIC, SID, PLA and the cartridge expansion plug. The R/\*W signal originates in the 6510 in the instruction decoder. It connects internally to the data bus buffers and then leaves the 6510 at pin 38. It is held high most of the time as the 6510 reads from the memory map. When it is time for the 6510 to write to RAM or an I/O port, the instruction decoder forces it low.

### \*IRQ Line

The \*IRQ line is an input to the 6510. It is held high through a 3.3K pullup resistor from +5 volts, as shown in Fig. 15-2. The 6510 can be interrupted if the line is forced low. A peripheral connected to a CIA can send a signal to make \*IRQ go low and

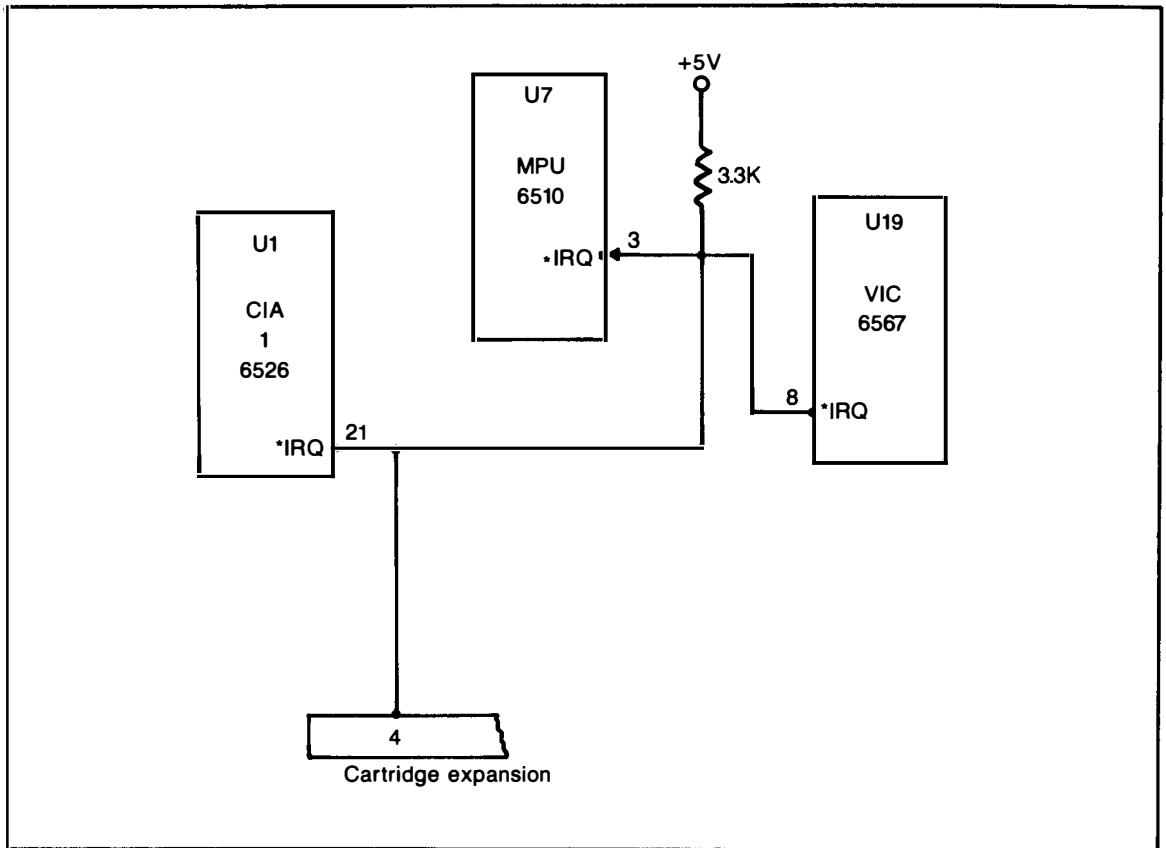


Fig. 15-2. The \*IRQ line originates in the VIC, CIA1 and the cartridge expansion plug. It will interrupt the 6510 when one of these circuits sends a low over the line.

interrupt the 6510. A signal from the VIC out of its \*IRQ output can interrupt the 6510. A low from the cartridge expansion plug can also interrupt the 6510.

The \*IRQ line is important when you are conducting machine-language programming. It performs automatically if BASIC is being used. For servicing, you need to know where the lines are on the print board and be looking for the high or low that will be present under your test conditions.

## Reset Line

The reset line in the 64 is used to get the computer underway when it is first turned on. The line originates at the 556 timer chip and starts up as +5 volts builds at the trigger input (Fig. 15-3). A low from the 556 circuit travels to the 6510, the CIAs, and the SID. The line also connects to the serial bus plug and the cartridge expansion plug. Reset is an input to the 6510 and CIAs. The signal either comes from the timer or some external circuit.

While reset is low, the 6510 is stopped. As the reset receives a high edge, it goes into the reset routine. At the completion of the routine, the processor is then able to resume its normal duties. While there is no pushbutton control for reset in the 64, you can get the circuit to go through its paces by forcing the line low by shorting it to ground.

## $\phi 0$ and $\phi 2$ Signals

$\phi 0$  is an input to the processor from the clock circuit. Refer to Fig. 15-4.  $\phi 2$  is an output that is the same frequency as  $\phi 0$ , about 1 MHz, but a different phase ( $\phi$ ). After  $\phi 0$  enters the processor at pin 1, it arrives at the internal timing control circuit. In the circuit,  $\phi 0$  is converted into the  $\phi 1$  and  $\phi 2$  internal clock frequencies that drive the processor. A  $\phi 2$  signal is also tapped off and sent to the rest of the computer through pin 39.

The  $\phi 2$  clocking has its own lines through the computer. It travels over the lines to some major chips that require the 1 MHz clock to perform. First of all,  $\phi 2$  is delivered to the two CIAs at pin 25. The CIA clocking keeps the CIAs in sync with the

6510 during the input and output processes.

$\phi 2$  is also sent to the SID chip to keep the music in the chip in time with the processor. Lastly  $\phi 2$  makes contact with the cartridge expansion plug. This gives any peripheral that is plugged in there the clock signal from the processor.

The only control lines left are the \*NMI, RDY and AEC. These were covered earlier and AEC will be discussed again in Chapter 17. To review, the \*NMI, nonmaskable interrupt, is used by the \*RESTORE signal from the keyboard. Refer to Fig. 15-5 and Fig. 15-6. RDY, ready, is a signal from the cartridge expansion plug that can cause the processor to be turned off when activated properly.

## TESTING THE BUS LINES

Since the bus lines in any microcomputer cover a lot of print board real estate, they have a lot of exposure to trouble. In the factory, when computers are made, the bus lines can be the source of a lot of problems. Most of the problems happen during the soldering operations. Sometimes the solder spreads all over and forms balls, threads and other odd shapes. The solder can drape itself over the top and bottom of the board and create havoc. The solder can get between traces, from trace to ground and in all the tiny cracks and crevices the components and chip sockets reside in.

In the early days of print board manufacturing, this was an expensive problem, but fortunately, modern techniques have eliminated a large percentage of the troubles. However, the situation still requires attention. When trouble strikes, the bus lines are prime suspects to be checked out. The solder shorting still happens on occasion, open traces can occur, and the components in the bus lines could fail.

As you can see, the bus lines are the wiring of the computer logic. During a trouble one of the first tests you could conduct is the continuity of the traces. With the aid of the location guide and schematic the bus lines can be tested with an ohmmeter one by one. However, there is one precaution you must take. The components and pin

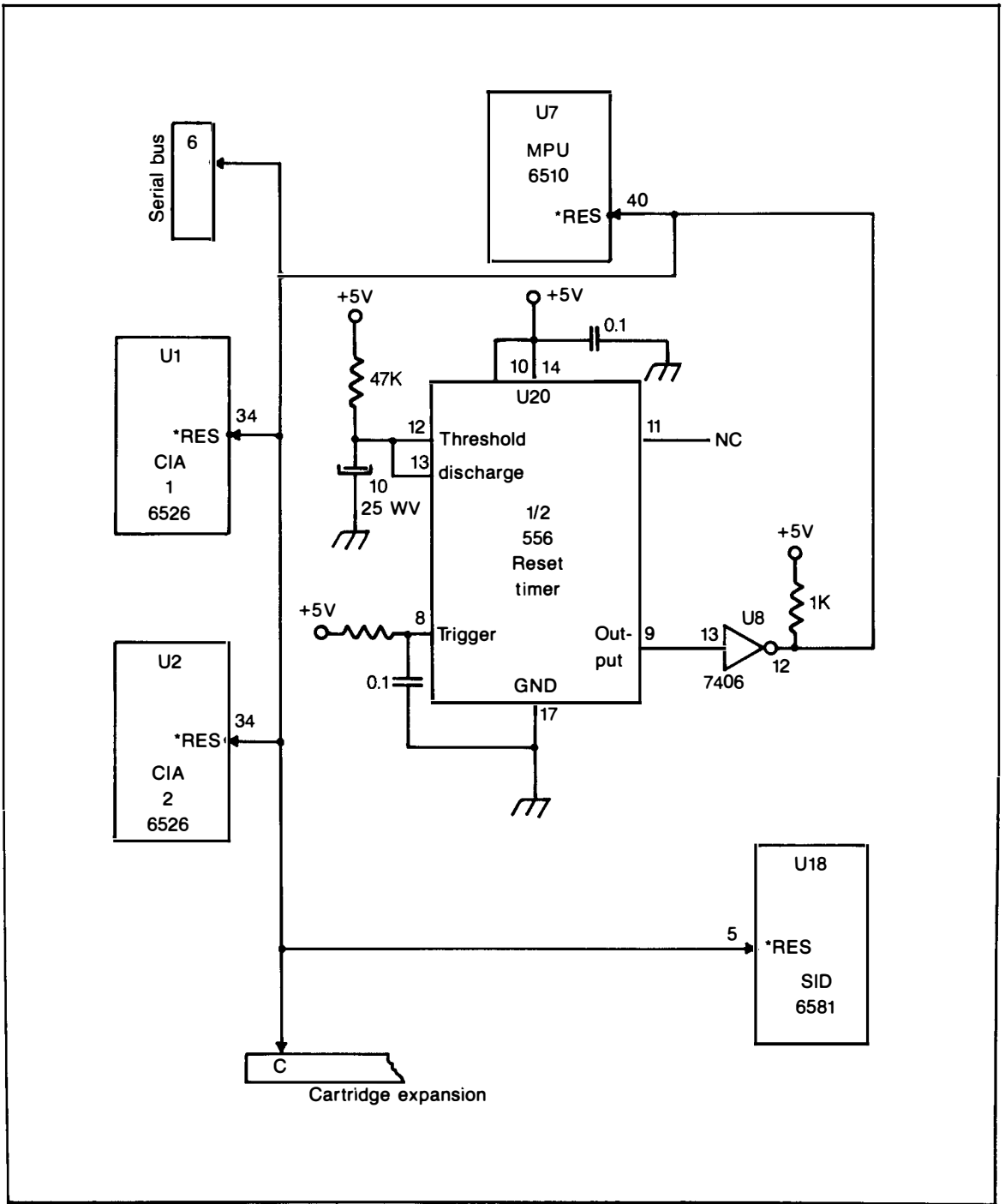


Fig. 15-3. The \*RESET line originates in U20, the 556 Timer chip. It is triggered on when the computer is first turned on and the power supply builds its output from zero volts to +5 volts. The \*RESET circuit gets the system underway.



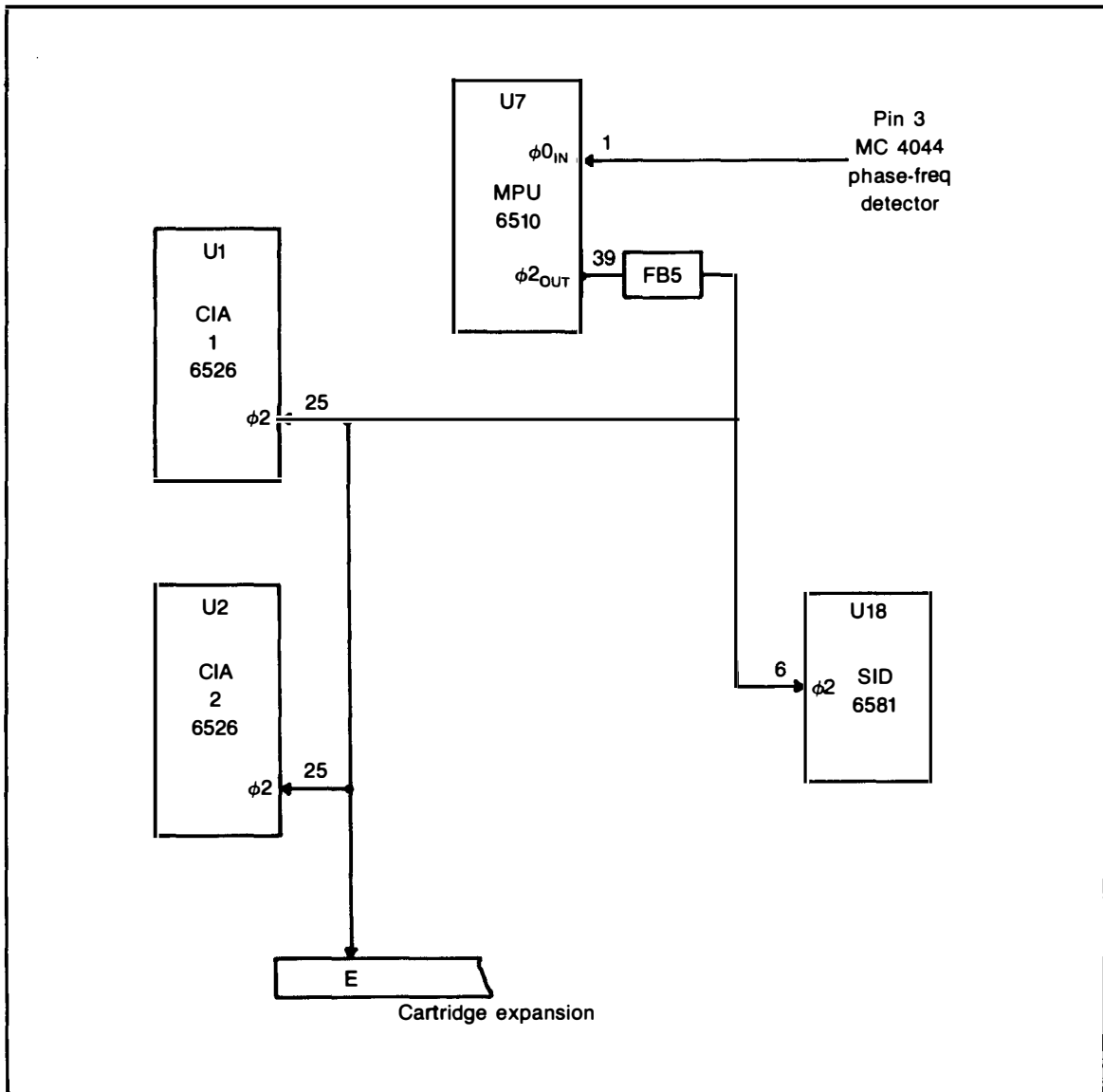


Fig. 15-4. The  $\phi_2$  signal is sent from the 6510 to the SID, the CIAs, and the cartridge expansion plus. It keeps these circuits in sync with the beating of the clock.

junctions of the transistors cannot handle the 1.5 volts the ordinary vom ohmmeter pushes through the circuits for the continuity test. To check out these circuits, you must use a low-voltage tester.

A typical low voltage tester circuit is shown in the Fig. 15-7. The parts of the tester are inexpensive and easily wired together. It will only push

about 1/4 of a volt through the sensitive circuits. The bulb will light when the probes are placed across a trace that is intact. If the trace is broken open, the tester will not light.

The tester will light if you find a trace that is shorted to ground. The tester will not light if a trace is not shorted to ground. Table 15-2 shows the go-

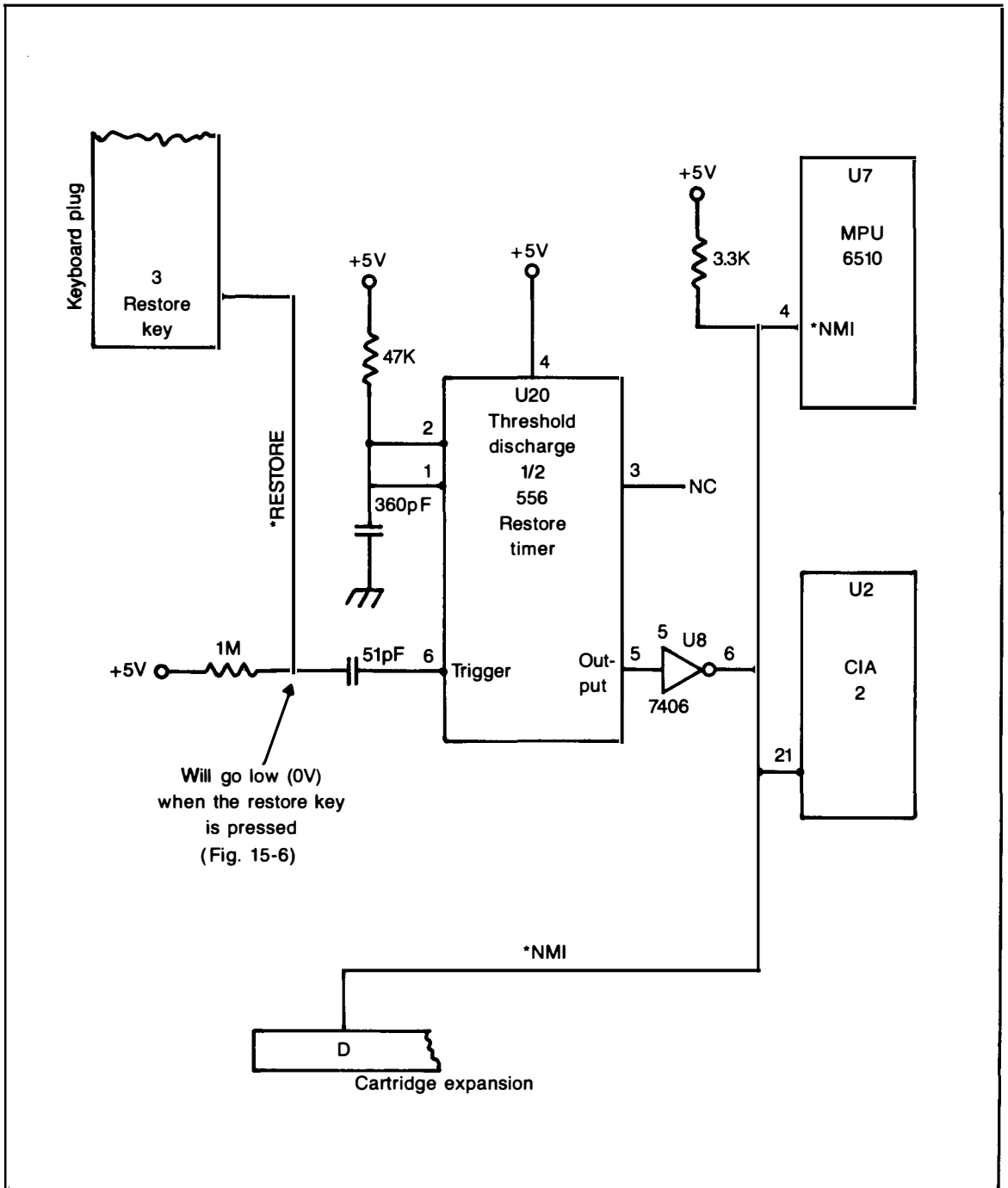


Fig. 15-5. The \*RESTORE signal is produced when you press the RESTORE key. You can test the action by reading pin 3 of the keyboard plug. As you press RESTORE, the pin should go low or to zero volts. If it does not, the key is not working. It probably is not making physical contact inside the keyboard.



Fig. 15-6. The RESTORE key is located on the right side of the keyboard.

For digital circuits  
low voltage continuity tester

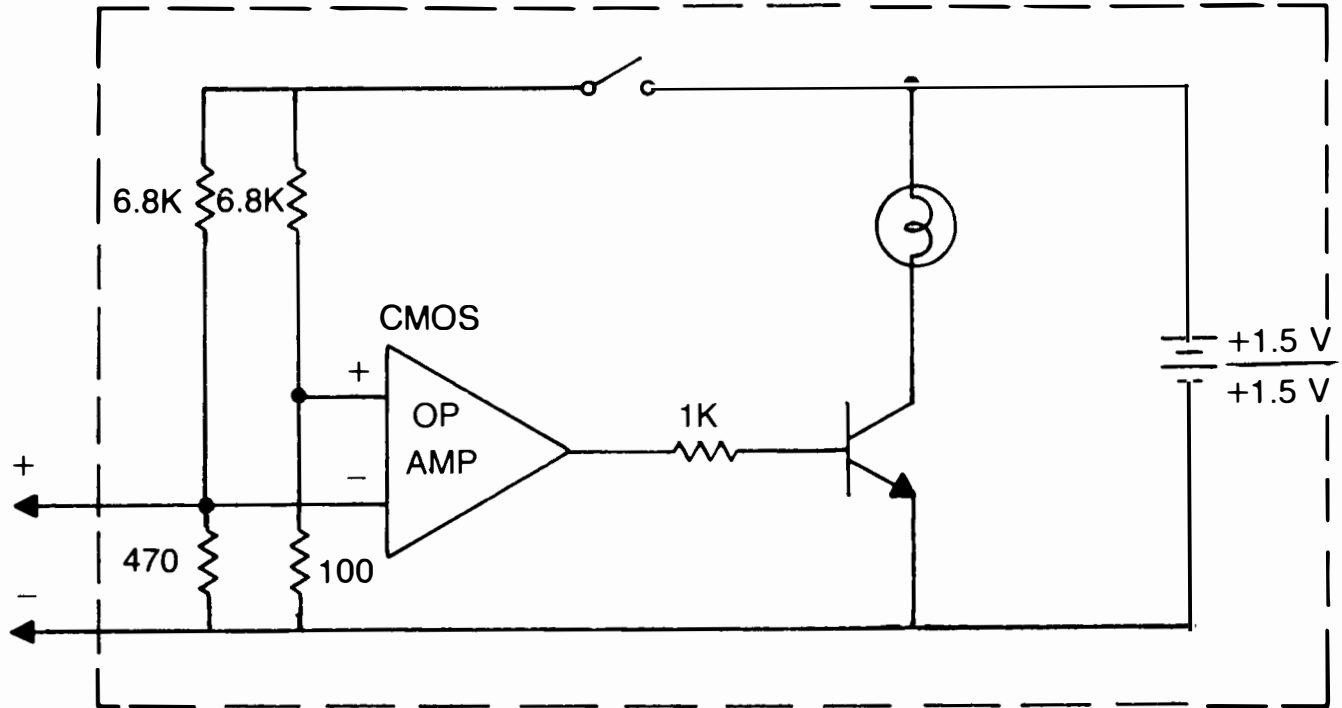


Fig. 15-7. It is risky to use an ordinary ohmmeter on the chips in the computer. It is best to use a low voltage continuity tester. With this easily assembled circuit, you can read continuity while only placing 1/4 volt through components rather than the 1.5-3 volts an ohmmeter uses.

**Table 15-2. Go-No Go Continuity Chart**

Service chart		
<u>SYMPTOM-CONSTANT PROGRAM ERRORS</u>		
RESISTANCE TEST	GO	NO-GO
POINT TO POINT	ALL POINTS CONTINUOUS	OPEN TRACE
TRACE TO GROUND	ALL RESISTANCES FAIRLY HIGH AND ALIKE	A TRACE READS SHORT TO GROUND
TRACE TO TRACE	ALL POINTS ARE HIGH RESISTANCE	SHORT BETWEEN TRACES

no go test results. With the tester, the bus lines can be checked out for opens or shorts very quickly.

### Using the Tester

The continuity tester is one of the most important and most used pieces of test equipment during troubleshooting. You can begin quick checking with the tester, use it off and on all during the repair, and finally check for faults after the repair is over. The tester will usually be the item that pinpoints the short, open or leak that caused the trouble.

As you use the tester, you must be sure the 64 is unplugged. Pull the power supply ac plug out of the wall. Also disconnect the power supply unit from the computer's plug. You do not want any energizing taking place during the testing. This could light the tester's bulb inadvertently and cause confusion.

The best place to view the circuit traces is from

the top of the print board. The bottom of the board, is for the most part, the solder connections. In the 64, a number of the chips are socketed. On the 64 I used as a model, the socketed chips have a yellow paint dot indicator while the soldered in chips do not. It is often useful to remove the socketed chips to get a good resistance test. With the chips out of there you are sure that the readings are of the board wiring and soldered in components only.

The first tests that should be made are point to point continuities. Once the 16 address lines and the 8 data lines are cleared for opens, you can then run tests for shorts. The method is not as foolproof as finding an open trace, but it is worth the effort. Measure the resistance to ground of all the address and data lines one by one.

The results should all be the same. All the trace to ground readings should be a high resistance under normal circumstances. There might be slight variations in resistance, but they should all be high. Should you find one of the traces quite different

than the other, that could be a valid clue that the trouble is nearby. Locate the circuit involved with that trace on the schematic. You might find a reason for the difference in resistance to ground, but I haven't encountered any in different models. Chances are, if one of the traces shows a dramatically lower resistance than the others, you have pinpointed a short that is causing the trouble.

Once the resistance to ground of each trace is tested, check the resistance between traces. Here again there should be a high resistance between all traces. Should you find a low resistance or a short indication, this is a trouble clue. There is probably some sort of short, like a sliver of solder, between the two traces that have developed the low resistance.

## PEEK and POKE Tests

The continuity tests can be run under any circumstances. The 64 can be completely dead or able to do some limited computing. If the computer is dead then you must use the static continuity test if you want to check out the address and data lines. If the computer is operating, but is not addressing properly, or delivering incorrect data, you could use PEEK and POKE tests.

Each line can be tested individually. For example, if you want to test the address lines all you need do is address certain locations and read the contents or write to data to the locations. These locations are chosen for the following reasons.

The address lines are made of 16 copper traces. Each trace is capable of transporting a high or low. A trace is deemed ok if you can send a high successfully from the 6510 to a memory location on that trace. Therefore, if you output an address on the bus that has all lows except for the trace you want to test, which will carry a high, and the address is accessed, then that trace is doing its job and is ok. If that address cannot be accessed, then the trace being tested is not transporting the high and could be the seat of the trouble.

To check out the address lines, the chart of binary addresses in Table 15-3 is used. Each address tests out one of the address lines. The line

that is carrying the high is the one being tested. Refer to Fig. 15-8. To get the addresses into the machine, all you need to do is convert the binary to decimal, and then PEEK or POKE the decimal into the 64. If you are testing an address that happens to land in RAM, you can POKE any number between 0 and 255 into the location and then PEEK to see if it arrived safely. If it did then the line carrying the high is intact.

Should you be testing an address that lands on a ROM you can dispense with the POKE. Just PEEK the address and read the contents. If the contents make sense, then the line carrying the address high is ok.

The chart shows the binary addresses, the decimal code for the binary, and the locations in the model of the 64 I used for the test. Newer models could possibly have different chips at those addresses.

The data bus can be tested in a similar manner. For the data lines though, you need to produce eight data bytes that have all lows except for the data line that is to be tested. That line will have a high.

Table 15-4 shows the eight bytes of bits that will test each data bus line. All that is needed is to POKE the eight bytes into RAM locations one at a time. Next the eight locations are PEEKed and the results of the PEEKs are printed on the screen. Refer to Fig. 15-9. If all of the POKEd bytes are correctly installed in their assigned locations, then the data bus lines are passing the data from the 6510 to the locations ok. Should one or more of the locations not contain the correct contents, the data bus line or lines that were to carry the high to that location could be shorted or open. If there are any latches, buffers, or gates in the indicated line, they are suspect too. For instance, if the binary byte LLLLLLLL did not arrive at the location POKEd then data line D6 has trouble.

## Control Line tests

PEEKs and POKEs can be used directly when the address and data bus lines are tested. PEEK is a BASIC routine that reads an address. POKE

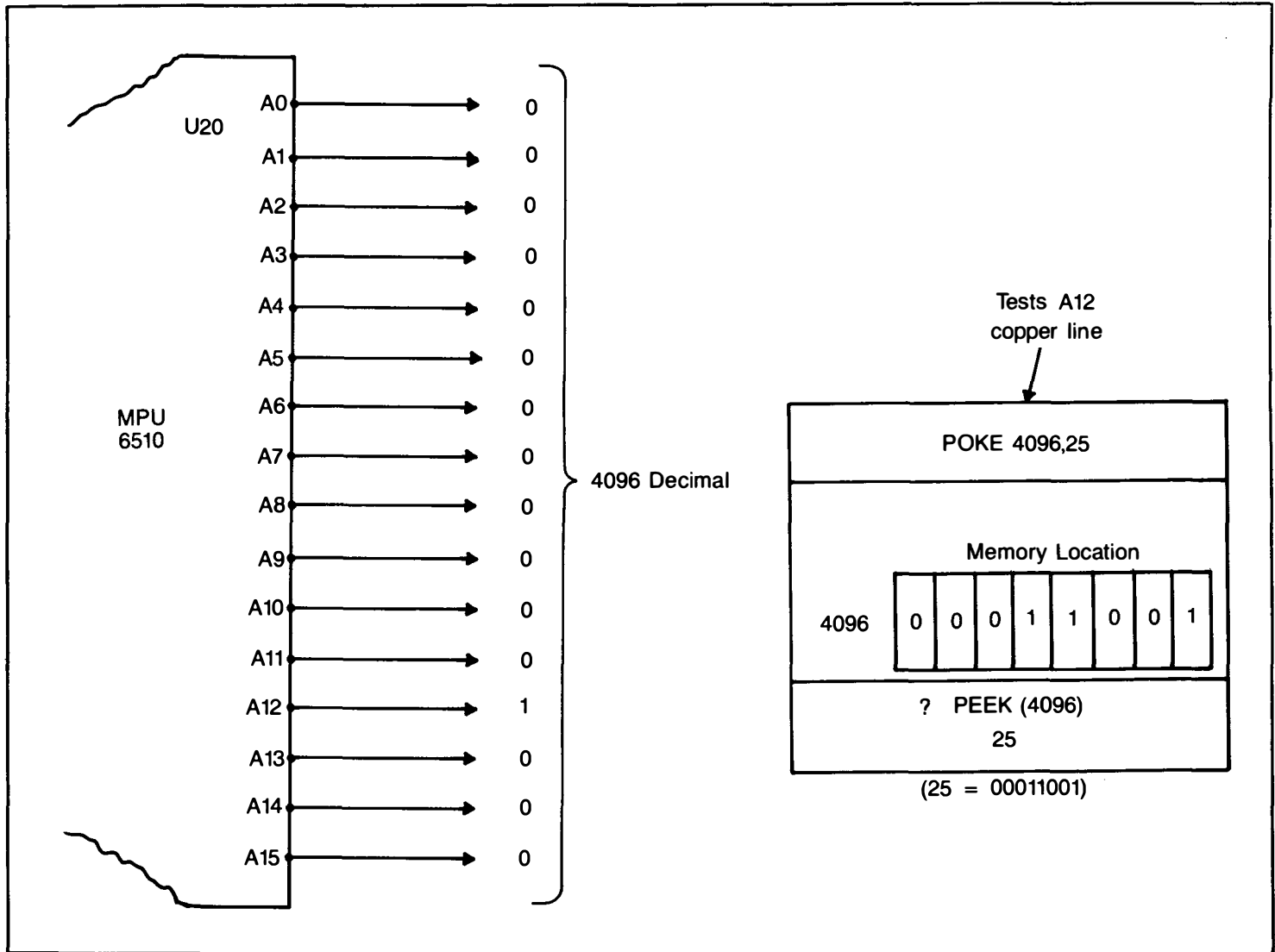


Fig. 15-8. This example is testing A12. If A12 is intact, the address 4096 (binary 0001000000000000) can be POKEd with a decimal 25 (00011001 in binary). After the POKE a PEEK will reveal whether the 25 ever arrived. If it did, A12 is ok.

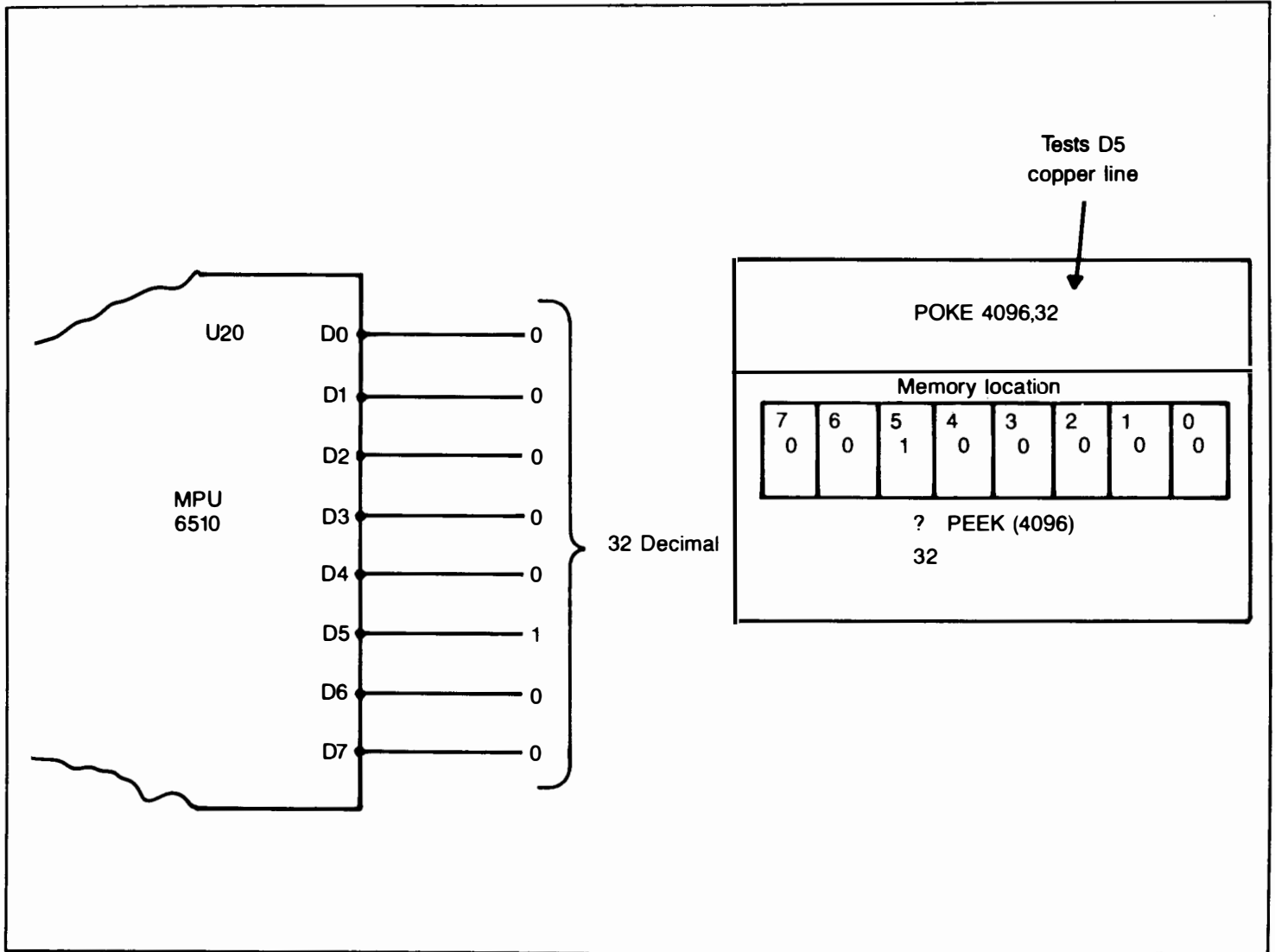


Fig. 15-9. To test D5, the number 32 (binary 0010000000) is sent to convenient location decimal 4096. If the location then returns a decimal 32, the data line D5 is ok.



**Table 15-3. Address Bus Test Chart**

<b>TEST Poke (Copper Address Line)</b>	<b>Binary Address</b>	<b>Decimal Address</b>	<b>TEST Peek</b>
A0	0000 0000 0000 0001	1	RAM
A1	0000 0000 0000 0010	2	RAM
A2	0000 0000 0000 0100	4	RAM
A3	0000 0000 0000 1000	8	RAM
A4	0000 0000 0001 0000	16	RAM
A5	0000 0000 0010 0000	32	RAM
A6	0000 0000 0100 0000	64	RAM
A7	0000 0000 1000 0000	128	RAM
A8	0000 0001 0000 0000	256	RAM
A9	0000 0010 0000 0000	512	RAM
A10	0000 0100 0000 0000	1024	RAM
A11	0000 1000 0000 0000	2048	RAM
A12	0001 0000 0000 0000	4096	RAM
A13	0010 0000 0000 0000	8192	RAM
A14	0100 0000 0000 0000	16384	RAM
A15	1000 0000 0000 0000	32768	RAM

**Table 15-4. Data Bus Test Chart**

<b>Copper Data Line</b>	<b>Binary Data</b>	<b>Decimal Data</b>
D0	0000 0001	1
D1	0000 0010	2
D2	0000 0100	4
D3	0000 1000	8
D4	0001 0000	16
D5	0010 0000	32
D6	0100 0000	64
D7	1000 0000	128

is another BASIC routine that writes to an address. If you are able to PEEK and POKE without any complications then the read/write control line is working ok. Should the line be in trouble, the two BASIC routines won't run. The exercising of PEEK and POKE is an automatic R/\*W line test. If you suspect R/\*W line trouble, you could follow up with a logic probe or vom test.

The rest of the control lines do not respond to PEEK and POKE tests since they do not have much to do with the operation of the address and data bus. The best way to test them is with the logic probe and vom. The predicted test results are shown in the test point charts. Any deviations from these results indicate that trouble is nearby.



## 16. Complex Interface Adapters

**T**HE COMPLEX INTERFACE ADAPTER IS PACKAGED as a 40-pin chip. It has the job of interfacing the digital circuits with all of the external devices that the computer has to deal with. These are inputs such as the keyboard and joysticks and outputs like the serial bus and user port.

Inside the digital part of the computer, the 6510, RAM, ROM, etc. all operate at matching speed, frequencies, voltages, and currents. The 6510 can interface with the rest of the digital circuits by simply attaching buffered pathways and begin operations. Unfortunately, the various peripheral devices do not usually have the same characteristics as RAM and ROM. The 6510 cannot just hook into the external devices. That is where the CIAs come in. The CIA has addresses and an 8-bit data bus connection on the computer side. The address lines and the data bus connect to the processor just as the memory chips do.

On the outside of the CIA, there are two byte size buses that interface with peripheral devices. Inside the CIA are registers that are built to receive

the bytes from the digital circuits, take care of any mismatching, and smooth the way for the streams of data to be output to the peripheral. The two outside buses also receive data from the peripherals and forward the data through the CIA to the internal data bus.

Besides these parallel I/O registers, the CIAs have a serial I/O register. This is a shift register that receives parallel data from the internal data bus and converts it to a serial output. The output leaves through a single pin. The serial register can also handle a serial input and convert it to parallel.

In addition to all that I/O work, the CIA is a dandy little timing device. It has registers that act as interval timers and more registers that provide an hours-minutes-seconds-tenths of seconds real time clock.

### ADDRESSING AND CONTROLLING

The CIA has four pins connected to the address bus, as shown in Fig. 10-1. They are pins 35-38. They are register selects RS3, RS2, RS1, and RS0.

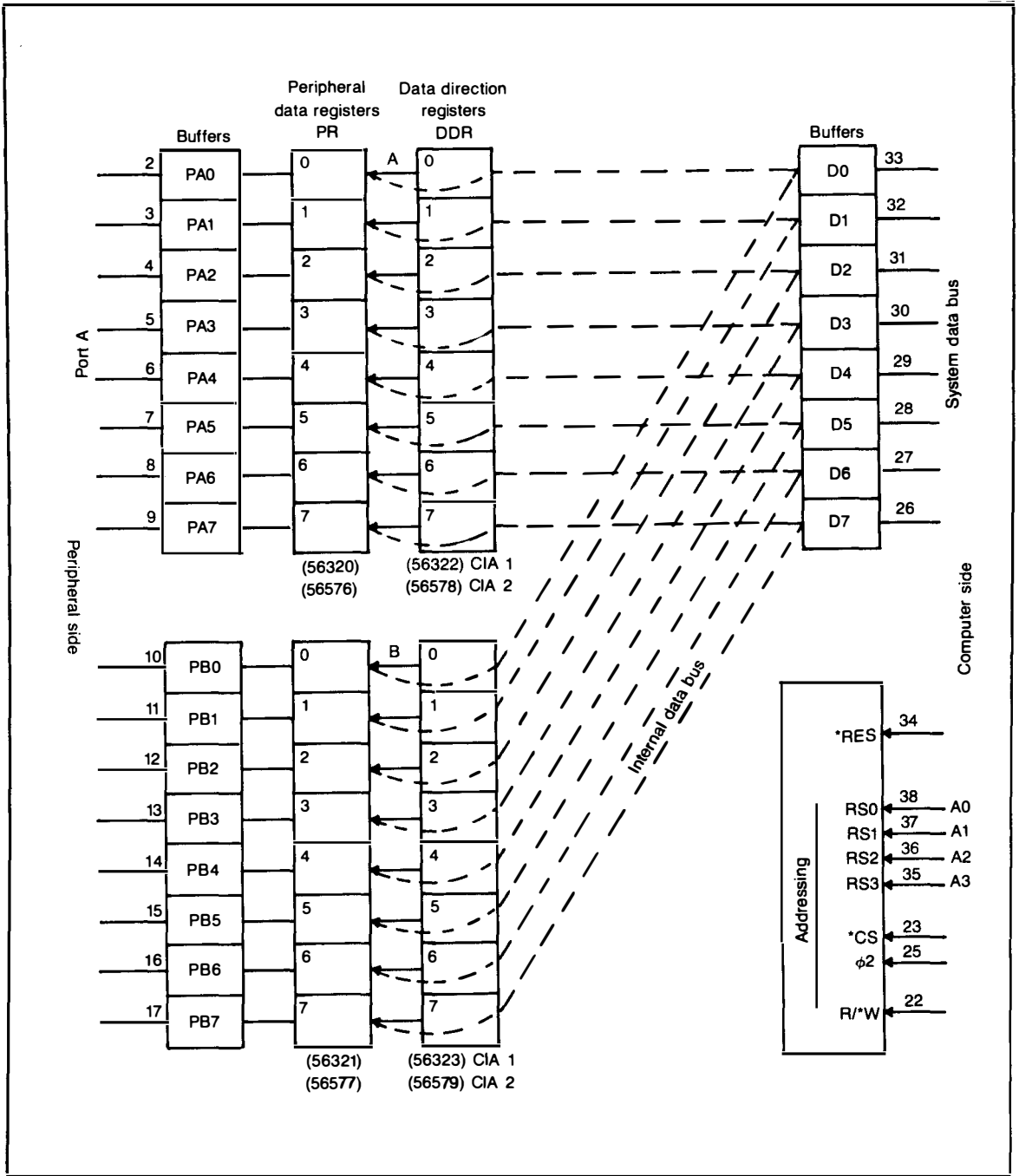


Fig. 16-1. The register selects of the CIA are connected to A3-A0. The chip select is made by the PLA and the 74LS239 decoder chip. However the chip select will not be enabled unless a  $\phi 2$  clock signal also arrives at the same time. Between the address bits that turn on the chip select and the address bits that choose a register, one of the 16 CIA registers can connect to the data bus. Four of the registers are shown here.

They tie into address lines A0-A3. These address lines choose among the registers in the CIA. There are four lines that can select from 16 registers. The 16 registers in the CIA have decimal addresses, 56320-56335 for CIA1 and 56576-56591 for CIA2.

In addition to the internal addressing of the CIA, there is a chip select at pin 23, \*CS. A low here will aid in selecting the chip. The signal for the chip select comes from the decoded signals out of the PLA and its accompanying chips (Master Schematic 10). The chip won't actually be selected unless a  $\phi 2$  clock signal at pin 25 is simultaneously high. This is a timing reference so the CIA can be in sync with the pulsing of the data bus.

When the \*CS is low and the  $\phi 2$  is high at the same time, then the CIA will go to work with the R/\*W line and the addressing bits. If R/\*W is high, the 6510 can read the CIA. If R/\*W on pin 22 is low, the 6510 is able to write to the CIA.

The \*RES, at pin 34 is usually held high and is not active once the computer is operating. If the need be, and the \*RES is forced low, all the internal registers in the CIA will be reset. The I/O pins are all set as inputs. The I/O registers are reset at zero. The timer control registers are made zero and the timer latches are filled with highs. All the other registers are cleared to zero.

## INTERNAL DATA TRANSFER

The CIA is connected at pins 26-33 to the data bus lines D7-D0. These pins are held in a three-state condition until \*CS is low and  $\phi 2$  pulses high at the same time. With the R/\*W line also high, the data from a peripheral will be pushed out onto the data bus and can be read by the 6510.

The CIA has an interrupt request line at pin 21 called \*IRQ. It is normally held high by a pullup resistor to +5 volts and is inactive while high. This lets the line have a number of interrupts connected together. If one of the interrupts forces the line low the interrupt acts as described later in the chapter in the ICR discussion.

## TIMING

The CIA responds to the addressing by the 6510

as the  $\phi 2$  clock cycle goes from low to high and then back to low. This takes about 1000 ns. During that time, the \*CS pin is forced low, the R/\*W line is made high for a read or low for a write, the RS pins receive a register address, and the data bus either inputs data or receives data. A timing diagram appears in Fig. 16-2.

The output pins in turn place data into the CIA for a read operation or send data outward if a write is taking place. All of this is shown in the timing diagram. The diagram is confusing at first glance, but taken step-by-step it can be comprehended.

The  $\phi 2$  clock cycle controls the timing of the data transfer. The cycle time is, as mentioned, 1000 ns. This coincides with the accessing by the 6510. The  $\phi 2$  signal has a high pulse that lasts 440 ns. Its low runs 420 ns. The rise and fall times are 25 ns each.

## Write Timing

When the 6510 writes to a CIA, the timing of each signal is measured. The write operation begins by the 6510 addressing and activating the CIA. The processor outputs an address over the address bus. Bus lines A15-A12 enter the PLA and cause the PLA to output \*I/O to pin 1 of the 74LS239 decoder. At the same time, address lines A11 and A10 enter pins 2 and 3 of the decoder.

The three inputs cause an output \*CIAS to exit pin 7 and enter pin 15. Meanwhile, A9 and A8 enter pins 14 and 15. The chip then can output the CIA select signals, \*CIA1 and CIA2. These signals go to \*CS of the two CIAs. The signals will enable \*CS when low.

As the write cycle begins,  $\phi 2$  goes high. For the first 58 ns, the addressing goes through an *address setup time*. The address bits are making their way through the PLA and decoder. Then the signal arrives at pin 23 of the CIA. The signal is a low. It turns on \*CS. The low stays there for the remainder of the time  $\phi 2$  stays high. That is 280 ns.

$\phi 2$  then falls to a low. As it falls, the low on \*CS goes through a quick 10 ns *address hold time* and then rises to a high.

While the chip selection is going on, the

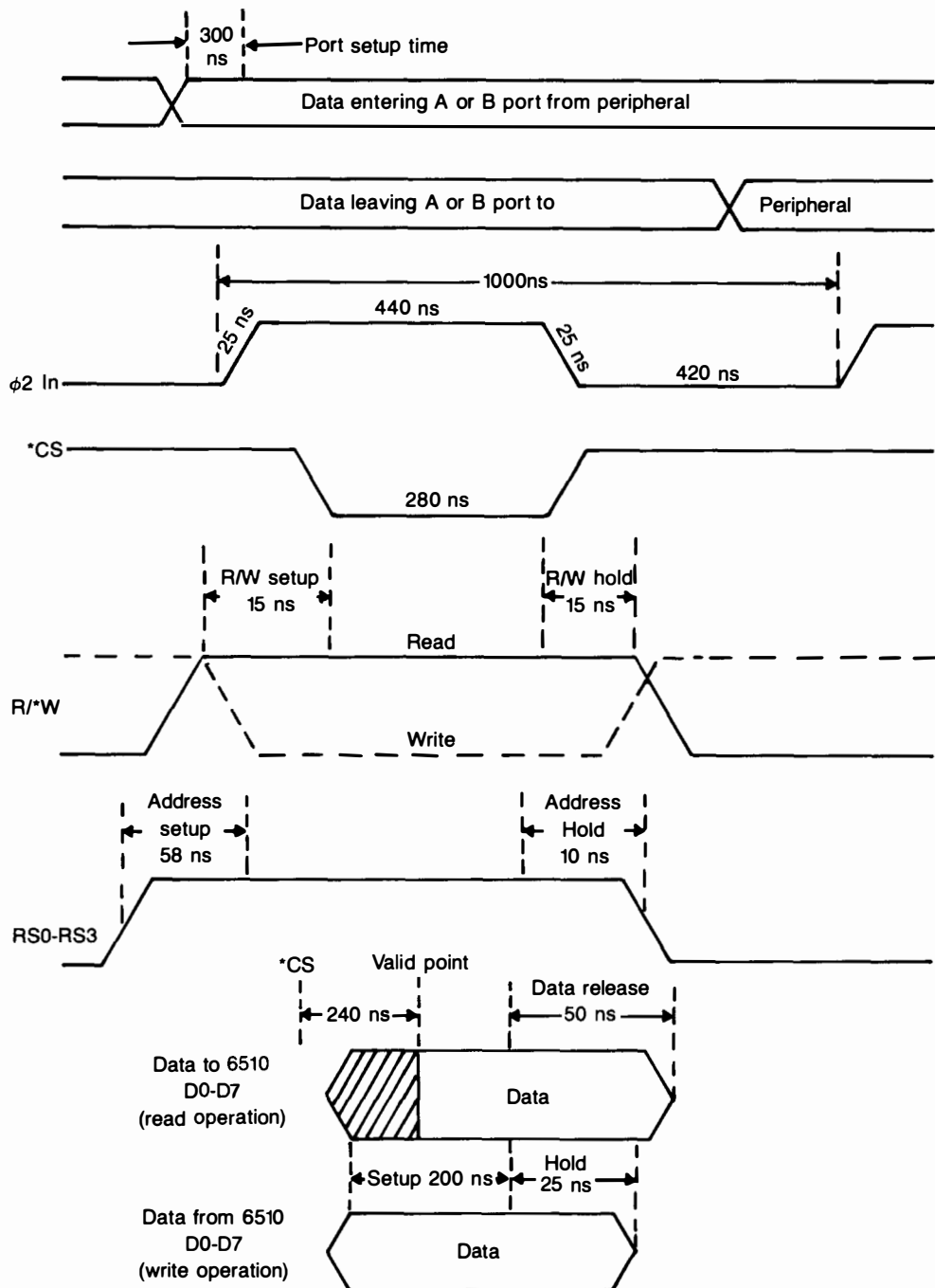


Fig. 16-2. The two top waveshapes are data entering or leaving one of the ports.  $\phi 2$  must be high when that happens. \*CS must be low. A read or write signal indicates which way the data must flow. The register select bits opens up one of the 16 locations. Valid data can then be latched as  $\phi 2$  falls.

registers that are to receive the data are addressed. Address lines A3-A0 open up one of the 16 registers in the CIA by putting address bits into pins 35-38, RS3-RS0.

Also during the  $\phi 2$  high, the 6510 sends the R/\*W signal to pin 22 of the CIA. The R/\*W signal starts off high but then falls as the write operation begins. The R/\*W signal goes through a 15 ns *R/W setup time* as soon as it goes low. It stays low all during the remainder of the  $\phi 2$  high. Then it has a 15 ns *R/W hold time*.

Once the chip is selected, the desired register addressed, and the R/\*W line made low, the data can be written to the CIA. The data leaves the processor, goes out on the data bus, and arrives at the data bus pins of the CIA. It arrives about halfway into the high of  $\phi 2$ . The data must have a data setup time of at least 200 ns during the  $\phi 2$  high. Then as  $\phi 2$  falls the data is strobed into the CIA. The data then must remain valid for another 25 ns, the data hold time.

Meanwhile, all during the operation, the data output pins are waiting for the data to be received by the CIA and pass through it and emerge to the peripheral. The data output is able to remain in waiting for 960 ns of the 1000 ns total cycle time. This is plenty of time for the data to get through and out as the 6510 writes to a peripheral device.

## Read Timing

When the processor wants to read from the CIA, it goes through a similar procedure. The main change is that the data will be traveling from the CIA to the 6510 rather than the other way around. During the read cycle, there are a lot of identical movements of signals and timings. The \*CS low remains 280 ns during the  $\phi 2$  high in the same way. The address setup and hold times are identical. The R/\*W setup and hold times are also the same. However for the read operation, the R/\*W goes high instead of low.

The read operation begins slightly before the  $\phi 2$  cycle. First, the port pins must be open and available to receive data from the peripheral. This is called the port setup time. The ports must be

setup for 300 ns. After the 300 ns stabilizing time the  $\phi 2$  clock will then go high. The data enters the CIA through the ports and is ready to be read by the processor.

The  $\phi 2$  clock at that point in time goes high. The CIA can then be selected, and the desired register can be addressed. As \*CS is forced low, a 240 ns time period begins. The data is let loose onto the data bus but will not be valid till that time elapses and the data settles in place. Once the 240 ns elapses, the data is valid and can be latched into the 6510. This happens as  $\phi 2$  falls to a low. The data then has a short hold time to ensure stability. The hold time is called *data release time* and lasts for 50 ns.

As the processor accesses the CIA, it must do so while \*CS is low and the register is addressed by RS0-RS3 signal bits.  $\phi 2$  must be high, and the data must be valid. If any of these requirements are not met, the data will not travel from the peripheral device to the CIA and over the data bus into the 6510.

## I/O PORTS

The \*CS pin,  $\phi 2$ , R/\*W, RS0-RS3, and D7-D0 pins all work on the computer side of the CIA and communicate directly with the 6510. On the peripheral side of the CIA, there are the pins that keep in touch with the external devices. There are two 8-bit I/O ports and a number of other pins. Let's examine the ports first.

It was mentioned that there are 16 addressable registers in each CIA. The registers are addressed internally through bits entering RS3-RS0. The first four registers in binary are LLLL, LLLH, LLHL and LLHH. These four addresses are to the four 8-bit registers that operate the ports. There are two ports, A and B. Each port has a Peripheral Data Register, PR and a Data Direction Register, DDR. Figure 16-1 shows which addresses contact what location.

The PR registers, through buffers, are connected to the port pins. PRA connects to pins 2-9, PA0-PA7, and PRB to pins 10-17, PB0-PB7. The

DDR registers do not have external pins on the chip. They are connected internally to the PR registers, bit by bit. For instance, DDRA bit 7 connects to the PRA bit 7.

The DDR register bits go one way to their PR register bits. Bits do not travel the other way from the PR to the DDR. The DDR controls the PR. The DDR bits make the decision of whether a PR bit is going to be an input bit or an output bit. If a bit in the DDR is set to a high then the corresponding bit in the PR becomes an output. Should a bit in

the DDR be reset to a low then the same bit in the PR turns into an input.

Once the DDRs are programmed and turn the PR bits into the desired input or output mode, they are not used again unless a PR bit must have its status changed. There are internal circuits that set up the input or output mode of the PRs. During a read operation, the PRs are inputs and will latch the incoming data from the peripheral. For a write operation, the PRs are outputs and will conduct the data right on out to the peripheral.

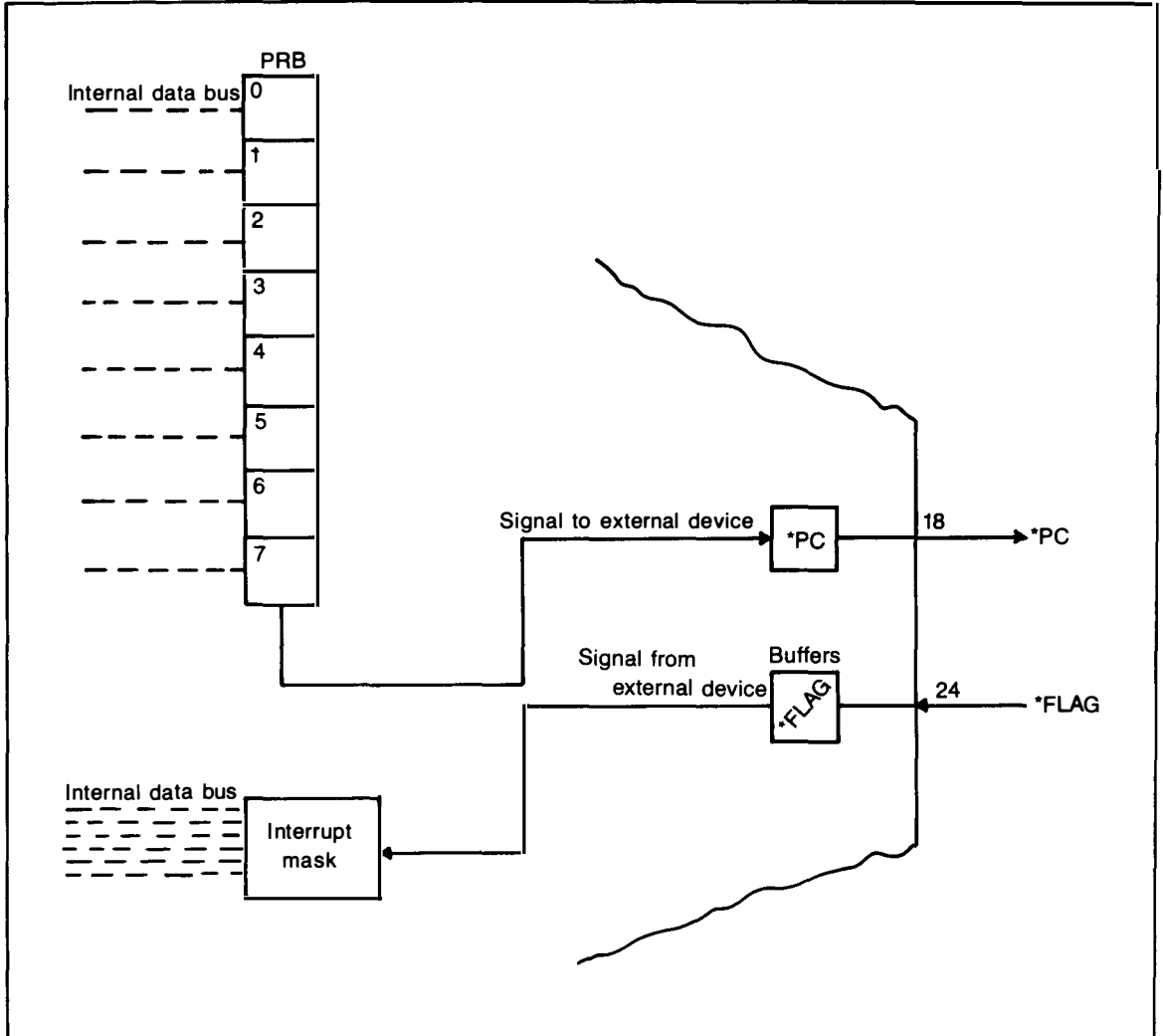


Fig. 16-3. \*PC and \*FLAG of the CIA are an output and an input that is used during handshaking.

## \*PC AND \*FLAG PINS

The CIAs are able to transfer data with handshaking. *Handshaking* is a fairly sure way of receiving data from a peripheral or sending data to the peripheral. Handshaking is a programming chore, but during servicing, there could be times when you need to understand what is happening at the ports as the handshaking takes place. Pin 18, \*PC, and pin 24, \*FLAG, are deeply involved with the handshaking.

\*FLAG is an input pin that connects to the internal data bus of the CIA and, therefore, to the inputs of both the A and B port circuits. \*PC is an output pin that exits only from the B port through a special \*PC buffer stage. \*PC will output a low for one cycle after any read or write of the B port. Refer to Fig. 16-3. During handshaking, this automatic low can be used to talk to another device. The signal could be code saying, "data is ready" or "data has been received."

The input to \*FLAG is also useful during the handshaking operation. If a low is sent to \*FLAG, the signal sets an internal interrupt bit in the CIA. Setting the \*FLAG bit can be used as a code signifying the readiness or reception of data too. As mentioned, handshaking is a programming job. If you want more details, check out a programming book.

## TIMERS

There are two 16-bit interval timer registers in the CIA. They need four addresses since they are 16 bits each, as Fig. 16-4 shows. Timer A has an 8-bit low register and an 8-bit high register. They have addresses of LHLL and LHLH. Timer B has a similar arrangement. Addresses LHHL and LHHH contacts the low and high bits of the register.

The timers have a number of modes. They are very useful in lots of applications. They can generate long time delays in a program, handle different width pulses, pulse trains, and waveform frequency changes. They can count pulses, measure frequencies and other characteristics of pulses that are injected into pin 40, CNT.

Each timer is controlled by another register in

the CIA. Timer A is controlled by an 8-bit register at address HHHL. Timer B is controlled by the register at HHHH. The control registers run the timers. The two timers are similar in operation but not identical.

The timers need two addresses because they use 16-bit registers attached to the 8-bit data bus. In order to read or write to a timer, the 6510 first addresses the low register and accesses it. Then it addresses the high register and accesses it. Actually each timer consists of two 16-bit registers. One is contacted during a write and the other when the processor does a read.

The register that is written to is a latch. The register that is read is the timer counter. Anything written to a 16-bit timer location becomes latched. When the same address is read the processor receives the current contents of the timer counter. Here again this action is important to the programmer. For servicing it is only of value to have an idea of what these registers do.

## REAL TIME CLOCK

There are four registers in the CIA that act as a real time clock. They are shown in Fig. 16-5. During actual programming applications, a real time clock is often needed. This one has four 8-bit registers to handle the timekeeping. Address HLLL counts 10ths of seconds. HLLH takes care of full seconds. HLHL is the minute calculator and HLHH is the AM/PM hours register.

With these registers, the CIA can do 24-hour timing with a resolution of 10ths of a second. The four registers are an electric clock. Like any electric clock it is plugged into the electric company and uses the 60 Hz voltage to time the clock. That is, 60 Hz in this country, other nations could use 50 Hz. The 60 Hz comes from the power supply and enters the chip at pin 19, TOD, time of day. The 60 Hz signal drives the count in the registers.

It is also an alarm clock. The alarm can be programmed to generate an interrupt whenever it is needed. The alarm is conducted by some other internal registers. These registers are at the same addresses as the TOD registers. The control register



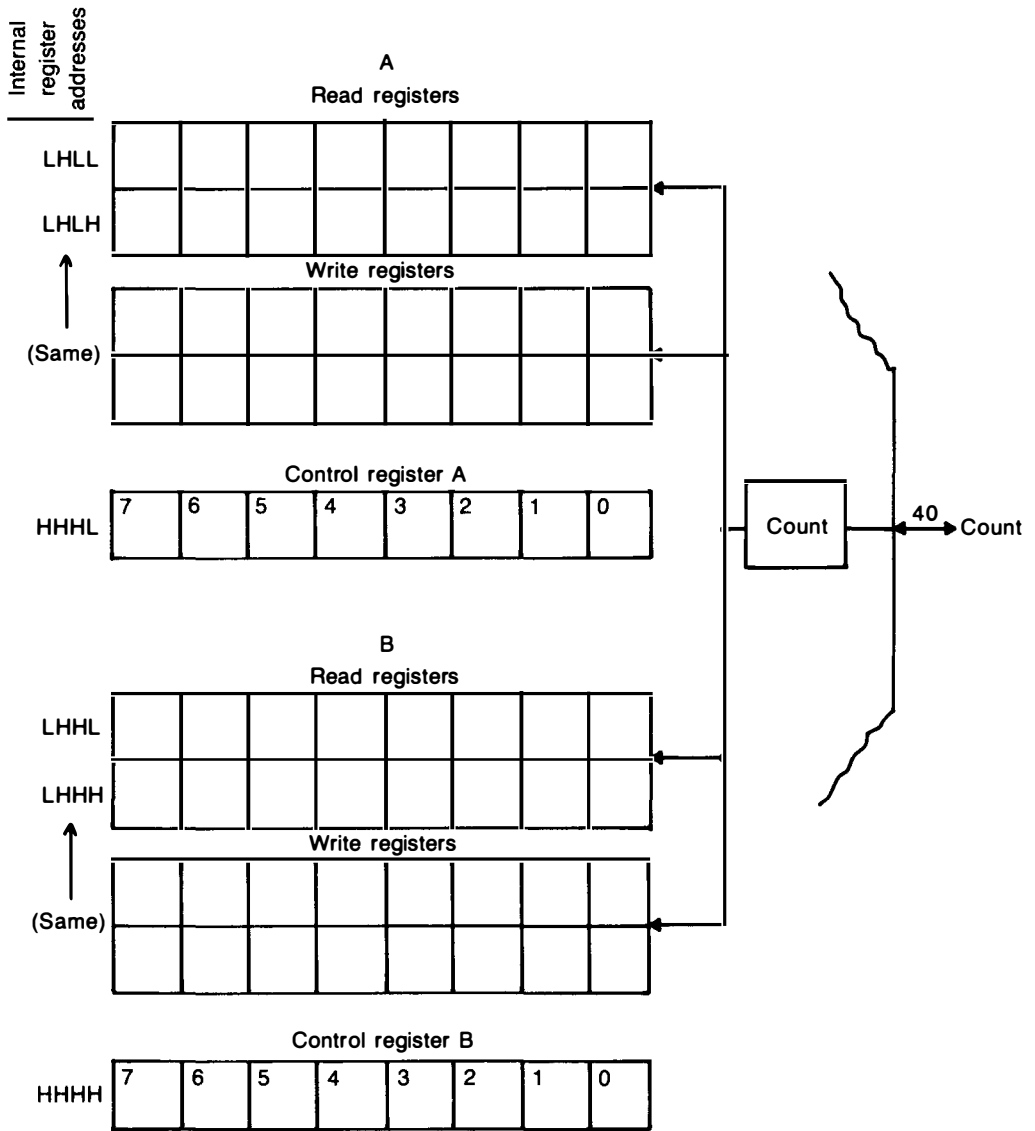


Fig. 16-4. The Timer registers can count pulses, measure frequencies, generate long time delays, and other things. They receive their data at Pin 40, COUNT.

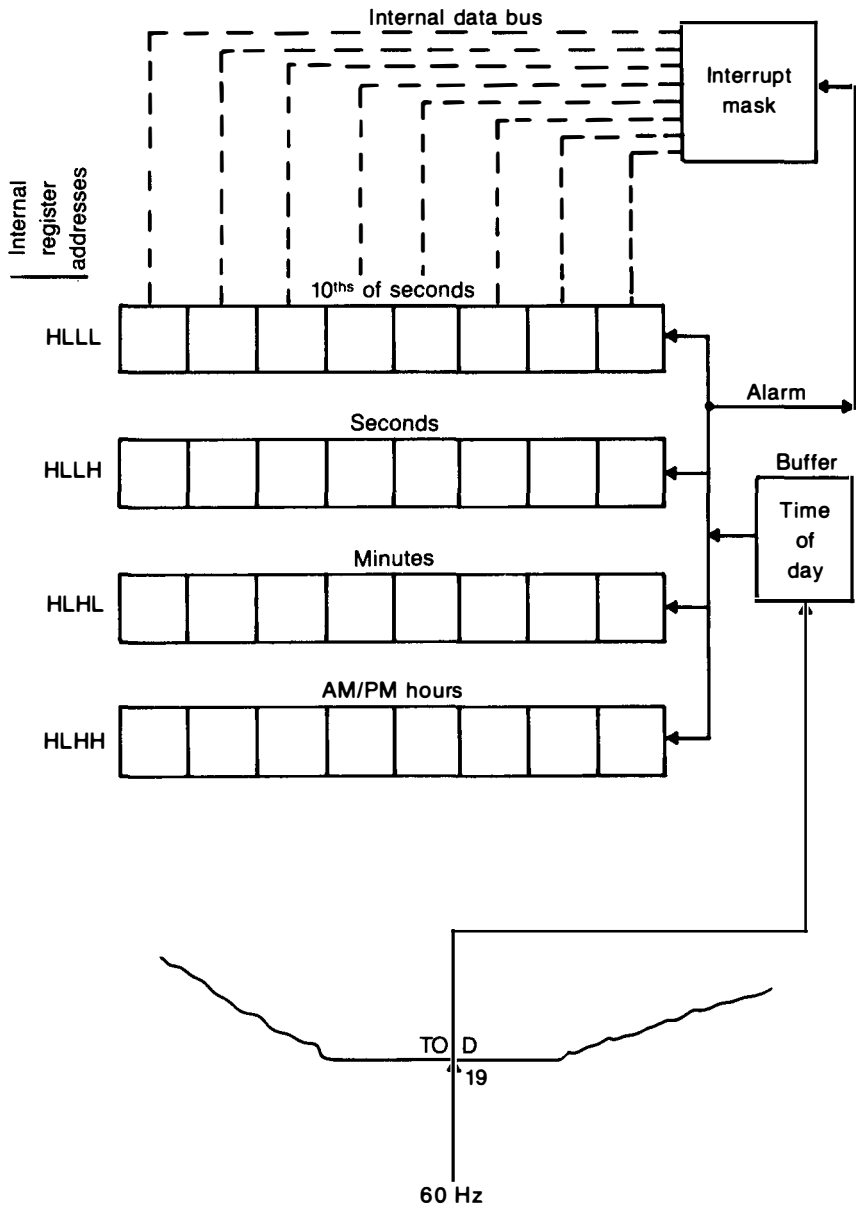


Fig. 16-5. Each CIA contains four registers that act as a real time clock. They count as fast as 10ths of seconds and are controlled by a 60 Hz pulse into pin 19 from the power supply.

for timer B can open up the alarm register. Bit 7 of the control register (Fig. 16-4), if set to a high, allows you to write to the alarm registers and set the alarm. When bit 7 is low, addressing the clock contacts the clock and not the alarm.

Using the real time clock registers is accomplished by careful programming. This is the realm of the programmer and not needed for routine servicing. The servicer should be aware of the clock registers in the CIA and in general what they do.

## INTERRUPT CONTROL REGISTER

The interrupts that leave the CIA as \*IRQ and go to the processor are vital to the operation of the computer. They alert the processor, and the 6510 then goes into its \*IRQ routine. There are five CIA internal inputs into the interrupt register of the CIA. Each type of interrupt is able to set one bit of the ICR.

The internal address of the ICR is HHLH. At the address are two 8-bit registers. There is one register that is contacted when the 6510 writes to ICR. This is the MASK register. The other one, the DATA register, can be reached only if it is being read. Refer to Fig. 16-6.

Inside the CIA, five circuits go to the Data/Mask register combination that is the ICR. Each of the five circuits is assigned its own bit in the Data part of the ICR. An interrupt pulse from any one or more of the circuits will set its individual bit in the Data register.

Each bit in the Data register has a corresponding bit in the Mask register. The mask bit can either let the interrupt through or block it off. When the mask bit lets the interrupt pass, the \*IRQ pin will go low and continue on to the processor. The 6510 will be interrupted and go into its IRQ routine as described in Chapter 12. Should the mask bit not let the interrupt pass, that is the end of the interrupt pulse.

The five circuits and their respective bit positions in the Data/Mask registers are as follows: bit 0 handles underflow from timer A; bit 1 handles the underflow from timer B; bit 2 is used to control the alarm on the time-of-day clock; bit 3

signifies whether or not the serial data register is full or empty; bit 4 indicates the condition of the \*FLAG buffer.

Bits 5 and 6 do not have much use but bit 7 is important and tricky. Bit 7 of the Mask register is called SET/\*CLEAR. It performs the following mask type job on the individual mask bits. Suppose you write to the Mask register and place a low in bit 7. The register will then place lows into any mask bits that have highs. Bits that already are low will remain that way.

On the other hand, if you write a high into bit 7 of the Mask register, the register will place highs into any mask bits with highs and leave the lows alone.

In order for one of the internal circuits to cause an actual interrupt, first a pulse from the interrupter must enter the Data register and set its bit. The Data register then checks the corresponding Mask bit. If the bit is a low, then nothing happens. Should the bit be a high the interrupt takes another step.

The interrupt will set bit 7 of the Data register. When bit 7 of the Data register gets set, pin 21, \*IRQ, goes low. The interrupt is then on its way.

## SERIAL DATA REGISTER

At CIA internal address HHLL, there is a complete I/O port. This is the Serial Port, SP. It connects to pin 39. Inside the chip the pin is attached to the SP buffer stage and then onto the serial port register. Refer to Fig. 16-7.

As discussed earlier, the serial port works as a shift register. It connects to the internal data bus and is one of the interrupt circuits that can make \*IRQ go low. The shift register has its parallel side connected to the eight bit data bus and the serial side to the SP buffer. It also attaches to the CNT buffer and the two timers.

The serial port can be placed into an input or output mode. Bit 6 of the control register A sets the mode (Fig. 16-4). If a high is put in bit 6, the serial port is an output stage. When a low is placed in bit 6, the port acts as an input.

In the input mode, the CNT pin 40 controls the

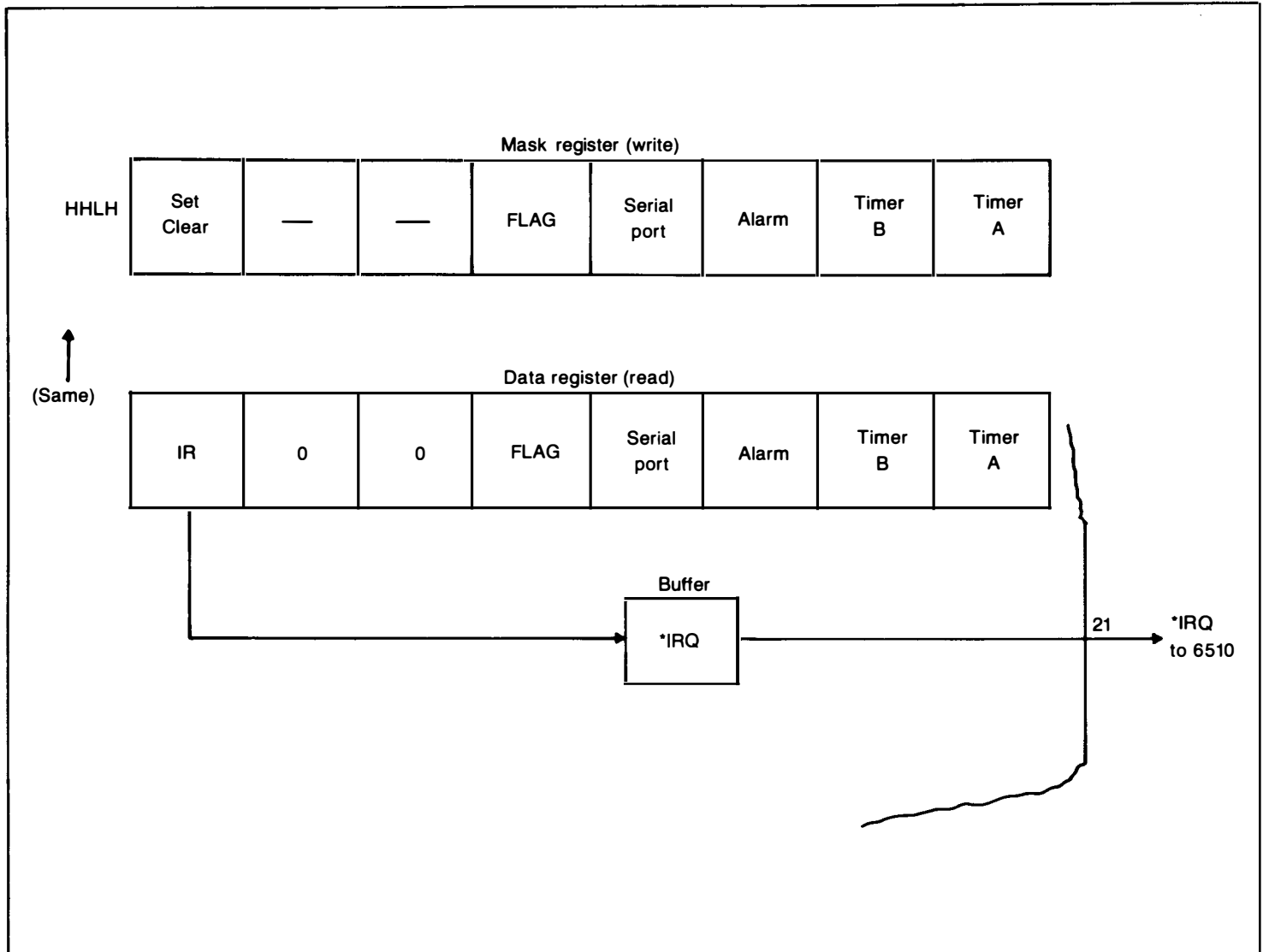


Fig. 16-6. The Interrupt Control Register is able to send the low signal \*IRQ to the 6510.

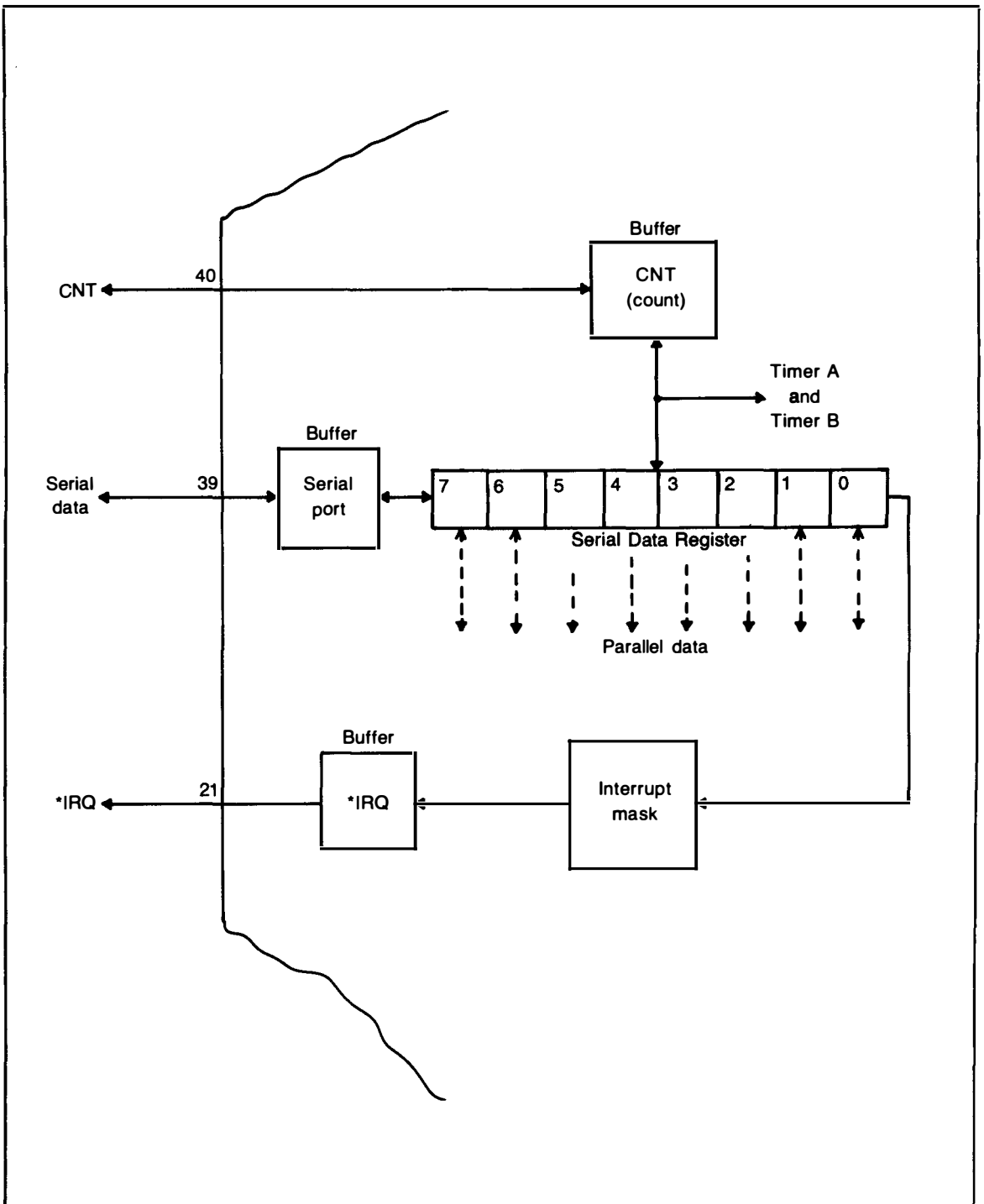


Fig. 16-7. A complete two way serial port is found at pin 39. It works well with \*RQ and CNT.

operation. When there is data coming from a peripheral and is waiting on the SP pin, a rising edge on CNT opens the SP pin, and the data can enter the shift register one bit at a time. Then after eight CNT pulses the data in the shift register is transferred to the Serial Data Register in a parallel fashion; all of the bits move at the same time. Once the data is in the register, an interrupt is generated. The 6510 takes over from there.

When the port is acting as an output, it needs a timer. Timer A sets a rate of data flow. This will be the rate of flow of the serial bits that leave the SP pin for a peripheral. This computer shifts the data out of the SP pin at 1/2 the underflow rate of timer A.

The 6510 can output data through the serial port by writing to the serial data register. The only requirement is that timer A is running in the continuous mode. During the write operation the clock signal from timer A is outputted from pin 40, CNT. The data that arrives in the serial register, in time with the clock, is then placed in the shift register and shifted out on pin 39, SP. The data is shifted out starting with bit 7, then bit 6 and so on, ending the byte output with bit 0.

The data is shifted out bit by bit. After eight CNT pulses a byte is outputted to a peripheral. At that time the SP circuit generates an interrupt and sends it to the interrupt Data register. The interrupt is designed to tell the processor that the previous byte has been transmitted and the serial register is ready for the next byte, if there is one.

The processor though runs one step ahead of the SP circuit. If it has more data, it will get the byte into the Serial Register immediately before the interrupt. The data is then shifted out of the SP pin. As long as the processor keeps one byte ahead of the SP circuit, the data output will be continuous. When there is no longer any data, after the eighth CNT pulse, CNT will go high and the transmission will cease.

## OPERATION

There are two 6526 CIA chips in the 64. The CIAs do jobs that require a digital-to-digital interface.

This is in contrast to the VIC and SID chips that convert digital signals to analog signals. These chips are covered in Chapters 17 and 18.

## CIA1

One of the CIAs is used to receive the keyboard input strikes. The same connections that receive the keyboard pulses are also attached to the two control ports. Refer to Master Schematic 1. Joysticks and other peripheral inputs can also enter here. There is no conflict, under normal circumstances. The keyboard and the joysticks are usually not made to perform at the same time. The peripherals, when off, present a high impedance to the CIA port pins and do not interfere. Even if both were activated at once, there probably will be no real harm done.

The keyboard operates into the CIA port pins in this way. Port B PB0-PB7, pins 10-17, are connected directly to eight columns of keys. Port A PA0-PA7, pins 2-9, are connected directly to eight rows of keys. When the computer starts up, one of the housekeeping jobs that the Kernal does is configure these CIA port pins. It writes to the Data Direction Register of the ports and makes the row connections outputs and the column connections inputs.

There are eight rows wired internally in the keyboard. The wiring bears no resemblance to the QWERTY layout of the keyboard. On each wired row, there are eight columns. At the intersections of the rows and columns, the keys are attached. When you strike a key, you cause a short between a row and a column. There are 64 possible shorts you can cause. The Kernal keeps a close watch on the keyboard to detect any key strikes.

The way the Kernal does this is by scanning the rows and columns continually using the port registers. The decimal address of port A in the memory map is 56320. The address of Port B is 56321. The rows are input at port A and the columns are input at port B. Bits 0-7 of port A will contain the state of row bits 0-7. Bits 0-7 of port B will contain the status of column bits 0-7.

The Kernal scans the columns and rows in this

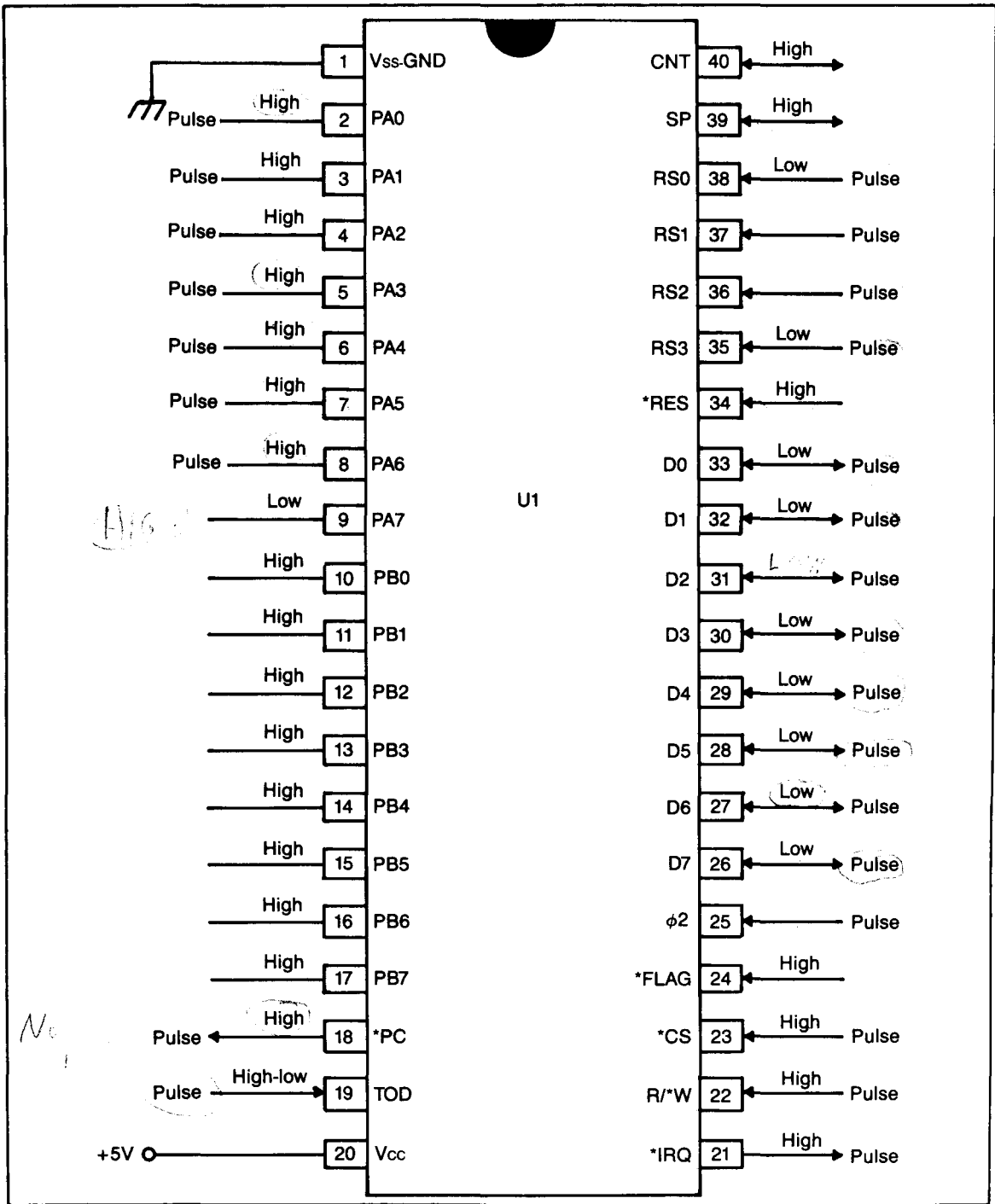


Fig. 16-8. When the computer is first turned on and displays READY and the blinking cursor, these highs, lows, and pulses should be present at U1.

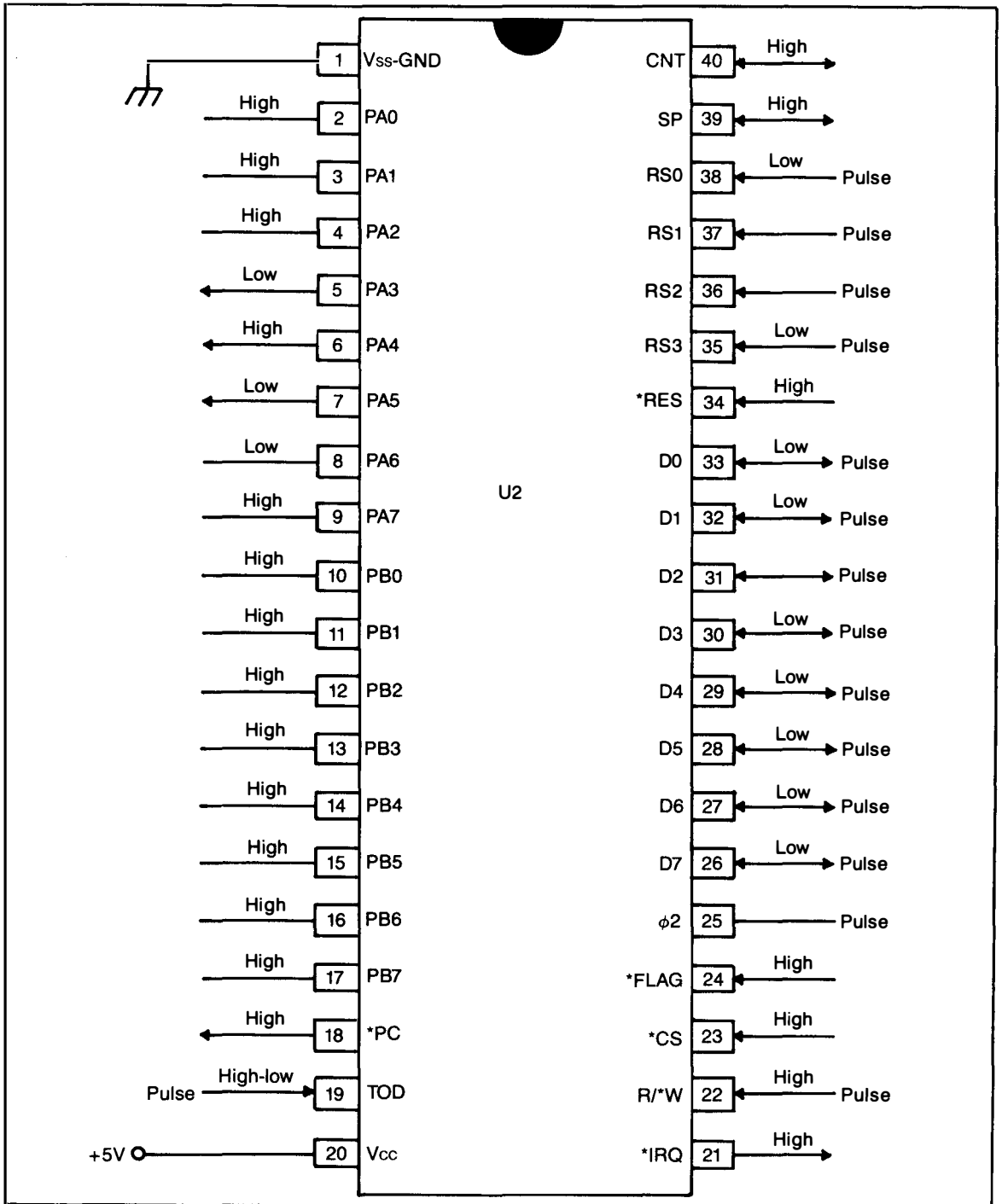


Fig. 16-9. Even though U2 is identical to U1 in construction, they are connected in different circuits and do not have the same signals on corresponding pins.



way. It writes a high to each column in turn. After each column write it reads the status of the rows. If a row and column have a key closure the Kernal will know which column it had written to and which row was shorted. The row and column information denotes one struck key out of the 64 possibles. The Kernal decodes the information and processes the value of the closed key.

While most of the port pins in this CIA are used for the keyboard and joystick type inputs, the serial port, SP and the timing devices in the 6526 are left to do other duties. Pin 40, CNT and pin 39, SP are wired to the User Port. \*FLAG is connected to the Serial Bus and the cassette interface.

## CIA2

The other CIA uses its parallel output port pins in entirely different types of operations. The port B PB0-PB7 pins are connected directly to the user port pins, C, D, E, F, H, J, K, and L. In addition, pin M is connected to Port A pin PA2 and pin B is connected to \*FLAG on the CIA. Refer to Master Schematic 5.

This arrangement gives a programmer complete input or output control over Port B of the CIA. The additional pins of PA2 and \*FLAG permits the programmer to use the port B register in a handshaking operation with a peripheral.

Pin 39 the Serial Port and its companion counter CNT at pin 40 join the other CIAs SP and CNT at the user port plug. The other handshaking line PC also goes to the user port.

PA7 is a data output to the serial bus plug. PA6 is a clock output to the same plug. PA5, PA4, and PA3 also are connected to the serial bus. The serial bus is the place where a disk drive or graphics printer can be plugged in. In this connector, up to five different devices can be connected at one time. The 64 is the controller of the bus. Each device is given a bus address. The addresses are numbered

from 4 to 31.

A programmer with the aid of the port A pins, PA7-PA3, is able to control, talk, and listen to any and all devices connected to the serial bus. Only one device can talk at a time, although all of the devices can listen all the time.

## TESTING

The two CIAs perform a lot of the complex I/O jobs. You can test to see if the CIAs are set up to do their duties by probing the CIA pins one by one with the logic probe. The test point charts in Figs. 16-8 and 16-9 show what state should be present on each pin when the computer is first turned on and displays the READY message and the blinking cursor. It is not necessary to know if you are reading a port pin or a control state. It is only required that you compare the voltage or logic state you find with what should be there during normal operation.

If you should find a test point that has a reading that does not match up with what should be there, that could be a valid service clue. Then it is useful to know more about the test point with the discrepancy. At that time, you find out what pin it was that has the wrong reading. Then if you have an idea of what the pin is doing and how the circuit involved is operating, you can intelligently come to some conclusions as to what could possibly be causing the trouble.

Most of the circuits in the CIA area of the computer have been buried in the CIA chip. There are no simple and easy ways, besides direct replacement of a chip, that will pinpoint a circuit trouble. Your technique must be an analysis of the input and output signals and states. The results of the tests can give you an idea of whether a state is entering properly, was processed and if it is exiting correctly. Your understanding of the workings will allow you to come to such a conclusion.



## 17. Video Interface

**T**HE 6510 PROCESSOR ACCESSES THE VIC JUST as it would any other I/O chip on the memory map. The VIC has 47 addressable registers, and they are all located on the map. The VIC can be reached via the address bits. It can transfer data through the eight bit data bus. It uses the R/\*W line as well as the clock and \*IRQ. Figure 17-1 defines all the pins.

Besides being a conventional type I/O chip, the VIC has some processor abilities. It is able to turn off the 6510 and take over control of the computer. It has address lines that connect directly to the computer's address bus. It also generates the dynamic RAM refresh signals, \*RAS and \*CAS as well as the Address Enable Control signal, AEC, that turns the 6510 off and on.

The VIC is not only an I/O chip, it is also the video peripheral device. The VIC outputs a conventional composite color TV signal that can be applied to video output stages and then drawn on a TV screen. The video output consists of the horizontal and vertical sync pulses, the black and white picture, and the components that color the picture.

### OPERATION

The VIC has a special relationship with the rest of the digital system. Under normal circumstances, the 6502 processor family, of which the 6510 is a member, is driven by  $\phi 2$  of the clock. When  $\phi 2$  is high, the data bus is used. During the low portion of  $\phi 2$  the data bus is dormant. The VIC takes advantage of the time when  $\phi 2$  is not using the bus. If the VIC is going to access memory, it does so when the 6510 pauses during the low of  $\phi 2$ .

The VIC needs to access memory for two important reasons. First of all, it must refresh the dynamic RAM. If it does not address the RAM at least once every 3.66 ms, the charges on the tiny capacitances that represent highs, will leak off. The second reason that the VIC will access memory is a normal fetch and execute operation. Memory is able to hold data for the characters that VIC outputs for display. The VIC must access memory for the data just as the 6510 accesses memory for its data.

In order for VIC to get control of the data bus during the  $\phi 2$  low period and then give the bus back

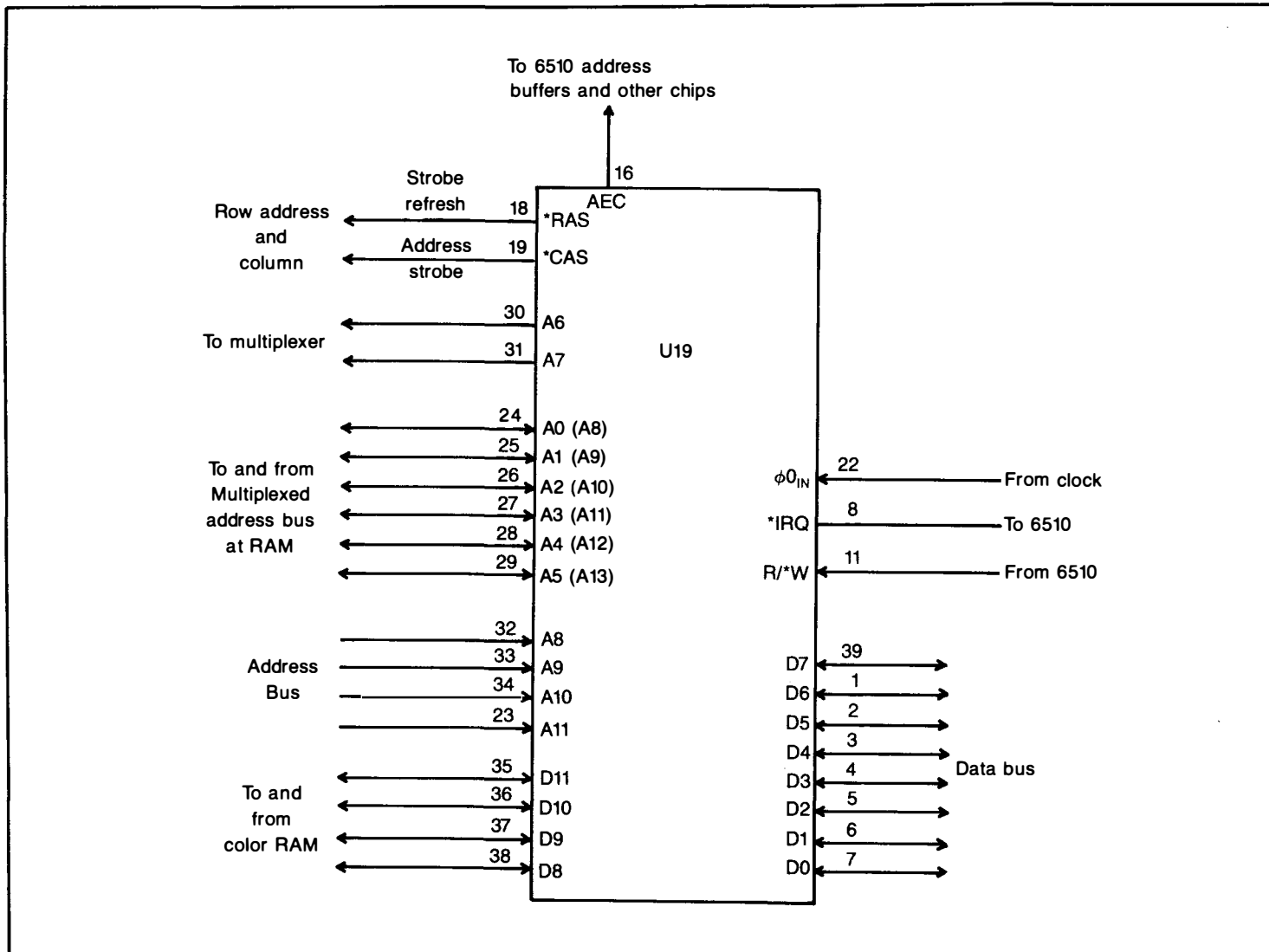


Fig. 17-1. The 6510 is able to access VIC in the same way it can access any I/O device. It uses address bits to select the chip and the 47 internal registers. It uses data bits to send or receive bytes over the data bus and it uses the R/\*W line,  $\phi 0$  and \*IRQ.

to the 6510 when the  $\phi 2$  goes high, VIC provides control signals. First of all, the AEC signal emerges out of pin 16. AEC is a high when the 6510 is in control. As the VIC takes over, AEC goes low.

When AEC goes low, it turns off most of the chips it is connected to. They are shown in Fig. 17-2 and are all concerned with the normal accessing of memory by the 6510. In addition, the AEC signal goes to the 74LS08 AND gate. When AEC is high, it outputs a high to pin 5 of the 6510. The high to the 6510 keeps the address bus buffers in operation. When a low arrives at the AND gate, the gate sends a low to the buffers. This shuts the buffers down. That closes off the program counter, and for all intents and purposes, the 6510 stops operation.

The AEC low pulse is designed to appear when VIC wants to use the data bus during the low of  $\phi 2$ . The low of  $\phi 2$  is only half a 1000 ns cycle. VIC then must act fast. It must access memory within the 500 ns restriction. That means it must arrange enough time to have an addressing take place, do the complete data accessing, and set up the data for a stable read operation.

The AEC signal then goes high and low repeatedly. It is high and not active while  $\phi 2$  goes through its high. The 6510 is in control while AEC is high. Then AEC goes low as  $\phi 2$  goes low. The 6510 relinquishes control at that time to the VIC. This procedure can accomplish a lot. However, the 500 ns time stretches are very restrictive to the VIC. It has two operations that cannot be completed in a half cycle. VIC cannot move the character pointers in the video part of memory in 500 ns. It also cannot do a read of the sprite data in this time. The access time for these items of data must be made longer. This means using the high time of  $\phi 2$  as well as the low period.

Coming out of pin 12 of the VIC is a signal called BA, bus available. BA is normally held high. When the VIC is accessing video memory or sprite data and while the  $\phi 2$  low period is on, the VIC brings BA low. This change in signal notifies the 6510 that VIC needs the  $\phi 2$  high time as well as the  $\phi 2$  low. The 6510 is then able to continue to use the  $\phi 2$  high three more times. This is usually enough access time for the processor to finish the

immediate job it was doing.

As the fourth  $\phi 2$  high occurs, AEC does not go high, it stays low. The 6510 remains in a three-state condition and VIC starts reading the data it wants. The BA line goes to the cartridge expansion port. It then comes out of the PLA and into another one of the 74LS08 gate sections. From there, it connects to the RDY input of the 6510. RDY will signal the processor that VIC is going to be using its  $\phi 2$  high for awhile.

## SPECIAL ADDRESS LINES

Figure 17-3 shows that there are address lines connected to VIC pins 24-31. A closer look shows pins 24-29 each have two address bits connected. For instance, pin 24 contains both A0 and A8. This shows that bits A0-A5 and A8-A13 are being multiplexed. Bits A6 and A7 are static type address bits that are needed when the VIC must address a location on a 2K ROM.

The A0-A5, A8-A13 multiplexing takes place during the control signals \*RAS and \*CAS. As \*RAS from pin 18 of the VIC goes low, the address bits A0-A5 are placed on pins 24-29. They go to the multiplex address lines of the memory. Then when \*CAS goes low the signals on the pins are replaced by bits A8-A13. With the two static bits A6 and A7, fourteen address bits are output by a type of program counter in the VIC. The fourteen bits are able to access 16K bytes of memory, as Fig. 17-3 shows.

## DATA BUS CONNECTIONS

The data bus, by its very nature, is bidirectional. The VIC is able to send or receive byte-sized data over the bus. The situation gets complicated though because the VIC can be accessed while the processor is in charge or the VIC can do its own accessing when it is exercising control over the system. There is one set of control signals when the processor is in charge and a second set of signals while the VIC does the controlling. The signals are \*CS, R/\*W,  $\phi 0$ , and AEC. They operate in the following way.

The AEC signal is an output from the VIC to

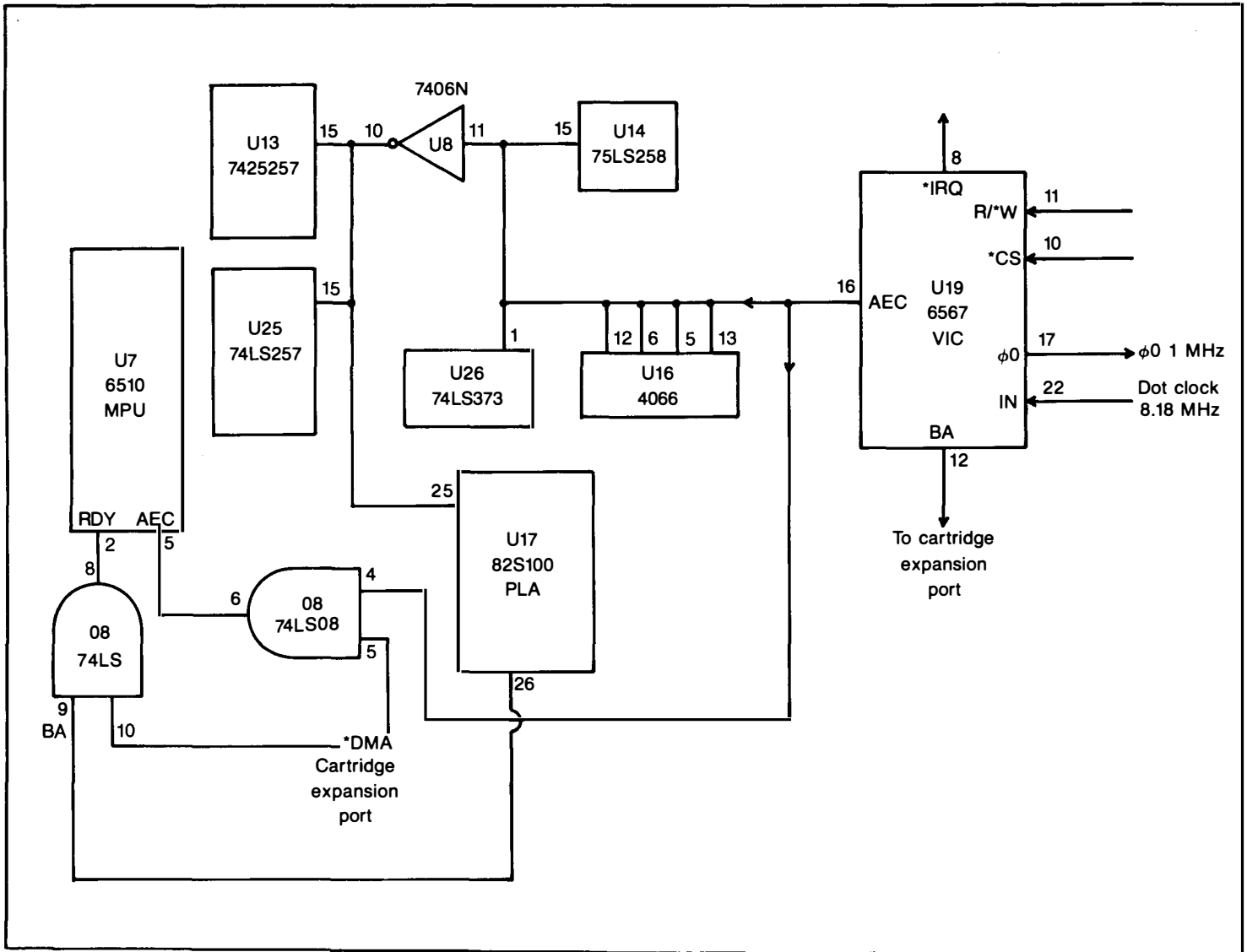


Fig. 17-2. When AEC from pin 16 of the VIC goes low, it exercises control over all the chips it is attached to.

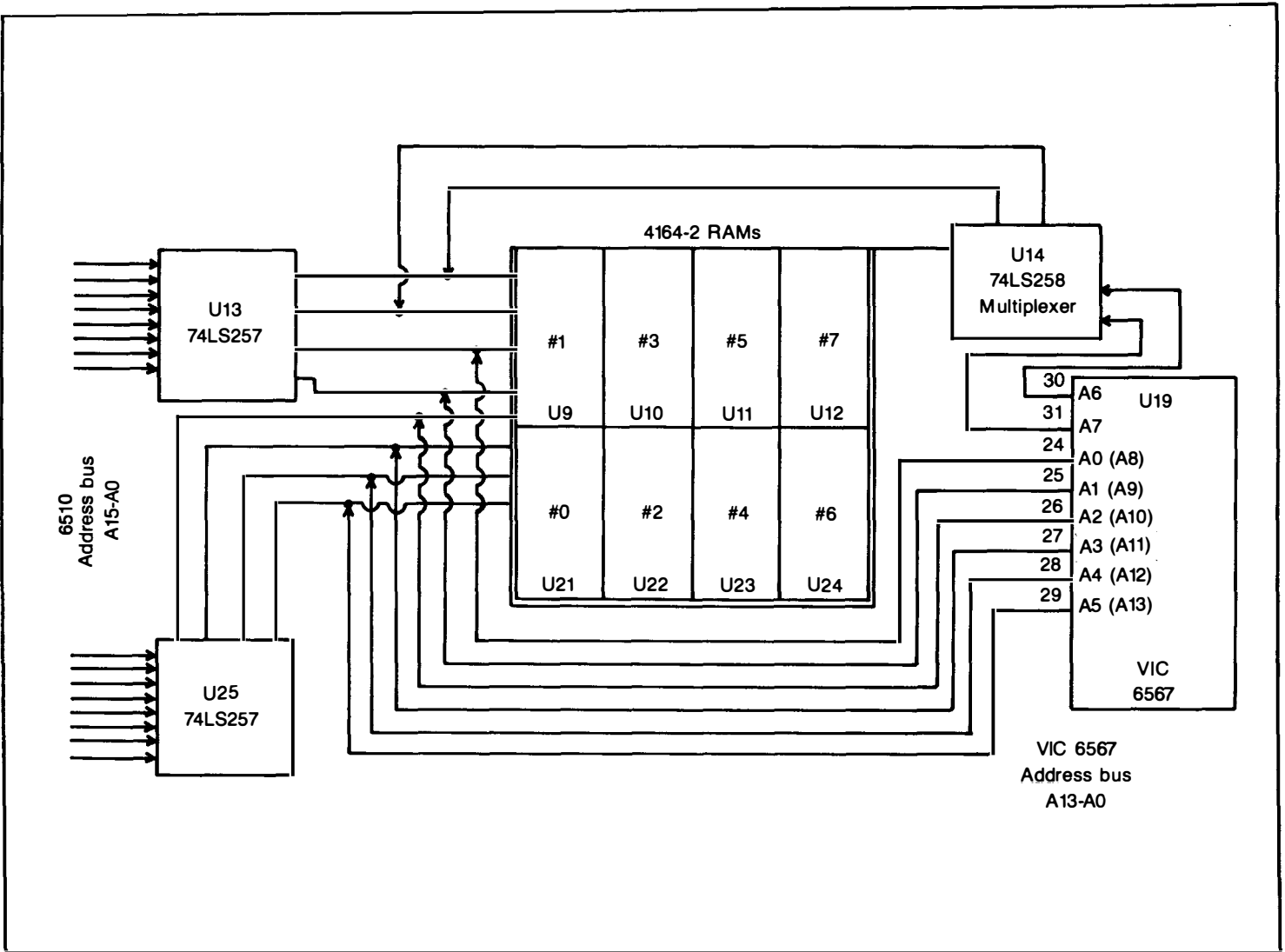


Fig. 17-3. VIC also is able to act like a processor in its own right. It is able to address and access RAM through its own address bits that connect directly to the RAM address input pins.

the address buffers in the 6510 and the other chips that must be controlled. Refer to Fig. 17-2. AEC leaves the VIC at pin 16. While AEC is high, it does not affect the processors address buffers. When AEC goes low, it shuts down the buffers. The VIC must shut the buffers down when it takes control of the system.

$\phi_0$  is the 1 MHz signal that the clock circuits manufacture. The  $\phi_0$  clock is the main reference frequency that all the rest of the 64's frequencies are derived from. At pin 22, the dot clock frequency (about 8 MHz) is input. Inside VIC, the dot clock is divided by eight. The resultant frequency is output at pin 17. This 1 MHz output is  $\phi_0$ .

When  $\phi_0$  is low, VIC is able to do jobs like read data in memory and refresh the dynamic RAM. As  $\phi_0$  is high the 6510 can read or write to and from the memory.

\*CS at pin 10 is the VIC chip select. The VIC is chosen when \*CS goes low. R/\*W at pin 11 is

the system read/write line that comes from the 6510 to the VIC. These four lines control whether the processor or the VIC is going to be in control.

With four control signals, each able to produce a high or low, 16 possible situations could take place. However, only four of the possibles produce computing activity. The rest of the combinations do not have any meaning to the system. The four signals are shown in Fig. 17-4. First of all there is the situation where AEC and  $\phi_0$  are both low. It doesn't matter what state \*CS and R/\*W are in.

When AEC is low, the 6510's address buffers are turned off. This of course means the processor is out of action during the low. If  $\phi_0$  is also low at the same time, the VIC is able to act during the low of  $\phi_0$ . It takes control of the system and is able to refresh the 256 rows in each of the RAM chips. At this time, the VIC is also able to read data from memory, if it so desires.

The next possible state combination is the

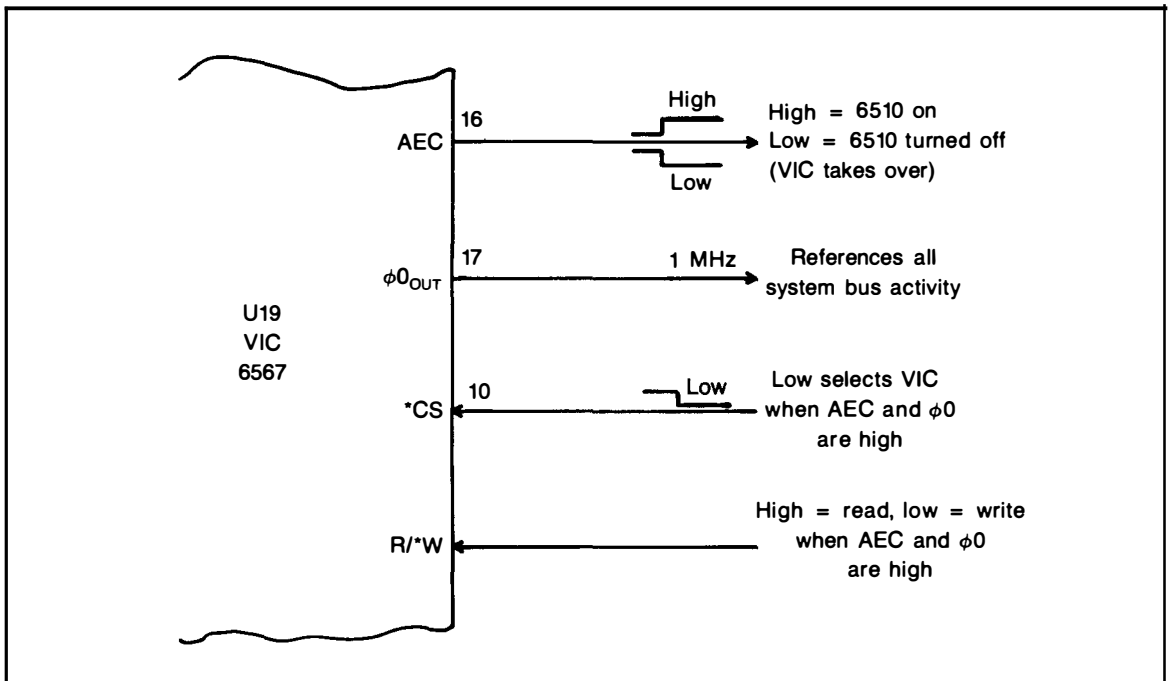


Fig. 17-4. The four control signals from AEC,  $\phi_0$ , \*CS and R/\*W. AEC and  $\phi_0$  are generated in the VIC. \*CS are the result of address bits while the 6510 originates R/\*W.

following. Again AEC is made low.  $\phi_0$ , though, is high for this activity. When AEC is low, the 6510 is again shut off, and the VIC is given charge of the system. Since the processor is now off,  $\phi_2$  can be utilized while it is high. The VIC is able to read data from memory during the  $\phi_2$  time period. It can leisurely read character pointers from the video RAM and obtain sprite data. Again, the states of \*CS and R/\*W are meaningless in the situation.

The last two activities were with the processor cutoff and the VIC in control of the computer. The next two situations have the 6510 back in action and the VIC as a memory map resident again. One of these situations occurs when AEC goes high. This frees the address buffers to pass the address bits from the processor's program counter.

$\phi_0$  goes high for this operation. This allows the processor to work the data bus during the  $\phi_2$  high. \*CS goes low. This chooses the VIC for accessing. Lastly R/\*W also goes low. This is the well known signal to a chosen chip that it is going to be written to. With the four signals attaining these four states, the VIC is in a position to be written to by the 6510.

The last way that the computer communicates between the 6510 and the VIC occurs with the following signals. AEC goes high which leaves the 6510 in control.  $\phi_0$  goes high which permits the 6510 to access the data bus during the  $\phi_2$  high. \*CS becomes low as the 6510 addresses the VIC. R/\*W develops a high state which lets the VIC know that it is to be read.

While the 6510 is accessing VIC, besides selecting the chip at \*CS, it must choose one of the 47 registers to read or write to. The six address pins, 23-29, A5-A0 are used for the register select. Six pins can address a possible 64 locations. The 47 registers can be accessed easily with plenty of address possibilities left over.

The six pins are two directional address lines. They are outputs when the VIC is in charge. They are six of VIC's 14 address pins. They are conventional chip input address pins while the processor is operating its normal reading and writing. The register select address bits enter the VIC at these

pins while the 6510 is accessing the VIC as an I/O chip.

## MODES

When the VIC is in charge, it can act in a number of graphic ways. The programmer can make it display whole characters or instruct it to control each dot on the TV screen. The character display graphics can be formed in one of a number of modes. The dot control also is able to operate in different modes. The modes are able to appear in many colors.

The VIC, in order to display the graphics, must work with the dynamic RAM, the Color RAM and the Character ROM, besides the usual operating system chips. A section of RAM is set aside for the VIC to be able to access. It is called video RAM. The Color RAM, the 2114 chip, is connected from it's pins, 11-14, D3-D0 to VIC pins, 35-38, D11-D8. These are four additional data bus type pins that the VIC possesses. The VIC can handle 12 data bits. The Character ROM is the place in memory where the characters are stored in silicon.

The VIC is put into the various modes by means of it's 47 addressable registers and uses these sources to form the desired graphics. Programmers must know the operations of the modes in order to do their job. Troubleshooters should know the way the modes access and use the data sources to puzzle out strange hardware problems.

## Displaying Characters

In a character display mode, the VIC places characters from the ROM onto the screen. The TV display is arranged in 25 rows of 40 columns each. One character space consists of 64 dots in an  $8 \times 8$  bit character into the  $8 \times 8$  dot matrix. Each ROM character is burnt into eight consecutive bytes. The VIC is designed to have it's choice of 256 characters in the ROM.

Each character in the ROM has eight addresses on the map. The VIC is able to access the ROM, obtain a character, and display the  $8 \times 8$  dot space on the TV screen.



The TV screen has 1000 character spaces that can be used. Video RAM is assigned 1000 bytes of memory reserved to coincide with the TV screen spaces. This arrangement is all well and good but each of the 1000 spaces need eight bytes of data to operate the 64 dots. Video RAM only holds one byte for each space.

This is because video RAM only needs to hold the first address of a character in the ROM. These addresses are the character pointers. When the VIC must display a character on the TV screen, it accesses video RAM for that character's address. Then it goes to ROM and obtains the eight byte character. Next it processes the  $8 \times 8$  character dots and arranges for them to light up or turn off the correct dots to form the TV display character.

## Character Fetch

VIC has a starting address on the map of decimal 53248. That is register 0 of the 47 registers. For a character fetch, register 24 must be contacted by means of a program line. The data to form the video RAM and Character ROM addresses is in register 24.

Video RAM must be addressed by the VIC to obtain the character pointer. The VIC has fourteen address bits to access video RAM. This will address a 16K portion of the total 64K RAM. Video RAM will be in the 16K section that the VIC is able to access. Sometimes special circuits must be installed to give the VIC two more bits to access 64K. Whatever the case, the VIC will be made to access the 16K with video RAM.

The VIC begins to form the address of the character pointer with the aid of register 24. Refer to Fig. 17-5A. In the highest four bits of 24 are the address bits A13-A10 of video RAM. The VIC outputs them as part of the address of the character in ROM.

Meanwhile, internal to the VIC is a counter circuit. It is constantly counting from 0 to 999, which is the 1000 video RAM locations and the 1000 TV screen spaces. The ten bits from this counter and the four bits from register 24 form the address in the character pointer. Refer to Fig. 17-5B. The

counter is constantly scanning the 1000 consecutive places. When a desired character is needed by the VIC, it outputs the ten additional bits through A9-A0 of its address pins. The video RAM is thus accessed and gives up the eight bits in the pointer address.

The next step is to form another 14-bit address so that the VIC can access the Character ROM for the eight bytes of dot information. The makeup of this address is shown in Fig. 17-5C. The three most significant bits of the address are found in register 24 in bits 3, 2 and 1. The VIC uses them as address bits A13-A11 to point to the Character ROM. The next eight bits of the pointer, A10-A3, are the bits VIC has just fetched from video RAM. That leaves A2-A0 to be filled in order to form the character address that will let VIC access the Character ROM.

To review, bits A13-A11 are the Character ROM chip select bits. These three bits are found by VIC in its own register 24, bits 3, 2, and 1. Bits A10-A4 are the register select to choose among the 256 different characters that are available to VIC from the Character ROM. That leaves A2-A0.

A2-A0 are a three bit counter. The three bits can count from 0 to 7. There are eight bytes to one of the characters. At each ROM byte location, the counter points to each byte in turn till all eight bytes have been addressed and accessed by the VIC.

In the character display modes, the activity can be summed up quickly. The VIC starts the memory accessing by first reading a character pointer out of the video RAM. The pointer is one byte and is an address. The address is a location in the Character ROM where the desired character is stored in eight bytes. The 64 bits in the eight bytes are processed by the VIC to light up the 64 dots in one of the 1000 screen spaces to form the character. The highs in the bytes light dots and the lows extinguish the dot light.

## Character Color

In the Color RAM chip that attaches to the VIC through four data pins, there are 1024 4-bit registers. These nybbles control the color of the

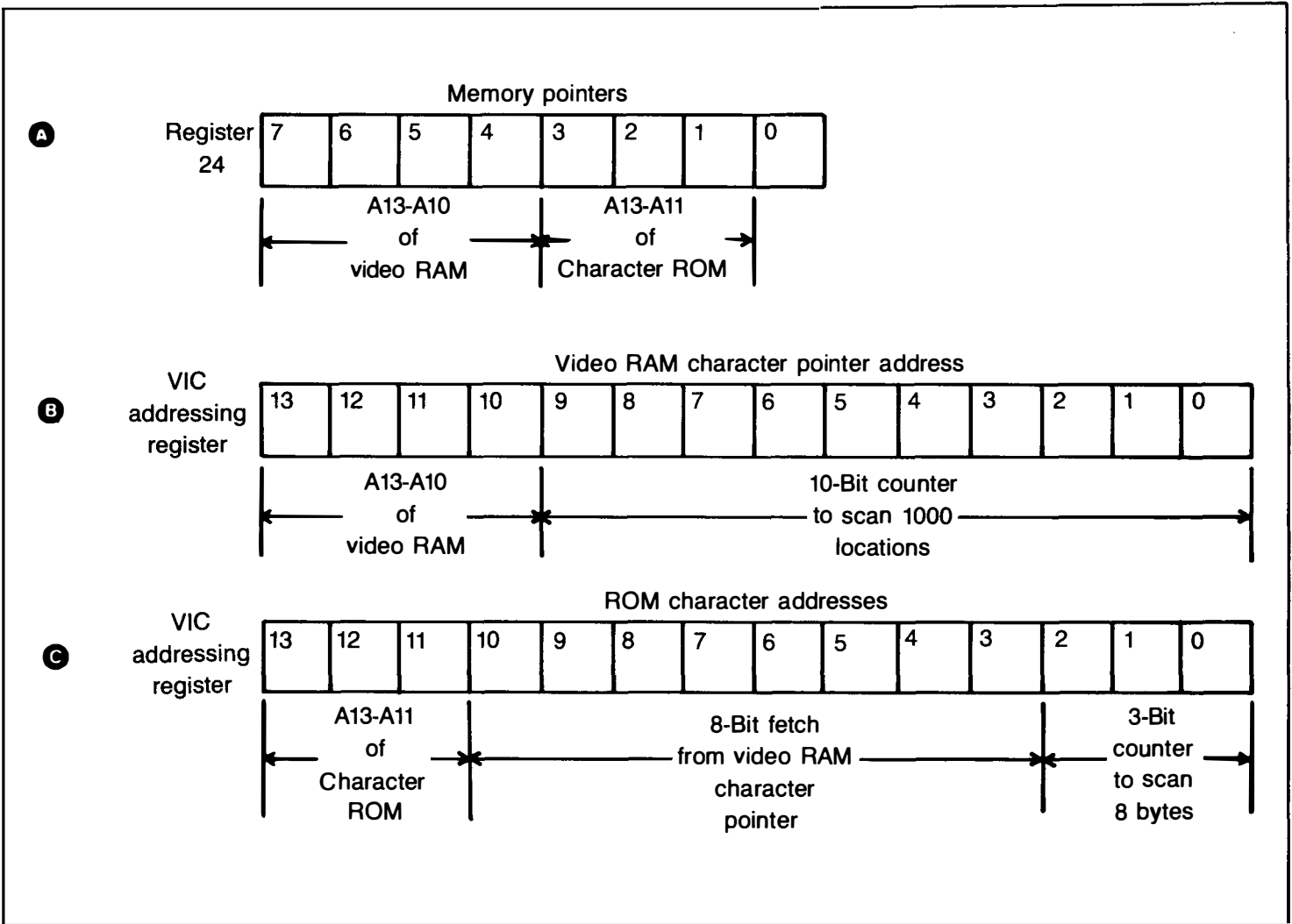


Fig. 17-5. In the highest four bits of register 24 are address bits A13-A10 of video RAM (A). They are combined with a 10-bit counter that provides the A9-A0 bits. VIC outputs the 14-bit address, accesses RAM, and has a character pointer returned (B). In bits 3, 2, and 1 of register 24 are address bits A13-A11 of the Character ROM. A10-A3 are taken from the character pointer just received from video RAM. A2-A0 come from a 3-bit counter in the VIC (C).

character that gets displayed. Since there are four bits in each nybble register and the four bits have 16 combinations, each nybble can code one of 16 colors.

The Color RAM has 1000 of its registers assigned to the 1000 bytes in video RAM. The wiring is such that when a video RAM location is addressed, the associated register in the Color RAM is also addressed. It is as if the video RAM is 12 bits wide, rather than eight bits. As the 12 bits are addressed, the eight bits from the video RAM enter the data bus bits D7-D0 and enter the VIC at pins D7-D0. The additional four bits from Color RAM leave the chip and enter the VIC at pins D11-D8. VIC then processes all 12 bits. At D7-D0, the character is received and at D11-D8, the color of the character is determined.

## Character Modes

Characters are displayed by VIC in three different modes. One mode, the Standard Character mode is the one that comes up automatically when you turn on your Commodore 64. The operating system sets up this mode during its housekeeping duties.

The VIC will adopt a specific character mode as three bits in its registers are set or cleared. The VIC is designed to respond with a mode as the following three bits are affected either by the operating system or by planned programming. The first bit is named MCM and is in register 22, bit position 4. The second bit is called BMM and is found in register 17, bit position 4. The third bit is called ECM and is also in register 17 but at bit position 5. Refer to Fig. 17-6.

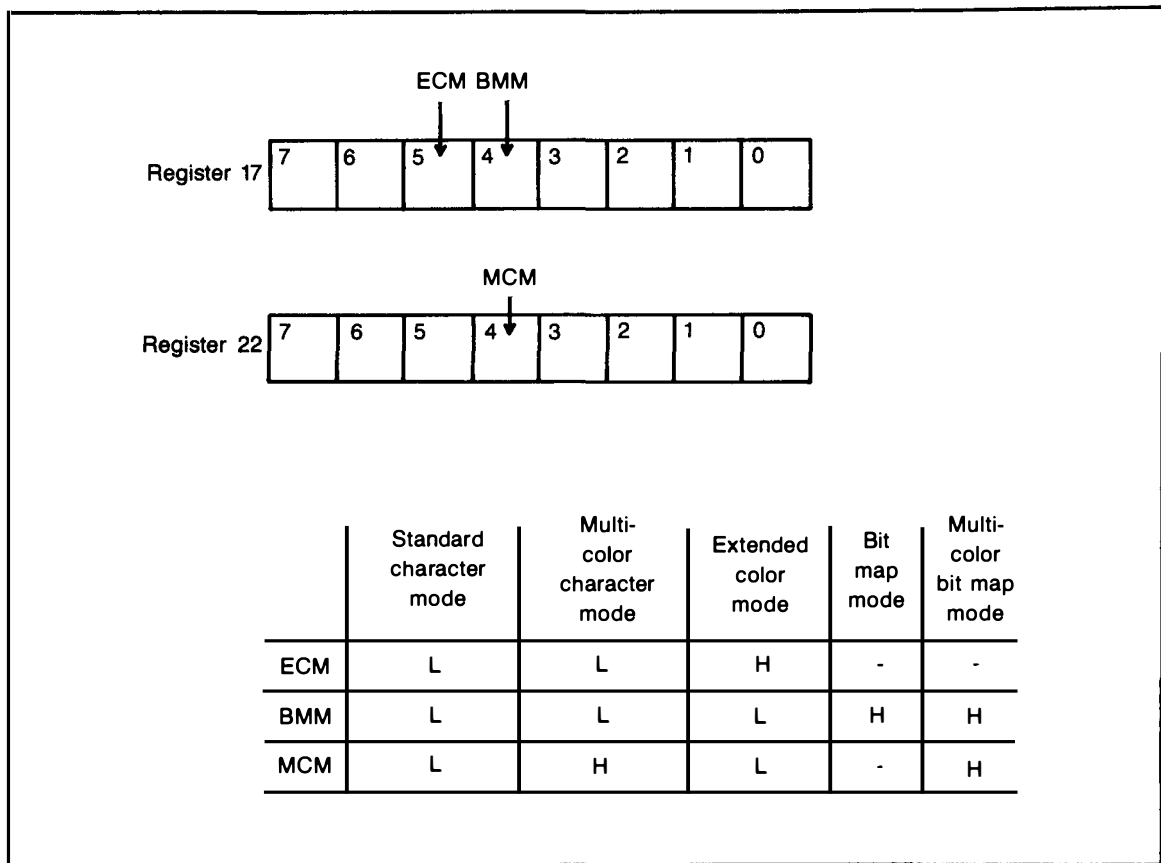


Fig. 17-6. The three bits that set up the various VIC modes are found in registers 17 and 22.

When your 64 is first turned on, all three bits are cleared and hold lows. This forces VIC to adopt the *Standard Character* mode. In this mode, when eight bytes of a character is fetched by VIC, all eight bytes are displayed directly onto the eight lines of each character space on the screen.

If you place 0s into these bit positions the background will be displayed as a result of register 33. When 1s are installed, the foreground color is selected by the color nybble. This is performed as desired by the programmer.

The *Multi-Color Character* mode is obtained by installing a 1 in MCM. BMM and ECM are left low. This mode allows the programmer to have up to four colors in each character space. However the resolution of the character suffers with the additional coloring. Two bits are needed to specify one dot color in this mode. This reduces the  $8 \times 8$  dot matrix to a  $4 \times 8$  matrix with each dot twice it's original horizontal size. The character is not quite as distinctive as in the  $8 \times 8$  matrix.

The *Extended Color* mode is produced by setting ECM to a 1 and leaving BMM and MCM as 0s. This mode lets the programmer have individual selection of background colors for each character space in the  $8 \times 8$  matrix.

The extended color feature is paid for in the number of characters that the VIC can use in this mode. Only 64 characters are available because two of the character address bits are being used for the color information. For further details on programming these three character modes, refer to a programming book that covers the VIC from this point of view.

## Bit Map Modes

Bit 5 in the VIC register 17 is BMM, Bit Map Mode. If the bit is set to a 1 the BMM is on (Fig. 17-6). When the bit is cleared to a 0 the mode goes off. In the character modes, VIC accessed video RAM to get a character pointer. This was an address to eight bytes in ROM that was storing a character. The character was installed in 64 light dots in a screen character space.

In the bit map mode, the VIC displays in the

same 64 light dot spaces but has more detailed control. The VIC, in bit map mode, is able to control each of the 64 light dots by itself. A bit in video RAM is assigned to each and every light dot on the screen. If the bit is a 1, the dot goes on. When it's assigned dot is a 0, the bit goes off.

There are 1000 character spaces on the screen. There are 64 light dots in each space. This means there are 64000 dots on the screen to be controlled. If one video RAM memory bit controls one dot, there has to be 64000 bits in video RAM to handle the screen. Eight bits to a byte means the bit map mode requires 8000 bytes of memory for the control.

The resolution of the mode is 320 horizontal dots by 200 vertical dots. The VIC accesses the 8000 bytes of video RAM in bit map mode in the same way it accessed the 1000 bytes of video RAM in the character mode. However, the VIC is not looking for character pointers. It needs direct color data.

The address that the VIC forms to access video RAM is also produced by a counter. The counter constantly scans the video RAM to keep the display up to date. A13 of the address VIC outputs is taken from register 24, bit 3 called CB13. Refer to Fig. 17-7. Bits A12-A3 are the outputs of the video RAM counter. The last three bits A2-A0 are the counter that covers the eight lines in each character space.

The addressing of the 1000 spaces on the screen is quite like the way the character pointers are addressed. A13 is the chip select for the video RAM start location. A12-A3 scans the 1000 spaces. It scans the 40 horizontal spaces eight lines at a time, A2-A0 then counts the individual eight lines in each space. That way each dot is individually addressed.

When BMM is a 1 the Standard Bit Map mode appears courtesy of the VIC. The color information is obtained only from the state of the individual dots in video RAM. The Color RAM chip has no effect in this mode.

The color of dots in one of the 1000 spaces is controlled by the screen memory byte of the space. For instance, decimal 1024 is the screen memory

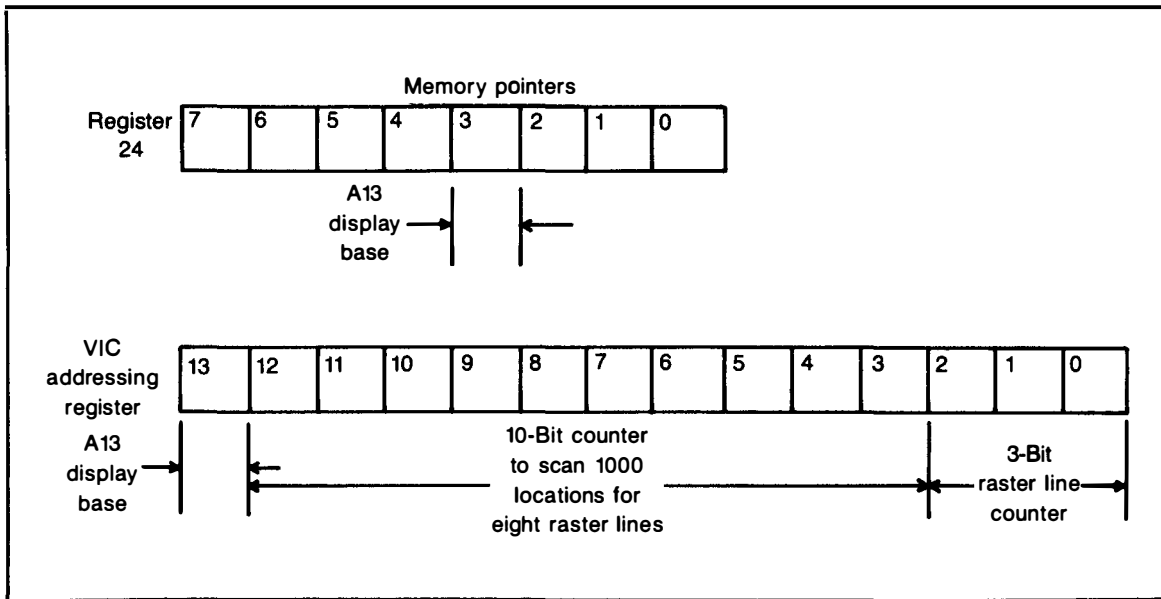


Fig. 17-7. In bit map mode, the VIC fetches data from memory on a one to one basis unlike the character type fetching. Bit 3 of register 24 is used as A13 in this mode. The rest of the address is formed with the 10-bit counter providing A12-A3 and the 3-bit counter contributing A2-A0.

address of the upper left hand space on the screen. In bit map mode, this pointer becomes the color controller of that single block. There are eight bits in the memory pointer.

The bits are arranged in four more significant bits and four lower bits. The four bits each become a color controller. In bit map mode, the higher four bits are the color code for all the bits in the block that are set as 1s. The lower four bits are the color code for all the bits in this block that are reset to 0.

The *Multi-Color Bit* map mode is produced by not only setting BMM to 1, but also MCM, bit position 4, in register 22 to a 1. The additional feature of more colors has a price to be paid too. Two bits are used to select the colors and the size of the horizontal dot has to be doubled reducing resolution to 160 horizontal dots by 200 verticals. However, three separate colors plus the background color can all be displayed in any of the 1000  $8 \times 8$  dot spaces.

## SPRITES

One of the exceptional abilities of the VIC is the manufacturing of sprites. The engineering spec

sheets on the VIC call them MOB for movable object blocks. The MOB is a character. The MOB is not found stored in a ROM like the keyboard characters. MOB's are conceived and designed by you. Once designed they are stored in RAM locations.

A MOB character is seen on the screen in a  $24 \times 21$  dot arrangement. This is much larger than the  $8 \times 8$  dot characters one of the 1000 spaces can display. A MOB needs 63 bytes in memory in comparison to the eight bytes that a keyboard character requires.

The VIC can display up to eight MOB's at one time. The large characters can be placed at any spot on the TV screen. They can be made in color. They can be magnified and moved around. The ability to use the MOB with its graphic capabilities makes the VIC an exceptionally good chip to display arcade type games. For further MOB programming details, there are a lot of books on the subject.

Once a MOB is programmed, the dot information is stored in 63 bytes of memory. Each three bytes of consecutive memory defines one line of the character. Three bytes contain the dot light infor-

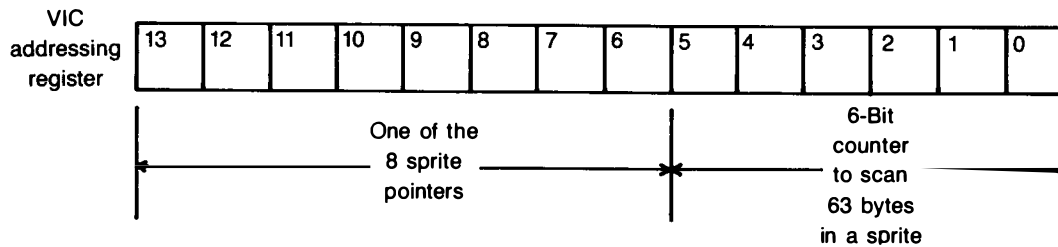
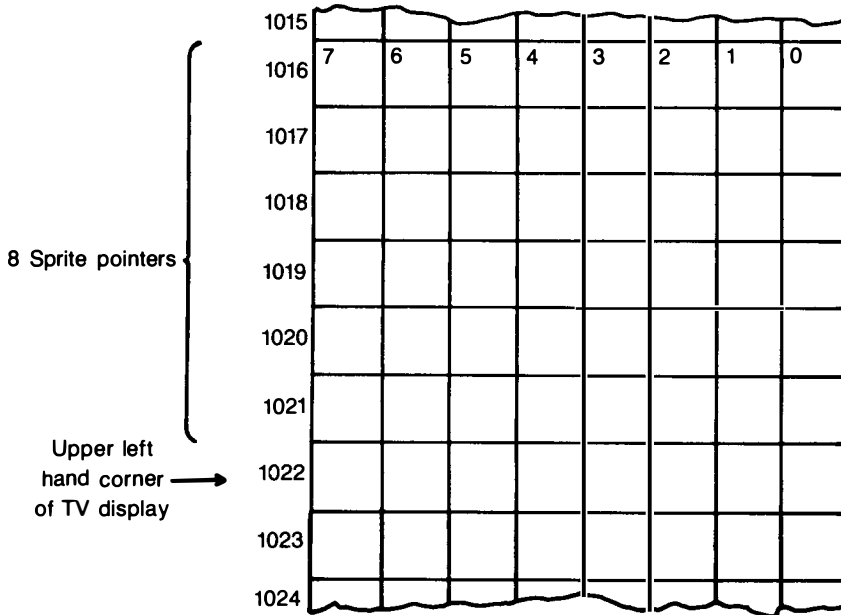


Fig. 17-8. In order to locate a MOB, the VIC must output an address consisting of one of the sprite pointers in A13-A6 and a 6-bit counter in A5-A0.

mation for 24 spaces. Since there are 21 lines in a MOB, the complete character requires 63 bytes to turn all the dots on and off.

To find a MOB in RAM, the VIC must output the correct 14-bit address. It forms the address with an 8-bit pointer that locates the start address of the 63 byte character and a six bit counter that steps through the 63 bytes. Refer to Fig. 17-8.

VIC gets the pointer from video RAM. It produces the counter from an internal register. If you look at the VIC's register map, there are 16 MOB

x and y position registers (Fig. 17-9): One x and one y register for each of the eight sprites. The VIC uses these positions with reference to the upper left hand corner of the TV screen to locate the MOB on the screen.

For the actual locating, VIC considers the TV face as having a resolution of 512 horizontal positions and 256 vertical positions. This includes the entire screen: border as well as display area. The visible screen area is from positions 23 to 347 horizontally and 50 to 249 vertically. With the x and

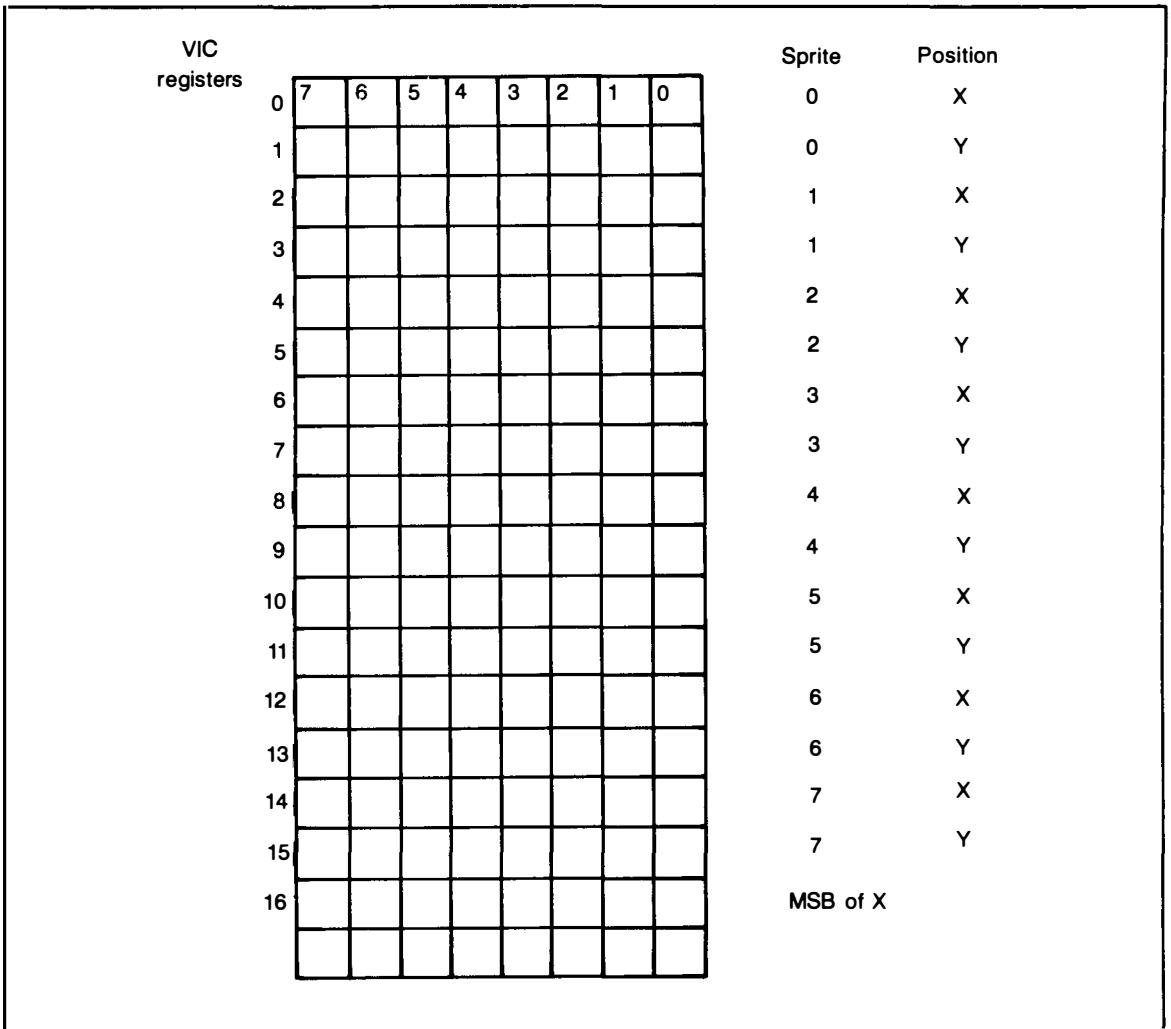


Fig. 17-9. 16 of VIC's registers are used to locate a sprite on the TV screen.

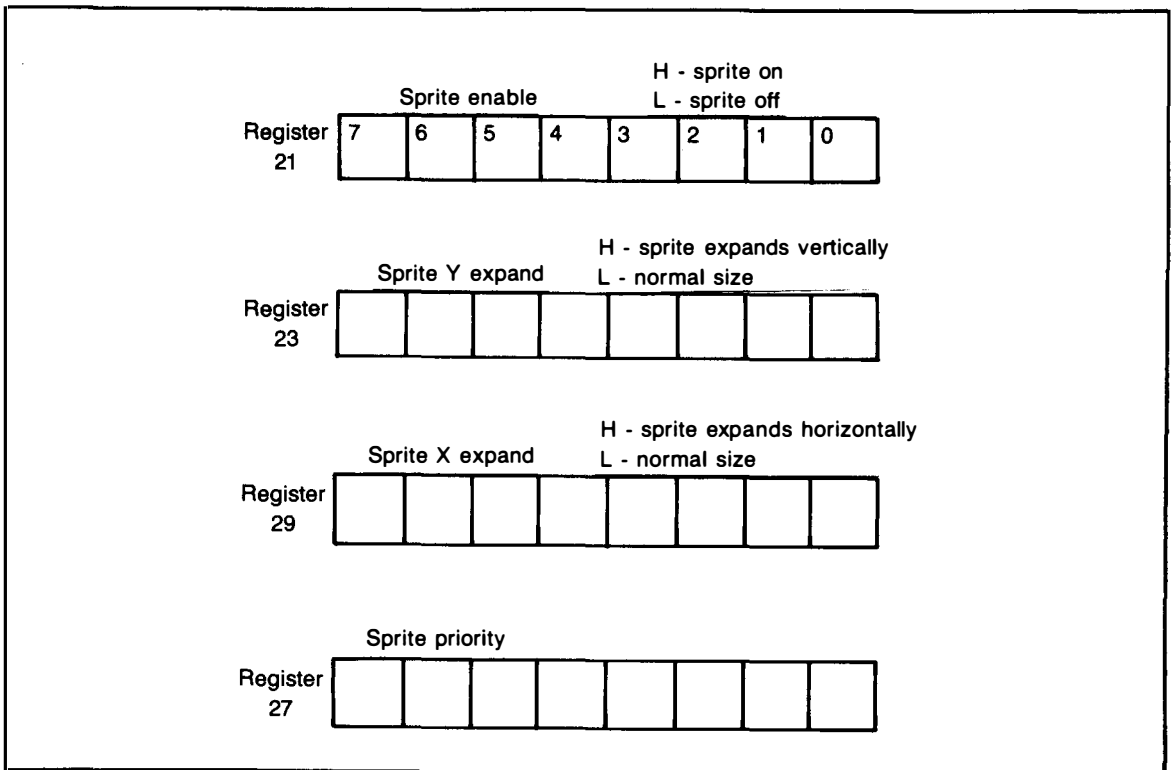


Fig. 17-10. Register 21 contains eight off-on switches, one for every possible sprite. Register 23 expands the sprites vertically while 29 expands them horizontally. Register 27 can place a sprite on top of any other display that might be on the screen.

y positions from the MOB registers for each of the eight sprites, the VIC can easily install the dot information on the TV face.

Register 21 in the VIC is the off-on switch for the sprites. There is one bit for each MOB. If the bit is set to a 1 the sprite will light up. A 0 will turn it off. Register 23 and 29 perform the magnification of a sprite. Register 29 expands it horizontally and 23 expands it vertically. A 1 does the expanding while a 0 makes it a normal size. If you want to display one sprite on top of another, display register 27 will do the job. Just install a 0 in the MOB's bit. A 1 in the bit will shift the display priority to the original display.

There are all sorts of tricks you can do with sprites, but here again we are getting into the realm of the programmer. It is important to comprehend all these workings of the VIC for servicing in case

a feature fails. If it does, you will be able to pinpoint where the trouble is by knowing how the system is supposed to be operating normally.

## OTHER VIC FEATURES

The VIC registers have some other jobs that they are assigned to perform. One such job is the ability to blank out the screen. In register 17, bit 4 is named DEN. It is usually set to a 1 which is the normal display mode. If you install 0 in the bit, the screen will be blanked out. When it does blank out it will assume the color prescribed by the bits in register 32, which sets the exterior color.

The usual display pattern on the screen is 1000 character spaces with a 40 × 25 layout. Bit 3, CSEL in register 22 and bit 3, RSEL in register 17 is normally set with 1s to produce this arrangement. If



you want to change this to  $38 \times 24$  install 0s into the bits.

The VIC provides either vertical or horizontal scrolling so game characters can be easily moved around the screen as programmed. Register 22, bits 0, 1, and 2 moves the display data an entire character space at a time in the horizontal position. Register 17, bits 0, 1, and 2 moves the display vertically.

### Raster Register

Register 18 is the raster register. The raster, of course, is the lines of light on the TV screen. The

VIC places the display precisely onto the lines of light. In order to do this, the VIC must know which dot of light is being lit or turned off at every instant of time. VIC is aware of the dot timing.

The raster register keeps track of the current raster position. If you read the register, the lower eight bits of the position are found in the register. The most significant bit of the position is then found in register 17, bit 7, RC8. This data is used by programmers to make timing changes in the display to get rid of flicker. The changes are made while the border is being scanned by the CRT cathode ray and not during the display window. The

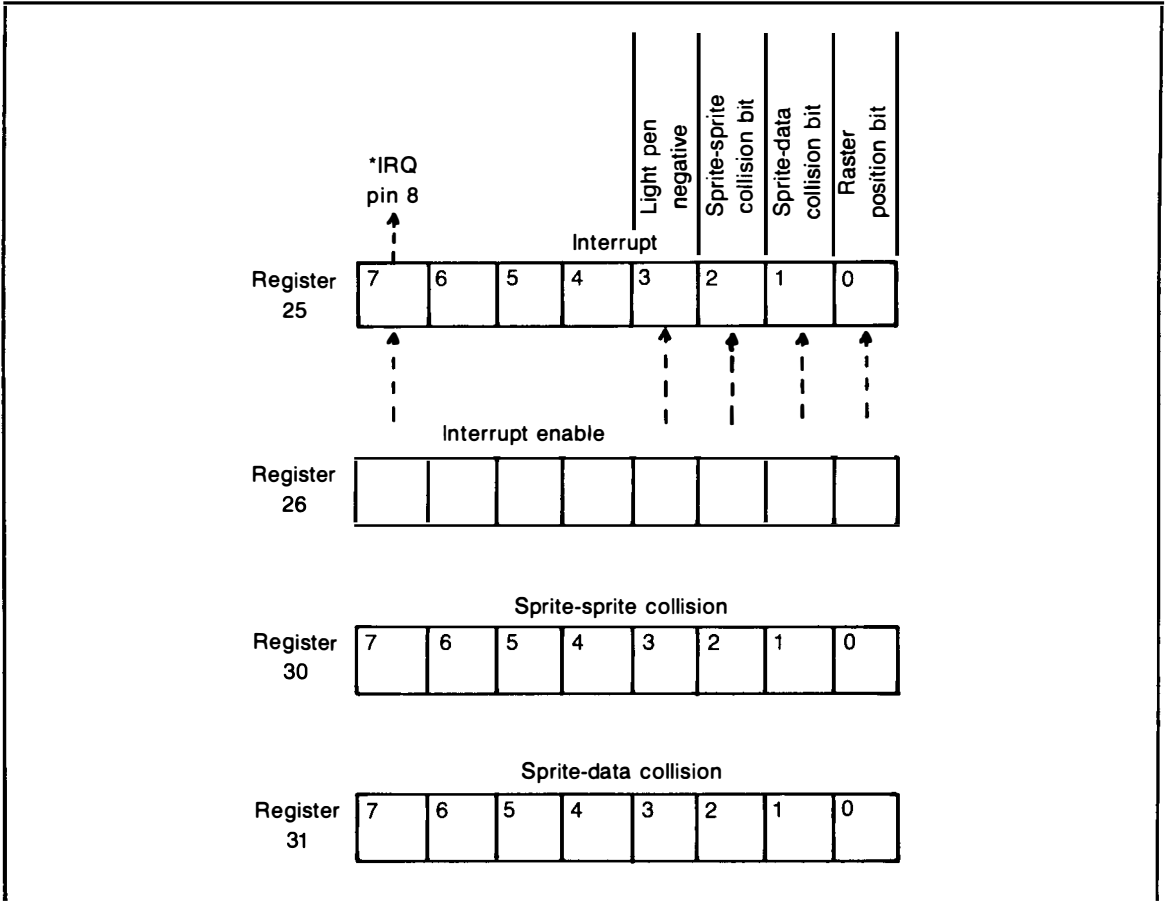


Fig. 17-11. Register 25 is the interrupt. When a collision occurs between a sprite and some display data, bit 1 becomes set. If the collision is between two sprites bit 2 is set. The interrupts are noted by registers 31 for the sprite-data collision and 30 for the sprite-sprite collision. The interrupt can be cleared by writing a 1 to the corresponding bit of register 26 the interrupt enable.

changes are made before the cathode ray reaches position 51 or after position 251.

While the raster register read is useful to eliminate flickering in the picture, a write to the raster register, including a write to bit 7 of register 17, performs another job. The data written gets latched in the registers. Then as the position of the raster on the screen changes, the position of the raster is compared with the register contents. When the position of the raster becomes the same as the eight bits in register 18 and the MS bit in register 17, an interrupt flag is thrown in register 25, the interrupt register.

## Interrupt Register

There are five active bits in the VIC Interrupt register 25. They are shown in Fig. 17-11. When the state of a bit changes from 0 to 1, it signifies that a flag is thrown and an interrupt takes place. The interrupts tell VIC that an event has taken place and that the VIC must take appropriate action.

The flag in bit 0 is the one that becomes a 1 when the actual raster position becomes the same as the raster position that has been latched in the raster register. Bit 1 becomes set when the first collision occurs between a sprite and display data. A sprite and display data collision is noted by register 31. Bit 2 is set when the first collision happens between two sprites. A sprite to sprite collision is noted by register 30. Once any of the flags are set, they stay that way till the programmer purposely resets them.

The resetting is accomplished by writing a 1 to the Interrupt Enable register 26. If you write the 1 to the corresponding bit, the same bit in register 25 will clear.

Bit 3 of the Interrupt register is set by the light pen during a negative transition of the input. Bit 7 is set automatically whenever one of the four sources of interrupts happens. This allows the interrupt register to be able to output a low through pin 8, \*IRQ of the VIC. Just because bit 7 gets set, that does not mean the low will leave pin 8 and head out into the system interrupt. In order for the low

signal to actually leave the VIC, bit 7 of the Interrupt Enable register 26 must also be set to a 1.

The two interrupt registers are very valuable to the programmer. They let him use the screen in a large number of useful ways. He is able to design split screen formats, install eight or more sprites, mix text and graphics and other important graphic techniques. It is important during troubleshooting and repair to have a good idea of the interrupts so that you can make PEEK and POKE tests to determine if the registers are operating properly.

## LIGHT PEN

Part of the light pen mechanism is inside VIC. The light pen is a device that gives the 64 a touch-pad ability. When the light pen is touched down on the TV screen, the exact position and dot it touches down on is recorded in two registers of VIC. The registers are 19 and 20.

The cathode ray scans the TV screen once every frame. Electrons impinge on every dot once every raster frame. When the electrons arrive at the spot that the light pen is touching, the pen is

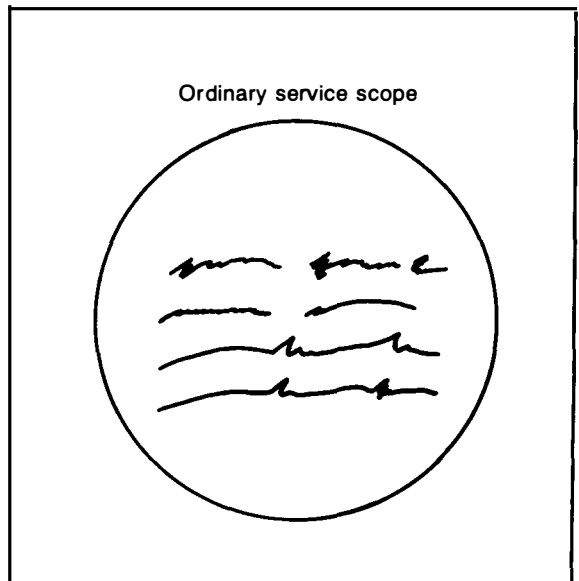


Fig. 17-12. The ordinary service scope can display the TV signals that exit the VIC easily. At pin 15 there should be this TV sync and luminance signal.

activated. Since the dot is only hit once every raster frame, the light pen latch is triggered only once every frame.

Register 19 latches the x position of the touchdown. The x position is defined by a 512 bit counter. This means there are 9 bits the counter keeps stepping through from 0 to 511. Register 19 latches the eight MS bits out of the nine. Bit 0 is not recorded.

Register 20 latches the y position of the light pen on the raster. The y position is defined by a 256 bit counter. This is the usual eight bits and the register is able to store them all.

## VIDEO OUTPUT

All of the VIC inputs and outputs shown so far have been digital highs and lows. The 47 registers can be read, written to, receive controls, and output controls. All of the processing has been in the digital circuits of the computer.

Internal to VIC are circuits that convert many of the input signals from a digital nature to analog video signals. The VIC is a digital-to-video device. If you test all the signals shown so far, the logic probe reveals their digital states. The same logic probe is useless at the analog output pins.

At the output pins, the ordinary TV service scope makes excellent tests. The output signals are each a portion of a composite color TV signal. At pin 15, the SYNC/LUMINANCE output is a composite signal containing the actual video, the vertical and horizontal sync signals, and the brightness or luminance signal that controls the intensity of the cathode ray. Refer to Fig. 17-12.

Pin 14, the COLOR output, contains all the color information. This includes the chrominance signal, the color burst and all the colors that the display is supposed to show. Refer to Fig. 17-13. When the two output pins have their signals mixed properly, the result can be input to a CRT for display.

The scope shows the signals well. You can see if the signal is getting out of VIC quickly with the scope. This is often one of the first test points covered during video type troubles.

Pin 15 is an open drain output and requires some sort of external pull up through 500 ohms. In the 64, pin 15 is connected to diode-npn circuit that provides the correct interface. Pin 14 is an open source and must be grounded through 1000 ohms. It is therefore attached to a 1000 ohm potentiometer that performs the ground return along with coupling the signal into a transistorized video output circuit.

The circuit shown in Fig. 17-14 consists of two PN2222 npn transistors, a video driver, and a video follower. The output of the npn's leave the emitter of the top transistor and connects to the emitter circuit of a third PN2222 that is transferring the SYNC/LUMINANCE signal from pin 15. The outputs of pin 14 and 15 are mixed in the emitter. The resultant composite color TV signal is then attached directly into the rf modulator circuit. It is then installed onto a TV channel three rf signal and output to the home TV.

Separate signals of either SYNC/LUMINANCE or COLOR are also tapped off before the emitter mixing and connected to the audio-video plug. These signals can be applied to a special TV

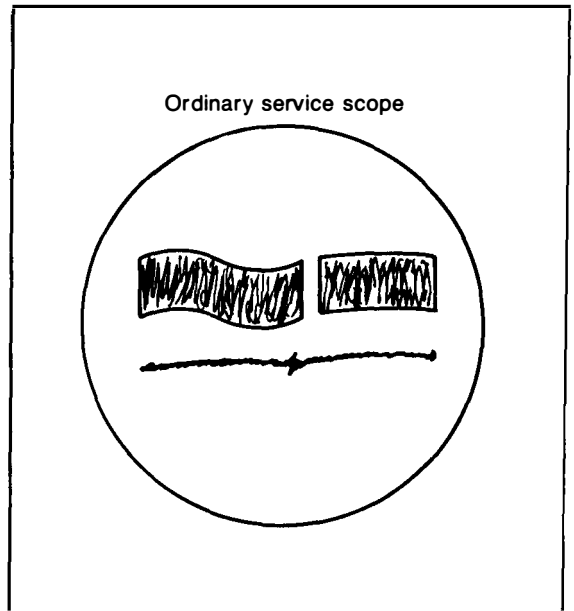


Fig. 7-13. At pin 14, the TV color signal can be viewed on the scope.

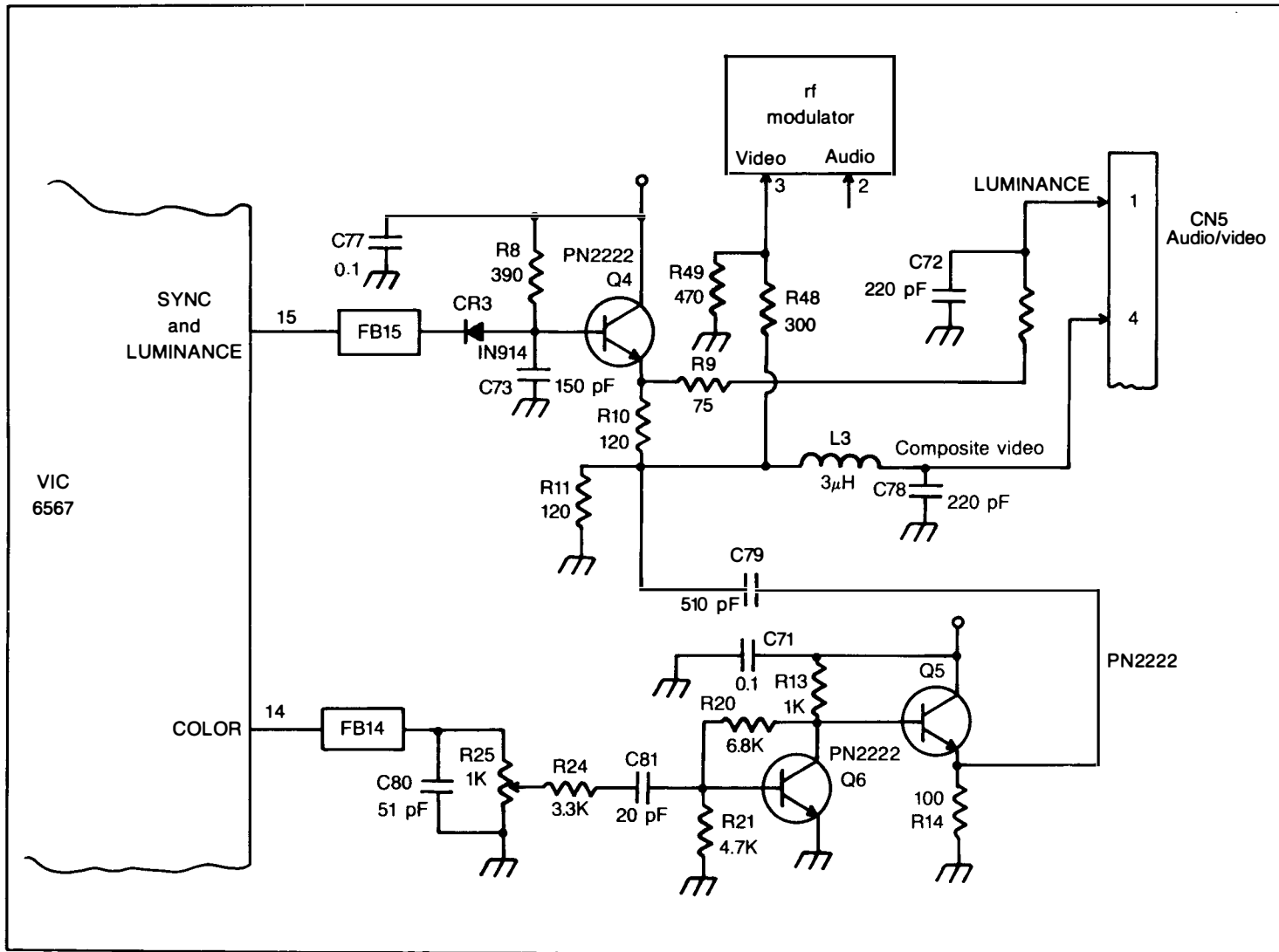


Fig. 17-14. The video output transistorized circuits are discrete transistors in early models of the 64. Later models contain these equivalent circuits in the rf modulator box.

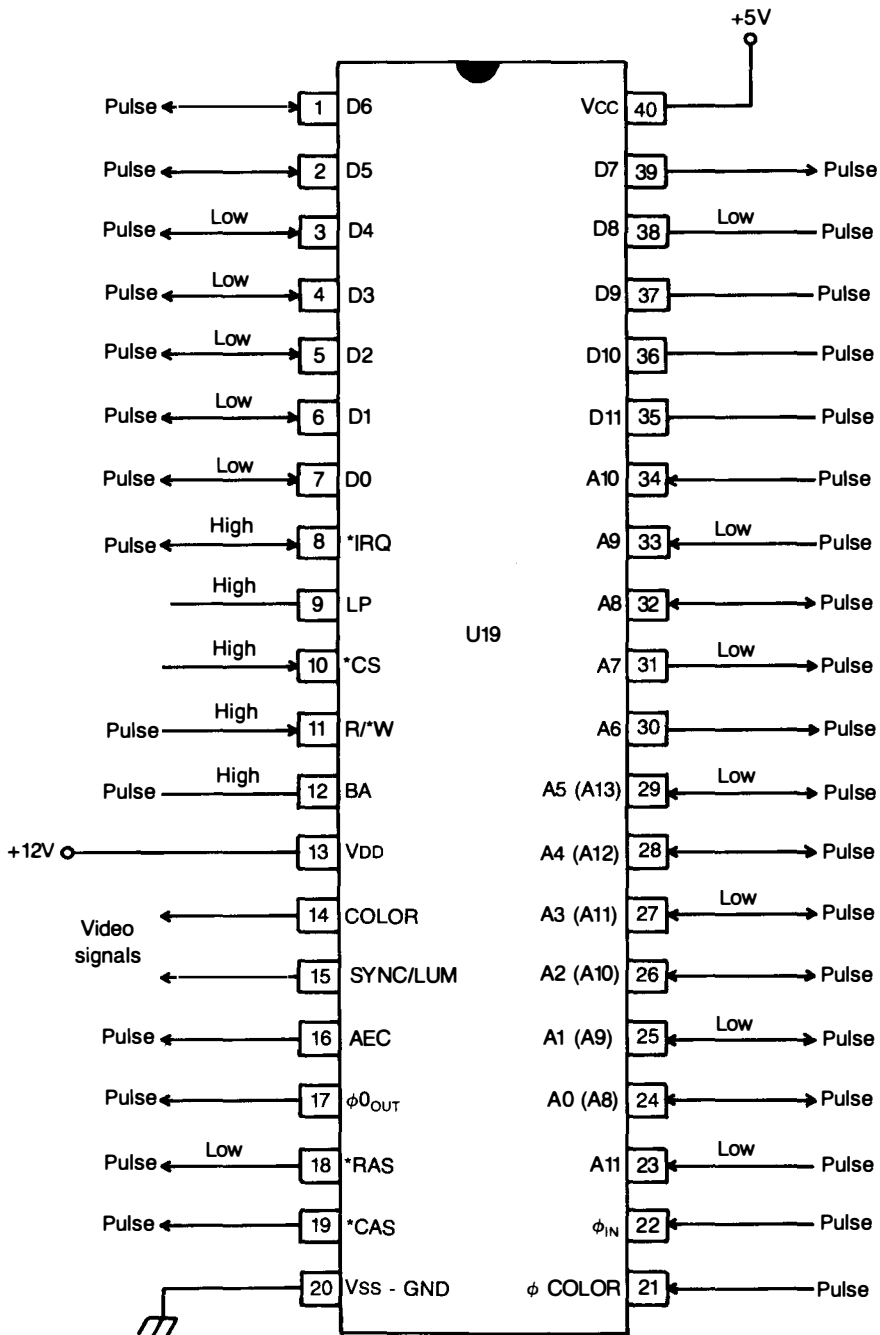


Fig. 17-15. The logic probe is useful on all the digital circuit pins. On the video analog output pins 14 and 15, the best test is with the TV service scope to display the video.

display monitor that does not require the rf modulation.

Once the signals exit the VIC they are in analog form and must be traced with the ordinary TV service scope. The signals that should be present are all ordinary TV video signals. Any loss of video could be a clue to the trouble you are searching out.

## **TESTING**

The test point chart in Fig. 17-15 can be used to compare your readings from the VIC. Any deviations from the chart indicates trouble. A logic probe or a vom and a service scope are needed to make all of the test.

## 18. Sound Interface Device

**T**HE SOUND INTERFACE DEVICE IS THE AUDIO companion to the VIC. The SID is designed to take bytes or bits from the processor, convert the bits into audio signals, and send them to the speaker in the TV monitor or a separate audio system. The created audio produces all the sound effects to further excite the graphics or to imitate musical instruments.

The SID has 29 addressable registers that create the sound effects. You can write to 25 of the registers with the BASIC POKE statement. The registers that can be written to cannot be read from. You can read from the remaining four registers with the PEEK function. The read registers are also one directional. They can only be read and not written to.

### PINOUT

The 6510 sees the SID as 29 addresses on the memory map. When the processor wants to access the SID it outputs a register address. The higher address bits are then sent over the address bus and

arrive at the PLA chip. The bits are decoded and an I/O pulse continues on to the 74LS139. The output is connected to the  $\overline{CS}$  pin 8 of the SID. When the SID is dialed up, a low enables pin 8. The pinout is shown in Fig. 18-1.

That one low though will not fully address the selected register. The SID needs more signals. It needs a clock driving pulse to open up the register. The pulse is  $\phi 2$  and enters the SID at pin 6. During the high of  $\phi 2$  SID can be accessed just as RAM or ROM is. Both the reads and the writes are performed during the high of  $\phi 2$ .

The SID knows to receive data or output data according to the input on pin 7,  $R/\overline{W}$ . If  $R/\overline{W}$  is high SID allows the processor to read the register. When  $R/\overline{W}$  is low SID accepts data bits from the 6510.

The registers themselves are addressed with the five lowest address lines, A4-A0. With five lines, a total of 32 registers could be addressed. Since the SID only has 29 registers, that leaves three extraneous addresses. Should one of the ex-

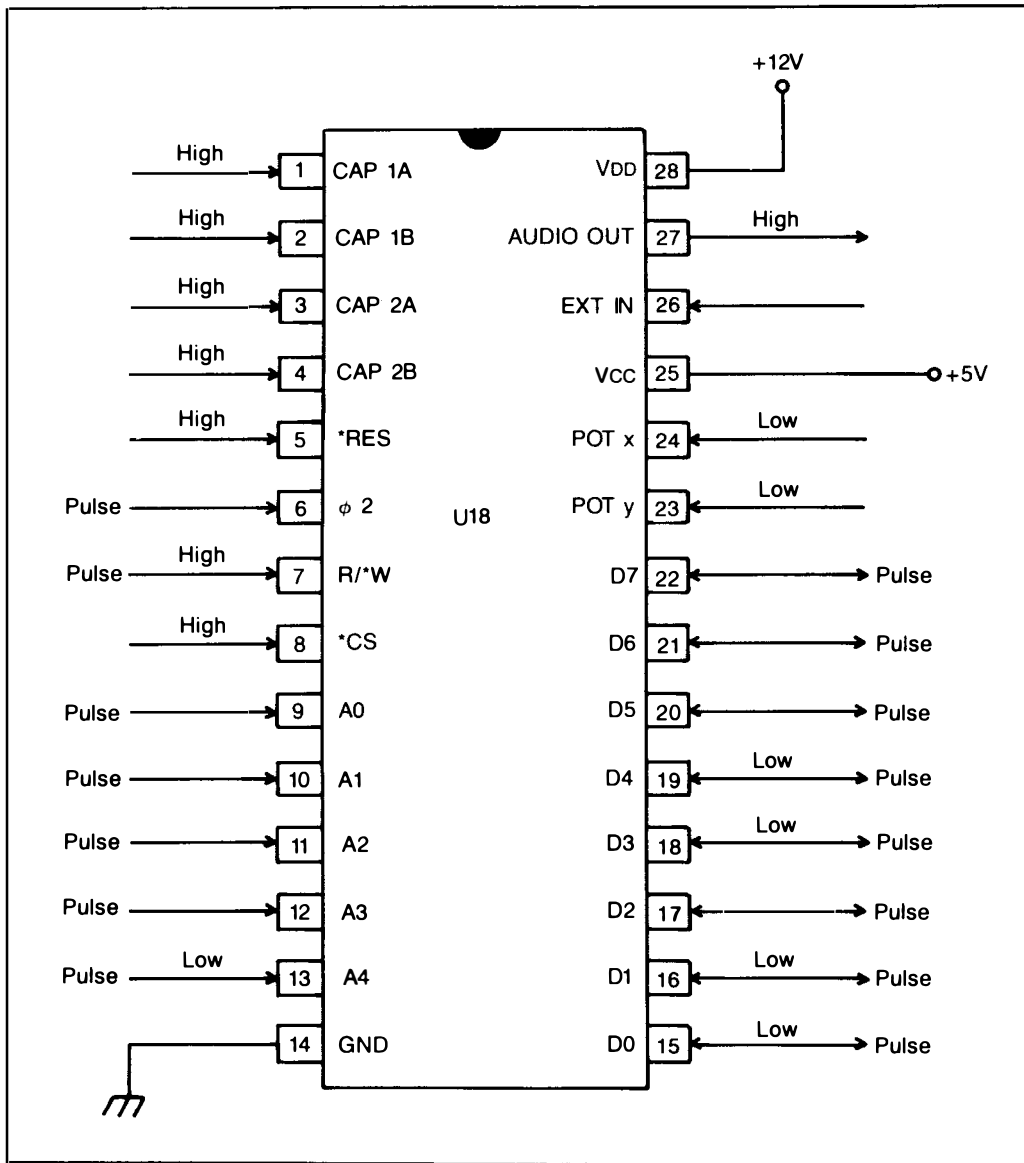


Fig. 18-1. The SID is a complex chip that needs two different supplies. There is +12 volts at pin 28 and +5 volts at pin 25.



tra registers be written to, SID simply ignores the data that arrives from the processor. If one of these registers are mistakenly read, data will arrive at the processor but it will be meaningless.

The data lines to the SID are the normal D7-D0 attachments in the system. They connect to pins 15-22 on the SID. There are data buffers in the SID to handle the data transfer. The three-state buffers are on when the 6510 writes data to the SID, or accepts data when the SID is read.

Pin 5 on SID is the \*RES input. If the reset pin is brought low for at least ten  $\phi 2$  cycles, the SID will reset all of its registers to zero. This stops any audio output that might be going on. The reset pulse is the same one that originates in the 556 and starts the processor off. The SID is started at the same time.

Pin 14 is ground. It is a good idea to always reconnect SID's ground to the power supply and not anywhere near the other digital circuits. The

VIC should be connected separately in the same way. Both the audio and the video outputs can be damaged by noise from the fast moving digital bits that are traversing the digital bus lines and chips.

Pin 28, VDD, is the +12 volt supply that energizes the drains of the FET's in the SID. VDD receives a separate power line and is filtered and bypassed thoroughly with three capacitors. They remove high frequency noise, medium frequency noise, and low hum type noise that might appear.

Pin 25, VCC, is the +5 volt supply that energizes the collectors of the transistors in the SID. This line too is separate in the effort to keep noise levels as close to non-existent as possible. There are three filter and bypass capacitors in this line too.

Pin 27 is the audio output pin. It exits from an open source buffer and contains all the outputs from all the audio generators in SID.

The total audio output energy is set by an in-

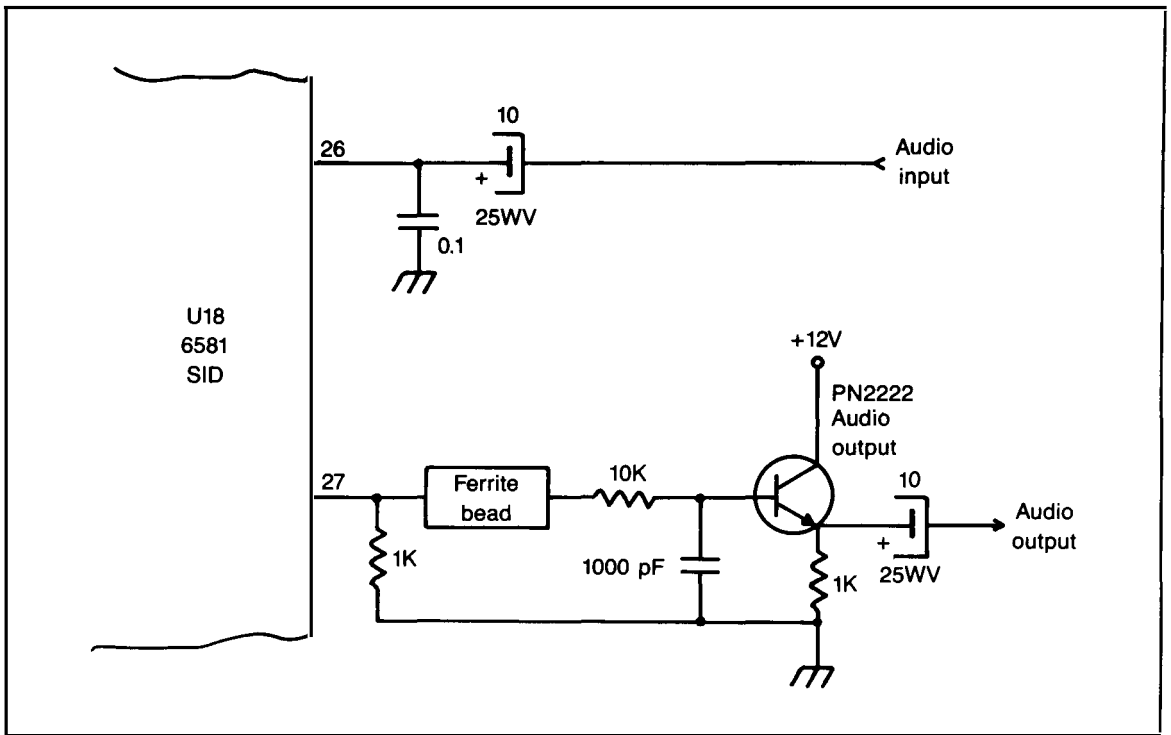


Fig. 18-2. The SID connects externally through pins 26 and 27. 26 can accept audio for processing in the SID and 27 is the computer's audio output.

ternal volume control. The volume is controlled by the programmer and the data he sends to the control. It has a peak-to-peak value of 2 volts at a dc level of six volts. A 1K resistor returns the source to ground.

The output signal in this machine is coupled to the base of an npn emitter follower in Fig. 18-2. The audio is removed from the npn through a 25 working volt, 10  $\mu$ F series filter. The 6 volt dc level is ac coupled in this manner. The signal is output to the rf modulator and also can be tapped off at the audio/video plug.

## SPECIAL INPUTS

The SID has some other very useful inputs. First of all there is pin 26. It will accept audio signals from an external source such as your voice or a musical instrument. Any signal that is input here should enter at a dc level of 6 volts and be larger than 3 volts peak-to-peak. This is accomplished in the same way the SID outputs audio. A series 10  $\mu$ F filter will perform the input chore. The input impedance at this pin is about 100K ohms.

The signal that is input at pin 26 is mixed with the audio that the SID is putting out. It is also passed through the filter. This input is valuable in case you want to use a number of additional SID chips. The only restriction to the number of additional chips you can input is the noise that each additional chip adds to the output. A large number of chips can be input. The volume control in the SID is positioned to act on the total input as well as SID's own output.

Two more special inputs are at pins 23 and 24. Refer to Fig. 18-3. They are called POT x and POT y. These inputs are connected over a pair of 1000 pF bypass capacitors. The two capacitors are designed to operate with a pair of external 470 ohm potentiometers. The pots connect to + 5 volts and varies the input voltage.

The pins are inputs to the analog to digital converters inside the SID. The A/D converters take the analog dc voltage and convert it to a digital set of highs and lows. The capacitor is designed at 1000 pF for the SID to keep pot jitter at a minimum.

The last special inputs are at pins 1, 2, 3, and 4. These hold the intergrating capacitors for the filter. All of these inputs are discussed in more detail later in this chapter. The SID has two of these capacitors installed. Each has a value of 2200 pF. One is installed between pin 1 and 2, while the other connects 3 and 4. They are fairly well matched. The capacitors are the polystyrene type.

The filter operates over the audio range between 30 Hz and 12 kHz. This is about the range the normal home TV audio outputs. In special audio applications, you could change the capacitors to different values to extend or reduce the frequency response.

## OPERATION

The SID has three sets of circuits that are all quite alike. Each circuit set is designed to output an individual *voice*. The choir operates as a total voice or each circuit can sing a solo.

Each circuit set contains a tone oscillator, a waveform generator, an envelope generator and an amplitude modulator. Refer to Fig. 18-4. The tone oscillator is variable and can be tuned to produce different frequencies. This is called the pitch of the voice. The pitch can be varied over the range the TV audio will reproduce.

The waveform generator is able to take the oscillator frequency and produce four different waveshapes at that frequency. The different waveshapes of the same frequency cause variations in the audio. The differences in sound are called tone color.

The loudness of the audio is controlled by the amplitude modulator. The envelope generator is able to create an envelope around the audio to control the volume. The volume is programmable due to the ability of the envelope generator.

The 25 write-only registers produce the sound effects, on order, as they receive data from the 6510. The processor is also able to read the four read-only registers. Two of the four registers contain the values of POT x and POT y. Another of the registers has data from the envelope register in the third set of circuits. The last register con-

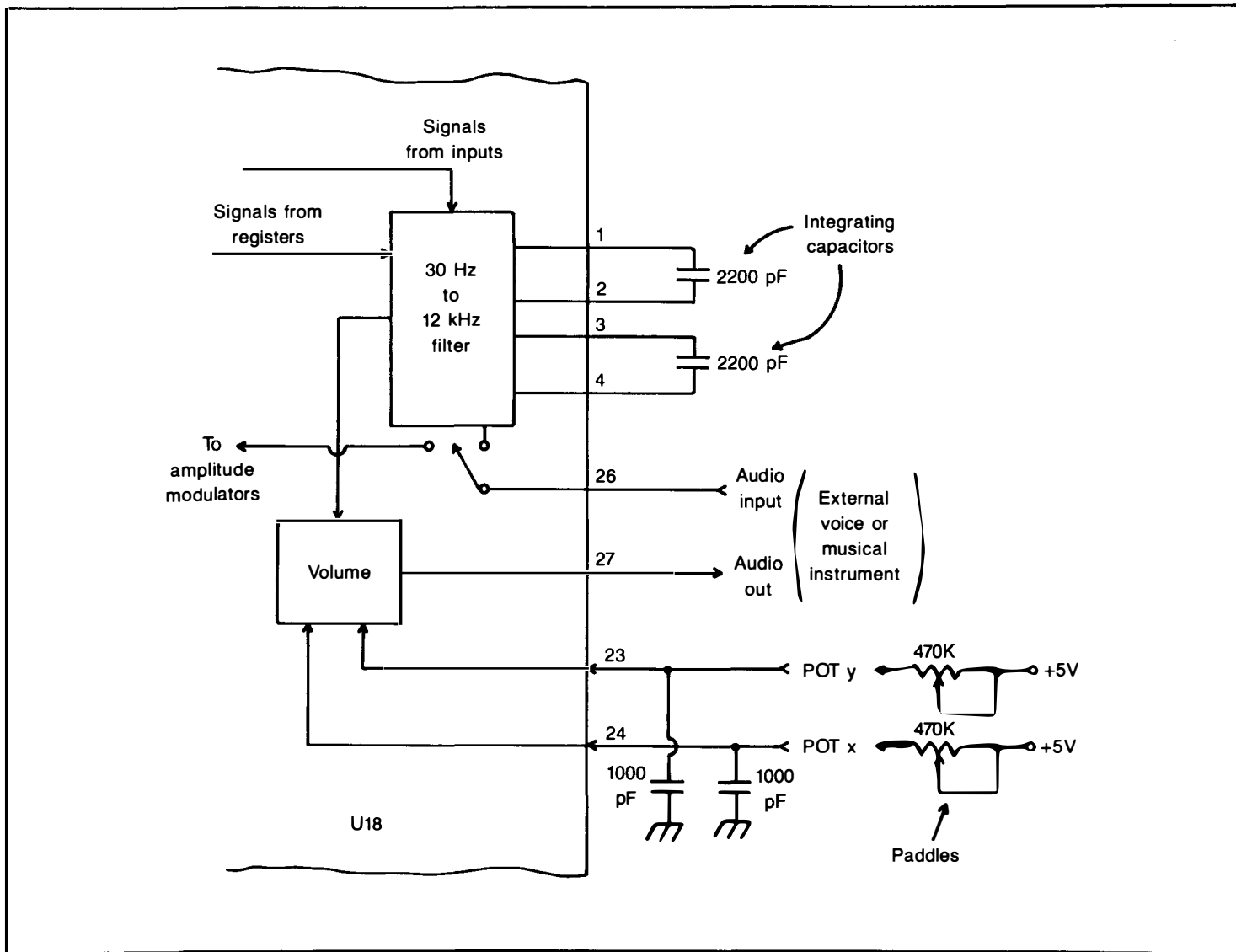


Fig. 18-3. Besides the audio input pin at 26, the SID is able to receive paddle inputs at pins 23 and 24.

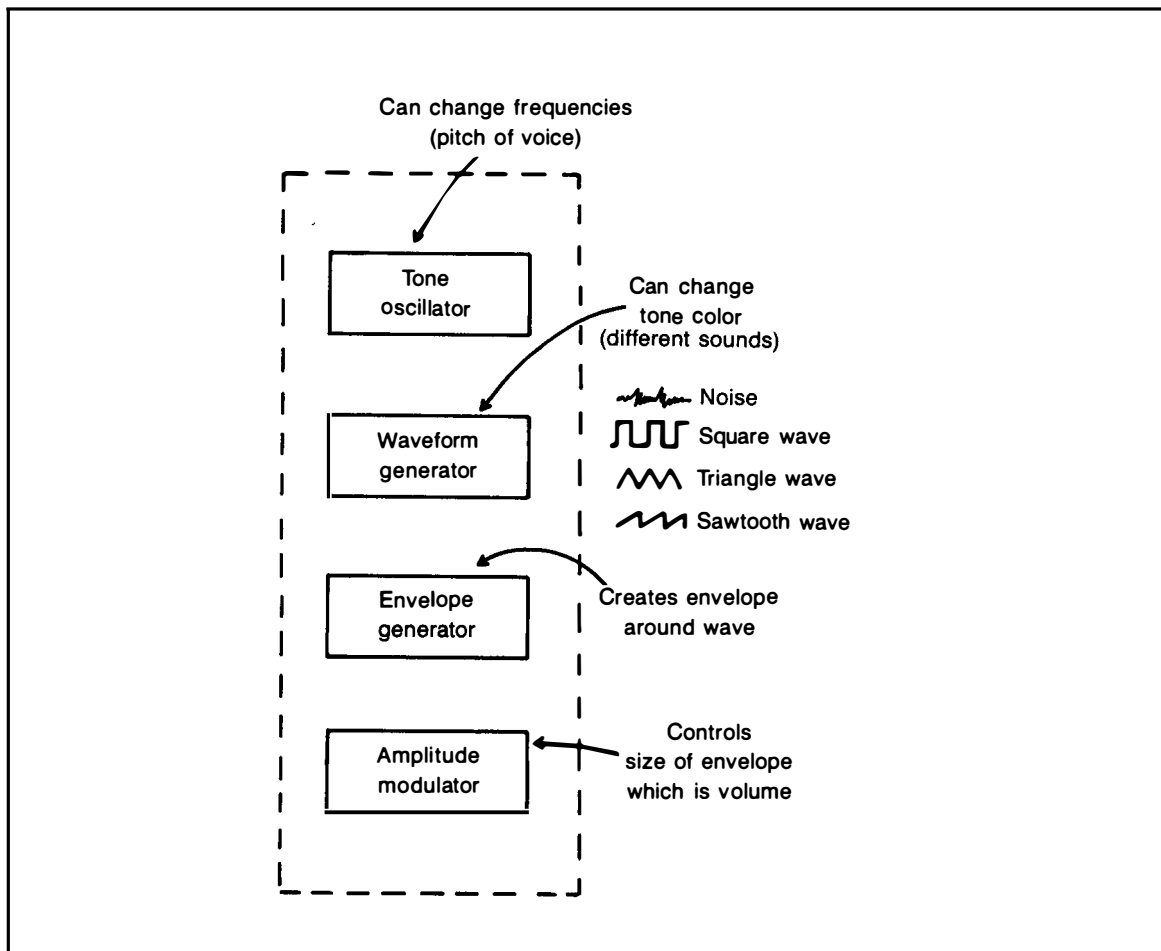


Fig. 18-4. Each voice is produced by means of four circuits working together. There is a tone oscillator, a waveform generator, an envelope generator, and an amplitude modulator.

tains the constantly changing output of the third tone oscillator.

These read outputs from the SID can be used in a number of ways. They can be mixed in with other audio to produce sound effects. The changing frequency of the third oscillator has a random movement. It is used to generate random numbers that can be used as the basis for some games. Let us examine the activity in the 29 registers as they operate.

## REGISTERS

The SID has five sets of registers. They are shown

in Fig. 18-5. There are three sets of voice registers, one set of registers for the filter and a set of four registers to handle the odd jobs. The voice and the filter registers are all write-only. The four odd job registers are the read-only types.

Each register is addressed internally by means of five bits that enter through address lines A4-A0. The registers are all byte size. The bits in the registers control the circuits that generate the audio output. Each bit is like a push button on a control board to produce sound effects and other related functions. There are seven registers assigned to each voice.

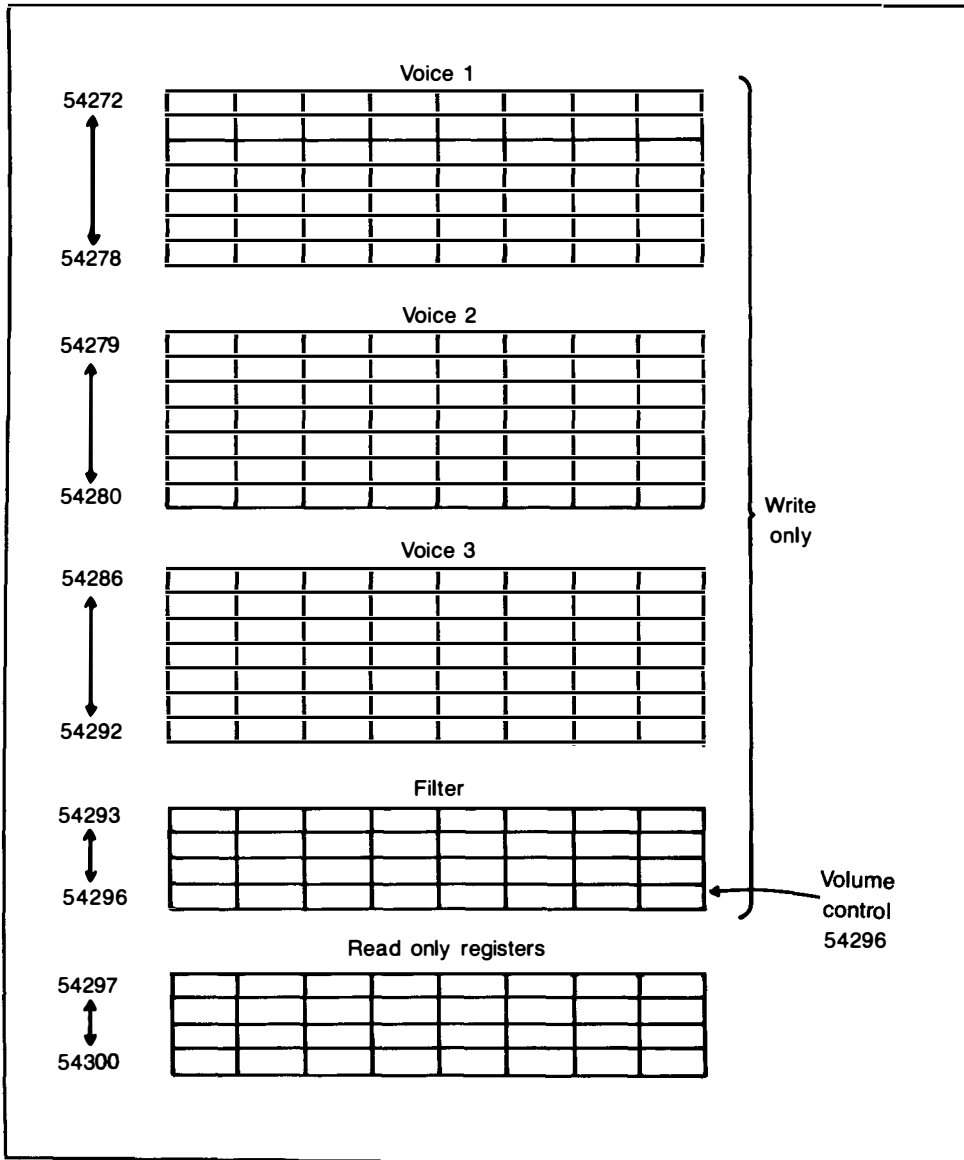


Fig. 18-5. The SID contains three sets of voice registers, one set of registers for the filter and one set for read only uses.

## First Voice Registers

Of the seven registers in Fig. 18-6, two are called Frequency High and Frequency Low. They are wired together and form a 16-bit register. The register controls the frequency of the tone oscillator in the first voice. If you POKE a decimal number into the 16 bits, the SID will generate a musical note. In your users guide, there is a table of the numbers and what musical notes they will generate when POKED.

Two more registers are called Pulse Width High and Pulse Width Low. These registers are also wired together to form a two byte register. However, only 12 of the 16 bits are used. Four of the higher Pulse Width High bits are unused. This 12-bit register controls the pulse width of the waveform that comes out of the tone oscillator. This allows the audio output changes to be smooth as the tone changes from frequency to frequency.

The pulse can vary from a constant dc output at one extreme to a square wave with equal highs and lows at the other extreme. To produce a constant low, 0000 0000 0000 is installed. To produce a constant dc high, 0000 1111 1111 is placed in the register. If you want a square wave, 0100 0000 0000 is installed. For pulses inbetween, other binary bits are placed in the registers.

## Voice Control Register

The eight bits in this register make the tone oscillator perform in different ways. Bit 0 is called the Gate bit. The actual details are discussed later in the chapter but the bit controls two functions. When the bit is set to a 1, that triggers the envelope generator and the ATTACK/DECAY/SUSTAIN cycle is begun. If the bit is cleared the RELEASE cycle takes over.

Bit 1 is the SYNC. When it is set to 1, tone oscillator 1 is synchronized with tone oscillator 3. This aids in producing desirable harmonic sounds.

Bit 3 is the TEST bit. When it is set to a 1 it resets and locks the tone oscillator at zero. It also resets the noise waveform of the oscillator and the pulse waveform output is placed at a steady dc level. This positioning is needed for program tests.

It also is useful for the programmer to be able to synchronize the oscillator with sound that is occurring outside SID.

Bit 4 when set to 1 is the control that turns the triangle waveform on in the tone oscillator. The triangle waveform is one of the ways the oscillator can run. As an audio output, it sounds something like a flute.

Bit 5 when set to 1 turns the sawtooth waveform on in the oscillator. It has sort of a brass instrument effect.

Bit 6 when set to a 1 turns on the pulse waveform in the oscillator. The Pulse Width can be adjusted by writing to the PW registers as discussed before. All types of interesting sounds can be produced from a hollow booming square wave to a squeaky reedy peep.

Bit 7 is the Noise output. Noise by its very nature is a true random device. The random noise can be adjusted from rumbling to hissing or whatever by the tone oscillator when bit 7 is set to a 1. This is the bit that can produce the explosions, rocket motors, windstorms, waves, drums, cymbals, and so on.

## Envelope Generator Registers

The ATTACK/DECAY register and the SUSTAIN/RELEASE register control the envelope generator circuits. When a sound begins and rises in volume, that is called the *attack* rate of volume. When the sound then peaks out and falls in volume, that is called the *decay*. There is an 8-bit register in the voice that controls the envelope generator that produces the ATTACK/DECAY. The four highest bits are attack bits. The four lowest bits are the decay bits.

The mid range of a sound that goes through attack and delay is called the sustain. *Sustain* is a volume level. When the sound quits, it falls to no volume at all, dead quiet. The rate at which the volume falls to nothing, is called *Release*. Figure 18-7 illustrates these four concepts.

A companion register to ATTACK/DECAY is the SUSTAIN/RELEASE register. For more details on how to use the registers during program-

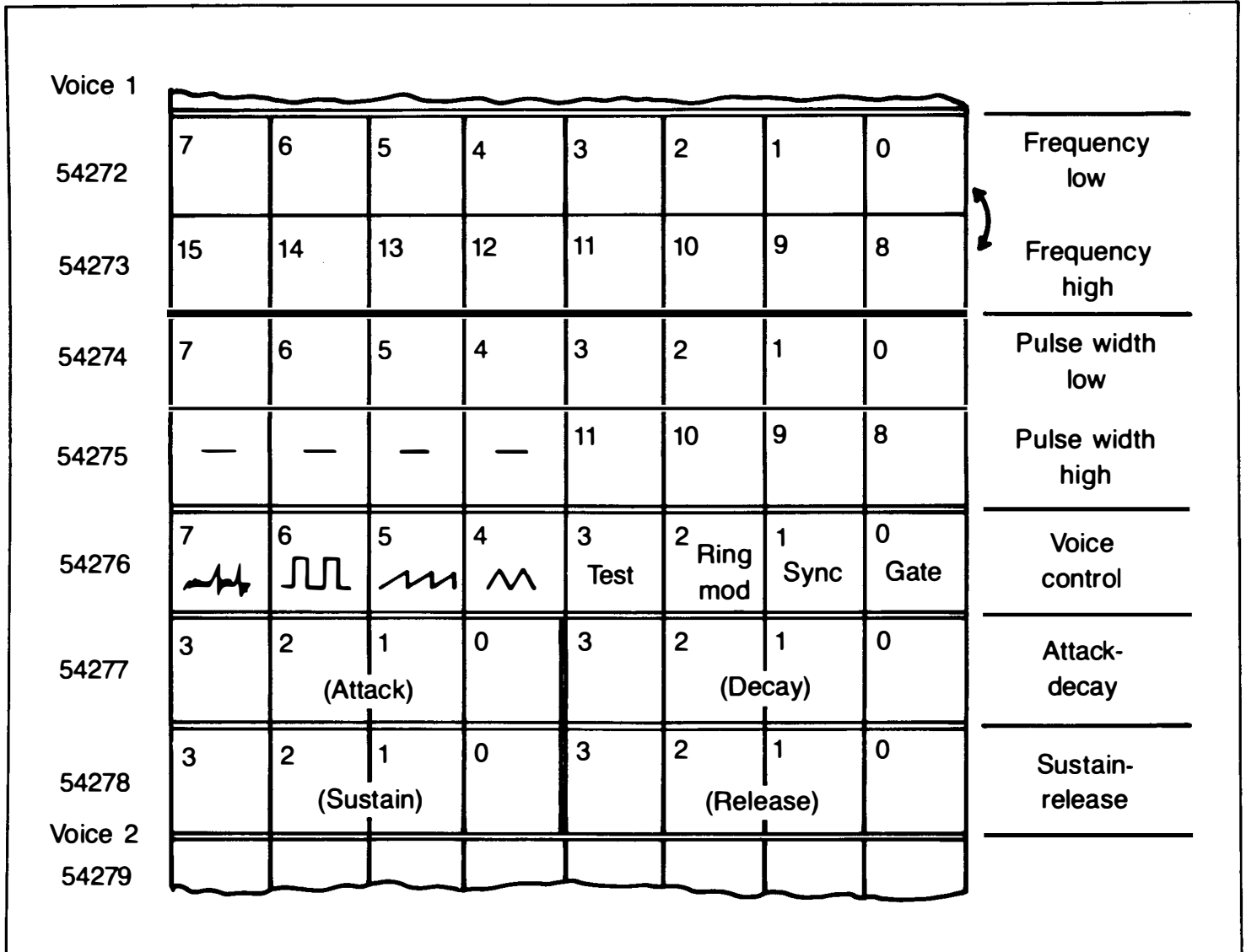


Fig. 18-6. In each voice, there are seven control registers.

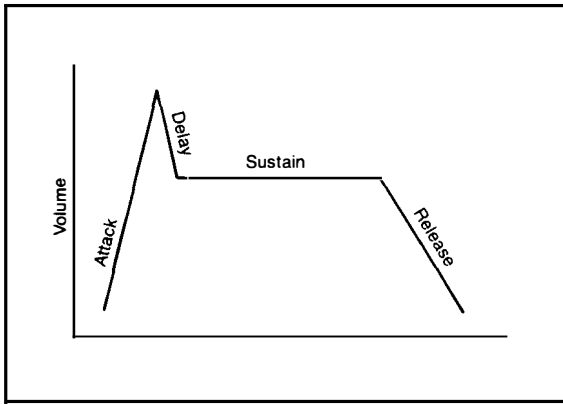


Fig. 18-7. When a sound begins and increases in volume, it is called the attack. As it peaks out and falls in volume, it is called the delay. The mid range of sound after attack and delay is known as sustain. When the sound quits, it goes into the release.

ming see the user manual that came with your 64. The bits in the SUSTAIN/RELEASE register are laid out like the ATTACK/DECAY register. The higher bits are used for the sustain control and the lower bits for the delay. When the programmer uses these registers, he must follow carefully all the rules of operation. Some truly amazing sound effects can be produced with attention to the details.

## The Other Voices

The two other voices and their seven registers are essentially identical to the first voice. The only difference between them are the synchronization of the oscillators and the ring modulated replacement of the triangle waveform.

When the voices are operated, you would have to select first of all the frequency you want to produce. Next there is the choice of the waveshape. Third you must pick out the effect that is needed. This is done with the SYNC and RING MOD bits. The envelope rates are decided on after that. When to turn the sound on and off and the length of time you want the sound to stay on is figured next. The voices can be used as solos, or as a choir.

## Filter Registers

The first two registers, FC LOW/FC HIGH in

the filter set, are combined to form a double-byte register. However, only 11 bits are needed for the job so bits 3-7 of FC LOW are not used. The job the registers perform is to set the cutoff frequency of the filter. The frequency used is 30 Hz to 12 kHz. The registers can be written to, which makes the filter a programmable circuit. Refer to Fig. 18-8.

The filter can be tuned to resonate at particular frequencies. When it is tuned to a frequency, sounds at that range have a much sharper sound. All the frequency components in that range are emphasized.

The higher bits of the RES/FILT register are wired so they can tune to a particular frequency range. Since there are four bits to the control, 16 resonant settings can be made. 0000 produces no resonant effect, and 1111 produces the finest tuning.

Bits 0-3 of the register serve a different purpose. Each bit acts as a switch to route four different signals through the filter. Bit 0 works on the route the first voice takes to the audio output pin 27. When bit 0 is cleared to 0, the voice bypasses the filter and goes directly to the audio output. When bit 0 is set to a 1, the voice is forced through the filter and is processed accordingly.

Bits 1 and 2 perform the exact same routing job for the second and third voice outputs. Bit 3 also routes a signal the same way. It handles the audio input that enters pin 26. A 0 in the bit bypasses the audio input around the filter. A 1 in the bit forces the audio input through the filter.

## Mode/Volume Register

Here again the eight bits in the register are separated into two nybbles, with the nybbles performing similar jobs with its four bits. The higher nybble, bits 4-7, decides on the mode of operation the filter will perform in. Bit 4 is called the LP bit, since it works with the low-pass qualities of the filter. When bit 4 is set to a 1, the low-pass output of the filter is used. All frequencies below 30 Hz are passed as is. The frequencies above 30 Hz are attenuated. This produces hearty sounds.

Bit 5 is called BP for bandpass outputs. The



Filter					2	1	0		FC low
54293	—	—	—	—					
	10	9	8	7	6	5	4	3	FC high
54294									
	3	2	1	0	External in filter	3 Filter 3	2 Filter 2	1 Filter 1	Resonance filter
54295		(Resonance)							
	Voice 3 off-on switch	High pass	Band pass	Low pass	Volume 3	Volume 2	Volume 1	Volume 0	Mode volume
54296									

Fig. 18-8. The filter sets the cutoff range of the audio between 30 Hz to 12 kHz.

passband for the audio output is between 30 Hz and 12 kHz. When bit 5 is set to a 1, the frequencies in the passband are allowed through and the frequencies above and below the band are attenuated. This mode produces normal sound with no harmonic content.

Bit 6 is the high-pass output (HP). All the frequencies that are above 12 kHz are passed as is, and all the frequencies below 12 kHz are attenuated when the bit is set to a 1. The sound produced with this filtering is tinny.

Bit 7 is the off-on switch for the third voice. When set to a 1, the third voice is disconnected from the audio output circuits.

The lower nybble of the MODE/VOL register is the VOL part. The higher nybble is the MODE section. Bits 0-3 are able to choose between 16 levels of volume that will emanate from the audio output. 0000 will kill volume altogether, while 1111 produces maximum volume. The bit layouts in-between produce varying volume levels.

## Writing and Reading

The preceding registers were all write-only types. The processor is able to write to all those registers and the individual bits to produce the

many and varied sound effects. BASIC, with its POKE statement, can work easily with all the registers. During troubleshooting each register could be POKEd at will to check its operation.

The next four registers are read-only. BASIC with its PEEK function can read all the registers and the bits.

## Pot Registers

The next two registers are the ones where the positions of the two POTs, x and y, are stored. The input to the register for POT x is pin 24 of the SID. As the potentiometer is varied a dc voltage is varied. The voltage enters an analog-to-digital circuit which changes the voltage from a single dc value to a relative binary number between 00000000 and 11111111. The zeros represent minimum resistance and the ones maximum resistance of the pot settings. Refer to Fig. 18-9.

The binary values are stored in the POT register x. The value is always the present pot setting and is updated every 512  $\phi$ 2 clocks. POT y works in exactly the same way except for the input that enters at pin 23. The 6510 can read the two POT registers and always know the positions the pots are set at.

## OSC 3/Random Register

This register is able to generate random numbers as one of its abilities. Noise is a true random sound. When the noise waveform is produced in the third voice by setting the noise bit 7 in the control register to a 1, this register stored the binary numbers that represent the random changing noise. The register stores the upper eight bits of the third voice's tone oscillator. These bits are the result of the changing noise waveform being generated. The random numbers can then be read by the processor and used in games. Refer to Fig. 18-9.

Besides producing random numbers, this register can be used for many timing or sequencing purposes. Since this register is constantly changing and recording the upper eight bits of the third voice, different waveforms will produce different types of changing number sets. You've seen how the register bits record the random wave shape. If you set bit 5 of its control register to a 1, the sawtooth waveform is generated by the third voice.

The OSC 3/RANDOM register will then have its bit change in time with the changing sawtooth waveform. Each sawtooth will increment the register by 1. The binary bits will count from 00000000 to 11111111, over and over again. When bit 4 of the control register is set to 1, the triangle waveform will drive the OSC 3/RANDOM register. The register will then count in a slightly dif-

ferent way. It will start the same and count from 00000000 to 11111111. Then instead of starting over and continuing its incrementing, it will simply start decrementing the count backwards till it reaches zero again. At that time, it will go back to the incrementing.

The register can record the action of the pulse waveform too. The square-wave type pulse goes from high to low and back and so on. The register counts 11111111 as it is high and 00000000 when it is at a low. Just as the square wave does not have any gradations between the high and low the register does not record any binary numbers that represent inbetween states.

The main function of this register though is as a modulation generator. The numbers that can be generated by changing the different waveshapes available can be used in a program to produce sound effects. Combinations of this registers numbers plus the bits generated by other registers can make the audio output sound like sirens, moaning, groaning, and all sorts of interesting effects. Other registers that will work smoothly with the OSC 3/RANDOM register are the pulse width registers, the filter and the oscillator.

## ENV 3 Register

The last read-only register is named ENV for Envelope. This register lets the 6510 read the out-

Read only									
54297	7	6	5	4	3	2	1	0	Pot x
54298	7	6	5	4	3	2	1	0	Pot y
54299	7	6	5	4	3	2	1	0	OSC3 RANDOM
54300	7	6	5	4	3	2	1	0	ENVELOPE 3

Fig. 18-9. All of the registers in the SID are write only except for these four which are read only.

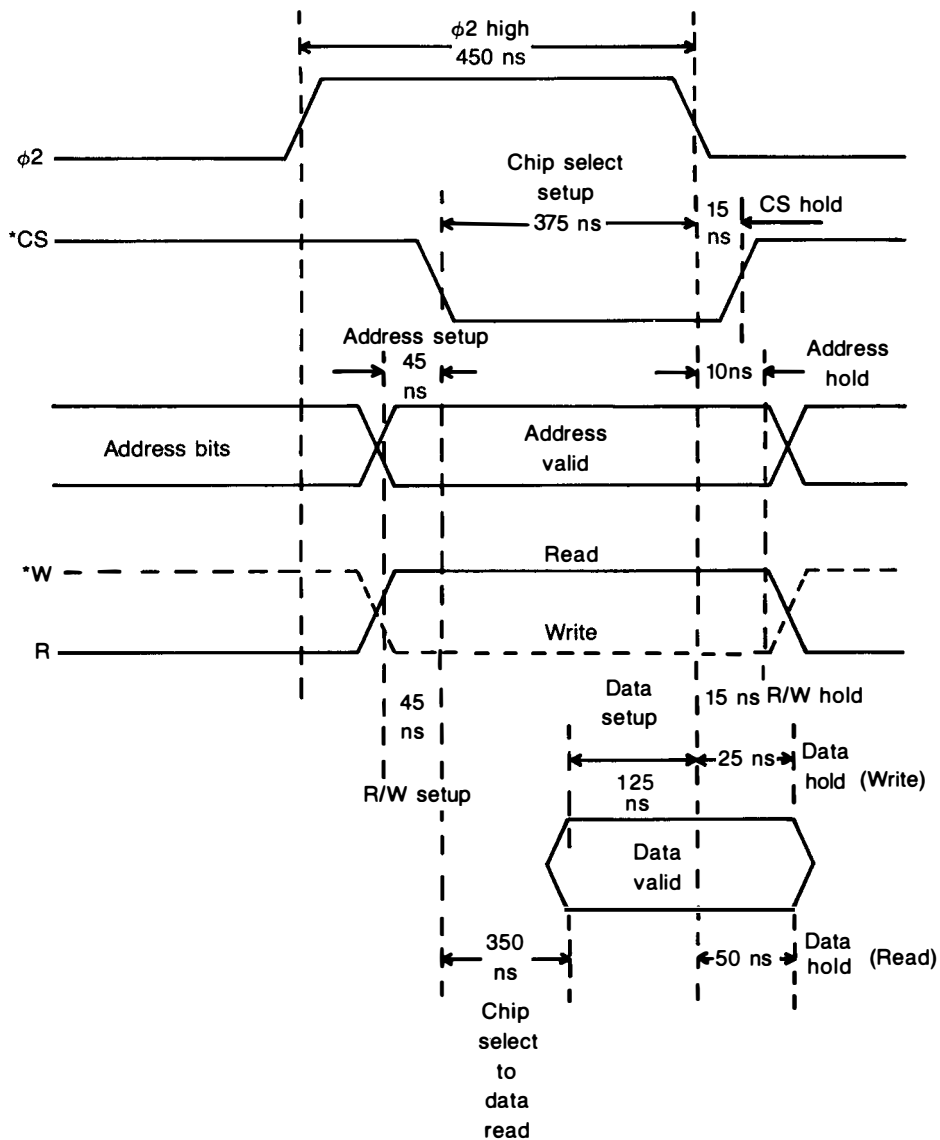


Fig. 18-10. SID is accessed only during the high of  $\phi 2$ . At that time  $*CS$  goes low, the address bits go through a setup time, a valid time, and a hold time. For a write,  $*W$  goes low and for a read R goes high. The data is valid near the end of  $\phi 2$  and is strobed into the SID's registers as  $\phi 2$  falls.

put of the third voice envelope generator. This register only records the output of the envelope generator when the EG is running. The numbers produced in this register can be added to other registers for more sound effects. Refer to Fig. 18-9.

## TIMING

The SID is accessed by the 6510 in a very straightforward way. The 29 registers look to the 6510 like ordinary locations. Most of the bits that travel back and forth are used to produce sound effects.

Your Commodore 64 is designed to run at near 1 MHz. This means a full cycle is 1000 ns. The SID is accessed only during the high of  $\phi 2$  which consumes 450 ns. When \*CS pin 8 is made low it must provide at least 375 ns of \*CS setup time during the  $\phi 2$  high. Once the data is accessed the \*CS stabilizing Hold time needs 15 more ns. Refer to timing diagram in Fig. 18-10.

As the addresses enter A0-A4, pins 9-13, the

bits need a setup time of 45 ns during the high of  $\phi 2$ . After the data is accessed, the address bits need 10 more ns as hold time.

The next signal that is also entering SID is the R/\*W at pin 7. It runs neck and neck with the address bits and needs the same setup time (45 ns) and the same hold time (15 ns).

The write-only registers receive their contents from the data pins when R/\*W is low. The registers need a setup time of 125 ns to settle the bits into place. Once the bits are stable, a hold time of 25 ns is needed to be sure.

The read-only registers place their contents onto the data pins when R/\*W is high. The total access time from when the chip is selected till the data is placed on the system data bus is 350 ns. Then it takes a data hold time of 50 ns to stabilize the bits in the data system.

## TESTING

You can check out SID quickly by POKEing the

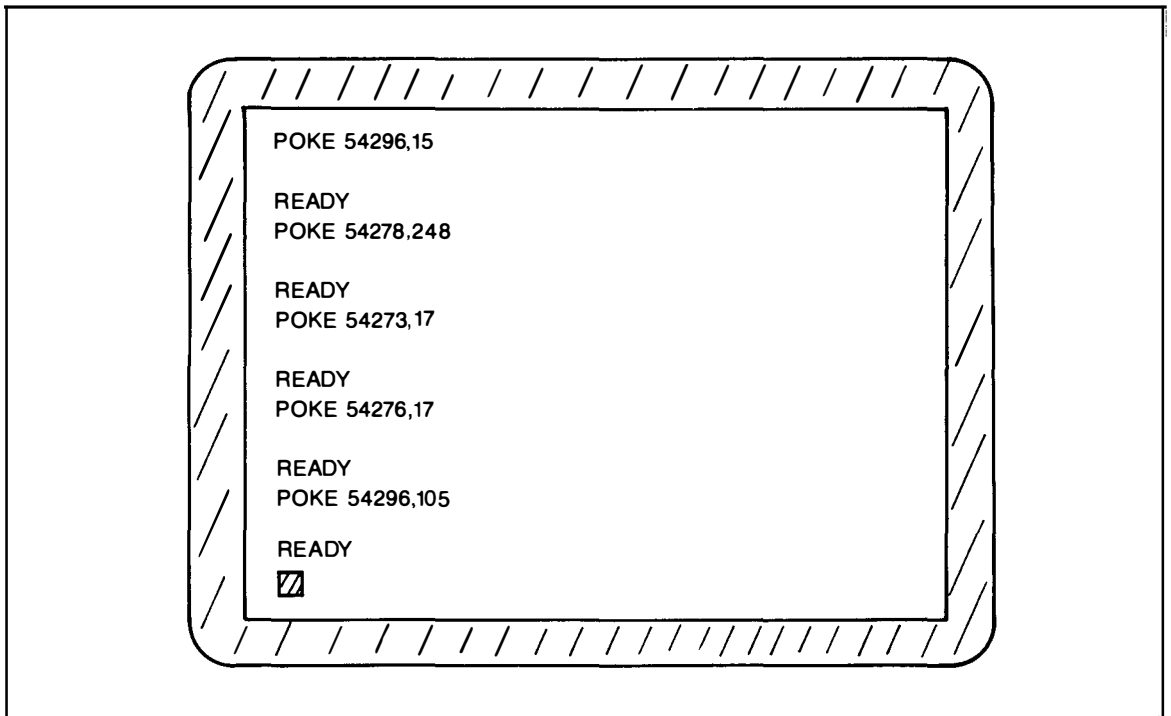


Fig. 18-11. You can check out SID quickly with these POKEs.

numbers in Fig. 18-11 into the SID. It will produce a continuous tone from voice 1. The first POKE sets the volume control register. The second POKE arranges the A/D S/R properly. The third POKE sets the note in the high frequency range. The fourth POKE sets the waveform to a sawtooth.

In a good SID, the note should come on loud and clear. If it does not the SID has trouble. To turn off the note, POKE 54296, 0. This shuts down the volume register. To check out the other voices or the high freq register, simply substitute the other register addresses.



## 19. Inputs and Outputs

**W**HEN YOU SIT AT YOUR COMPUTER YOU are part of the system. Your fingers input the data from your brain. The keyboard codes your data into the binary bits the digital circuits must work with. Your fingers on the keyboard are your input into the computer. You are an external device.

The VIC processes your input data and displays it onto the TV screen. You read the screen, and you can tell from the way the data becomes assembled how you should continue your computing. The data on the screen is the input, through your eyes, back into your brain. The computer and you are a system. Either the computer is an extension of you, or you are an extension of the computer.

Besides interfacing with you, the 64 is able to receive inputs from joysticks, paddles, and a light pen. It is also able to output data directly to the cassette, disk drive, printer, and modem. The 64, in addition, has an expansion plug that can connect external ROM cartridges and other devices into the memory map.

There are seven port plugs to handle the inputs

and outputs. There are two 9-pin input control ports that are important for game playing. They input joystick positions and pushbutton impulses.™ A 44-pin cartridge expansion plug is able to connect to the internal buses for cartridge ROM insertions. The five pin audio/video plug is able to output audio and video as well as input external audio. A 6-pin serial input/output connection can send data to a printer or communicate with a single disk drive. The 12-pin cassette I/O slot controls the cassette motor as well as the tape reading and writing. A 24-pin user I/O port allows connections to a printer, a modem, or another Commodore 64.

Any easy way to check out all the input and output ports and devices is to let the computer test itself. A diagnostic program can be used. You can write one for yourself if you are capable or purchase the software. One such program that I've seen is called *64 Doctor* and is found in software stores. It is the work of Computer Software Associates, Randolph, MA 02368. It comes on a disk or on tape. The *64 Doctor* will check out the keyboard, the TV

monitor, the three voices of SID, the disk system, the printer, the cassette, and the joysticks. It tells you whether the device and the ports are operating correctly or not. The *64 Doctor* or a program like it can be useful in isolating a defective peripheral system.

## CONTROL PORTS

When a joystick, paddle, or light pen refuses to work either the device is broken or the computer has a defect. If available, a new input device should be tried. When the new joystick restores the activity, then the old one was defective. Should the new one still not operate, then the computer becomes the suspect. The place to begin troubleshooting is at the input control port. There are nine pin ports. Each port can connect to a joystick or pair of paddles. Port A is also able to have a light pen plugged into it. These ports are shown in Fig. 19-1.

On each port, pins 7 receive +5 volts and pins 8 are ground. Pins 1-4 are the joystick inputs. Port A connects its joystick to the port pins PA0-PA3 of the keyboard CIA. Port B connects to the port pins PB0-PB3 of the same CIA. Even though the keyboard columns and rows are also connected to the same pins, there is no conflict since the keyboard and joysticks are rarely if ever on at the same time.

The buttons on the joysticks are connected to PA4 and PB4 of the CIA. The four joystick positions and the button are all switches. The position switches are used to direct the image on the screen, and the button is a fire switch. The switches connect to the lower five bits of the locations in the CIAs. The switches are held high till a direction is chosen or the button is pressed. Then the switch forces the bit to a low.

The quick way to check out these ten inputs on the two ports is with the logic probe. A high should be present when the computer is first turned on and the keyboard is not touched.

The other two inputs on the control ports are for the pot x and pot y inputs that go to the SID chip. These two inputs are checked easily with the logic probe. They are both lows and remain low all

the way through to SID.

The light pen shares a pin with the A fire button input. If the button is working, a good light pen will too. Test for the high that the button uses to operate with.

## EXPANSION PORTS

The expansion plug is the large 44-pin connector with 22 connections on the top side and 22 more on the bottom. This port is the most convenient interface in the 64. It permits the user to attach to almost every important signal in the machine. The most important access is to the address and data bus lines. Also available are the  $\phi 2$ , R/\*W, \*IRQ, \*NMI, \*RESET, and many others. As a result of this convenient access, this port is also a good place to test for most of the signals that are supposed to be coursing through the computer.

To aid you during point by point checking, the female edge connector has the upper 22 pins numbered 1 through 22. The lower edges are labeled A through Z with G, I, O and Q missing. There are four ground connections 1, 22, A and Z. Pins 2 and 3 are the +5 volt supply. They are filtered with a 10  $\mu$ F, 25 WV capacitor.

The eight data bus lines are connected to pins 14-21. The 16 address lines are at pins F-Y. R/\*W is on pin 5, and \*IRQ on 4. The bus lines should all show logic-probe pulses, and the two control lines are normally held high. To turn on the interrupt, it must be forced low. Pins 8 and 9, \*GAME and \*EXROM, are external device inputs to the PLA and are held high when not being used. Pins 11 and B are \*ROML and \*ROMH outputs from the PLA and are also held high when not being used. They go low when active. Pin C is \*RESET an input or an output. It is also held high when off. It goes on as it is forced low. \*NMI is an unused input and output at pin D. Pins 7 and 10 are both outputs from the decoder chip 74LS239.

$\phi 2$  is on pin E. Pin 6 is the video dot clock that runs at about 8 MHz. All of the system timing is the result of that frequency. Pin 12 is the bus available signal BA that originates in the VIC. It is held high until the VIC takes control of the bus

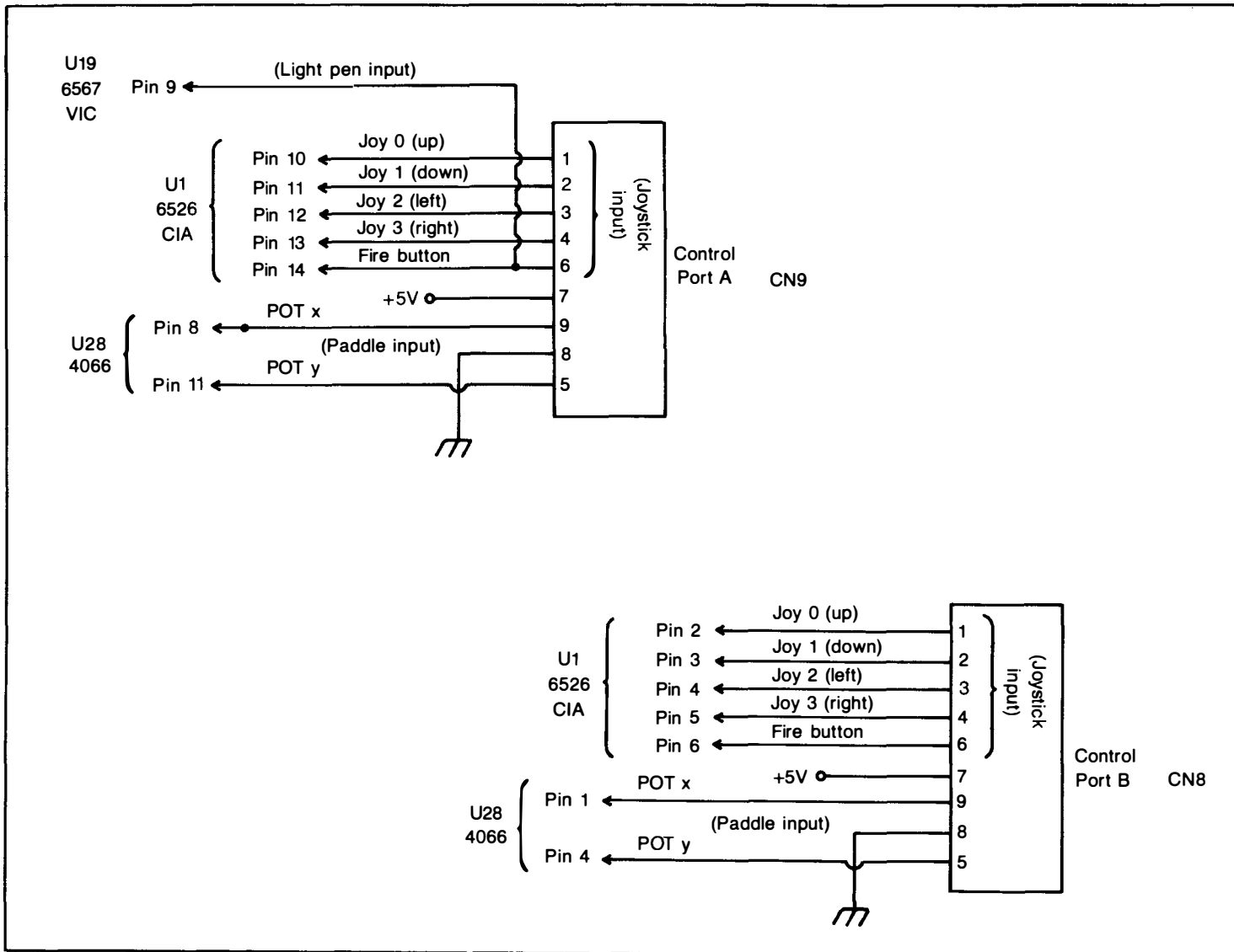


Fig. 19-1. The two control ports are able to input data from the joysticks, paddles and the light pen.



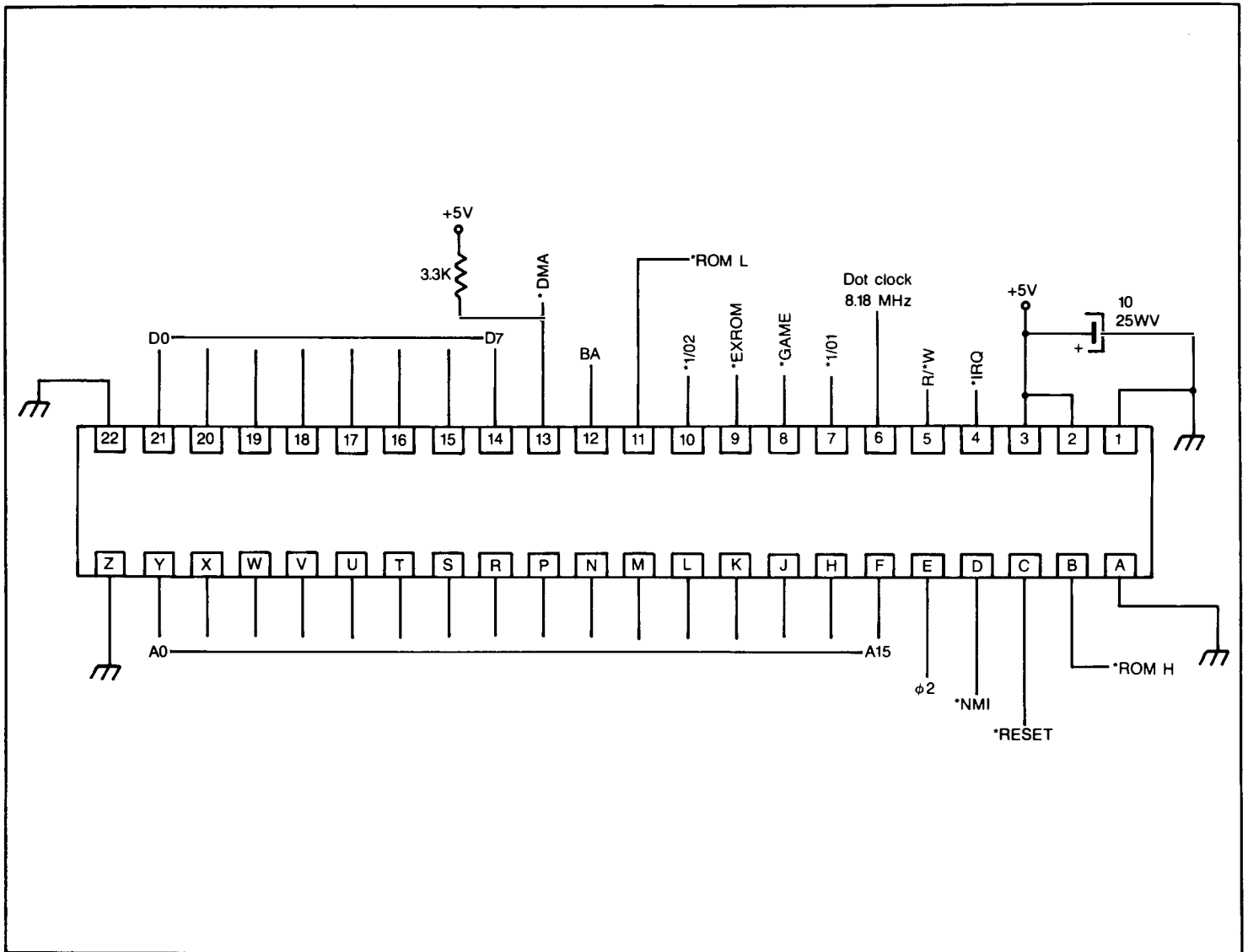


Fig. 19-2. The 44-pin cartridge expansion port allows access to every important signal in the computer.

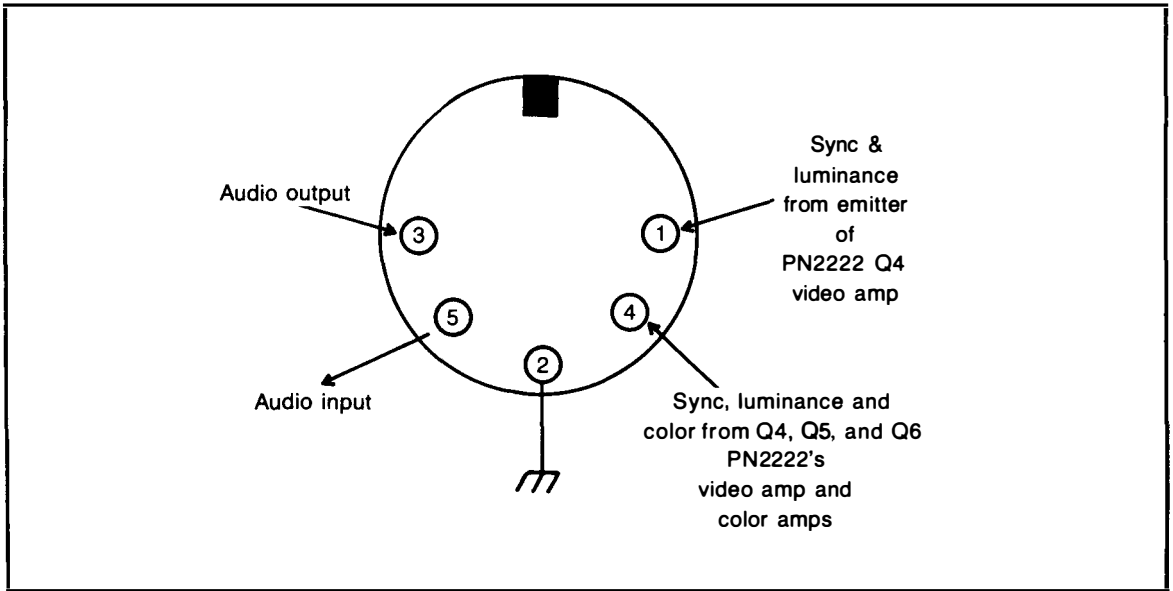


Fig. 19-3. The audio-video port is an alternative interface for the rf modulator circuit. It can output audio and video or input external audio.

lines. It goes low three cycles before the VIC goes into action and stays low till the VIC is finished accessing and displaying the TV signals.

Pin 13 is the direct memory access, DMA. It is held high through a 3.3K pullup resistor. An outside microprocessor can be connected to this interface and force this line low during a  $\phi 2$  low. When that happens, the R/\*W line, the address bus, and the data bus all three-state. The outside processor can then time up with VIC and run the 64. When not in use, it should be high.

## AUDIO-VIDEO PORTS

In most of the 64's in use, there is a five pin audio-video port. Refer to Fig. 19-3. Pin 1 of the port receives the sync and luminance signal that has come out of pin 15 of the VIC and was passed through the video amplifier (a PN2222 transistor). Pin 4 of the port receives the composite video that was formed from the VIC's color signal. These signals are easily viewed here on the ordinary TV scope.

Besides these signals being available individually at pins 4 and 5 of the port, they are mixed and

sent to pin 3 of the rf modulator. Here again the signal can be traced right to pin 3 of the modulator.

In some newer models of the 64, the video amplifiers are placed into the rf modulator and disappear as discrete components. The VIC outputs are then injected directly to the rf modulator. There is still an audio-video port but it is an extension of the rf modulator box and not a separate entity.

Getting back to the audio-video plug, pin 3 is the audio output and pin 5 is a convenient audio input. At pin 5, some external device can insert audio for mixing in the SID chip. Pin 2 is ground for the port.

## SERIAL I/O PORT

The serial bus has six pins. Pin 2 is ground, and pin 6 is connected to \*RESET. The other four pins perform the serial transmission of data. The data is capable of going in both directions. This port is illustrated in Fig. 19-4.

Pin 1 is the serial service request in, \*SRQ IN. The \*SRQ IN line can be connected to a number of external devices. \*SRQ IN is held high by a 3.3K pullup resistor. If a device wants service, it pulls

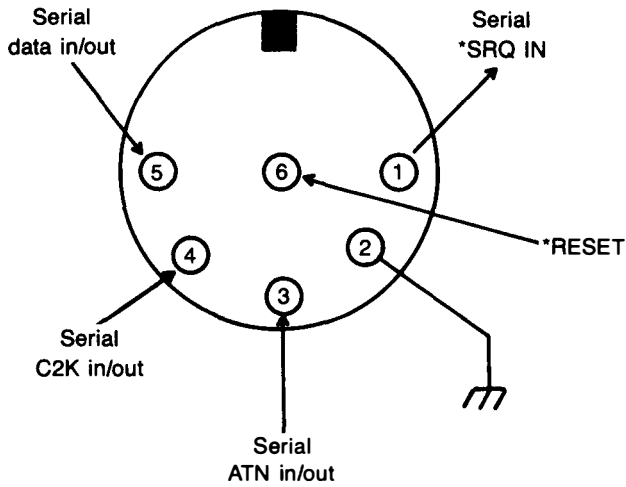


Fig. 19-4. The serial I/O port performs both the input and output serial transmission of data.

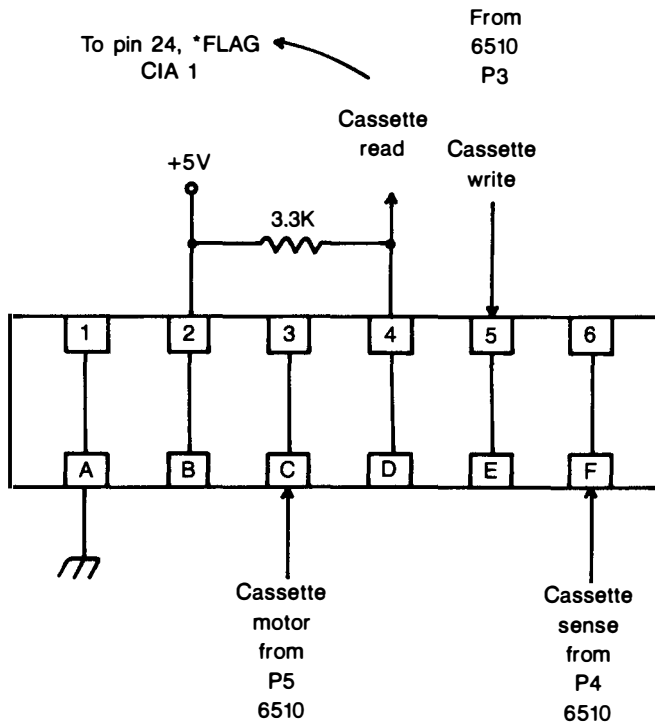


Fig. 19-5. The cassette I/O slot is used exclusively to receive and transmit data to the cassette recorder. It also carries the off-on signal to the cassette.

\*SRQ IN low and the line will know it needs service.

Once pin 1 goes low, the 64 acts. It brings pin 3, serial attention in/out, SERIAL ATN IN/OUT, low. If there are any devices on the line, they are alerted. The devices on the line can do three things. They can listen, talk, or control. The devices, items like a disk drive, graphic printer, or others, are all connected on the same line. They can all listen at the same time, but only one can talk at a time. The 64 controls the one it wants to talk.

When the 64 brings pin 3 low, all devices listen, and the one told to talk will respond. The data that the device sends to the 64 comes in serial fash-

ion, a bit at a time, over pin 5, serial data in/out. Pin 4 carries a clock signal to sync the timing of the external devices with the processor. Pin 4 is called serial clock in/out, SERIAL CLK IN/OUT.

### CASSETTE I/O SLOT

The apparent 12 pins on the cassette slot boil down to only six since the corresponding top and bottom pins are tied together. Refer to Fig. 19-5. That is, A and 1, B and 2, and so forth. When you test A, you are testing 1 simultaneously. A-1 is the ground connection and B-2 is the +5 volt supply.

C-3 is the motor switch, E-5 is the write output line, and F-6 is the cassette sense. These lines

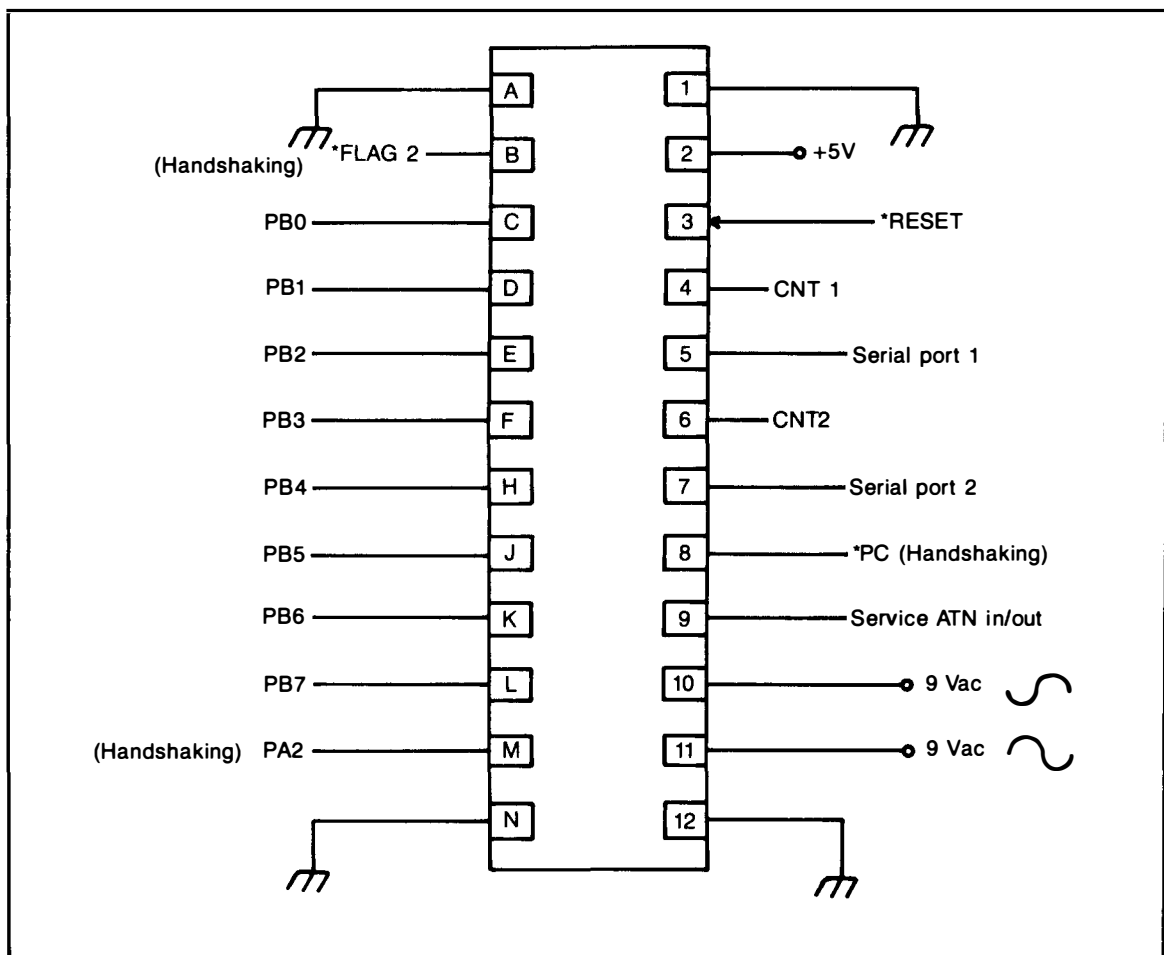


Fig. 19-6. The user I/O port interfaces CIA2 to external devices.

were all discussed in the 6510 chapter. They are coming from the 6510 pins 26, 25, and 24, which are P3, P4, and P5 respectively. They are connected to the special I/O register of the 6510. The setting of the register bits 3, 4, and 5 decide what action these lines are to take. The 6510 test point chart shows their normal standby states. P3 is low, P4 is high, and P5 is held high.

The only remaining pin is the cassette read, D-4. It is held high by a 3.3K pullup resistor. When the processor wants to read from the cassette, it forces D-4 low and the read will take place. D-4 connects to pin 24 of the keyboard CIA. Pin 24 is \*FLAG. When it is pulled low, it signals the chip that a data transaction is about to take place.

## USER CONNECTOR

The 24-pin user connector is the I/O connector for the second CIA. The one set of connections is numbered 1-12 and the other set is labeled A-N, leaving out G and I. Refer to Fig. 19-6. The numbered pins connect to all sorts of circuits. The lettered pins are used with the B port of the CIA. The connectors can interface the 64 with many dif-

ferent types of external devices. The connectors not only input and output signals, but they also supply power to some devices.

Pins 1, 12, A, and N are ground. Pin 2 is the +5 volt supply. Pin 3 is attached to the \*RESET circuit. \*RESET is held high. If you force it low by shorting it to ground, the processor will restart and reinitialize the entire machine. This is a good service test.

Pins 10 and 11 come from the 64's power supply and contain 9 Vac. Pin 9 is another connection from service attention in, at the serial bus. Pin 8 is a CIA \*PC output pin needed for the handshaking type of data transfer.

Pins 5 and 7 are the two serial port I/Os that the 64 possesses. Pin 5 is SP1 from the keyboard CIA, and Pins 7 is SP2 from the second CIA. The two serial ports on the one user connector can be very convenient for many applications.

The lettered pins are a complete I/O port for the B side of the second CIA. The eight bits of port B connect to pins C through L. \*FLAG on pin B and bit 2 from the A port are used for handshaking. With proper programming, the port can handle many different devices.

## 20. Power Supply

**T**HE POWER SUPPLY IN THE 64 HAS TWO SECTIONS. Some of the circuits are mounted right on the print board in the computer case. The rest of the supply is in the plastic casing that looks like a giant calculator ac adapter. The ac adapter, of course, plugs into the side of the 64.

The computer circuits need two very well regulated dc voltages to operate. They require sources of +5 volts and +12 volts. The two actual dc voltages should be within  $\pm 5\%$ . If you find an input voltage that is out of that range there could be trouble nearby.

The most demanding voltage is the +5 volts. The supply is able to produce a full amp of the +5 volts which is more than enough. The +12 volt requirements are small and are easily handled. There are six inputs to the computer from the supply: four dc and two ac. There are two +5 V types, one +9 V unregulated, A +12 V and two 9 Vac.

Power supply troubles are among the most common in the computer. Typically, the machine simply goes dead as the source voltages go bad. The fuse could blow, rectifiers short, regulators

open up, the transformer or another component could start smoking, and other things. Aside from the symptom of smoking, which provides instant identification of the defect, the best way to check out a defective supply voltage is step-by-step.

### TROUBLE LOCATION

When the supply voltage ceases, the first question to ask is, "which power section has failed; the ac adapter or the print board components?"

Plug the ac adapter into the 120 Vac house current but do not plug the adapter into the side of the 64. There is no off-on switch in the adapter so it comes on when plugged in. If the adapter is ok, there will be two voltages at the pins of the plug that goes into the computer. At the two pins on either side of the keyway, there should be about 9 or 10 volts ac. At the remaining two pins, there should be about + or - 5 volts dc. Refer to Fig. 20-1. It will be +5 volts if you get across the two pins one way or -5 volts if you measure the other way.

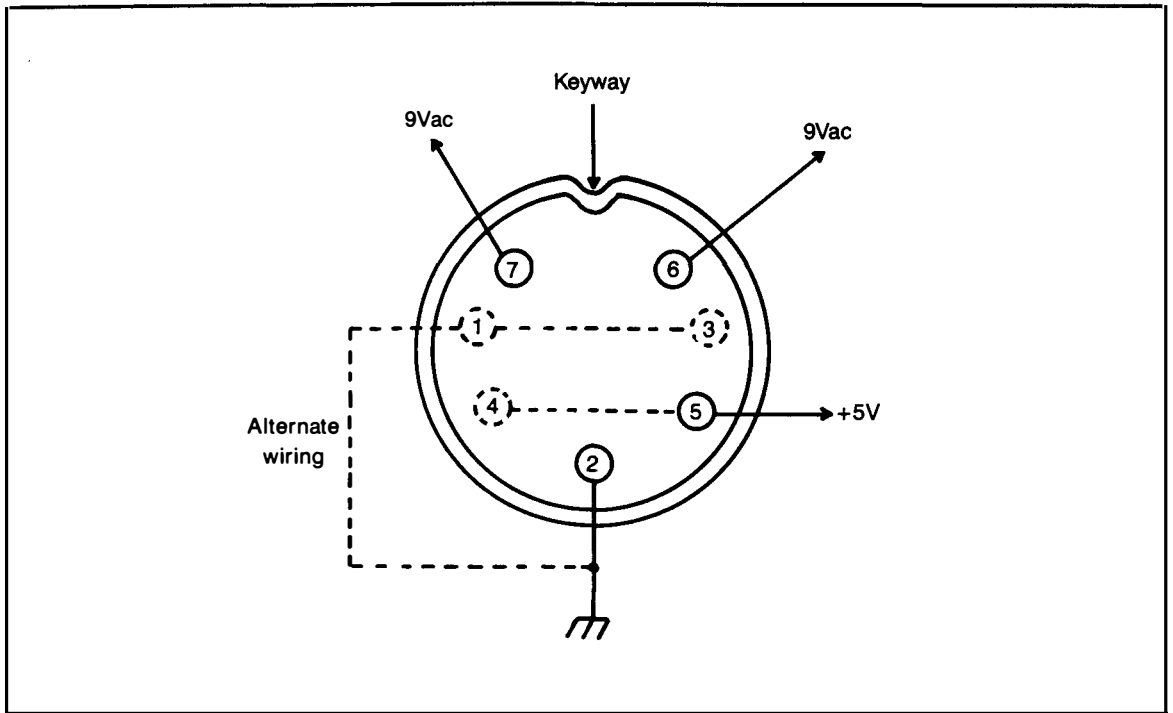


Fig. 20-1. The ac-dc plug exiting the ac adapter has test points for vom voltage tests. Pin 5 should have +5 volts when the unit is plugged in. Pins 6 and 7 each should read 9 Vac.

If either the ac or dc voltage is missing or way off, there is a defect in the ac adapter. Commodore recommends replacing the entire adapter unit. They can be purchased from Commodore for about \$40. It is the easiest and probably the most practical solution.

Inside the typical adapter casing are a power transformer, rectifiers, a 5 volt regulator, and some assorted components. They are shown in Fig. 20-2. The transformer receives the 120 Vac from the house receptacle into its primary windings. The core of the transformer is grounded to the third wire of the ac plug that goes into the wall socket.

The secondary of the transformer has two windings. The windings are centertapped to ground. One winding puts out 9 volts ac across its two sides. The other winding has its two sides attached to the anodes of a pair of diodes. The cathodes of the diodes are tied together, and their output is fed to the input of a 3052P regulator.

The input and output sides of the regulator are

bypassed to ground. The common of the regulator is connected to ground through a 37 ohm resistor. The wiring of the 3052P regulator is conventional and the output is rated at +5 volts and is able to supply one ampere.

When you test the ac adapter and find the +5 volts is missing and the 9 Vac is present, chances are good that the regulator has failed. If you are able to gain access to the regulator, however, you'll find that it is not easily available. It is easier to simply spend the \$40 and get a new reliable adapter.

The adapters are semisealed by Commodore. If you want to take one apart it is possible, although it is also possible to ruin it in the process. The first step in taking it apart is to make sure that it is not plugged in! It does contain dangerous voltages when it is plugged in.

With a thin screwdriver you could gingerly pry off the nameplate side of the case. Once you get the nameplate off you will be confronted with a thick layer of epoxy. You will have to break off the

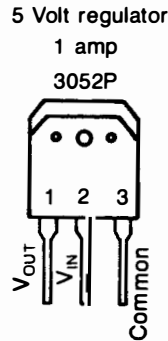
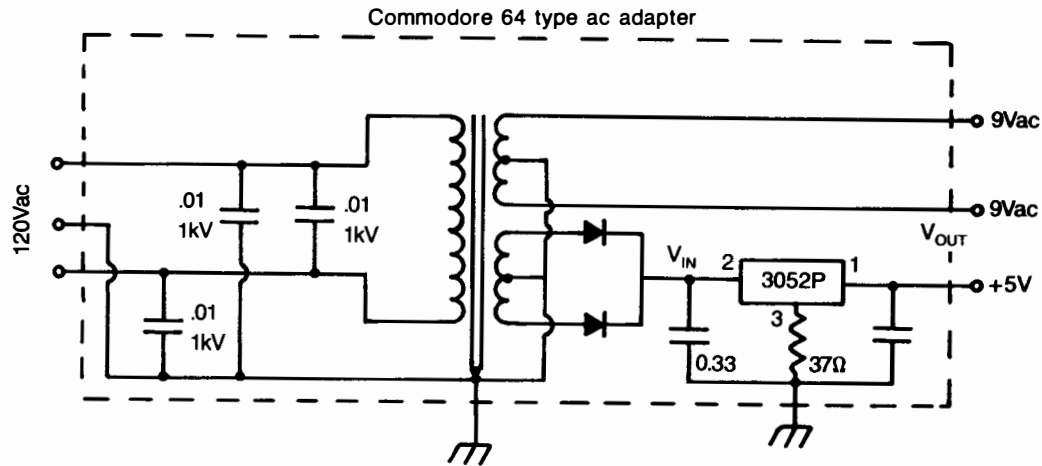


Fig. 20-2. The typical ac adapter that produces voltages for a computer like this one is based around a power transformer with two secondaries. The loss of +5 volts is often due to failure of the 3052P regulator in one of the secondary circuits.



epoxy layer, being careful since the epoxy contains imbedded wiring. The components will be buried beneath the epoxy.

Should you be able to safely repair the wiring faults, retrace your steps till you have the unit back together. Again let me mention that the adapter unit was designed as a throwaway and it might not be practical or reliable to repair it unless you are an experienced electronic wireman.

When the 9 Vac and the +5 volt dc are present, the ac adapter is considered good and the rest of the supply inside the computer is to be checked next.

## FUSE CONSIDERATIONS

Once the adapter is deemed ok but the computer appears dead, the next move is to look at the fuse, a 1 amp, 250 V, 3AG type. When the fuse is open and it is replaced, the computer will probably start operating. If it does not and the fuse blows again, there is a short circuit in the input circuit. This circuit is shown in Master Schematic 9.

The fuse is in the input line that connects 9 Vac to the bridge rectifier network. A short in any of the components in that line could possibly blow the fuse. Test the bridge rectifier, the 7805 voltage regulator, and the four bypass capacitors in the input and output of the regulator.

When the fuse is blowing continually and a checkout of the 7805 regulator line bears no fruit, there is an outside possibility that the tap from the fuse connection to the user port might have a short to ground. Follow the copper trace from one end to the other.

Another reason for the 64 to be dead is a faulty off-on switch. The switch has two sections. One section connects the 9 Vac to the board and the other section attaches the +5 volts to the print board. If either section of the switch opens permanently, replace the switch.

## SOURCE CHECKING

When the computer is dead and the fuse and off-on switch are intact, the voltmeter is needed. Touchdown at the bridge rectifier inputs. There

should be 9 Vac there. If the ac voltage is missing, work back across the components to the input where the 9 Vac is entering the board. The components are some bypass capacitors and the input coil. An open coil, defective capacitor, or board short could kill the correct voltage.

There are a number of different supply configurations in various 64 models. As time goes by, there could be more circuit changes. Basically though, they all use similar principles. Typically, the bridge diodes receive 9 Vac at junctions 4 and 6 of the four diode junctions. The junctions are opposite to each other. Refer to Master Schematic 9.

Junction 3 is grounded. The fourth junction, 1, outputs about +12 volts dc. Test for this voltage. If it is missing the bridge circuit could be bad, or one of the two capacitors in the regulator input line could be defective. The +12 volt output is not able to deliver much current, but it is handy for circuits that need the voltage potential but do not draw much current. The circuit ends up as three supply lines that originate out of the bridge diode unit. Refer to Master Schematic 10.

One line receives the diode output over top of two capacitors into a 7805 regulator device. As the number implies, it is a 5 volt regulator. A quick test of the line is made at the output of the regulator. There should be +5 volts there. If it is not and the voltages up to that point were ok, chances are good that the regulator has quit. If the regulator turns out to be good, the next suspects are the two capacitors in the output line: a 10  $\mu$ F at 25 WV and a 0.1  $\mu$ F. Should they all check out fine, then there might be a board short or a defect in one of the circuits that the line is energizing.

When this supply line checks out ok, look for the +12 volt line. It originates out of the bridge too. It has a series filter of 470  $\mu$ F at 50 WV, a series diode, and a pair of input bypass capacitors. The resulting voltage, about +21 volts is put into the input of a 7812, 12 volt regulator.

The voltage can be tested at the input and output of the regulator. The input should be about +21 volts dc and the output about +12 volts dc. It is needed in circuits like the audio output to power the PN2222 transistor. If either voltage is incorrect

or missing, any of the components in the line could be bad with the regulator being the prime suspect.

A third line is tapped off in this same area after the series filter. A series diode is the only component in this line. The output is an unregulated +9 volt dc. This voltage is used in the cassette motor control circuit to power the collectors of three discrete transistors.

### **THE MAIN LINE**

Most of the other circuits are energized by a well regulated +5 volts. The regulator and its circuit components are confined in the sealed ac adapter. The regulated +5 volts enters the 64 through the

power input plug. The dc input line is composed of a series coil, L5, with two 0.22  $\mu$ F bypass capacitors on either side of the coil, one side of the off-on switch, and two capacitors performing some filtering. The line connects to practically every circuit and there are numerous bypass capacitors all over the board connected to the line. Refer to Master Schematic 9.

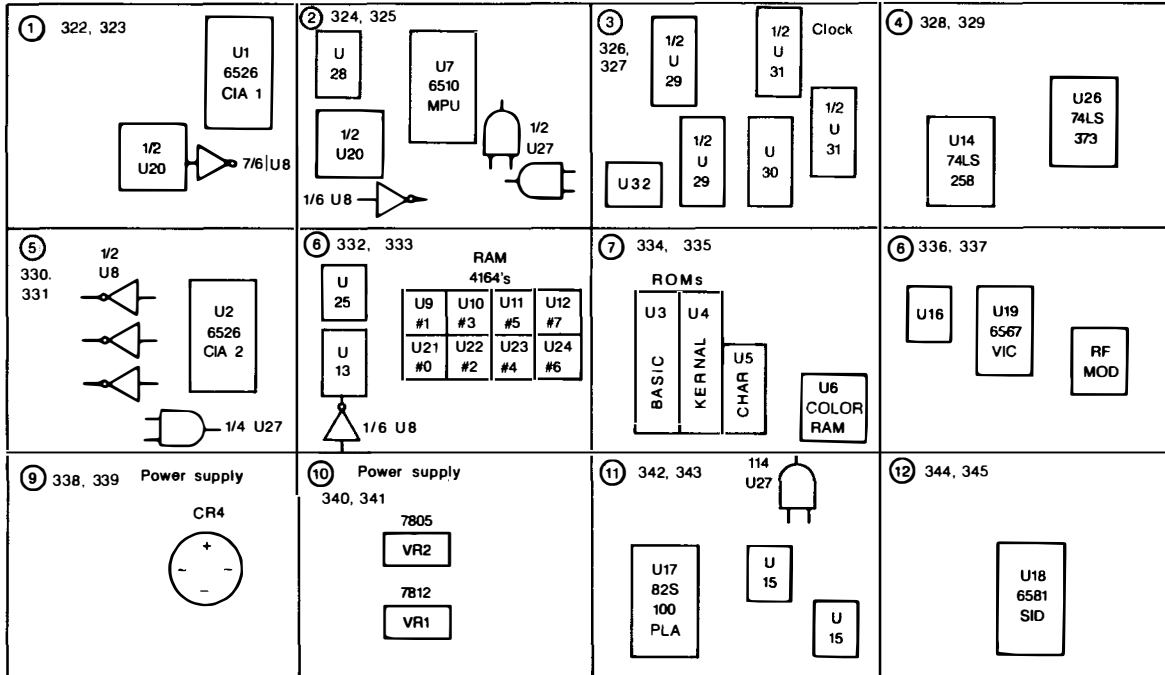
The voltage can be tested from the input pin to any destination. If you start at the input pin and the voltage is there, check out the line by crossing over component by component. If the voltage suddenly disappears, you probably have just passed over the defect.



# **Appendix**

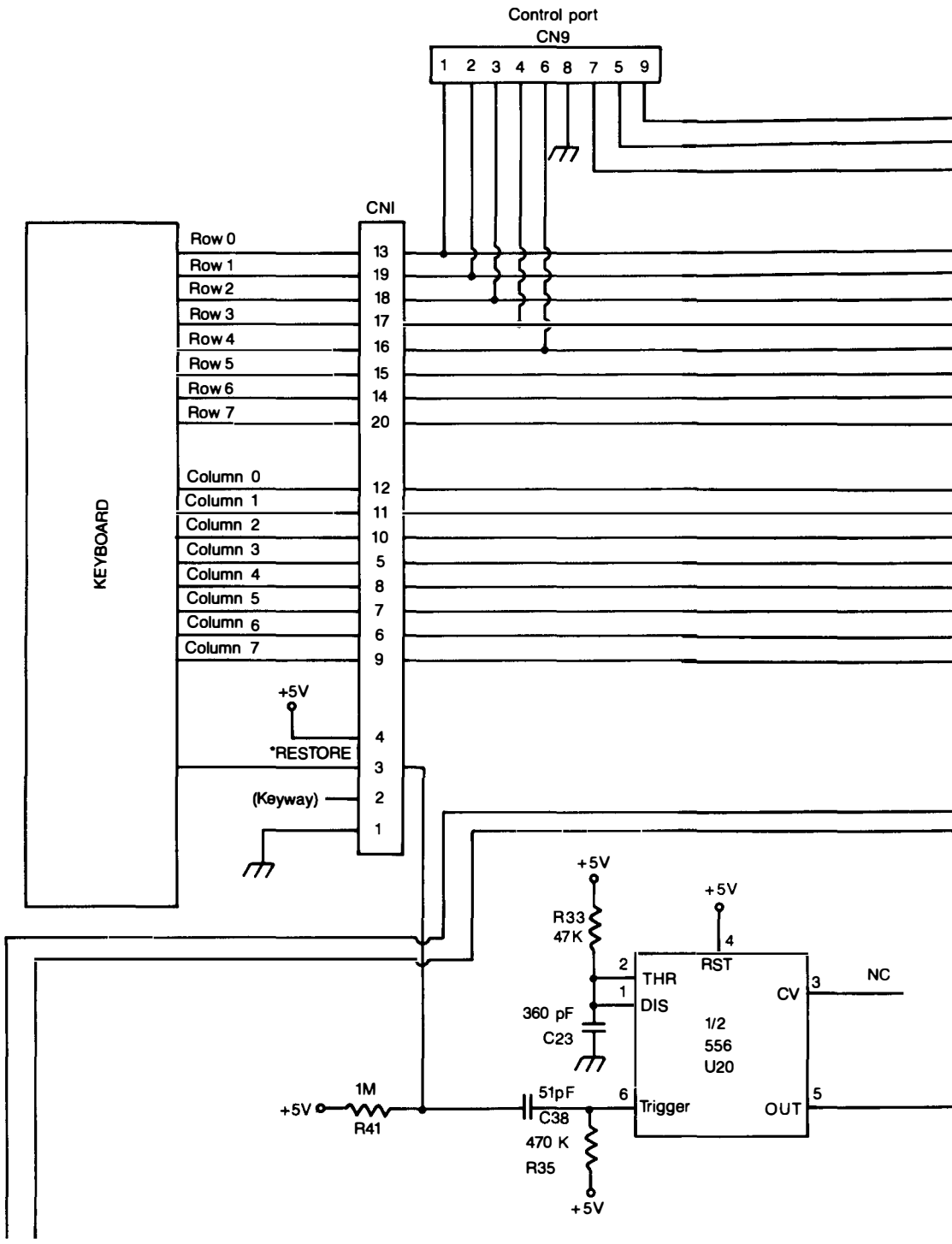
## **Master Schematic**

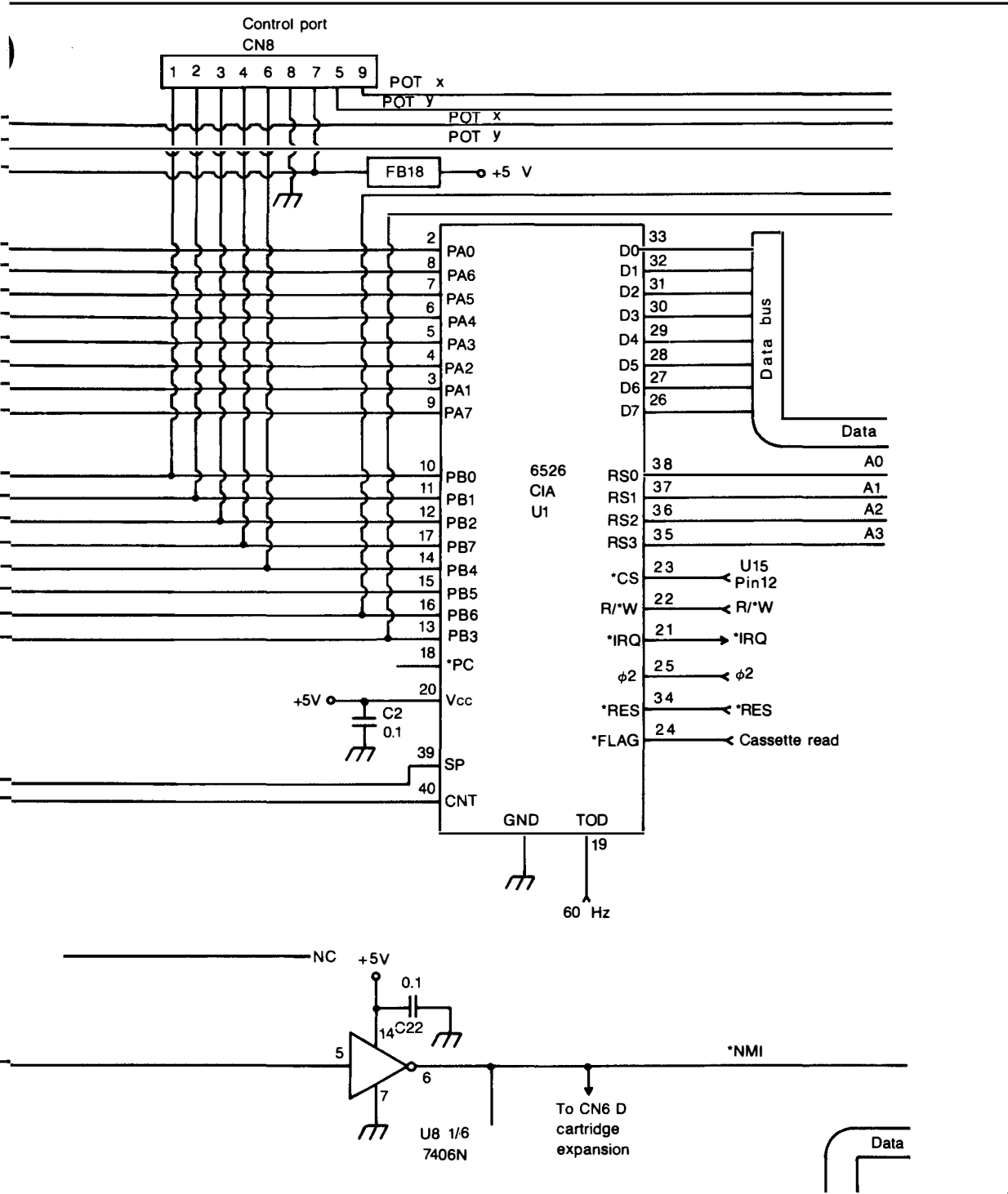




Chip number	Guide sheet number
U1	1
U2	5
U3	7
U4	7
U5	7
U6	7
U7	2
U8	1, 2, 5, 6
U9	6
U10	6
U11	6
U12	6
U13	6
U14	4
U15	11
U16	8
U17	11
U18	12
U19	8
U20	2
U21	6
U22	6
U23	6
U24	6
U25	6
U26	4
U27	2,5,11
U28	2
U29	3
U30	3
U31	3
U32	3

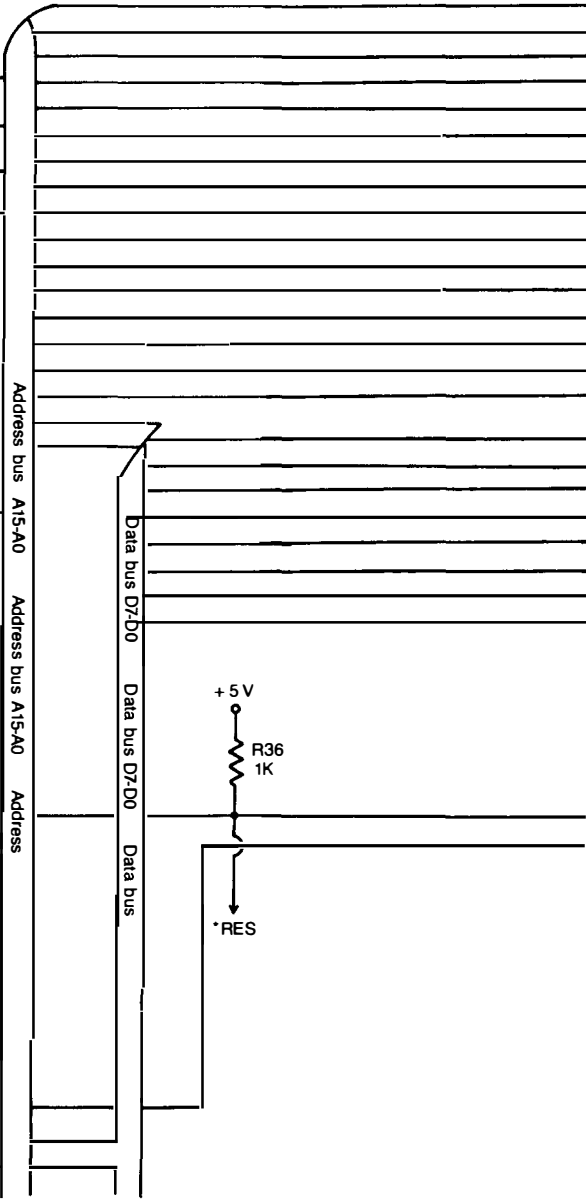
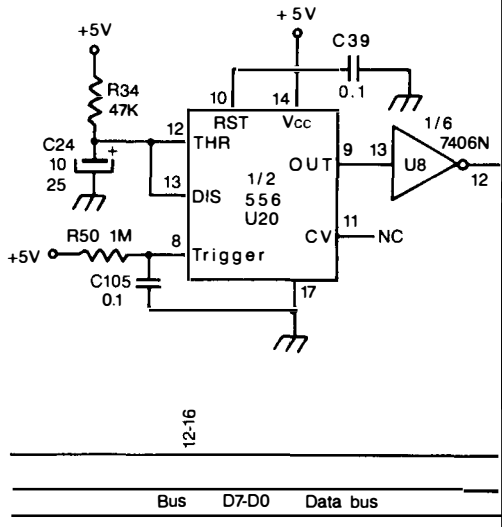
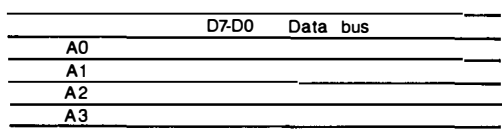
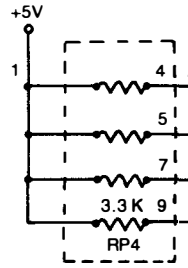
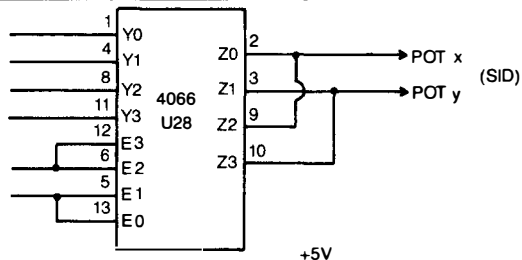
1

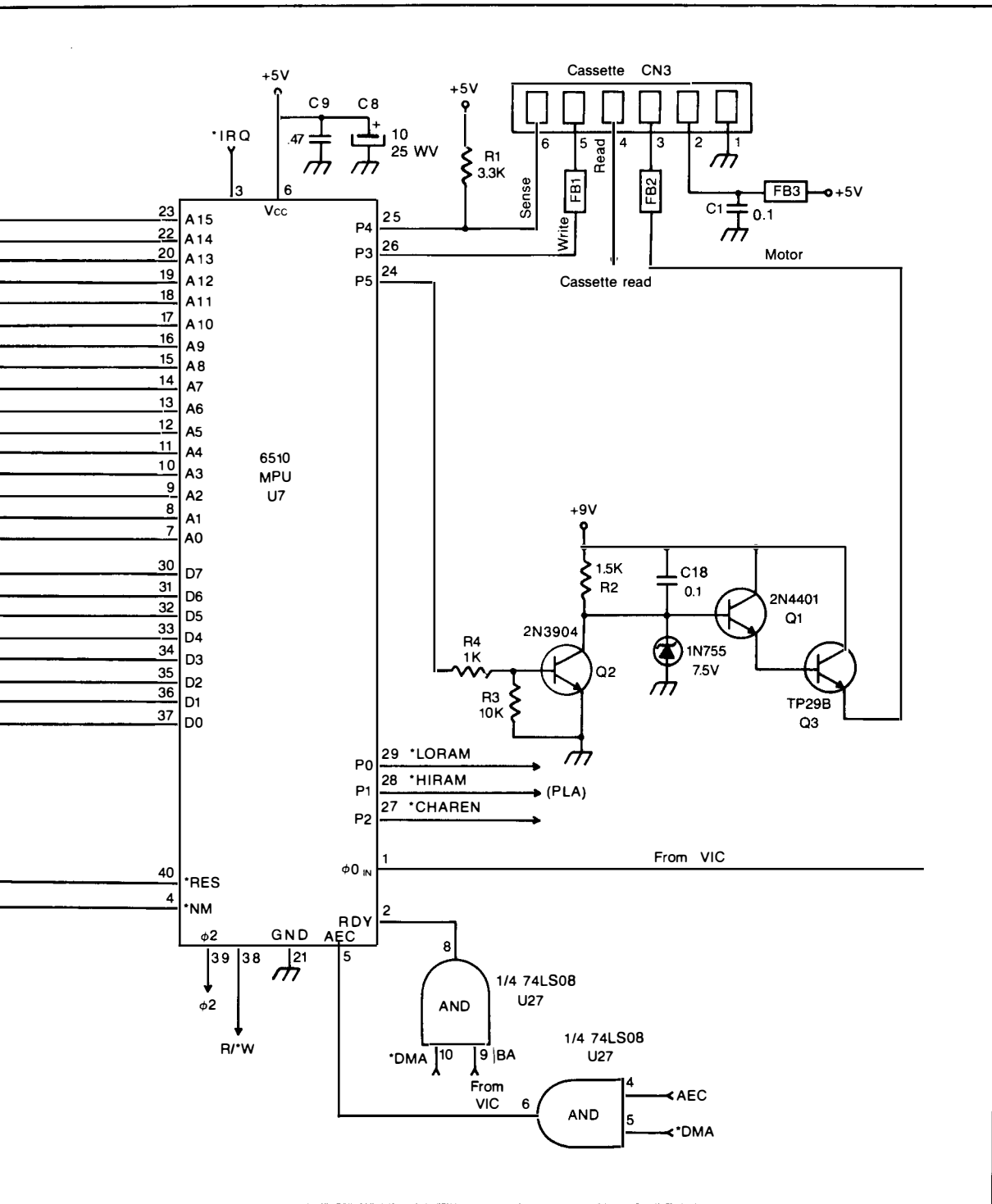




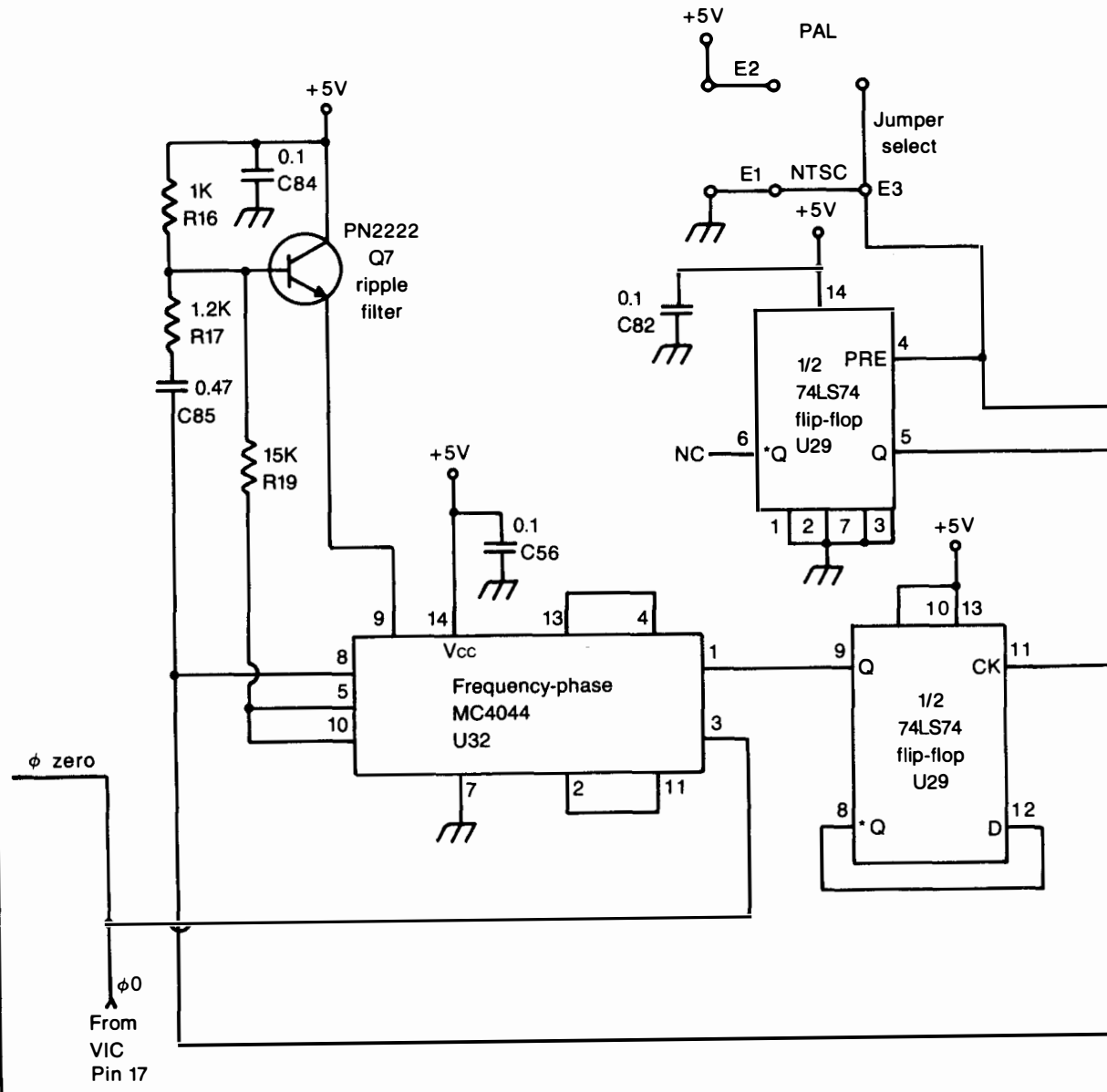


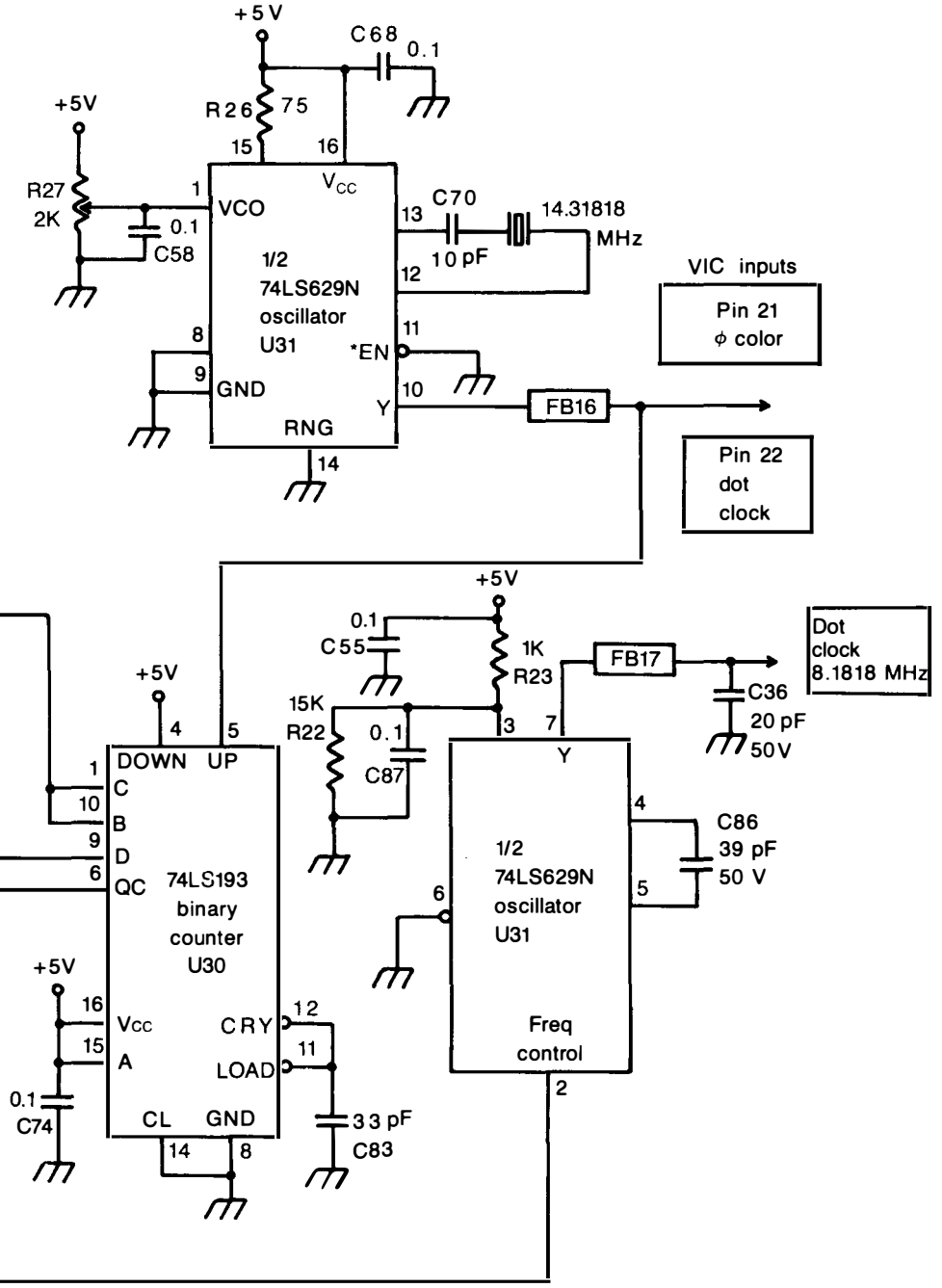
2

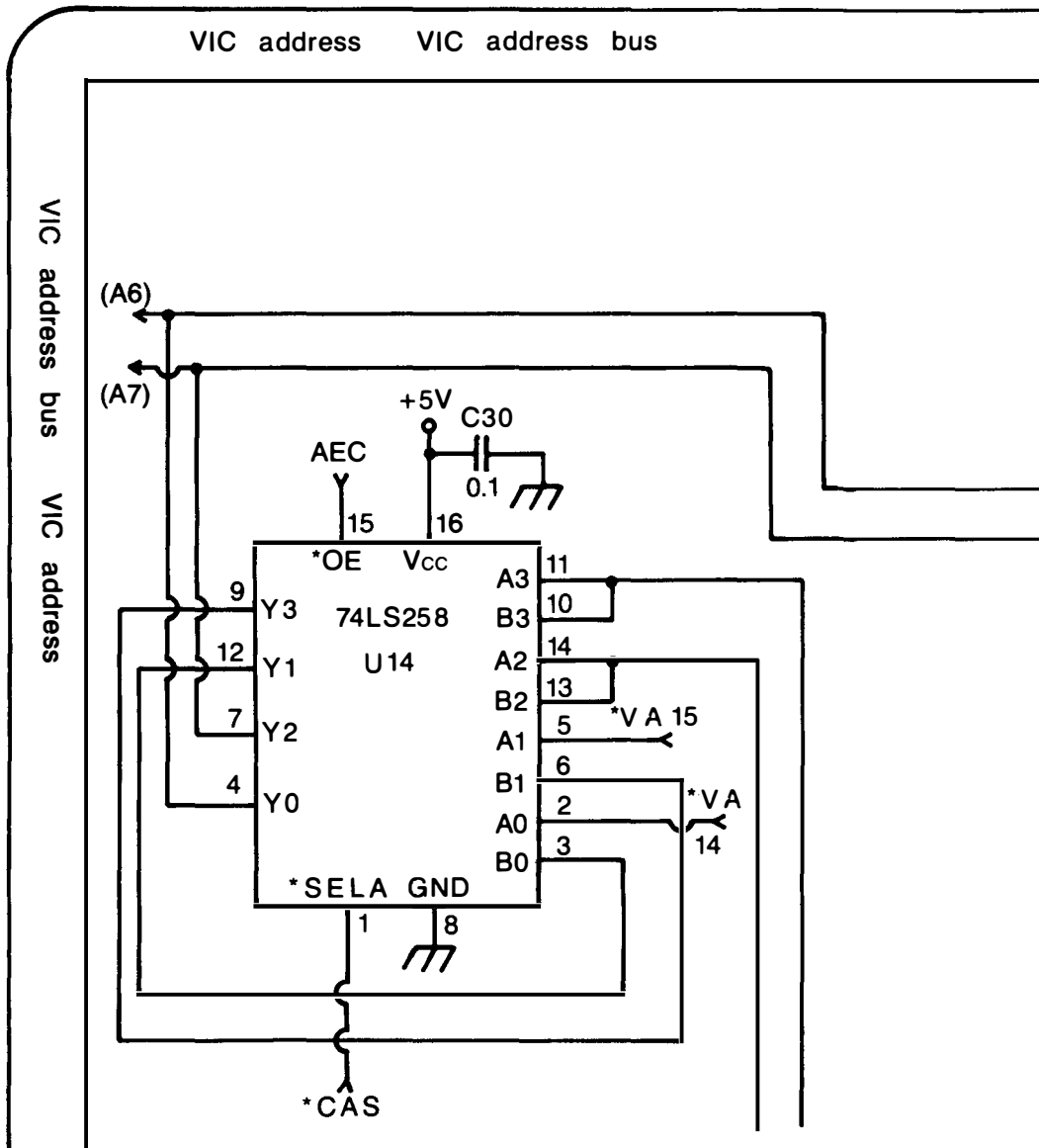


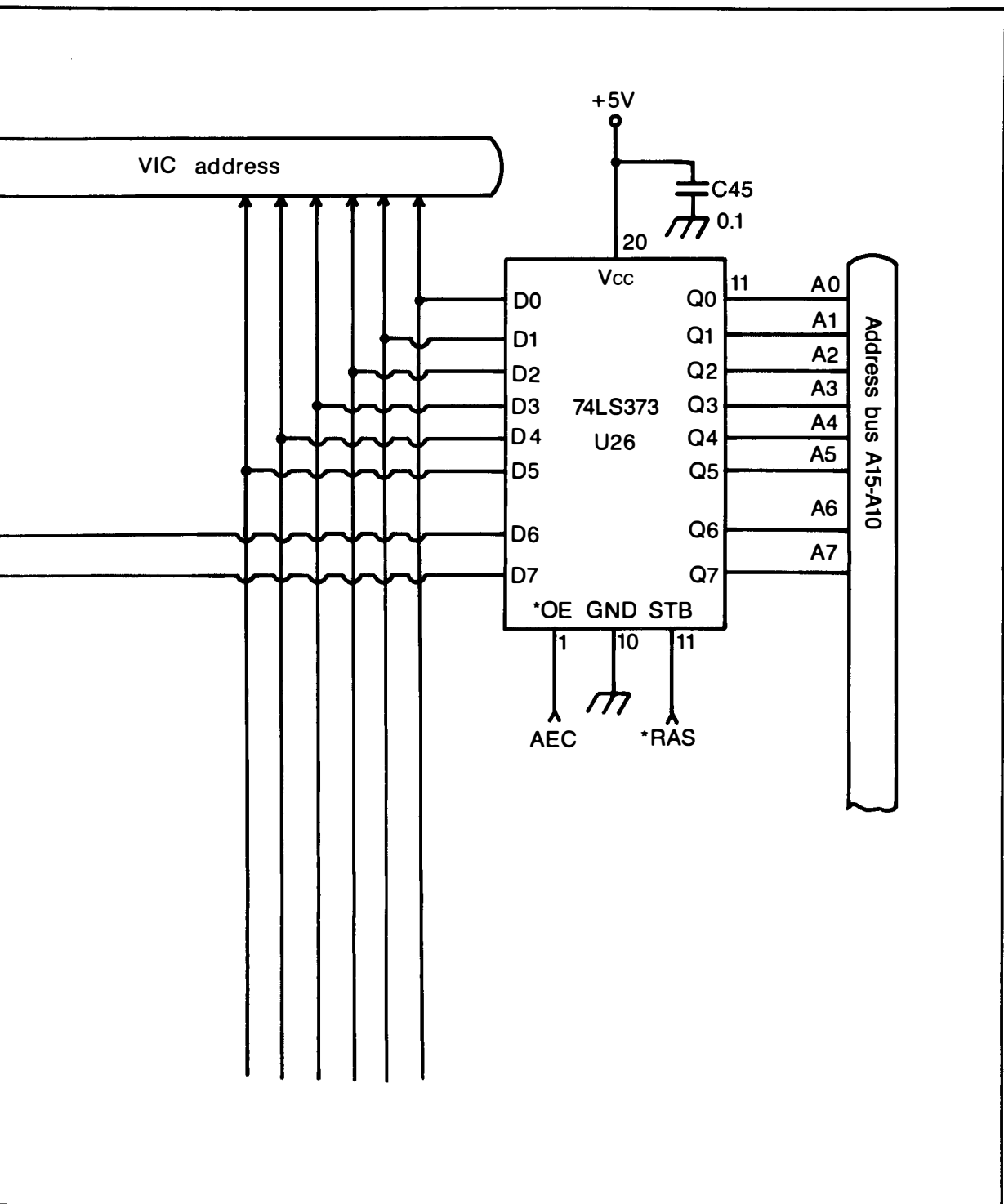


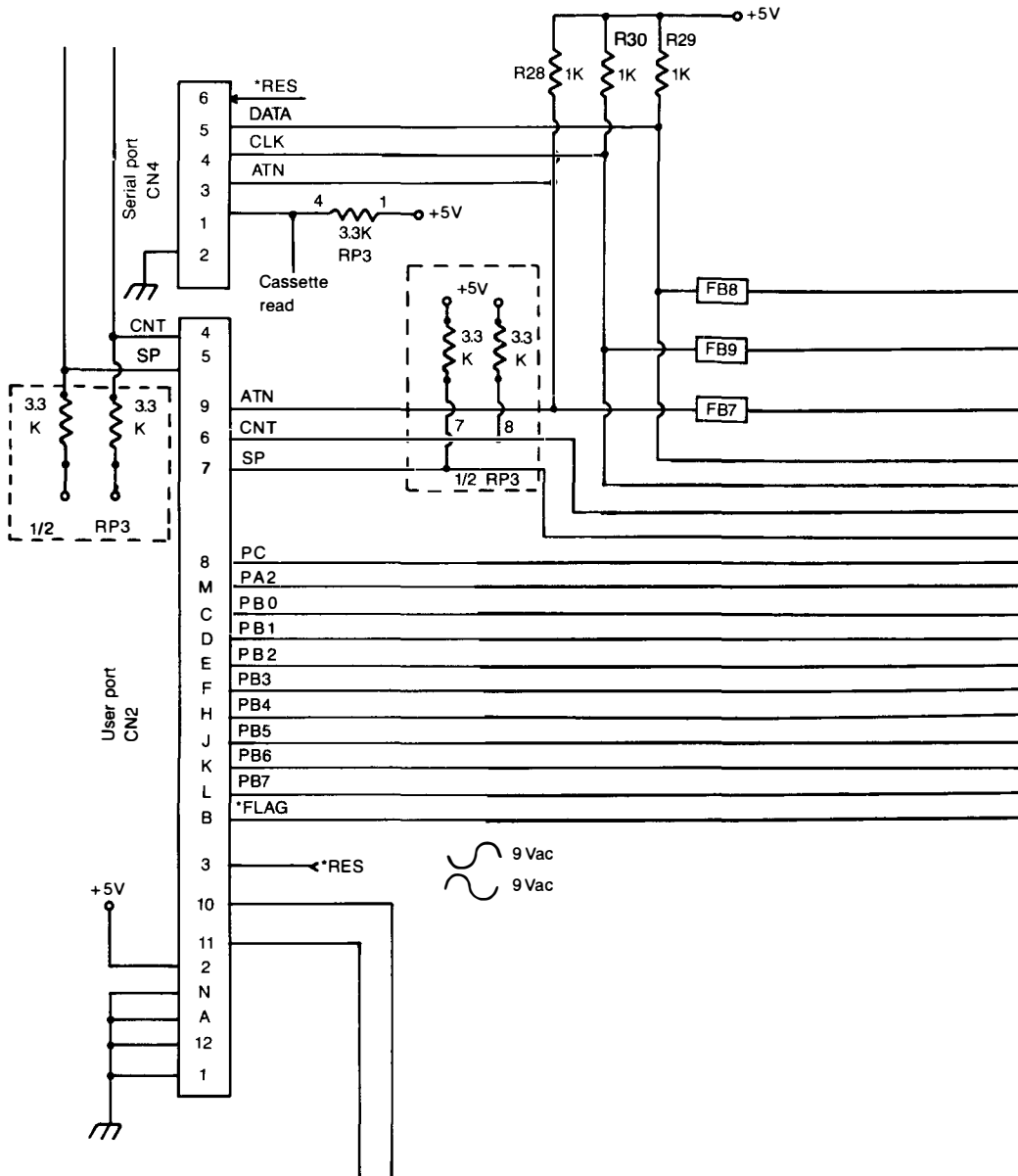
Clock

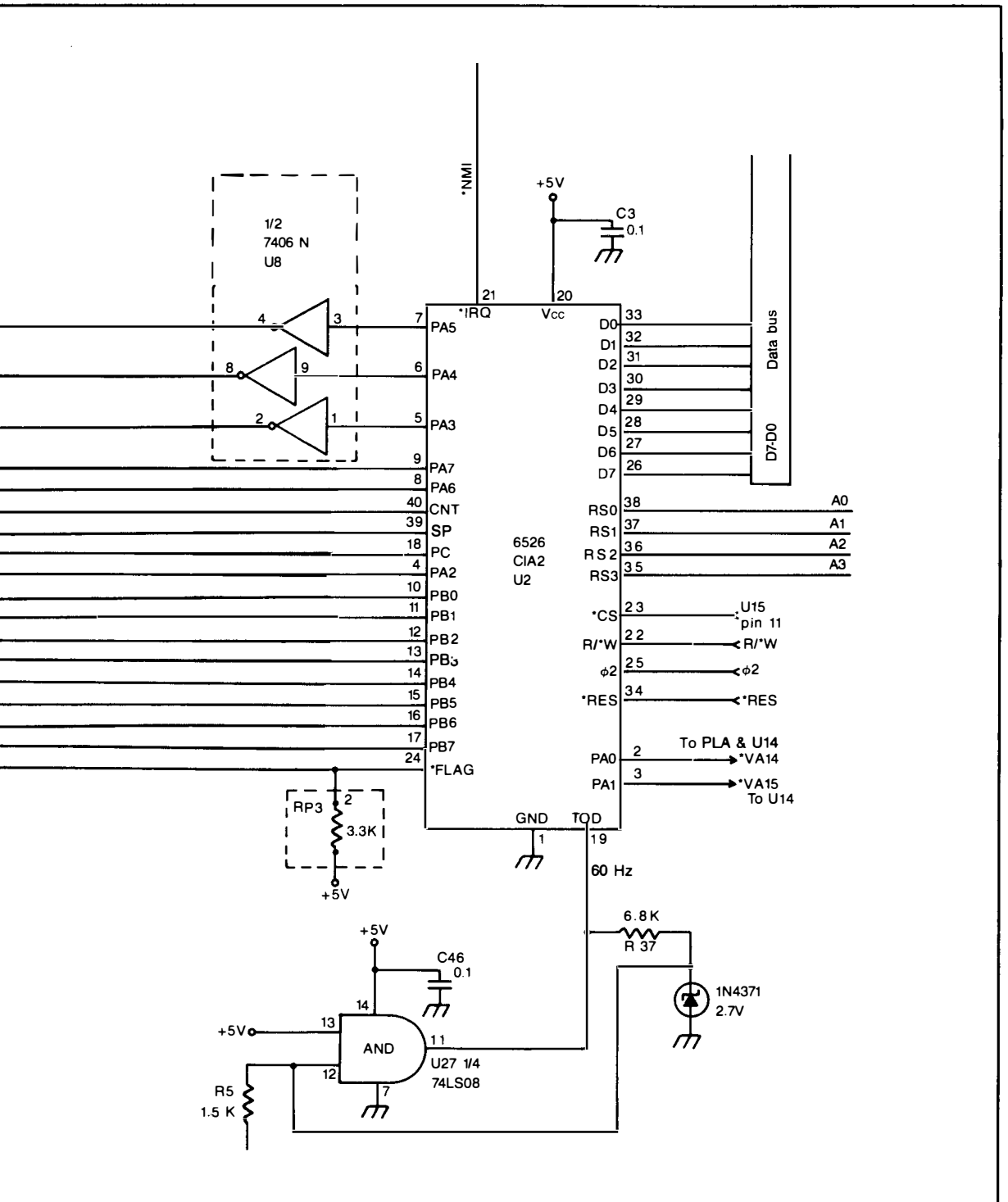




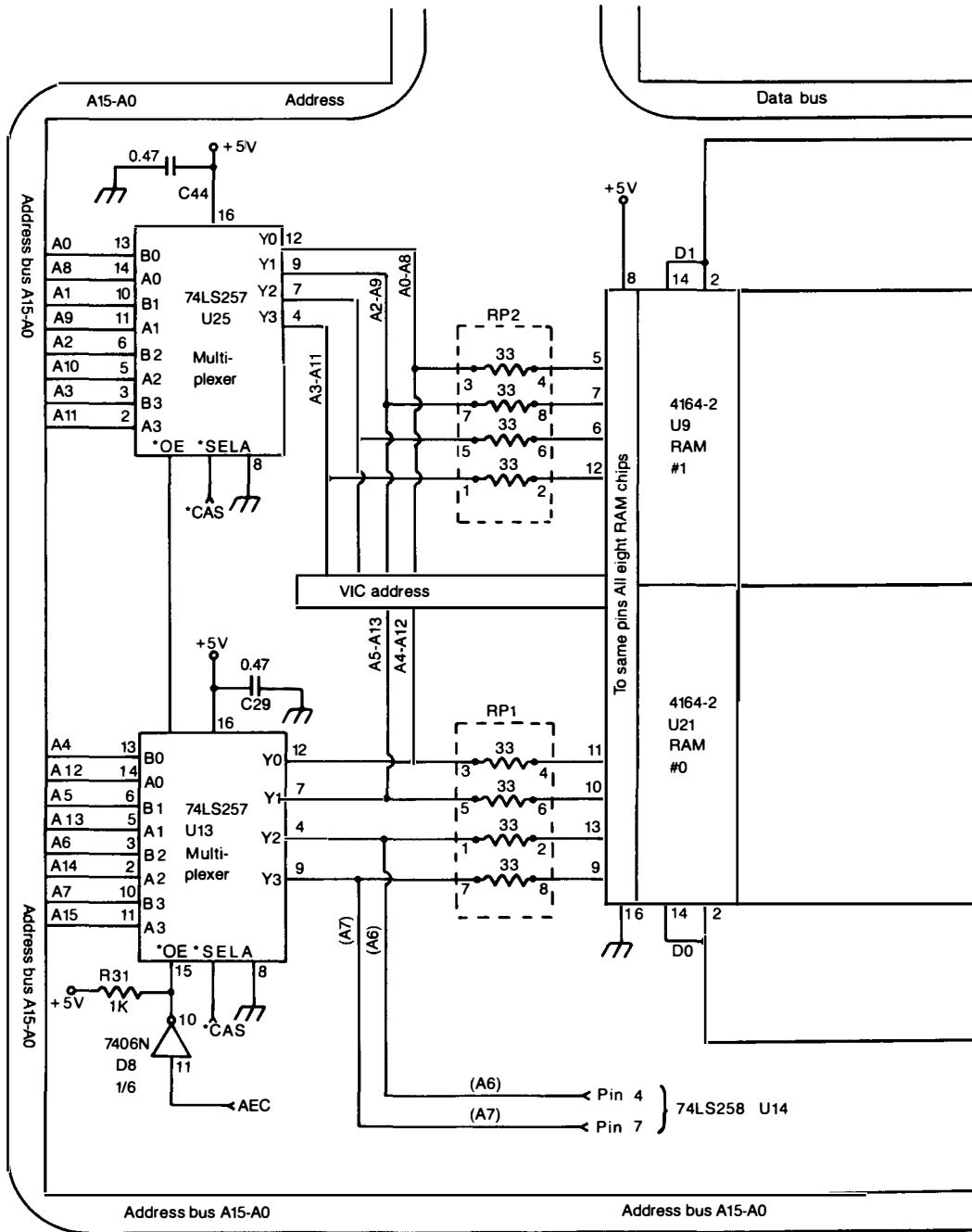


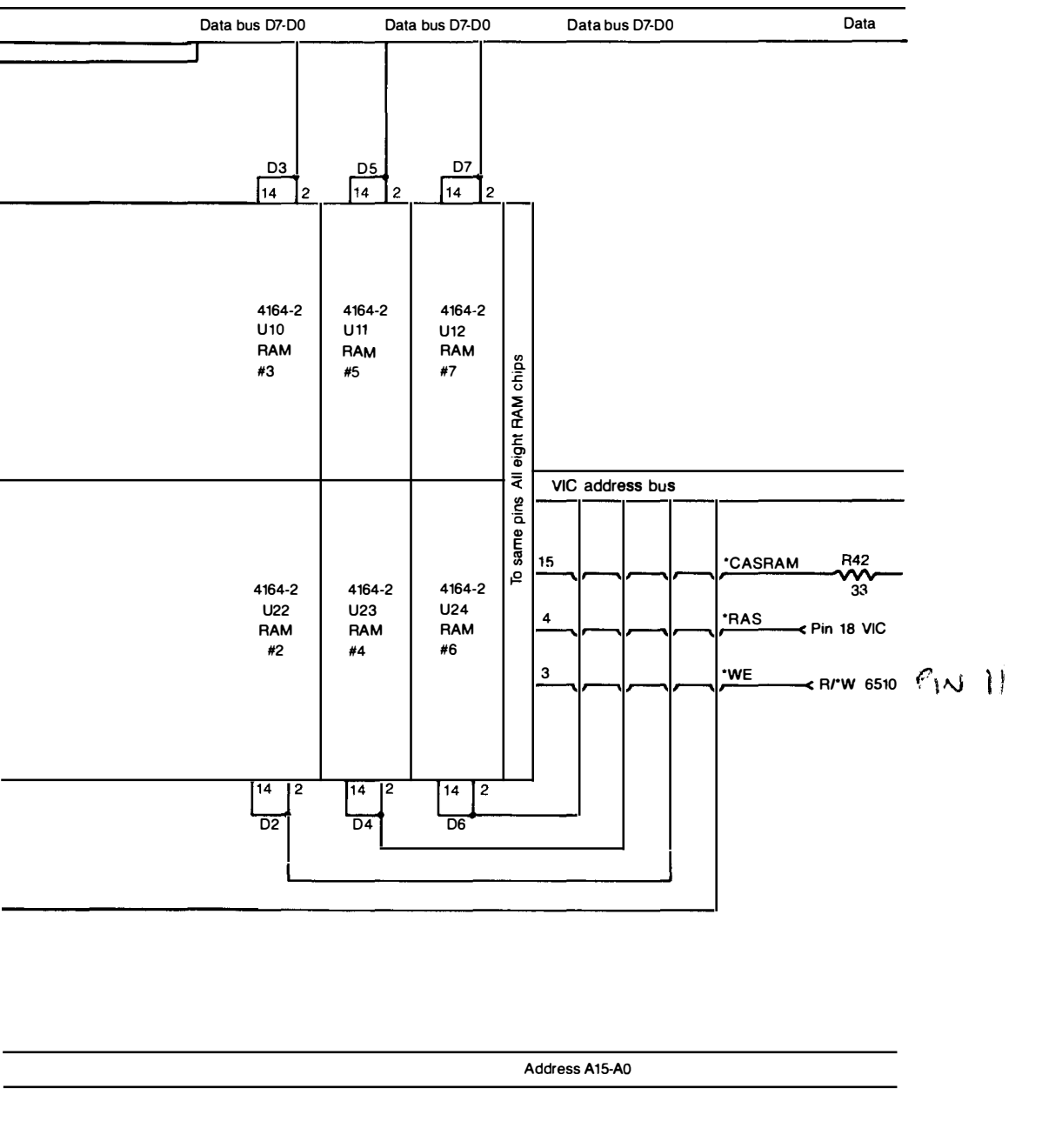


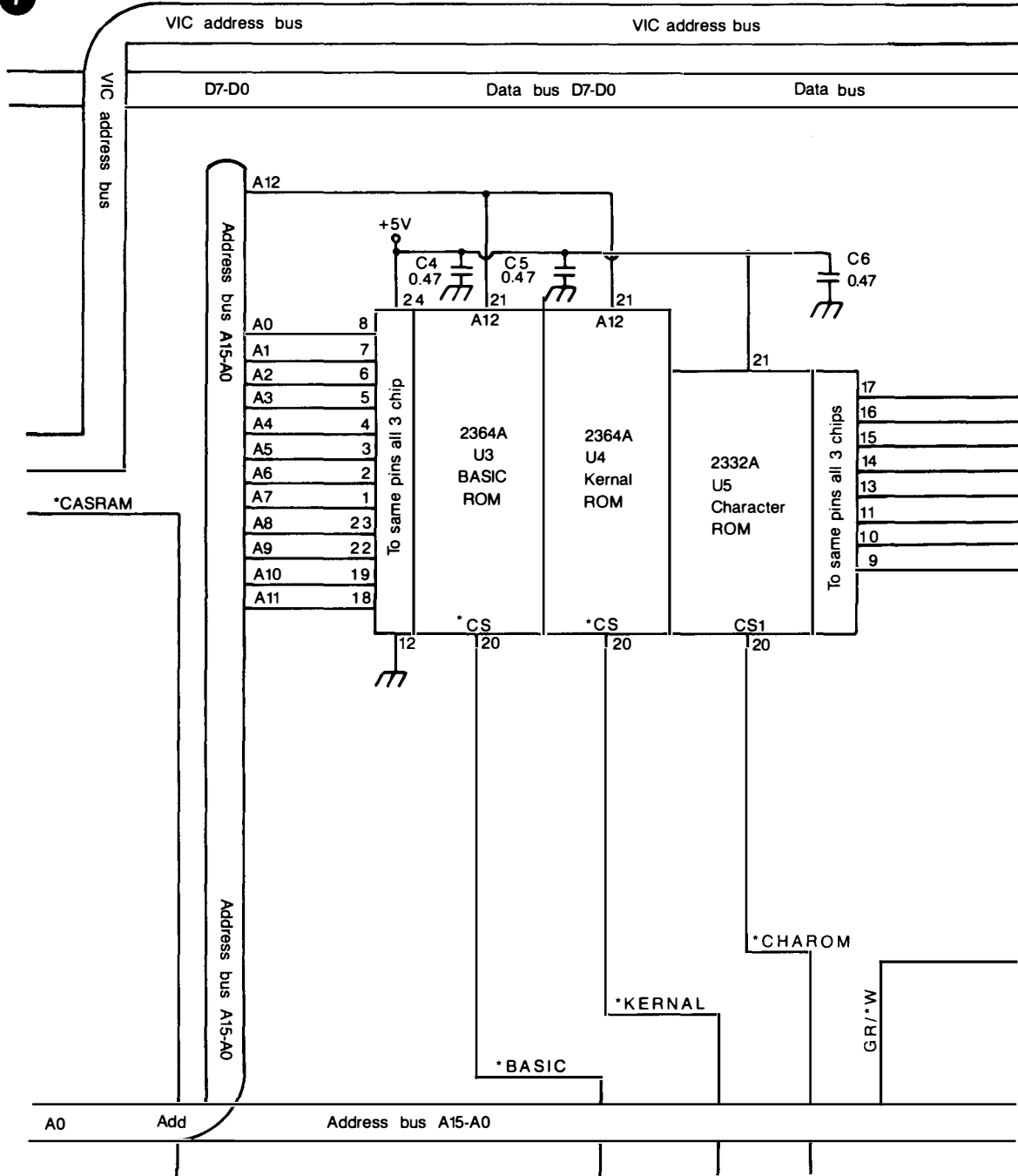


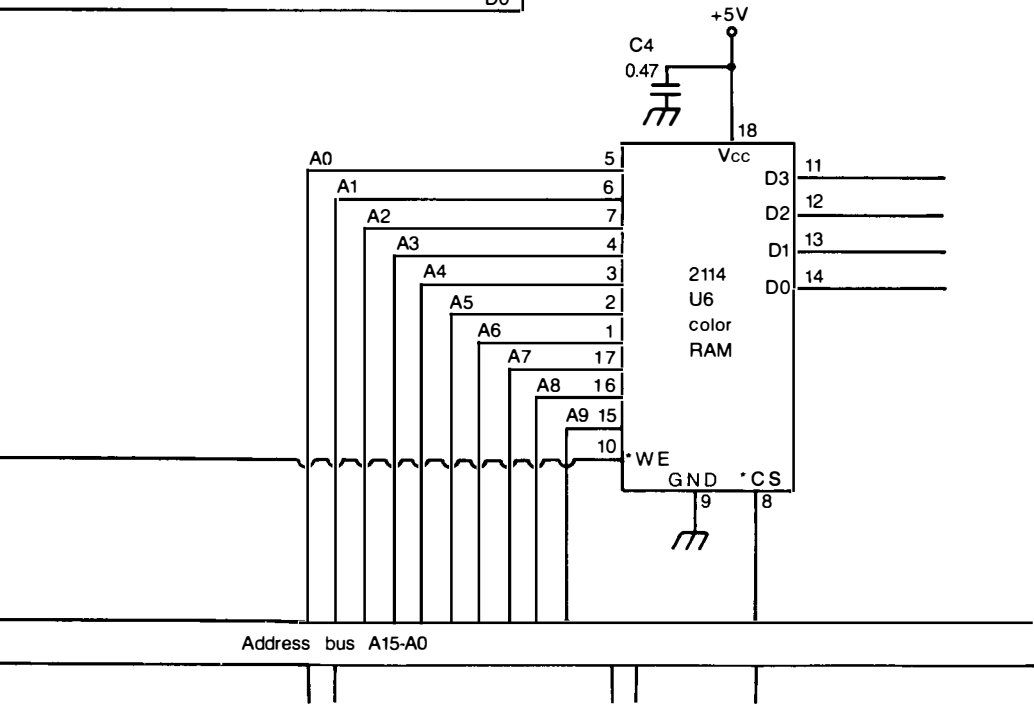
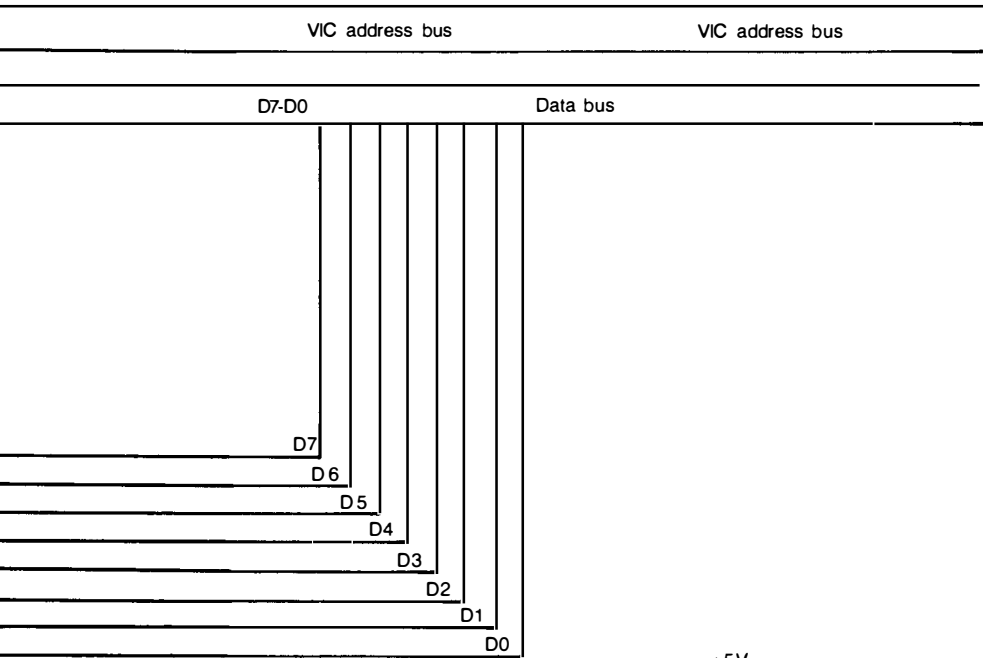




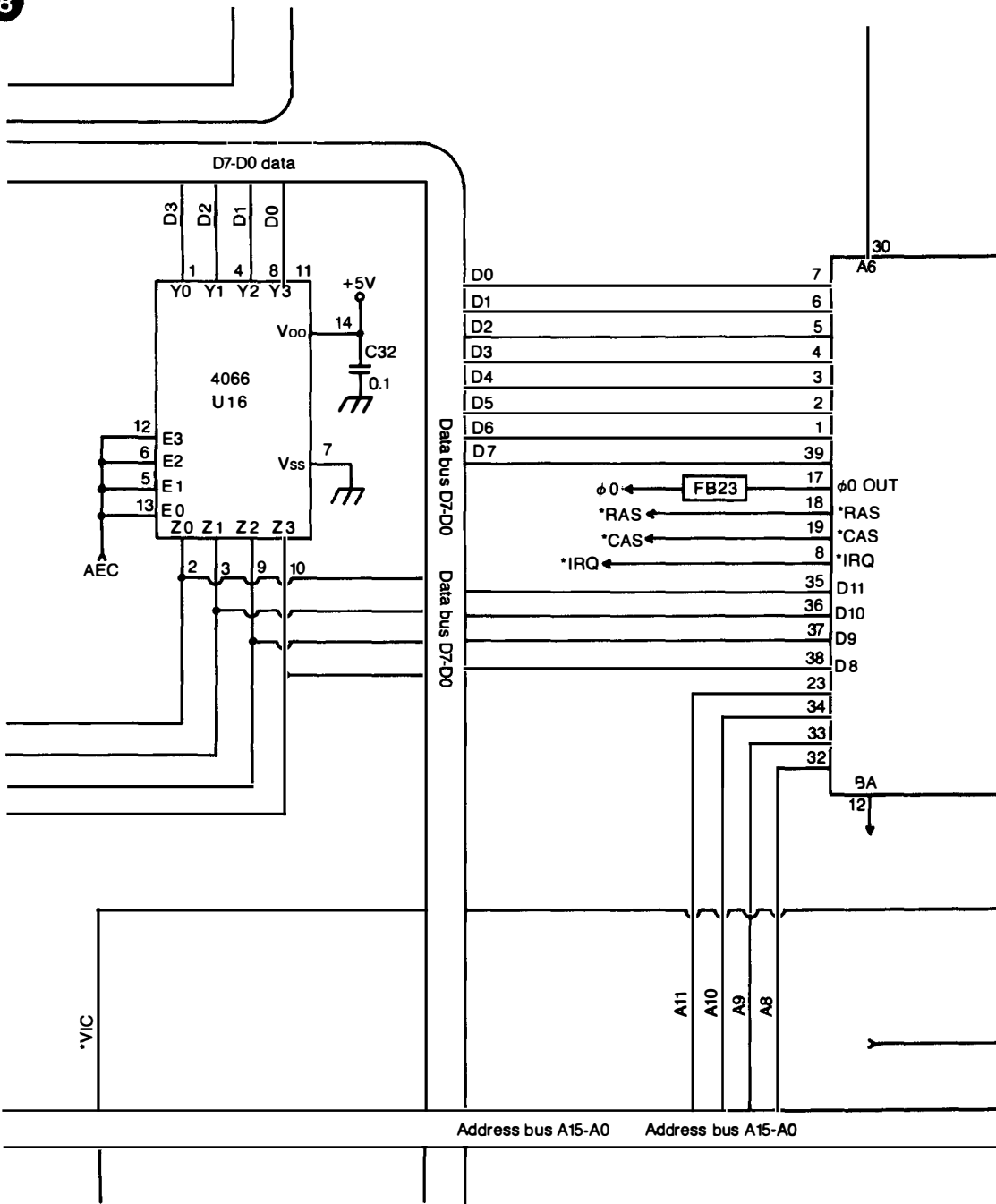


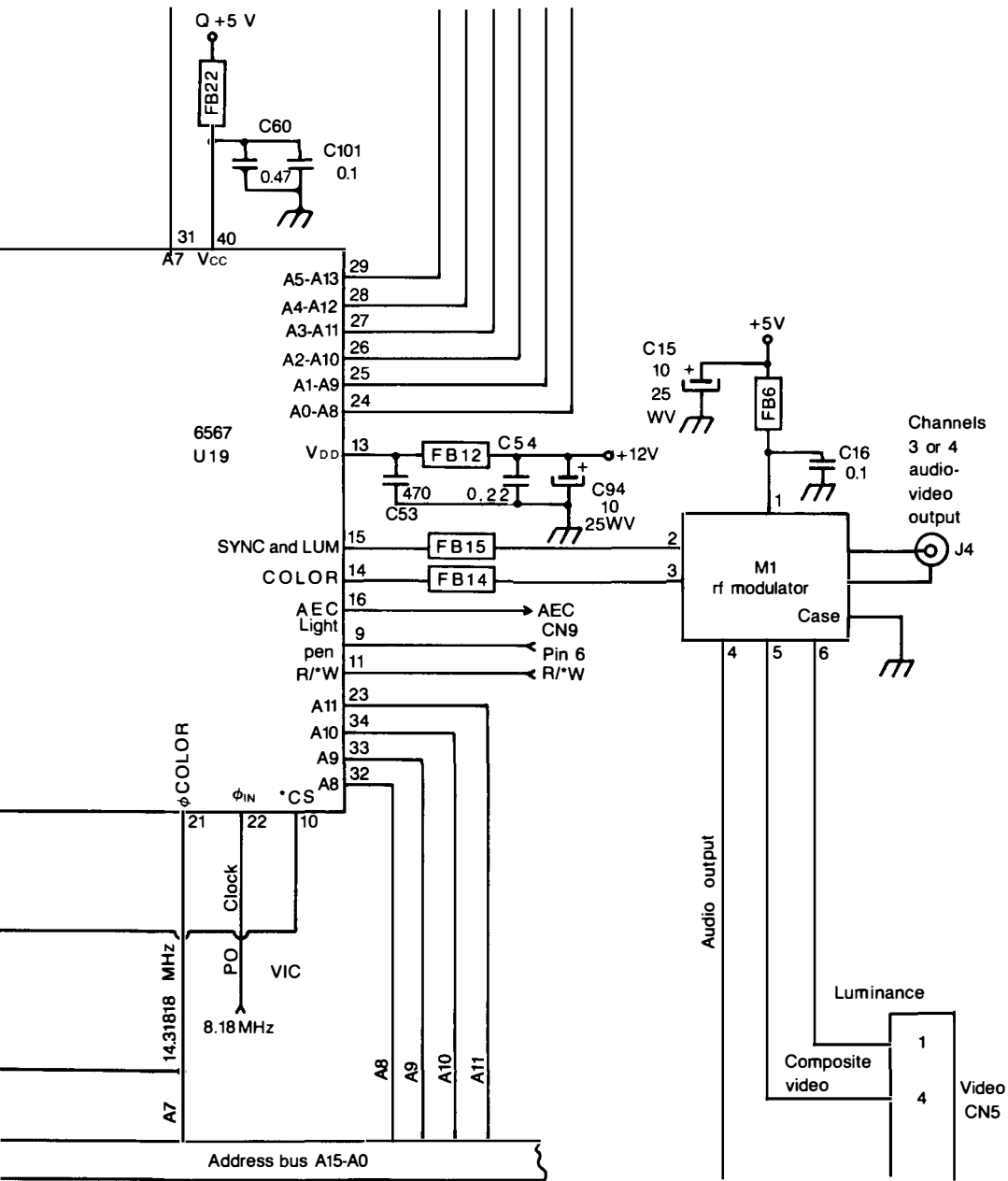




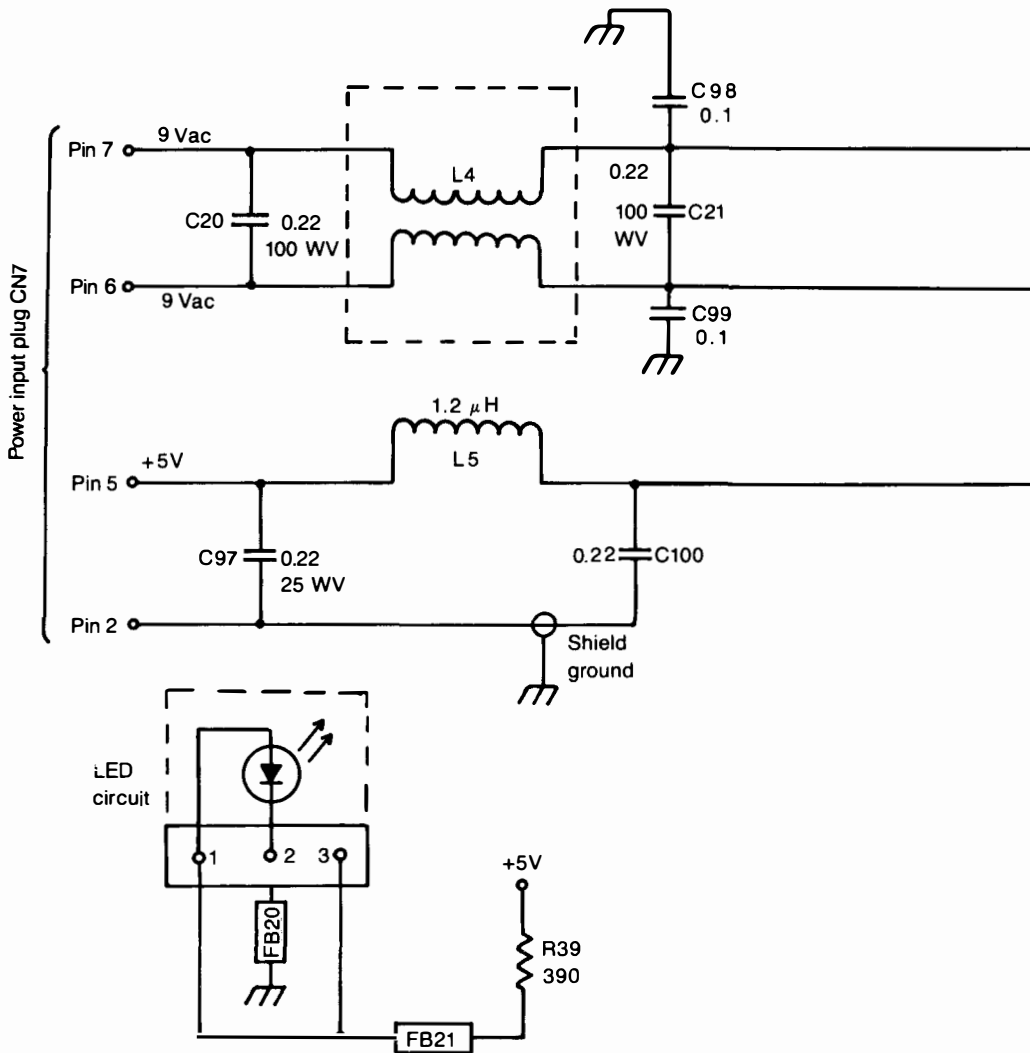


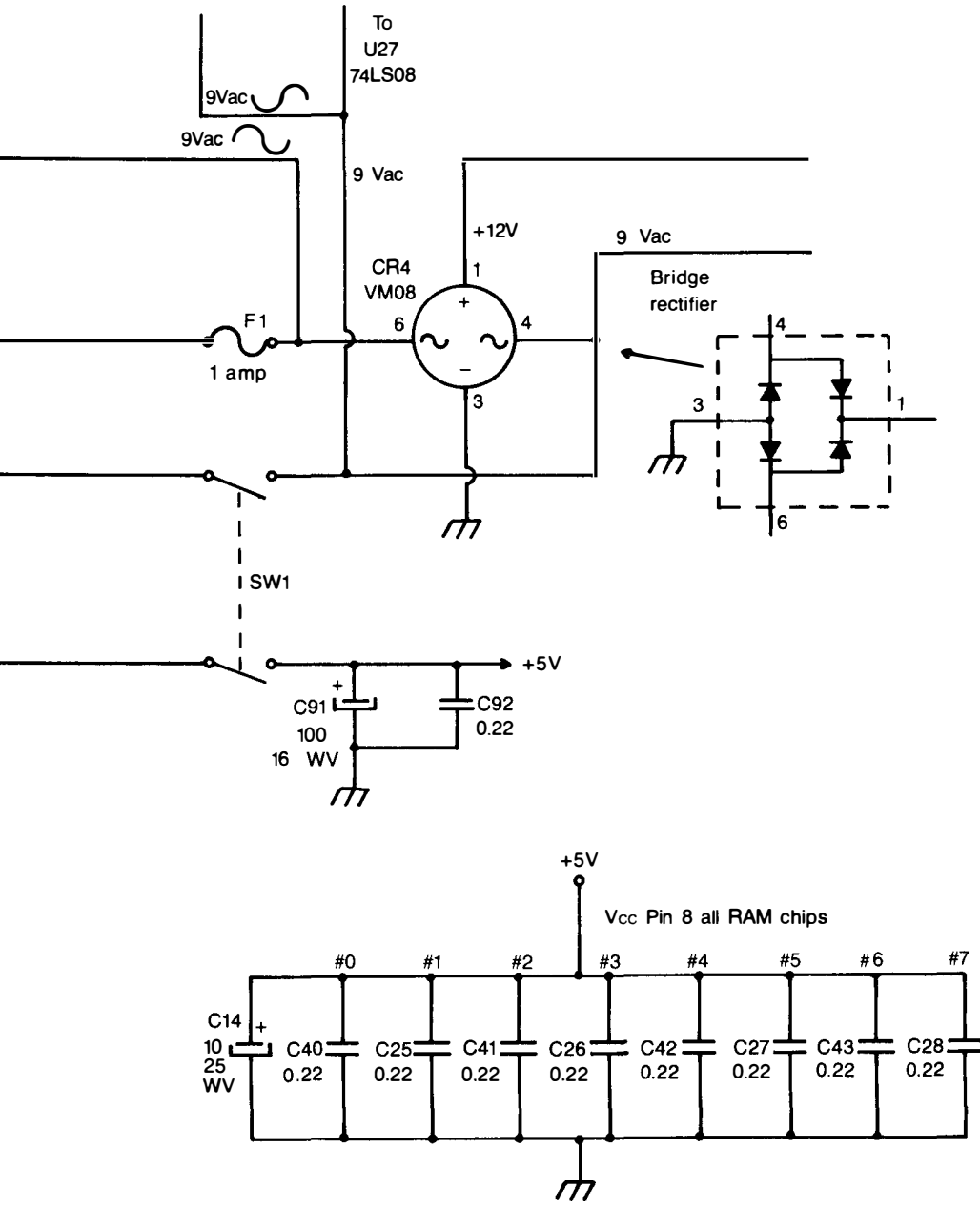
8





# Power supply

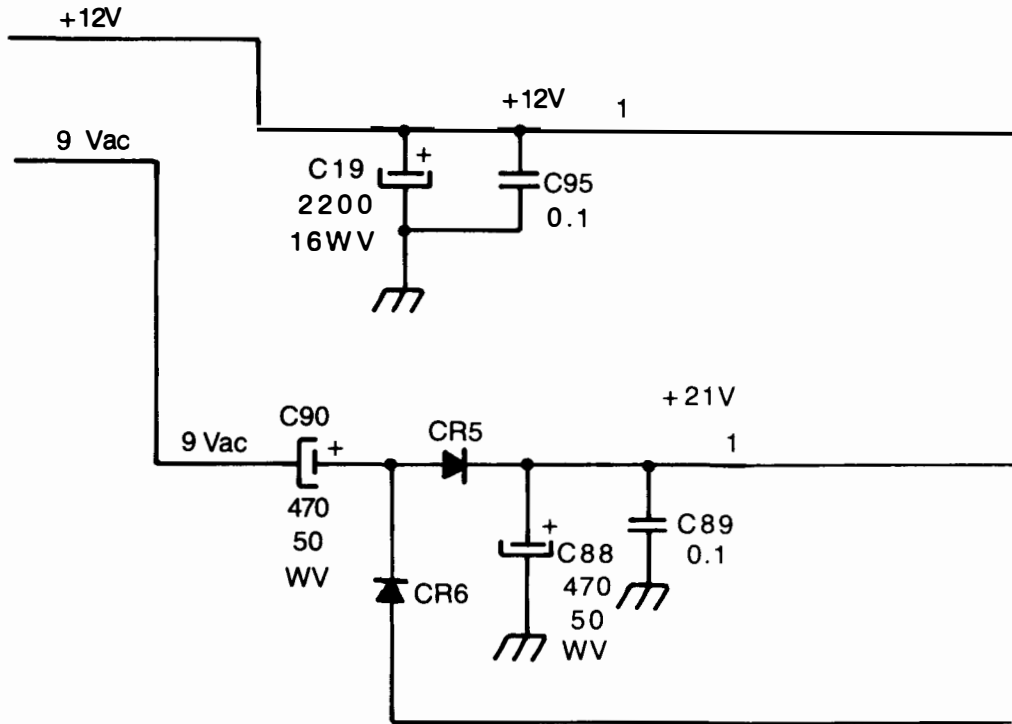


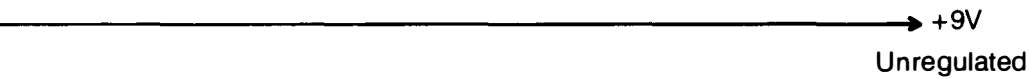
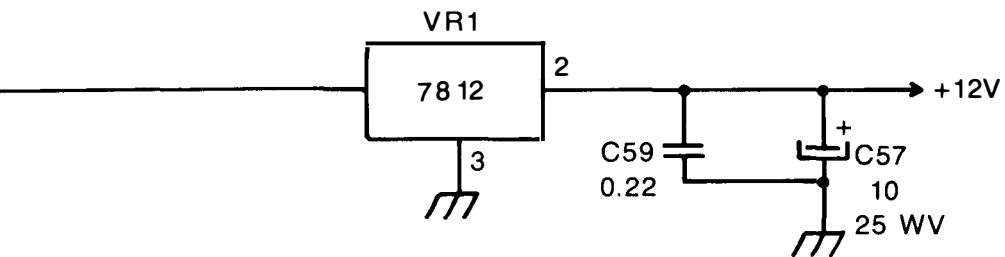
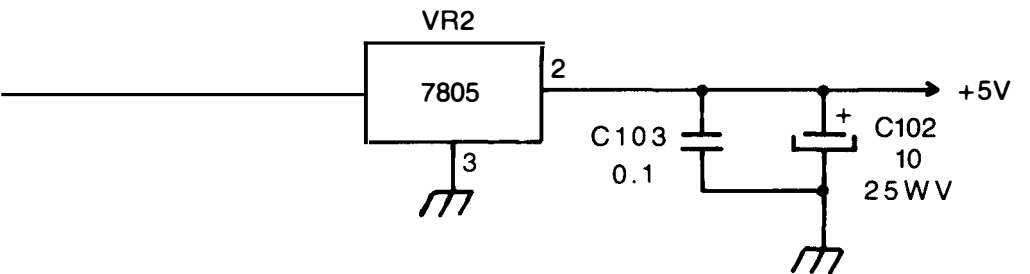


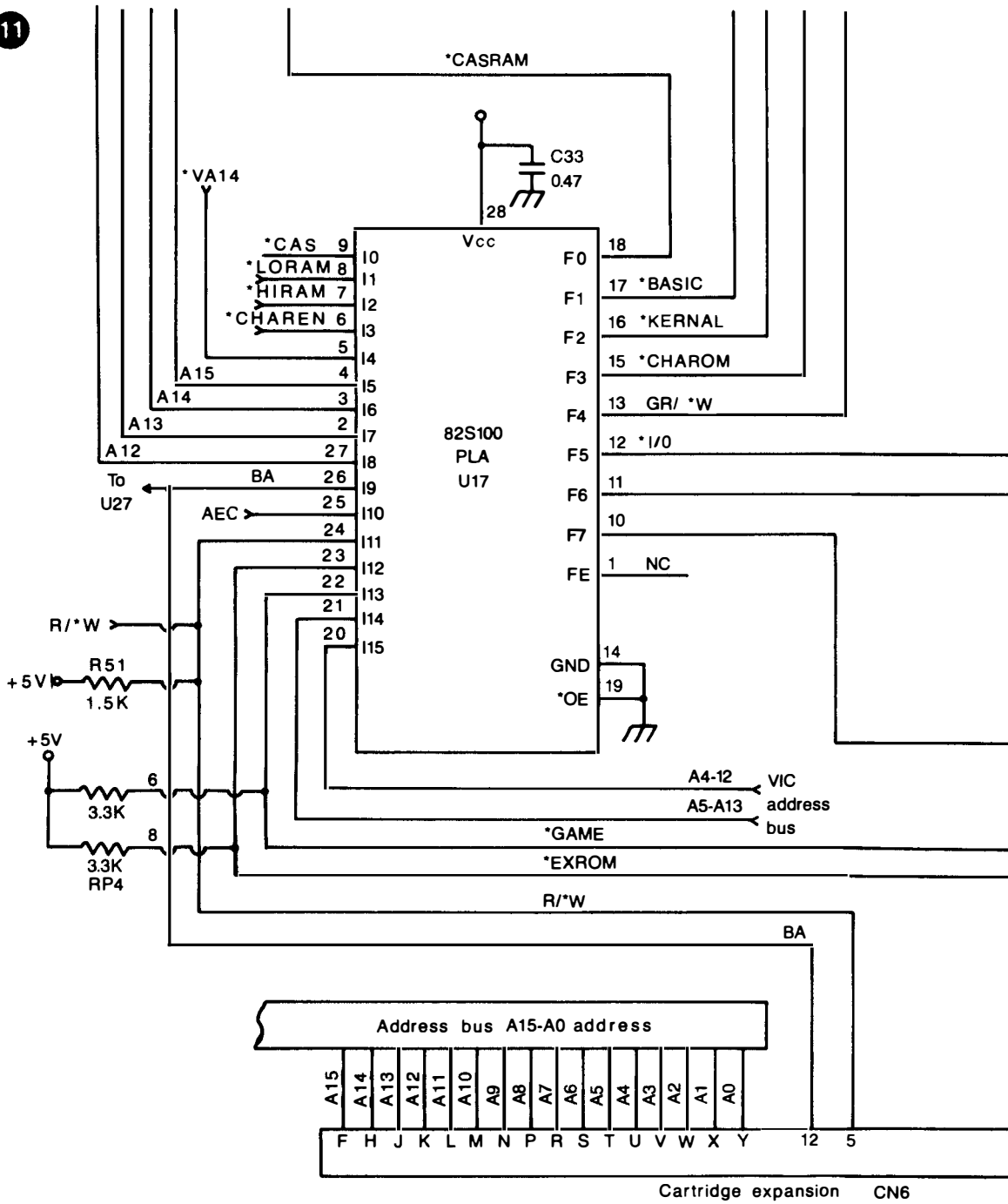


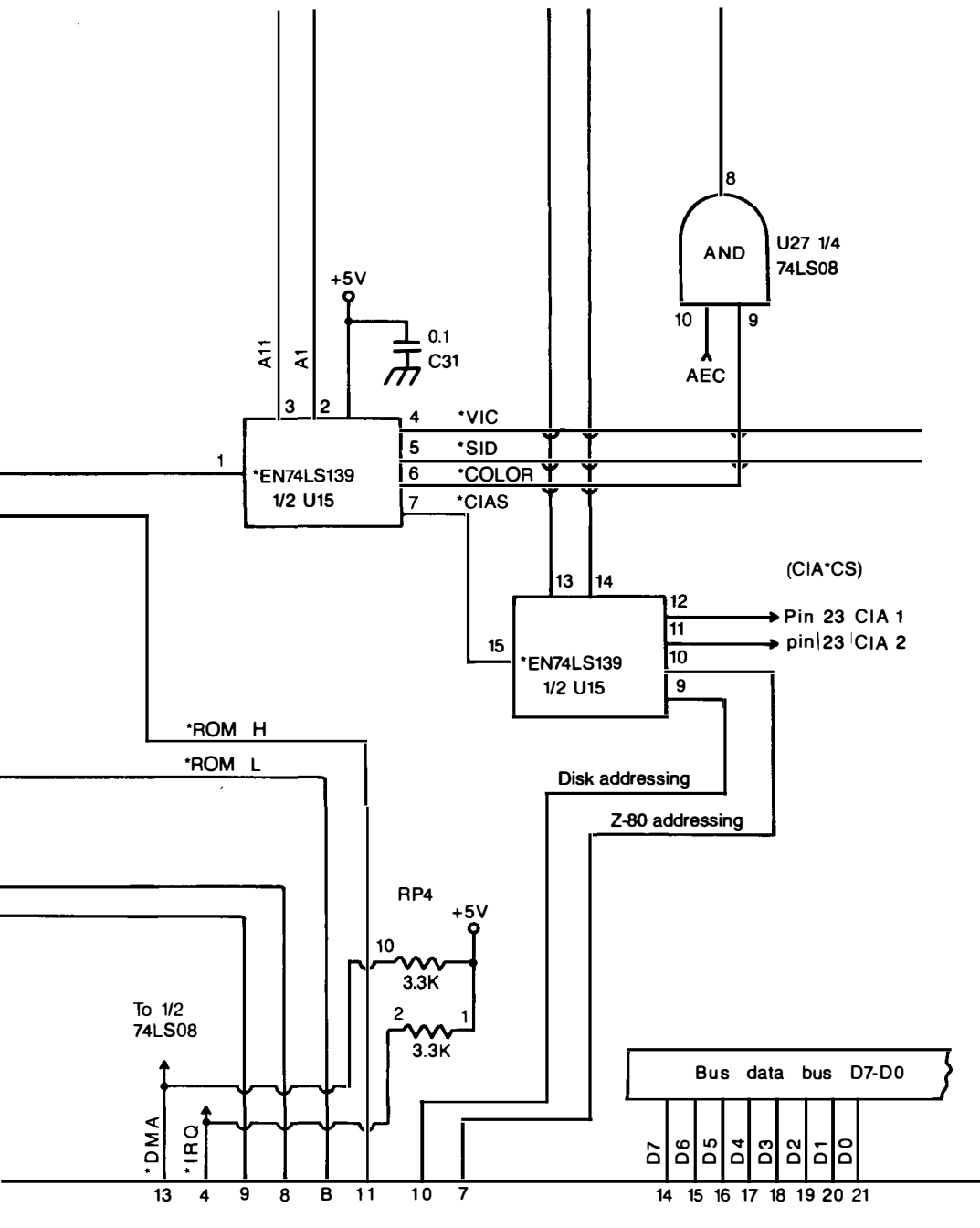
10

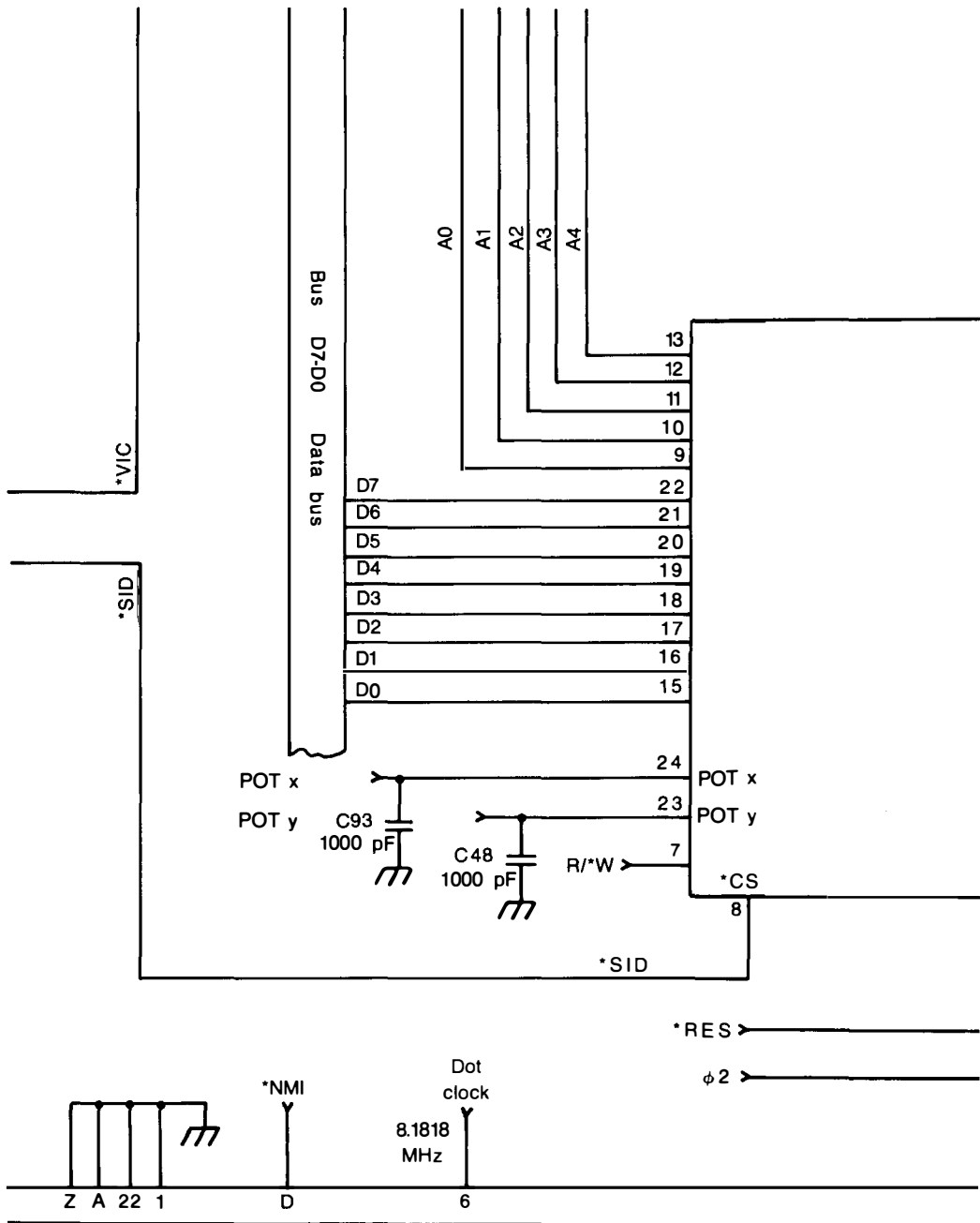
Power supply

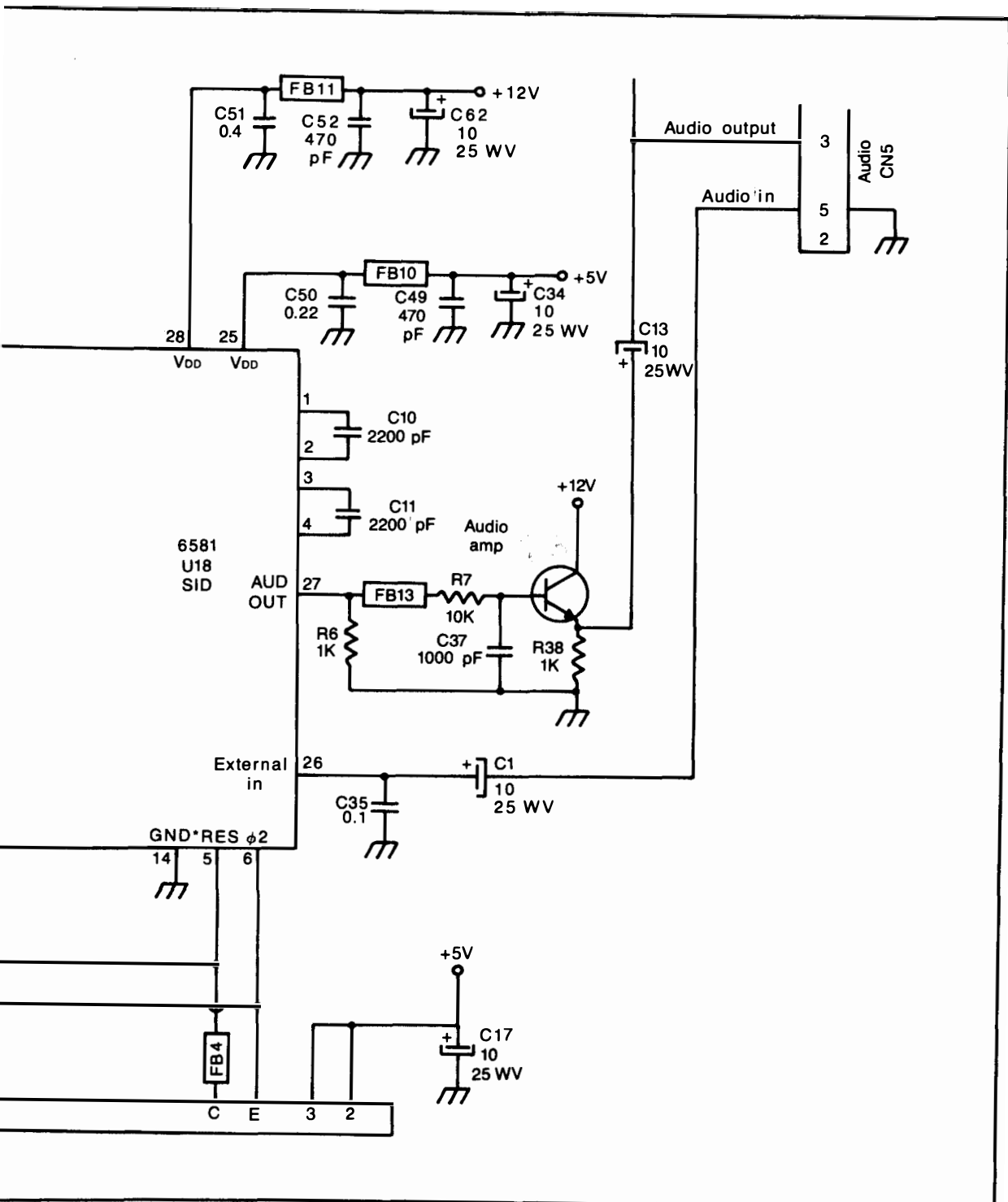














# Index





# Index

---

## A

access time, 232  
accessing peripherals, 233  
accumulator, 200  
address assignments, 237  
address bus connections, 237  
address bus, 58, 236  
address signals, 232  
addressing the decks, 221  
addressing, 60, 192, 253  
ALU, 195  
AND, 111, 165  
ANDing, 188  
arithmetic logic unit, 195  
audio/video ports, 309

## B

BASIC ROM, 37, 90, 97, 99  
binary, 153  
bit map modes, 279  
block diagram, 138  
board defects, 27  
buses, 236

## C

cassette I/O slot, 311  
cassette, 4, 61  
character color, 276

character fetch, 276  
character modes, 278  
character ROM, 37, 68, 90, 96  
chip inserter, 54  
chip location guide, 31, 40  
chip removal, 53  
chip replacement, 39, 42, 52  
chip sockets, 39, 52  
chips, soldered-in, 55  
CIA addressing, 253  
CIA controlling, 253  
CIA data transfer, 255  
CIA operation, 265  
CIA read timing, 257  
CIA timers, 259  
CIA timing, 255  
CIA write timing, 255  
CIA, 8, 34, 61, 253  
circuit board, 8  
cleaning, 28  
clearing, register, 185  
clock signals, 242  
clock, 224  
clock, real time, 259  
CMOS, 50  
color missing, 4  
color RAM, 37, 68, 75  
color signal, 66, 286

color, character, 276  
complementing, register, 187  
complex interface adapter, 8, 34, 61, 253  
condition code register, 195  
continuity tester, 248  
control bus, 58, 236, 241  
control line tests, 249  
control lines, 58, 236, 241  
control ports, 306  
counter, up/down, 116, 179  
CRT, 1

## D

D-type latch, 126  
data bus, 9, 58, 236, 239, 271  
data timing, 232  
dead computer, 1  
decimal, 153  
decoder, 2-of-4, 118  
decoding, 203  
decrementing, register, 188  
default map, 217, 222  
delay time, 233  
desoldering, 75  
diagnostic programming, 5  
diagnostic programs, 7  
DIP package, 50

disassembly as a cure, 24  
disassembly, 16  
disassembly, removing the circuit board, 19  
disassembly, removing the top, 16  
disassembly, tools, 16  
disk system, 4  
displaying characters, 275  
DTL, 42  
dynamic RAM operation, 84  
dynamic RAM timing, 86

## E

82S100, 38, 52, 102, 237  
electrostatic damage, 28  
empty display block, 4  
envelope generator registers, 297  
expansion ports, 306  
extended color mode, 279  
external device troubles, 4  
extraction tool, 53

## F

failures, common, 1  
falling edge, 225  
fan-in, 43  
fanout, 43  
FET, 43, 48  
fetch and execute cycle, 61, 203  
filter registers, 299  
first voice registers, 297  
556 timer, 131, 182, 210  
flag register, 207  
flip-flop, 114, 174, 176  
4044, 130, 136, 226, 239  
4164, 36, 53, 73, 83  
freeze aerosol liquid, 41  
fuse considerations, 316

## G

garbage display, 3, 40  
graphics, 275  
ground plane short, 26  
ground plane, 24

## H

handshake, 62, 64  
handshaking, 259  
heatsink, 56  
hex inverter, 109  
hex, 156  
hexadecimal, 156  
hold time, 232, 233, 255

## I

I/O port, 214, 217, 257  
I/O signals, 217  
incrementing, register, 188  
index register, 206  
inputs, 305

instruction byte, 202  
instruction set, 202  
interrupt control register, 262  
interrupt register, 285  
interrupt request line, 241  
interrupt, 195, 209  
\*IRQ line, 241

## J

joysticks, 4, 61, 265  
jump table, 100, 101  
jumping, 188

## K

kernal ROM, 37, 90, 97, 100  
keyboard, 4, 8, 61, 265

## L

light pen, 285  
logic gate testing, 172  
logic gates, 153, 158  
logic manipulations, 197  
logic probe, 47  
LSI chips, 57  
luminance signal, 66, 286

## M

main line, 317  
masking, 198, 263  
master schematic, 321  
memory decks, 218  
memory layout, 82  
memory map, 12, 214, 236, 237  
memory page, 218  
memory refresh, 82  
microprocessor, 9, 34, 52, 57, 101, 140, 143, 147, 151, 192, 214, 228  
mode/volume register, 299  
MOS, 28, 48, 55  
MOSFET, 79  
MPU and CIA interface, 143  
MPU and dynamic RAM, 140  
MPU and ROM, 143  
MPU and SID, 151  
MPU and VIC, 147  
MPU timing, 228  
multi-color character mode, 279  
multiplexer, 121, 122

## N

NAND, 45  
NMOS, 50  
NOR, 172  
NOT, 163

## O

offset, 206  
operating system, 97, 101  
OR, 169  
ORing, 188

oscillator chip, 224  
outputs, 305

## P

paddles, 4  
page, memory, 218  
PEEK and POKE tests, 249  
PEEK, 6, 158, 223  
 $\phi 0$ , 226, 228, 235, 242  
 $\phi 1$ , 89, 228, 230, 232, 233  
 $\phi 2$ , 89, 228, 230, 232, 233  
PLA, 38, 52, 102, 237  
PMOS, 50  
POKE, 6, 158, 223  
portability, 101  
power supply troubles, 313  
printer, 61  
problems, common, 1  
program counter, 153, 203, 206, 214  
programmable logic array, 38  
programs, 202

## R

R/\*W line, 241  
RAM dynamic, 77  
RAM, 7, 9, 36, 73  
RAM, color, 37, 38, 53, 73, 75, 82  
RAM, dynamic, 36, 53, 73, 83  
RAM, static, 38, 53, 73, 74, 75, 82  
raster register, 284  
read/write line, 241  
reading the decks, 211  
refresh timing, 88  
register clearing, 185  
register complementing, 187  
register shifting, 184  
registers, 158, 174  
registers, computing, 183  
reset line, 242  
resister decrementing, 188  
resister incrementing, 188  
resoldering, 56  
rf modulator, 38  
rising edge, 228  
ROM, 7, 9, 90  
ROM, BASIC, 37, 90, 97, 99  
ROM, character, 37, 68, 90, 93, 96  
ROM, kernal, 37, 90, 97, 100  
RTL, 42

## S

schematic for the C64, 321-345  
screen editor, 100  
serial data register, 262  
serial I/O ports, 309  
service charts, 107  
setup time, 232, 235  
7406, 109  
74LS08, 111  
74LS74, 114, 176  
74LS139, 118, 290

74LS193, 116, 176, 179  
74LS257, 121, 140, 169  
74LS258, 122, 178  
74LS373, 126, 176, 178  
74LS629, 137, 224, 226  
7805, 316  
7812, 137, 316  
shifting, 197  
shifting, register, 184  
SID inputs, 293  
SID operation, 293  
SID registers, 295  
SID timing, 303  
SID, 15, 35, 69, 290  
sine wave, 224  
64 Doctor, 7, 305  
6502, 52, 58  
6581, 15, 35, 69, 290  
6567, 34, 65, 147  
6510, 9, 34, 52, 57, 101, 140, 143,  
147, 151, 192, 214, 228  
6526, 34, 52, 61  
6581, 69, 151  
sockets, 52  
solder suckers, 56  
sound interface device, 15, 35, 69,  
290  
sound missing, 4  
source checking, 316  
special address lines, 271  
sprites, 280  
square wave, 224

stack pointer, 194  
stack, 100, 194  
standard character mode, 279  
static electricity precautions, 28, 30  
static electricity, 28, 48, 55  
switch, bilateral, 130  
symptoms, common, 1  
sync signal, 66, 286

#### T

testing the bus lines, 242  
testing the clock, 233  
testing the SID, 303  
testing the VIC, 289  
testing, 211  
testing, CIA, 268  
testing, logic gate, 172  
testing, MPU, 211  
tests, control line, 249  
tests, PEEK and POKE, 249  
three-state, 46  
timer, 131, 182, 259  
timing, CIA, 255  
timing, data, 232  
timing, MPU, 228  
timing, refresh, 228  
trouble location, power supply, 313  
troubleshooting charts, 15  
TTL, 28, 42, 48  
TV display, 1  
2114, 37, 38, 53, 73, 75, 82  
2364, 37, 90, 97, 99, 100

2332, 37, 68, 90, 93, 96

#### U

user connector, 312

#### V

vector addresses, 211  
VIC operation, 269  
VIC, 9, 12, 34, 38, 52, 65, 269  
video display modes, 275  
video interface chip, 9, 12, 34, 38,  
52, 65, 269  
video missing, 4  
video output, 286  
video/audio ports, 309  
visual inspection, 26  
voice control register, 297  
vom, 47, 107

#### W

working frequency, 225  
wrist strap, 29  
writing to the decks, 211

#### X

XNOR, 172  
XOR, 171

#### Y

YES, 161



# Other Bestsellers From TAB

## **BUILD YOUR OWN IBM® COMPATIBLE AND SAVE A BUNDLE—Aubrey Pilgrim**

Now you can have your own PC, XT, AT, or compatible for hundreds, even thousands, of dollars less than comparable, factory-built systems. This book shows how to buy the needed components and peripherals—disk drives, mother board, keyboards, hard drives, printers, video adapter boards, etc. Then, following the author's instructions, you'll be able to connect all these parts to produce your own customized microcomputer! 224 pp., 108 illus.

**Paper \$14.95** **Hard \$22.95**  
**Book No. 2831**

## **COMPACT DISC PLAYER MAINTENANCE AND REPAIR—Gordon McComb and John Cook**

Packed with quick and reliable answers to the problems of maintaining and repairing CD players, this illustrated, do-it-yourself guide takes the apprehension out of first-time repairs. Master repairman Gordon McComb takes away the mystery that surrounds these seemingly complicated devices and gives you the confidence you need to repair minor malfunctions (the cause of more than 50% of CD player problems). 256 pp., 193 illus.

**Paper \$12.95** **Hard \$18.95**  
**Book No. 2790**

## **COMPUTER PERIPHERALS THAT YOU CAN BUILD—2nd Edition—Dr. Gordon W. Wolfe**

Includes state-of-the-art interfacing techniques and peripheral devices for all of today's most popular computers! You'll learn how to create an interface for the IBM® PC and all of its "plug-compatibles," all Apple® products including the Macintosh™, all Tandy/Radio Shack products, and the Commodore 64™/128™. And you'll build peripherals ranging from sense switches, clocks, and interrupt timers to four different types of analog-to-digital converters, an x-y plotter for data output, a mouse, and bar code readers. 304 pp., 262 illus.

**Paper \$16.95** **Hard \$22.95**  
**Book No. 2749**

## **THE ILLUSTRATED DICTIONARY OF MICROCOMPUTERS—2nd Edition—Michael Hordeski**

Little more than a decade after the introduction of the first microprocessors, microcomputers have made a major impact on every area of today's business, industry, and personal lifestyles. The result: a whole new language of terms and concepts reflecting this rapidly developing technology . . . and a vital need for current, accurate explanations of what these terms and concepts mean. Michael Hordeski has provided just that in this completely revised and greatly expanded new second edition of *The Illustrated Dictionary of Microcomputers!* 368 pp., 357 illus. Large, Desk-Top Format (7" x 10").

**Paper \$14.95** **Hard \$24.95**  
**Book No. 2688**

## **COMMODORE 128™ DATA FILE PROGRAMMING—David Miller**

Of all programming topics, file handling is the one that most intimidates beginning and intermediate programmers alike. Now, David Miller has developed a guide for C-128 users that takes the misery and mystery out of learning to use the C-128 file structure. After completing this book, you will fully understand what files are and how to use them—plus you'll be able to create your own sequential and relative access files! 290 pp., 4 illus.

**Paper \$16.95** **Hard \$21.95**  
**Book No. 2805**

## **TROUBLESHOOTING TECHNIQUES FOR MICROPROCESSOR-CONTROLLED VIDEO EQUIPMENT—Bob Goodman**

With this excellent introduction to servicing these "electronic brains" used in everything from color TVs and remote control systems to video cassette recorders and disc players, electronics service technicians, engineers, computer service technicians, even advanced electronic hobbyists can learn how to get to the heart of most any problem and solve it skillfully and confidently. Includes dozens of handy hints and tips on the types of problems that most often occur in microprocessors and the easiest way to deal with them. 352 pp., 232 illus.

**Paper \$16.95** **Hard \$24.95**  
**Book No. 2758**

## **30 CUSTOMIZED MICROPROCESSOR PROJECTS—Delton T. Horn**

Here it is! The electronics project guide that you've been asking for—a complete sourcebook on designing and building special purpose computer devices around the Z80 microprocessor! This is where you'll find all the information you need not only to construct the specific custom-dedicated CPU projects supplied by the author, but also how to customize these devices for your own individual applications! 322 pp., 211 illus.

**Paper \$14.95** **Hard \$22.95**  
**Book No. 2705**

## **COMMODORE 64™/128™ GRAPHICS AND SOUND PROGRAMMING—2nd Edition—Kroute**

Here's all the hands-on, learn-by-doing information you'll need to start taking full advantage of your Commodore's exceptional graphics powers—sprite, character, and bit-mapped graphics. Plus, you'll find out how to utilize your machine's advanced three-voice music synthesizer chip. Best of all, you'll find a whole collection of ready-to-run programs to demonstrate how each concept works! 272 pp., 157 illus. Large Format (7" x 10").

**Paper \$14.95** **Hard \$22.95**  
**Book No. 2640**

# Other Bestsellers From TAB

**THE COMPUTER SECURITY HANDBOOK—Richard H. Baker**

Electronic breaking and entering into computer systems used by business, industry and personal computerists has reached epidemic proportions. That's why this up-to-date sourcebook is so important. It provides a realistic examination of today's computer security problems, shows you how to analyze your home and business security needs, and gives you guidance in planning your own computer security system. 288 pp., 61 illus. 7" x 10".

**Hard \$25.00** **Book No. 2608**

**PRACTICAL INTERFACING PROJECTS WITH THE COMMODORE™ COMPUTERS—Robert H. Luetzow**

Hands-on techniques for transforming your C-64, C-16, Plus/4, VIC-20 or the new C-128 into an accurate controller for science, engineering, or home and hobby electronics applications. Includes over 80 different software programs, plus an introduction to using machine language for controlling I/O projects. This sourcebook will have you using your Commodore in exciting new ways. 256 pp., 256 illus.

**Paper \$16.95** **Book No. 1983**

**COMMODORE 64™ EXPANSION GUIDE—Gary Phillips**

Far more than just a product listing or a rehash of manufacturers' sales brochures, these are the best of the hundreds of hardware accessories currently on the market . . . each one chosen for value and performance after exhaustive testing and examination. You'll find in-depth background on each type of device—printers, disk drives, modems, monitors, and photographic details. 288 pp., 31 illus.

**Paper \$16.95** **Hard \$22.95**  
**Book No. 1961**

**ONLINE RESEARCH AND RETRIEVAL WITH MICROCOMPUTERS—Nahum Goldmann**

This time-saving guide shows you how to turn a micro into an invaluable research "tool" for digging up information from databases across the country. Using Nahum Goldmann's "Subject Expert Searching Technique," businessmen, engineers, physicians, lawyers, professors, and students can quickly and easily retrieve information in the comfort of their work station. 208 pp., 119 illus.

**Hard \$25.00** **Book No. 1947**

**COMMODORE 64™ ADVANCED GAME DESIGN—Schwenk**

Professional game designers George and Nancy Schwenk reveal their winning formula for creating stimulating, professional-quality microcomputer games for family fun and even profit! Using three fully-developed C-64 games to illustrate game design, this innovative tutorial provides an informative and practical look at the conceptual and implementation techniques involved. 144 pp., 14 illus. 7" x 10".

**Paper \$10.95** **Book No. 1923**

**DATA COMMUNICATIONS AND LOCAL AREA NETWORKING HANDBOOK—Britt Rorabaugh**

With data communications and LANs being the area of greatest growth in computers, this sourcebook will help you understand what this emerging field is all about. Singled out for its depth and comprehensiveness, this clearly-written handbook will provide you with everything from data communications standards and protocols to the various ways to link together LANs. 240 pp., 209 illus., 7" x 10".

**Hard \$25.00** **Book No. 2603**

**TROUBLESHOOTING AND REPAIRING SATELLITE TV SYSTEMS—Richard Maddox**

This first-of-its-kind troubleshooting and repair manual can mean big bucks for the professional service technician (or electronics hobbyist looking for a way to start his own profitable servicing business) . . . *plus big savings for the TVRO owner who wants to learn the ins and outs of maintaining and servicing his own receiver!* Includes service data and schematics! 256 pp., 479 illus.

**Paper \$18.95** **Hard \$26.95**  
**Book No. 1977**

**101 PROGRAMMING SURPRISES AND TRICKS FOR YOUR COMMODORE 64™ COMPUTER—Heiserman**

This exciting new collection of games, novelties, and programming marvels is fresh, literate, and packed with all kinds of downright amazing ways to have fun with your C-64. And unlike other programming books, it makes no attempt to instruct you—instead, the object is to entertain and be entertained. 224 pp., 12 illus., 7" x 10".

**Paper \$11.95** **Book No. 1951**

**COMPUTER TECHNICIAN'S HANDBOOK—2nd Edition—Art Margolis**

Cash in on the multi-million dollar computer repair business! Written by a successful microcomputer repairman, the *Computer Technician's Handbook* starts with the basics and takes you one step at a time through the process of breaking down and repairing a personal computer! Here's all the electronics background you'll need, particulars on equipment required, and clear, detailed schematics. 490 pp., 284 illus.

**Paper \$17.95** **Book No. 1939**

**MICRO MANSION: USING YOUR COMPUTER TO HAVE A SAFER, MORE CONVENIENT HOME—David B. Bonyng**

Here's where you'll find everything you need to know to get the benefits of a computer home control system in the most economical, trouble-free way whether you only want to install a simple light, or you want a computer home control system that has it all—complete control of heating and cooling and improved security—this book is for you! 192 pp., 115 illus. 7" x 10".

**Hard \$18.95** **Book No. 1906**

# Other Bestsellers From TAB

**COMPUTER USER'S GUIDE TO ELECTRONICS—Art Margolis**

Assembly language will be easy to learn . . . you'll be able to interface with peripherals . . . and most importantly, you can perform simple repair procedures yourself. In fact, the savings realized by changing one bad chip yourself will more than pay for this invaluable handbook! 320 pp., 250 illus. 7" x 10".

**Paper \$15.95 Book No. 1899**

**TROUBLESHOOTING AND REPAIRING YOUR COMMODORE 64™—Art Margolis**

Cure virtually any problem that plagues your Commodore 64 with this hands-on repair guide for do-it-yourself owners and professional servicemen alike! It's chock full of maintenance tips, troubleshooting procedures and do-it-yourself repair techniques that will save you time, money, and frustration. Even includes names and addresses of where you can obtain parts. 368 pp., 291 illus.

**Paper \$16.95 Hard \$22.95  
Book No. 1889**

**MASTERING THE 68000™ MICROPROCESSOR—Phillip R. Robinson**

Delve into the heart of today's new 68000-based microcomputers—Apple® Macintosh, ATARI®, and others. Here's a software-oriented approach to understanding and programming the Motorola family of microprocessors and support chips including the 68008, 68010, and the 68020. It takes you inside the architecture of the 68000 including registers, flags, and more! 272 pp., 121 illus., 7" x 10".

**Paper \$16.95 Hard \$22.95  
Book No. 1886**

**FROM FLOWCHART TO PROGRAM—Richard G. Todd**

Master the skills of effective, "bug-free" programming with this practical approach to program design and development. Using the flowcharting system of structuring a program, you'll learn step-by-step how to break down the problem logically, enabling you to tackle even large-scale programs that are easier to debug, easier to change and expand, easier to understand, and faster in execution. 192 pp., 190 illus., 7" x 10".

**Paper \$12.95 Book No. 1862**

**1001 THINGS TO DO WITH YOUR COMMODORE 64™—Mark R. Sawusch and Tan A. Summers**

Here's an outstanding sourcebook of microcomputer applications and programs that span every use and interest from game playing and hobby use to scientific, educational, financial, mathematical, and technical applications. It provides a wealth of practical answers to the question—what can my Commodore 64 computer do for me? Contains a gold mine of actual programs! 256 pp., 47 illus.

**Paper \$10.95 Book No. 1836**

**INTERFACING YOUR MICROCOMPUTER TO VIRTUALLY ANYTHING—Joseph J. Carr**

Here, at last, is a sourcebook that's literally packed with practical interfacing techniques, plus a wealth of useful projects. You'll find projects such as a single-ended amplifier, a differential amplifier, a universal rear-end, and a precision 10-volt reference power source. All of these building block circuits can be used in a variety of applications. 336 pp., 212 illus.

**Paper \$13.95 Hard \$21.95  
Book No. 1890**

**MASTERING THE 8088 MICROPROCESSOR—Lanny V. Dao**

This is a must-have handbook for every creative programmer who works with an IBM PC®, PCjr®, a PC-compatible micro, or any other machine based on an 8088 or compatible chip such as the 8085, 80286, or 80186 (the chip used in the new Tandy 2000)! Here is your source for instant access to all the advanced programming capabilities offered by the Intel 8088, complete with actual program examples. 336 pp., 142 illus.

**Paper \$15.95 Hard \$22.95  
Book No. 1888**

**ARTIFICIAL INTELLIGENCE PROJECTS FOR THE COMMODORE 64™—Timothy J. O'Malley**

This uniquely-exciting guide includes 16 ready-to-run, fully-explained projects illustrating a wide variety of artificial intelligence techniques; a plain-English introduction to artificial intelligence, robotics, and LISP; a complete glossary of artificial intelligence terminology; easy-to-follow examples and show-how illustrations; plus a quick-look-up index for fast reference. 160 pp., 15 illus., 7" x 10".

**Paper \$12.95 Book No. 1883**

**THE BASIC COOKBOOK—2nd Edition—Ken Tracton & Thomas A. Wells**

Covers every BASIC statement, function, command, and keyword in easy-to-use dictionary form—highlighted by plenty of program examples—so you can cook up a BASIC program in just about any dialect, to do any job you want. Whether your interests are business, technical, hobby, or game playing, this revised 2nd edition of our all-time best-selling BASIC guide contains exactly what you want, when you want it! 168 pp., 57 illus.

**Paper \$7.95 Book No. 1855**

**SERIOUS PROGRAMMING FOR THE COMMODORE 64—Simpson**

Serious programming means writing programs that are user friendly, well documented, and designed to take full advantage of all of the resources offered by the BASIC language, your DOS (disk operating system), and assembly language routines. And that's what you'll find here—everything you need to get more programming power from your C-64! 208 pp., 124 illus. 7" x 10".

**Paper \$9.95 Book No. 1821**



# Other Bestsellers From TAB

□ **BASIC COMPUTER SIMULATION**—Lawrence L. McNitt

Use your computer to find answers to questions that simply don't have neat, easily found solutions. Now, this exceptional sourcebook introduces you to the how-to's of modeling and creating simulations, and programs written in a universal subset of BASIC that can be used on any BASIC microcomputer. Loaded with easy-to-follow explanations, detailed illustrations, and specific programming examples! 352 pp., 63 illus., 7" x 10".

Paper \$15.50

Book No. 1585

□ **25 GRAPHICS PROGRAMS IN MICROSOFT® BASIC**—Timothy J. O'Malley

Plot two-dimensional graphs, draw maps and diagrams, plot in three dimensions, using either high- or low-resolution graphics, construct a model or a function, create an interactive space shuttle simulation, generate computer art including animated graphics! This collection of graphic programs written in MICROSOFT BASIC can easily be adapted to any BASIC micro system! 160 pp., 92 illus., 7" x 10".

Paper \$11.95

Book No. 1533

□ **TROUBLESHOOTING AND REPAIRING PERSONAL COMPUTERS**—Art Margolis

Save time and money when your personal computer breaks down, even get in on one of today's fastest growing business opportunities—home computer maintenance and repair! Everything you need is included in this hands-on guide to troubleshooting and repairing all types of personal computers—from the ZX81 to the IBM PC, the Apples, TRS-80, ATARI, TI-99, and more! 320 pp., 293 illus. 7" x 10".

Paper \$16.95

Book No. 1539

□ **MAINTAINING & REPAIRING VIDEOCASSETTE RECORDERS**—Robert L. Goodman

This is the most practical handbook yet on troubleshooting and servicing *all the popular brand home videocassette recorders!* From a look at the history of video tape recording to specific information on test equipment set-ups, servicing techniques, and maintenance . . . here's all the hands-on advice, practical tips, and specific how-to's anyone needs to keep VCRs in A-1 operating condition! 416 pp., 348 illus.

Paper \$15.95

Book No. 1503

\*Prices subject to change without notice.

---

---

Look for these and other TAB books at your local bookstore.

---

---

TAB BOOKS Inc.  
P.O. Box 40  
Blue Ridge Summit, PA 17214

---

---

Send for FREE TAB catalog describing over 1200 current titles in print.

OR ORDER TOLL-FREE TODAY: **1-800-233-1128**

# Troubleshooting & Repairing Your Commodore 64™

Art Margolis

**An invaluable time- and money-saving tool for every C-64 owner . . .  
as well as an unbeatable technical reference for service professionals!**

If you own a C-64, this sourcebook is probably the most important "add-on" you'll ever purchase! Packed with maintenance tips, easy troubleshooting procedures, and do-it-yourself repair techniques, it can save you untold time, money, and frustration over the lifetime of your micro. The simple chip-changing instructions, alone, will enable you to cure at least 50% of the problems that cause micro breakdowns. Plus, Margolis gives you as much technical detail and advanced computer repair guidance as you're interested in acquiring! Advanced hardware enthusiasts and professional service technicians will both appreciate the in-depth examination of every C-64 circuit—including timing diagrams, the use of PEEK and POKE commands to signal-trace circuits, and more.

And, if you're a programmer interested in getting maximum performance from your machine, there's a wealth of information to help you gain mastery of your micro—including complete memory maps and thorough descriptions of the C-64's operating characteristics. Just some of the highlights include:

- Tried and true service procedures for pinpointing trouble spots.
- A Chip Location Guide—a printboard layout of all 32 chips.
- Electronic and mechanical techniques for chip replacement.
- 21 Test Point Charts showing exact pinouts, the name of each pin, arrows indicating direction of signal flow, and readings you should receive using a logic probe or VOM.
- A master schematic of the Commodore 64 complete with all parts numbers—an essential tool for more difficult repairs.

With the high cost of professional computer repairs, just one chip replacement can cost you more than the price of this excellent sourcebook. Plus, your better understanding of how the C-64 works will make you a better programmer. And, who knows, you may decide to put your new found troubleshooting and repair skills to work in your own full- or part-time C-64 repair business!

Art Margolis is a computer professional who has written several bestselling books for TAB on electronics and computer troubleshooting and repair.

---

**TAB TAB BOOKS Inc.**

Blue Ridge Summit, Pa. 17214

---

Send for FREE TAB Catalog describing over 1200 current titles in print.

FPT > \$16.95

ISBN 0-8306-1889-9

PRICES HIGHER IN CANADA

1660-0885