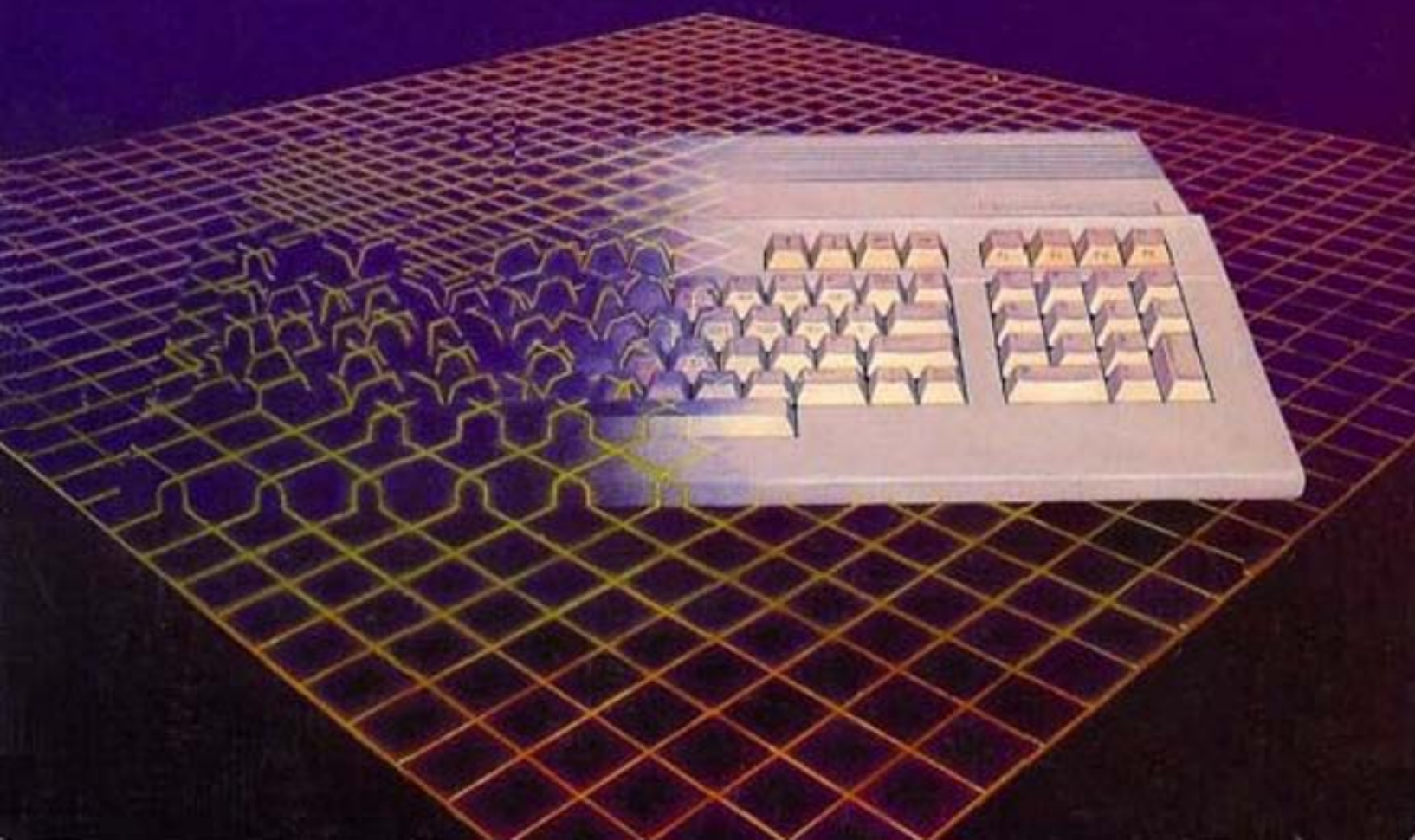
3083

ELECTRONIC PROJECTS FOR YOUR COMMODORE 64™ AND 128™

John Iovine



Electronic Projects For Your Commodore™ 64 and 128

To my father who was always there for me:
Frank L. Iovine
Sept. 28,1925 - Aug. 21,1980

To my mother who still is:
Ann M. Iovine

Electronic Projects For Your Commodore™ 64 and 128

John Iovine

TAB BOOKS Inc.
Blue Ridge Summit, PA

FIRST EDITION
FIRST PRINTING
Copyright © 1989 by TAB BOOKS Inc.
Printed in the United States of America
Reproduction or publication of the content in any manner, without express permission of the publisher, is prohibited. The publisher takes no responsibility for the use of any of the materials or methods described in this book, or for the products thereof.

Library of Congress Cataloging in Publication Data

Iovine, John.
Electronic projects for your Commodore 64 and 128 I by John Iovine.
p. cm.
Includes index.
ISBN 0-8306-0483-9 ISBN 0-8306-9383-1 (pbk)
1. Computer interfaces-Amateurs' manuals. 2. Commodore 64 (Computer)-Amateurs'manuals. 3. Commodore 128 (Comp□ -Amateurs' manuals. I. Title
TK9969.I581989
004.6-dc19 89-4320
CIP

TAB BOOKS Inc. offers software for sale. For information and a catalog, please contact TAB Software Department, Blue Ridge Summit, PA 17294-0850.

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

Reader Inquiry Branch
TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0214
Cover illustration by Mia Bosna

Contents

Introduction vii

1 Computer Fundamentals 1

Memory · Bank Switching · Input/Output Devices · Interfacing · Building Projects · How to Use This Book

2 User Port Fundamentals 7

6526 Chip · Memory Mapped I/O · Binary · DDR Register · Peripheral Data Register · Input · Output · Circuit Construction Logic · Setting a Bit with " OR " · Electronic Logic · Parts List

3 Speech Synthesizer 22

Speech Synthesis · The Speech Chip · A Little on Linguistics · Circuit Construction · The Program · Example · Basic Crunch · Conclusion · Parts List

4 Analog to Digital Conversion 34

Analog Events · Digital Events · Serial A/D Converter Chip · Test Program · Resolution · Programming · Transducers · Light · Applications · Temperature · Applications · Toxic Gas Sensor · Applications · Biofeedback · Applications · 60 Hz Interrupt Vector Demo Interrupt · A/D Interrupt

5 Digital Audio Recording and Playback 55

Applications · Sound Sampling · Digital Oscilloscope · Circuit Description · Programs · Applications · Parts List · Advanced Sound Digitizer Program

6 Subliminal Communication **73**
History · Orwellian Mind Control · Audio · Visual · Subliminals and the Law · Circuit Construction · Hookup · Circuit Operation · Message Screens · Troubleshooting · Bibliography · Parts List

7 Appliance Controller **82**
Real World Environment · Inductive and Resistive Loads · Dc Loads · Ac Loads · Circuit Construction · Test · Program · Smart Control · Parts List

8 Monitor Projects **91**
80 Column to TV · Circuit Construction · Screen Saver C-128

9 Digital Camera **96**
D-Cam Chip · Photoelectric Effect · Matrix Unscramble · Limitations of System · Low Resolution Screen · Extended Field of View · Black and White Camera · Gray Scale · 256 Shades of Gray · Coloration · Timing · Construction · Lenses · Preassembly Test · Final Assembly · Lighting · Program Operation · Hi-Resolution Digital Camera · Artificial Vision · A Little History · D-Cam · Conclusion · Parts List

10 Dynamic Equations **139**
Dynamic Equations and Nature · Beginning of Chaos · Population Growth Model · Graphing Programs · Order Out of Chaos · Self-Similarity · The Butterfly Effect · Nature · Usefulness · Fractals · Why Now?

11 Mandelbrot Graphics **150**
Complex Numbers · Plotting · First Mandelbrot Picture · Program Features · Program Operations · Advanced Operations · Classic Fractals · Properties of Complex Numbers · Algebraic Operations · Computer Choke

12 Additional 6526 Functions **167**
Timers · Control Registers (ICR) · Interrupt Control Register · Time of Day Clock (TOO) · Frequency Counter · Frequency Generator

Index **177**

Introduction

Commodore computers are versatile machines that represent a landmark in the era of personal computers. The use of large-scale integrated (LSI) circuits in these computers enabled millions of people to purchase Commodore computers as their first home computer.

The purpose of this book is to show you how to build electronic projects that will enhance your computer. But more than this, the information can provide a foundation upon which you can build. Perhaps this book will provide the spark to ignite your own creativity. Many projects lie on the frontiers of technology. The projects represent a starting point for the basic tools with which you can explore on your own, and perhaps make a contribution to emerging technologies.

As an example, the sound-digitizing project can be expanded for work on speech-recognition programs. The digital camera can be used for machine vision and recognition.

I'm sure you will be delighted to learn how simple and inexpensive it is to build these projects. This is a "learn-by-doing" approach. You will also see how certain avenues are taken to accomplish various feats; this will help resolve your interfacing tasks when they arise.

In the design of their computers Commodore engineers gave the computer an extremely versatile user port. Although the user port is not the only port capable of interfacing, it is the one on which I concentrate.

This book assumes no previous electronic or digital interfacing experience.

The projects can be constructed by novices. The schematics

are kept simple, block components-such as ICs that function Iike complete electronic circuits-where possible. In addition, ICs are drawn as they physically appear, rather than in standard electronic schematic form. This is to make it easier for you to wire the circuits.

For those of you with prior electronic experience, you will enjoy a feast of circuits that are not only simple in construction, but perform as good or better than their commercial counterparts. Quality has not been compromised in order to keep the circuits simple.

Many of the programs used to run the projects are written succinctly, with little embellishments. This approach to the programming is intentional; it allows you to enter these programs as subroutines into your own programs. In many cases, machine language (ML) subroutines became essential, as BASIC is too slow to adequately handle some of the projects.

All the ML routines are written with a BASIC loader. A BASIC loader is a small BASIC program that pokes an ML program into memory, where it can be called upon (SYS XXXX) by another BASIC program.

All the circuits within this book have been constructed and tested. I have tried to anticipate difficulties that might be encountered and provide troubleshooting information. Naturally wiring errors, bad components, and programming errors will prevent the projects from functioning properly. So do take your time, double check your work, and you will find the rewards well worth it.

Most of the projects are open ended (meaning that they can be built on). Descriptions of practical applications of circuits are given where applicable. Avenues of research are opened for those of you who wish. Neural networks, character and pattern recognition, fractals, and speech recognition are just a sampling of the types of research you can pursue. The tools are enclosed.

1

Computer Fundamentals

Each one of the computers we will be interfacing, the C-128, C-64, and Vic-20 has a 8 bit microprocessor. The MicroProcessor Unit (MPU) is the heart and brain of the computer. The MPU performs the following functions; internal math and logic operations, directs data to and from memory, and can execute a sequence of commands, previously stored in memory, that we call a program. The microprocessor can also be called the Central Processing Unit (CPU).

Although all the computers we will be working with use an 8-bit MPU that functions pretty much the same, they are not the same MPU chip. The C-128 computer which has the latest and most enhanced MPU, a 8502, the C-64 MPU is a 6510 and the VIC-20 MPU is a 6502.

Each of these microprocessors has 16 address lines (see Fig.1-1) that are capable of addressing up to 65535 unique memory cells or locations. Also each MPU has 8 data lines to read/write information in memory that has been addressed by the addressing lines. Each memory cell or location can contain one byte (8-bits) of information, which matches eight of the data lines (one line per bit). The 16 addressing lines are commonly called the Address bus, similarly the 8 data lines are called the Data bus.

MEMORY

Memory can be divided into two main groups, RAM and ROM. RAM is an acronymn for Random Access Memory. RAM memory is a read/write

Fig.1-1. Internal computer structure.

memory. When you type in a BASIC program it is held in RAM. As long as power is maintained the program will remain in memory. When you turn off the computer, anything that was written into RAM will evaporate.

ROM is an acronymn for Read Only Memory. ROM is permanent memory. The program or data in ROM has been burned in. New information cannot be written into ROM because the old information is permanently programmed in. The MPU can execute a program that is stored in ROM. And this is what happens every time you power-up your computer. The ROM contains a bootstrap program that initializes the computer and becomes the operating system (OS) called the Kernal.

BANK SWITCHING

In the C-128 and C-64 the total number of memory cells exceed the 65535 capacity of the address lines. To access this additional memory, a technique known as bank switching is employed. How bank switching works is simple. Imagine two sets of memory, each containing 65535 identically numbered memory locations. Label one set of memory cells Bank #1 and the other Bank #2. By using a computer controlled electronic switch (logic circuits) you can read or write into either one of the memory banks, (but not both at the same time). I admit this is an over simplification of the actual process, but it is correct.

INPUT/OUTPUT DEVICES

Our computers come with standard interfaces that communicate with the outside world. The two main devices are the keyboard (an input device) and the monitor (an output device). All Input/Output (I/O) devices including the projects in this book are connected to the MPU as if they are memory. By accessing these memory locations the computer can either instruct (write operation) an interface to do something, or read (read operation) to see if anything happened.

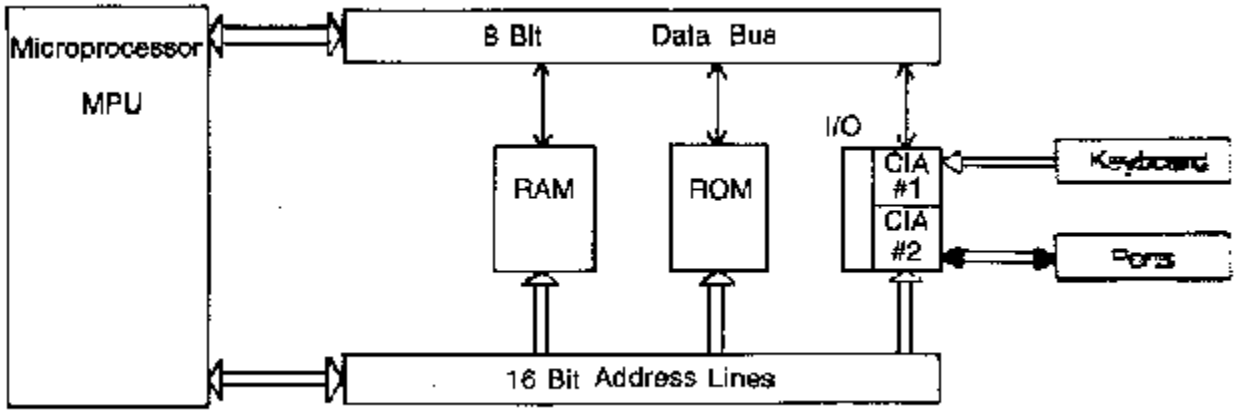


Fig. 1-1. Internal computer structure.

Fig.1-1. Internal computer structure.

memory. When you type in a BASIC program it is held in RAM. As long as power is maintained the program will remain in memory. When you turn off the computer, anything that was written into RAM will evaporate.

ROM is an acronym for Read Only Memory. ROM is permanent memory. The program or data in ROM has been burned in. New information cannot be written into ROM because the old information is permanently programmed in. The MPU can execute a program that is stored in ROM. And this is what happens every time you power-up your computer. The ROM contains a bootstrap program that initializes the computer and becomes the operating system (OS) called the Kernal.

BANK SWITCHING

In the C-128 and C-64 the total number of memory cells exceed the 65535 capacity of the address lines. To access this additional memory, a technique known as bank switching is employed. How bank switching works is simple. Imagine two sets of memory, each containing 65535 identically numbered memory locations. Label one set of memory cells Bank #1 and the other Bank #2. By using a computer controlled electronic switch (logic circuits) you can read or write into either one of the memory banks, (but not both at the same time). I admit this is an over simplification of the actual process, but it is correct.

INPUT/OUTPUT DEVICES

Our computers come with standard interfaces that communicate with the outside world. The two main devices are the keyboard (an input device) and the monitor (an output device). All Input/Output (I/O) devices including the projects in this book are connected to the MPU as if they are memory. By accessing these memory locations the computer can either instruct (□vrite operation) an interface to do something, or read (read operation) to see if anything happened.

INTERFACING

Connecting a computer to an external device or circuit is known as interfacing. Interfacing can involve controlling, reading or exchanging data from your computer to an external device, circuit, or another computer. Commodore computers use a number of standard interfacing devices to communicate to the outside world. A keyboard allows you to input information, a monitor to output or display information. Additional I/O devices available are a disk drive for permanent program and file storage, and retrieval, modems to connect to mainframe computers or information services, joysticks, mouses, etc.

We will build our own interfaces, whose applications range from computer control of appliances and electronic devices, computer vision systems, robotics, and biofeedback devices. You will learn to utilize simple transducers with an analog to digital converter to allow the computer to monitor, measure, or react to light, sound, temperature, pressure, vibrations, and more. The applications are limited only by your imagination and determination.

BUILDING PROJECTS

The number of tools needed to construct the projects in this book are minimum, they are:

- prototype breadboard
- wire cutters/strippers
- soldering iron

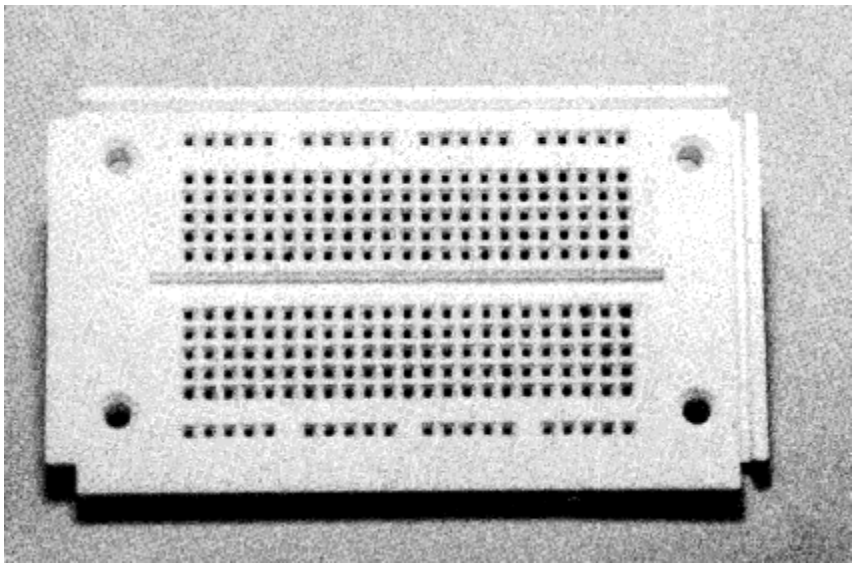


Fig.1-2. Experimenters breadboard.

The prototyping breadboard is important, it allows us to construct the projects with a minimum amount of soldering. (See Figs.1-2,1-3, and 1-4.)

The prototyping breadboard is solderless, 22-gauge wire is used to connect the components on the board. If you have never used a prototyping breadboard the holes on the board are plug points. The plug points are internally wired as shown in Fig.1-3 and 1-4. T'he column points on each half are connected as shown as well as a row on each half. The rows are usually used as power supply connections, one being ground the other the positive voltage supply.

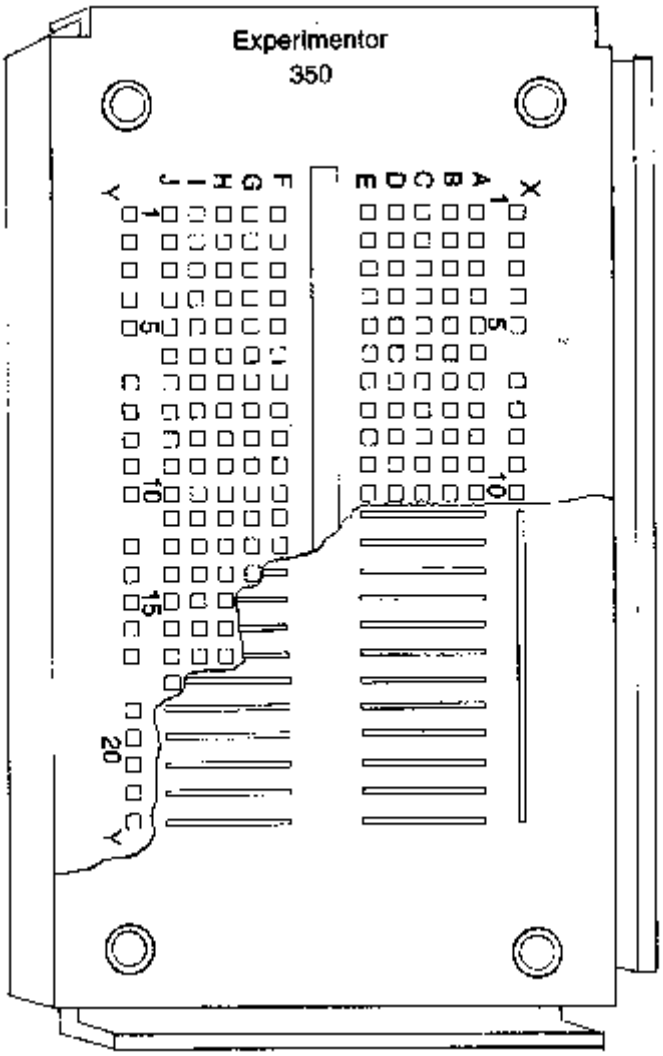


Fig.1-3. Cut away view of experimenters breadboard.

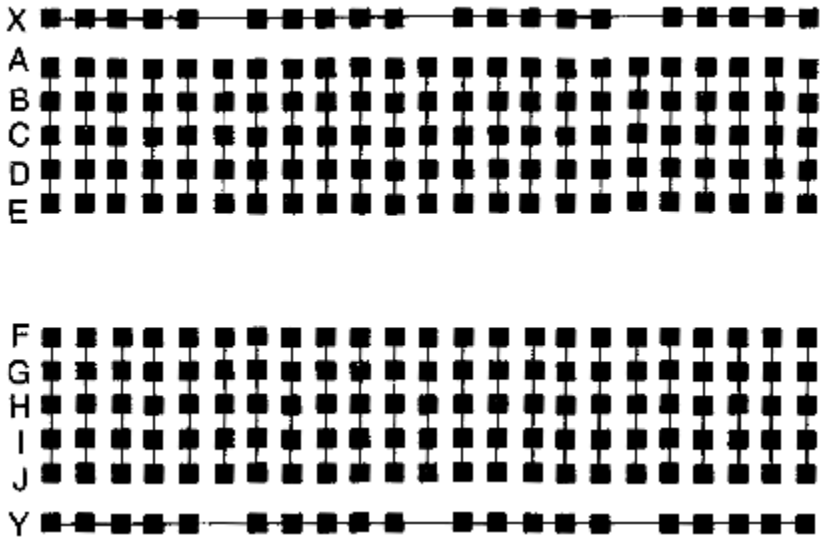


Fig.1-4. Internal wiring of experimenters breadboard.

The board is excellent for constructing the projects in this book. By assembling the circuits first on the board, wiring changes, and/ or corrections are easily accomplished. After you have finished a project and have it working you can decide whether you wish to make that project permanent. If you do decide to do this you will find it easier to transfer the project onto a standard printed circuit board and solder it together. At that point you could add any components such as power switches or phono-jacks that will enhance the operation of a permanent version. You will also have the knowledge that the project works perfectly beforehand.

The breadboard is reusable, after you have finished a project you can simply pull the components out and begin on your next project. Although our board is solderless, some of the components that are used must have wires soldered to them. The largest component I can think of is the card connector that plugs into the commodore user port, but there are also switches, battery holders and an odd assortment of other parts.

As you continue working with electronic projects you will find a VOM and a digital logic probe helpful aids. Although neither of these tools are required to construct the projects in this book. You will find you'll suffer less brain damage when troubleshooting a circuit if you have the ability to check if a circuit is receiving power with your VOM, or check that the computer is really outputting logic signals on the proper wire with your probe. This will become even more important as you advance beyond the projects in this book and design your own interfaces.

HOW TO USE THIS BOOK

This book provides a step by step guide to build electronic enhancement projects for your computer. The projects are arranged so that each chapter builds upon information in previous chapters. No previous electronics experience is assumed, the projects are outlaid so that novice beginners can construct these projects. Pictorial views of ICs are used instead of the standard schematic drawings to help beginners wire the projects. A working knowledge of BASIC language is assumed. Each project requires a program to run the interface circuit. In some cases it became imperative to use machine language (ML) subroutines to utilize the projects to their fullest capabilities. In these cases BASIC program loaders that poke the ML subroutines into memory are provided.

2

User Port Fundamentals

Commodore engineers have been generous in their design of computers, allowing users access to various I/O (input/output) ports. This I'm sure, is a major reason for Commodores immense popularity and profusion of aftermath hardware and software. Our concentration will focus on the user port located at the back of the following computers; the VIC-20, C-64 and C-128. These computers have similar user ports (see Fig. Z-1) that are functionally about the same.

We will be accessing Port B of the user port labeled PBO through-PB7. This is an 8-bit parallel port. Each bit on the port is bidirectional. Programmable as either an input or output bit. Each bit on the port can be programmed independently from all the other bits.

6526 CHIP

Commodore computers use an integrated circuit chip between the central processing unit (microprocessor) and the I/O ports. The C-64 and C-128 computers both use a 6526 CIA (complex interface adapter) chip. The Vic-20 uses a 6522 VIA (versatile interface adapter) chip.

(In order to avoid confusion, further descriptions of the 6526 CIA chip will be the only one given, and should be assumed to be the same for the 6522 chip unless otherwise noted.)

All input and output requests and functions transmitted by the MPU are buffered by the 6526 chip. Each 6526 CIA chip contains 2 parallel,

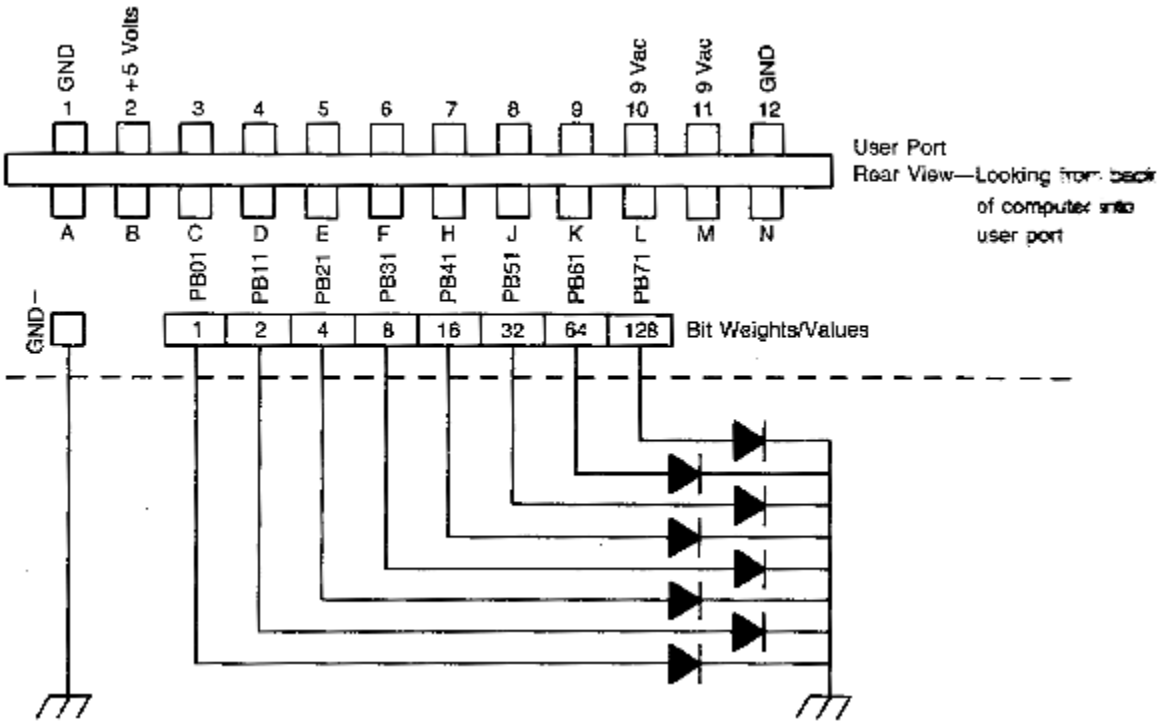


Fig. 2-1. User port.

8-bit input/output ports, two 16-bit counter/timers, clock, and a serial shift xegister. The chip is responsible for the 60 Hz interrupt routine, keyboard scanning, game port reading, and serialized data input. As you can see, in addition to being a buffer the CIA has its own specialized functions that compliment the microprocessors and relieve it of many housekeeping chores.

We will learn to use the timers, interrupt routines, and serialized data input later on in the book. For now we will concentrate on the basic functions.

MEMORY MAPPED I/O

All accessing of our user port is through the 6526 chip. To access the user port we must be able to set and read bits on the 6526 chip registers. A register is a one byte (8-bit) memory location providing direct access to the 6526 chip. By placing specific numbers into the register we can program the chip to perform the functions we require.

The registers on the CIA chip are memory mapped. Meaning that the chips registers are located in the computers memory at certain addresses. This is an important concept. We can read or write the registers on the CIA from BASIC with simple peeks and pokes. We are control-

ling our interfacing device with the same BASIC commands peek and poke.

In order to program the registers on the CIA chip to perform the functions we require, we must be able to read and write to the 8 individual bits (one byte) that make up a register. This isn't as difficult as it may sound, but it does require a basic understanding of the binary number system.

BINARY

Binary means "based on two" as in two numbers, 0 and 1. Or like an electrical switch that has two values off (0) and on (1). In binary a digit is called a bit, which stands for "binary digit." A byte is a digital expression containing 8 bits.

The microprocessor used in the computers we're working with are 8-bit microprocessors. The registers we will read from and write to are 8-bit registers.

We will investigate the binary relationship controlling various I/O functions. All the information, however, is applicable to controlling other chips in Commodore computers. A fuller understanding of the binary number system can be acquired by reading any of the many fine books available on machine language.

Examine Fig. 2-2, for each progression of the binary "1" to the left, the power of 2 is increased by 1. These are relevant numbers, because

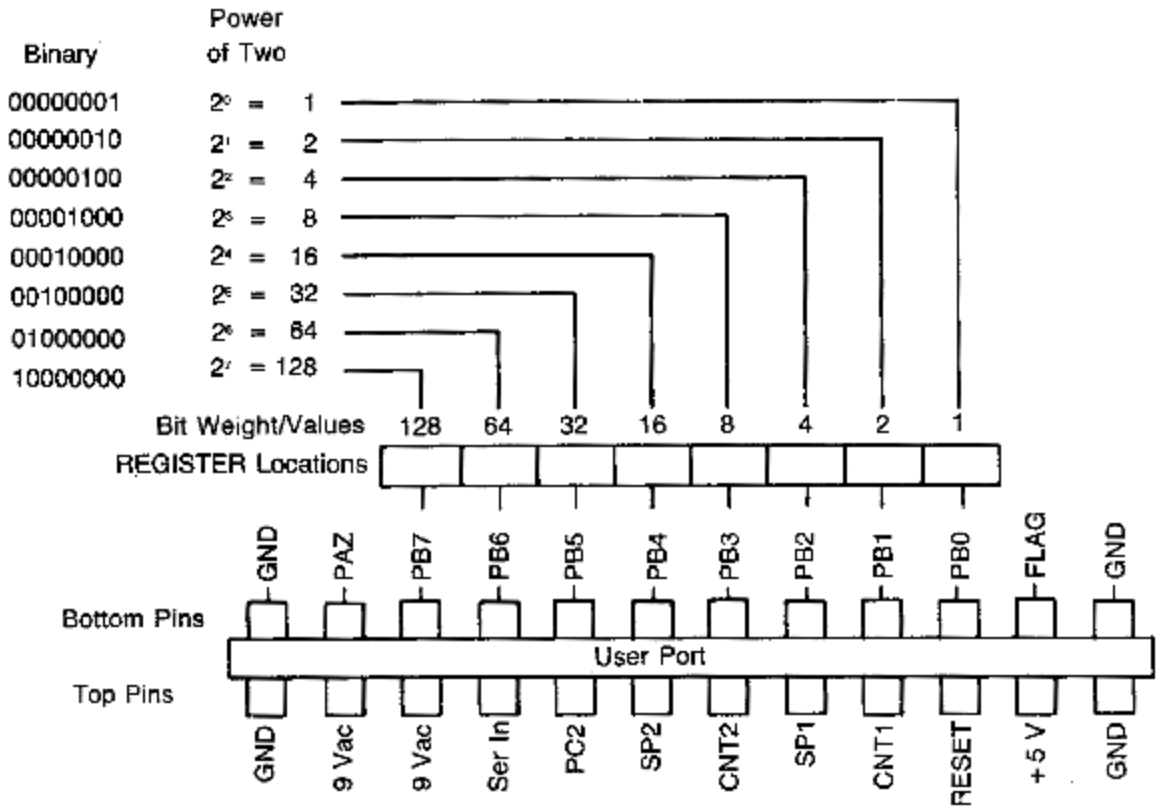


Fig. 2-2. Binary numbers relating to user port.

each progression identifies a bit location and weight. Notice the correlation between the user port lines, bit weights, and register location, we will be using this often. There is a one to one correspondence between bit number and the PB line. Bit 0 is equal to PBO, bit 1 is equal to PB1 on through to bit 7 is equal to PB7.

Examine Table 2-1. This table shows the binary numbers 0 through 20. This chart will come in handy later on. When a bit in the port is configured for input, the computer uses electrical voltages present at the pin/bit to determine whether that bit is set (on) "1" or clear (off)"0". Binary "1" can also be called "high" and a binary "0" called "low". A binary "1", is equal to a voltage level between 2 and 5 volts, and a binary "0", is equal to a voltage level between 0 and 0.8 volts. Voltages between 0.8 and 2 volts are undefined. When a bit in the port is configured as an output, the computer will output 5 volts.

DDR REGISTER

The DDR (data direction register) is a programmable register on the 6526 chip that controls the direction of the lines in Port B (input or output). The direction refers to whether you will be reading information off a line (inputting) or writing information to the line (outputting). A binary "1" placed at a bit location will turn that bit into an output bit. Conversely, a binary "0" will make that bit an input bit. Most interfacing tasks require a combination of input and output lines. In addition we have the ability to switch lines from output to input or input to output in the middle of a program.

The DDR occupies one byte in memory. The location of the DDR for Port B on the user port is 56579 for the C-64 and C-128, for the Vic-20 the location is 37138 (see Table 2-2).

Table 2-1 Binary Numbers.

Decimal	Binary	Decimal	Binary
0	= 00000000	11	00001011
1	= 00000001	12	00001100
2	= 00000010	13	00001101
3	= 00000011	14	00001110
4	= 00000100	15	00001111
5	= 00000101	16	00010000
6	= 00000110	17	00010001
7	= 00000111	18	00010010
8	= 00001000	19	00010011
9	= 00001001	20	00010100
10	= 00001010		AND SO ON
		255	11111111

Table 2-2. DDR Registers and PDR Registers.

Register	Computer	Memory Location
Data Direction	Vic-20	37138
Peripheral Data	Vic-20	37136
Data Direction	Com-64 & Com-128	56579
Peripheral Data	Com-64 & Com-128	56577

We use our bit weights (Fig. 2-2) to output binary 1's at the corresponding pins to create output pins. Any pins that aren't programmed as outputs automatically have 0's placed at their bit location and are configured as input pins. POKE 56579,20 would turn PB2 and PB4 into output bits, as PB0,PBl,PB3,PB5,PB6 and PB7 automatically became input bits. To see this more clearly, transfer the binary equivalent of 20 into the empty register location spaces on Fig. 2-2. The binary 1's are in PB2 and PB4 bit locations.

POKE 56579,3 makes PBO and PB1 output bits transfer the binary equivalent of 3 into the location spaces. Doesn't the number 3 in binary place binary 1's at the location of PBO and PB1? As you can see by pokeing this location with various bit weights we can configure any pin in the port to be an input or output bit in any combination we might require. Any unused bits can be ignored.

To summarize let's state by pokeing a binary 1 in the DDR corresponding to a bit, it turns that bit into an output bit. Likewise, pokeing a binary 0 will turn the bit into an input.

PERIPHERAL DATA REGISTER

Only after we have configured our port with the DDR, can we start using it. The user port lines are configured as inputs on power up, if this is the configuration you need you can begin immediately without setting up the DDR register. The peripherals data register (PDR) memory location 56577 is where we poke and peek to physically output or input (pull data) off the pins.

The procedure for the PDR is the same as described far the DDR. Only now when we output a binary "1" at a pin location a +5 volt signal will be present. (Provided we configured that pin as an output pin.) We will build a display circuit to demonstrate input and output procedures and techniques.

INPUT

Examine the diagram of the user port again. Beneath the user port are labels PB0, PBl, PB2, . . . PB7 corresponding to the pins on the user port. Under the port is the corresponding bit weight for each pin. Let's

configure all the bits on the port as inputs:

```
Poke 56579,0   Data Direction Register (DDR)
               Places binary 0's at all bit locations.
```

Now we apply +5 volts to pins PB2 and PB4. By applying the 5 volts to these pins we are inputting a binary 1 at each pin, if we then peek the port:

```
Print Peek (56577)   Peripheral Data Register
```

the number 20 would be returned. This is the added bit weights (4 + 16 = 20) of pins PB2 + PB4. Look at Table 2-2, transfer the binary equivalent of the number 20 into the bit locations on Table 2-1 it is the same. The binary 1's are in the same bit positions we inputted. If we applied +5 volts to just PB5 then peeked the port the number 32 would be returned. This is true for all pin/bit combinations.

OUTPUT

Let's reconfigure the user port so that all the bits are now outputs:

```
Poke 56579,255  Data Direction Register (DDR)
                Places binary 1's at all bit locations
```

Now poke the number 20 into the port.

```
Poke 56577,20   Peripheral Data Register
```

What do you think will happen? If you reasoned that +5 volts would appear on PB2 and PB4 you are right! By pokeing the number 20 into the port we are essentially outputting a binary 1 at those two pins.

It is important to understand that the voltage being outputted is a signal voltage and has very little power. It cannot be used to run a device. But by adding a simple circuit later you can use the signal to control most any electrical appliance you'd like.

If you feel a little confused at this point don't worry it will all come together very quickly once you gain_ some practical experience by utilizing and experimenting with the port. In order to do this you will need to build the demonstration circuit.

CIRCUIT CONSTRUCTION

Look at Figs. 2-1, 2-3, and 2-4. This is a simple circuit that we can use to experiment inputting and outputting information on the user port. Figure 2-3 is the same as Fig. 2-1. In Fig. 2-3 a pictorial rather than a schematic is used. This shows how the components are arranged on the

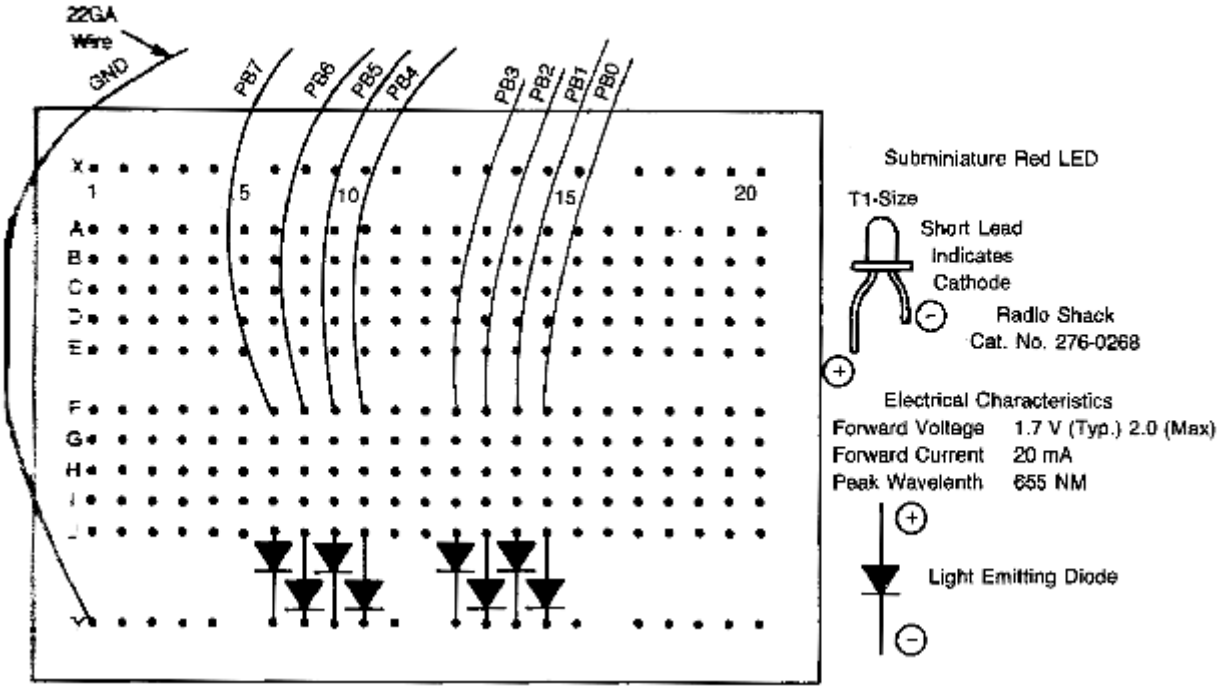


Fig. 2-3. User port wiring diagram.

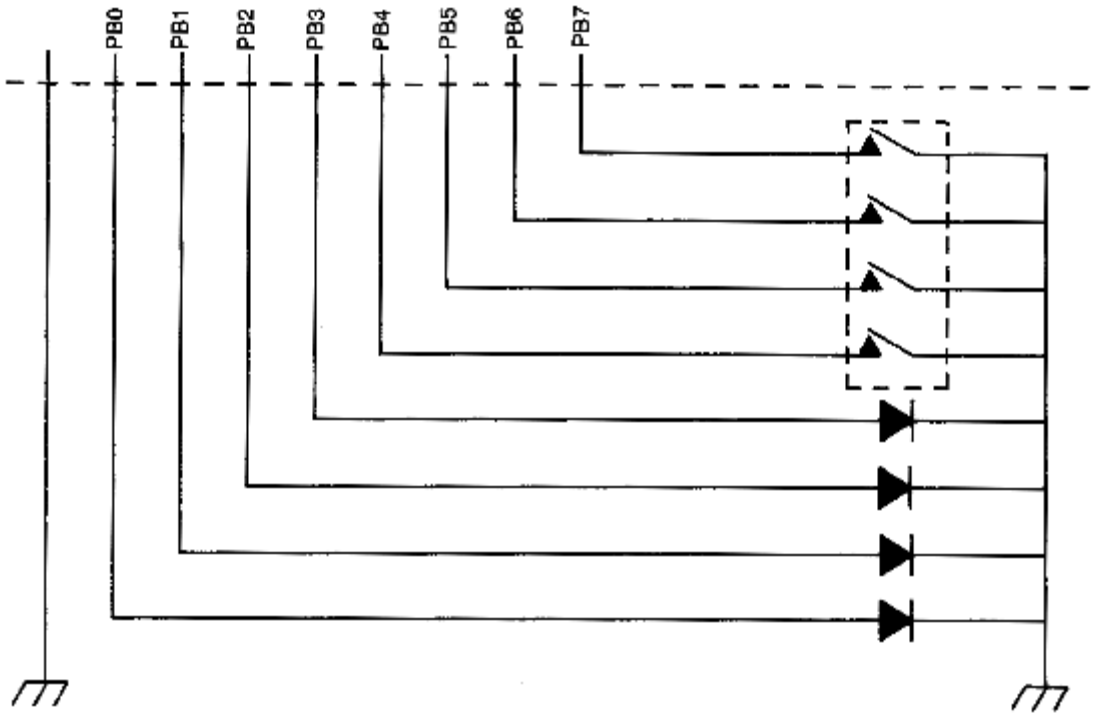


Fig. 2-4. User port connections.

prototyping breadboard. When wiring use Fig. 2-1. Most of the soldering has been eliminated by using our experimenters plug-in breadboard (see Fig. 2-5). By using this board we can simply plug in our components and control lines. This board also facilitates changing the circuit by being able to pull out the components and replacing them with different ones.

Most of the parts required are available at your local Radio Shack store, except for the 12I24 card connector. This is available from Mousser Electronics (see parts list). You can utilize a 44-pin connector that is available at Radio Shack by cutting off one end, leaving 12 pins.

Begin by soldering 22-gauge stranded wire to the card connector. Solder a wire to each PB line and to one of the grounds (see Fig. 2-6). If you have any difficulty matching what card connector terminal corresponds to the PB lines hold the card connector to the drawing of the user port (Fig. 2-1). The drawing is a rear view and is how the user port looks from the back of the computer.

Do not substitute the LEDs specified. The LEDs are microminiature, these were chosen because they don't require much power, and can be lit directly from the current available at the port. When you have completed the soldering and wired the circuit as shown (see Fig. 2-5), we are ready to continue. Turn off your computer (if it is on) and plug the card connector into the user port (see Fig. 2-7). Power-up your computer, all the LEDs should be dimly lit. If they are not, turn off the computer immediately, you've made a wiring error. .Recheck your wiring and make sure the LEDs are in properly, facing the right polarity. (The reason the LEDs are dimly lit is that, although the computer configures

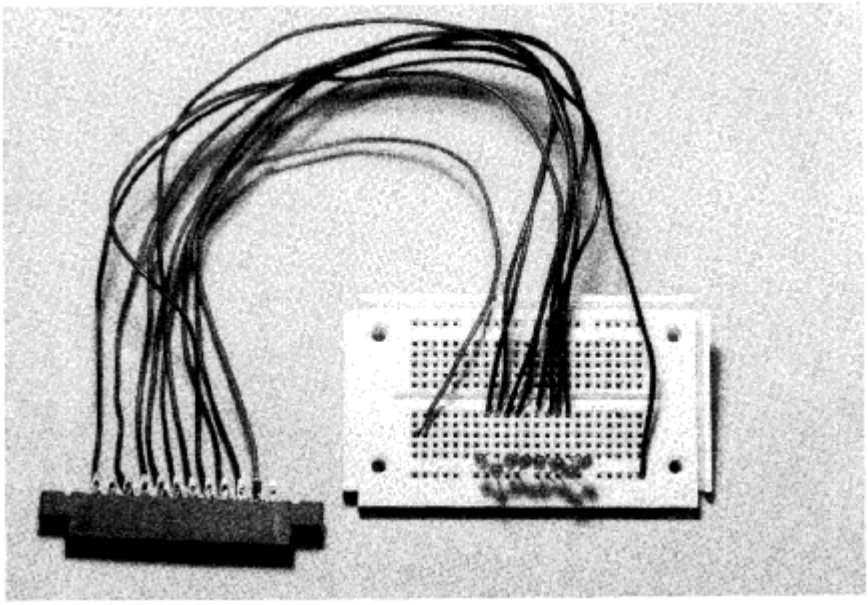


Fig. 2-5. Basic completed project.

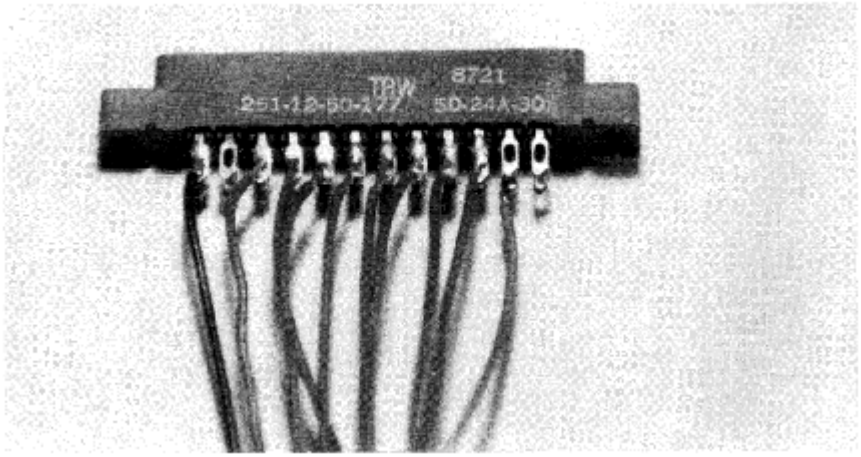


Fig. 2-6. Wiring for user port connector.

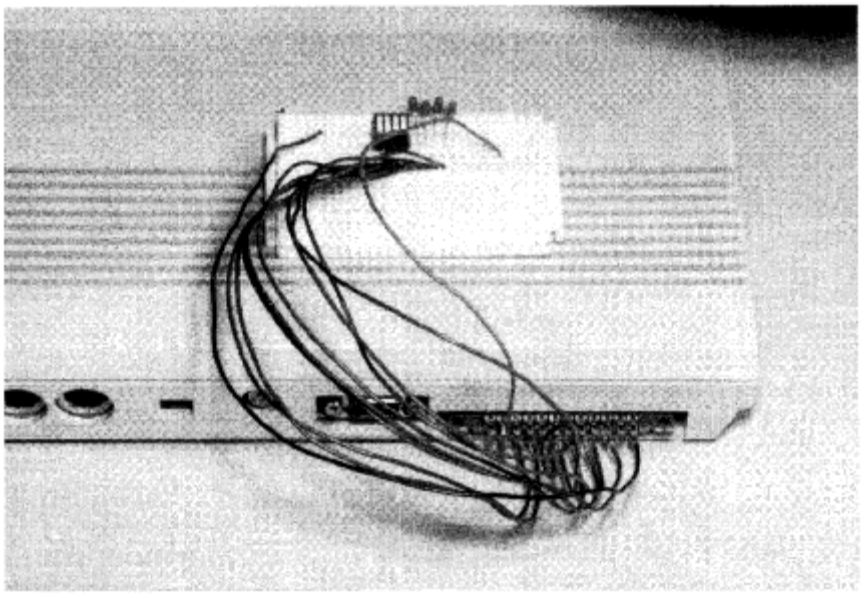


Fig. 2-7. Advanced project connected.

all bits as inputs on power up, the 6526 chip pulls the bits to 5 volts through a 10 k resistor. There is sufficient current to dimly light the LEDs.) If everything checks out you're ready to continue. Enter Command:

```
  Poke 56579,255  Set up DDR (data
                  direction register) This turns all the bits
                  into output bits
```

```
Poke 56577,20   peripherals data register lights
                LEDs connected to PB2 and PB4
Poke 56577,0    Turns off LEDs
```

In order to become familiar with the bit weights and their correlation to the pins, type in this simple program. Any number you input will light the LEDs corresponding to the bit weight.

```
10 POKE 56579,255
20 INPUT"[down 4) INPUT BIT WEIGHT ";BW
30 PRINT "[CLR], down 4 THE NUMBER " BW " IS
          BEING DISPLAYED IN BINARY ON YOUR INTERFACE"
50 POKE 56577,BW
60 GOTO 20
```

This second program will count in binary. To make it run faster or slower change the value of T accordingly. To count to a value less than 255 change X accordingly.

```
10 POKE 56579,255
20 FOR X=0T0255
30 POKE 56577,X
40 FOR T=1T0255; NEXT T
50 NEXT X
60 G0T030
```

We now have some experience outputting binary 1's. Now let's reconfigure the port. First turn off the computer. Remove the four LEDs connected to PB4 through PB7 and replace it with the 4 position dipswitch (see Fig. 2-4 and Fig. 2-8). Enter command:

```
Poke 56579,15   DDR set-up
                configures PBO through PB3 as outputs; PB4
                through PB7 as inputs
```

Turn all the switches on. Enter command:

```
Print Peek (56577) < return >
```

A "0" will be returned. Turn off the switch connected to PB4 and reenter command:

```
Print Peek (56577) < return >
```

Now the number "16" has been returned. You should know by now that the number "16" represents the bit weight for that pin. But the question

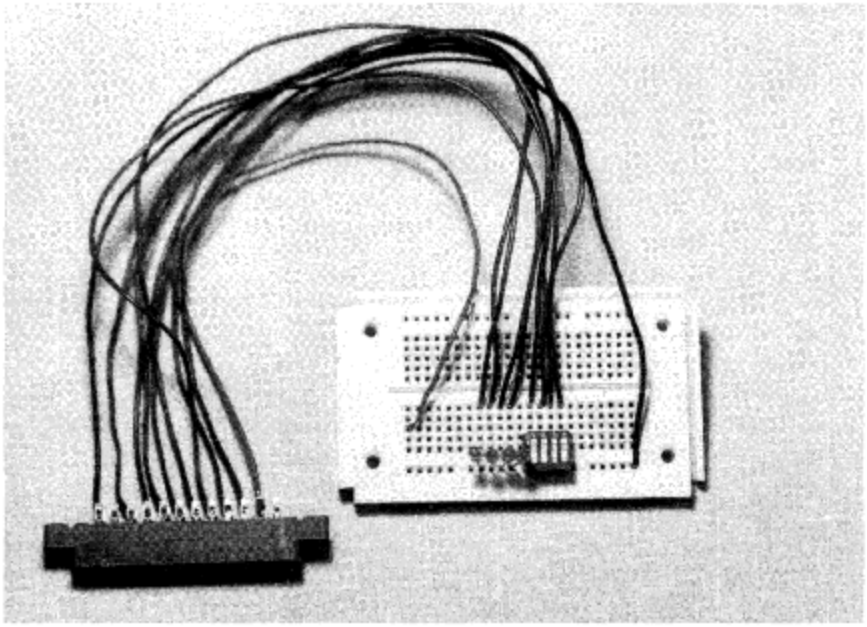


Fig. 2-8 Advanced project.

begs to be asked, "Why is it when you turn the switch off that the computer reads a binary "1" at the bit location?" The answer is the same as why the LEDs are dimly lit on power-up. When our switch is turned on it is connected to, and pulled down to ground (0 volts). When the switch is turned off, the line is pulled up to +5 volts (binary 1) by the 6526 chip though a 10 k ohm resistor. To become more comfortable with inputting enter the following program:

```
10 Poke 56579,15
20 D=Peek (56577)
30 Print "clr/home 7 right,7 down" D "This
        is the bit weight of your interface"
40 GoTo2O
```

By turning various switches on and off, the bit weights are displayed on the screen. Let's do something a little interesting press the RUN/ STOP key and enter: Poke 56577, 5. The LEDs connected to PBO and PB2 are lit, enter RUN notice the bit weight of 5 is added to the display.

Now let's try something a little more interesting; enter the following program. When bit 8 (dipswitch connected to PB7) is turned on, the LEDs connected to PBO to PB3 will start counting. No other bit has any effect. This is accomplished by masking all bits except bit 8, and reading its bit weight.

```
10 Poke 56579,15
20 For x=0to15
30 D=Peek(56577)ANDl28
40 If D=OGoTo3O
50 Poke56577,X
60 For T=1to100:nextT
70 NextX
80 GoTol0
```

To fully understand this we must look at some logic instructions. You may skip this section if you feel it's too difficult, go directly to the circuits. But do come back after you feel comfortable with all the other material presented. This logic section will enrich your ability to design and configure the user port to your needs with the minimum amount of instructions.

LOGIC

There are two logic instructions available to us from BASIC, that can be used to set (binary 1) or clear (binary 0) specific bits on the port without affecting the other bits. They are " AND " and " OR " instructions.

When using these instructions, we are comparing the number in the register to the number we PEEK or POKE in the register. The results , can be used to make useful decisions and perform functions. For each set of bits compared there are four possible combinations.

0	0
0	1
1	0
1	1

Refer to Table 2-2 to see the results of these two instructions. By studying the table, two conclusions can be drawn. The results of an "AND" instruction is "1" only if both bits are "1", otherwise the results are "0" . The results of an "OR" instruction is "0" only if both bits are "0", otherwise the results are "1". Our computer uses eight bit binary numbers, examine the following examples.

AND		OR	
11010011	register	11010011	register
10000101	AND 133	10000101	OR 133
10000001	RESULT	11010111	RESULT

In the fourth program we used the "AND" instruction to test a bit. Then made a decision based upon the results. Let's analyze how the program accomplishes this.

```
30 D=Peek(56577)ANDl2B

OXXXXXXX    register   X= any Value 0 or 1
_________   AND 128
00000000    RESULT     PrintPeek(56577)ANDl28=0
```

This instruction compares the AND 128 with the number in the register. The only bit that can have an impact on the result is bit #7. Because all other bits are "AND" with 0, their results are 0. By setting bit #7 we have the following scenario.

```
lXXXXXXX   register    X = any value 0 or 1
10000000   AND 128

_________
10000000   RESULT      PrintPeek(56577)ANDl28=128
```

With these two possible results, we cari use a familiar basic decision command:

```
40 If D=0 Then GoTo 30
```

SETTING A BIT WITH "OR"

We can use an "OR" instruction to set various bits. It is very useful when we wish to set specific bits without disturbing the status of the other bits on the port. Examine Table 2-2, any number that is ' `OR" with a "0" remains unchanged. Therefore, if we wish to set bit 4, we can "OR" bit 4 with a binary "1" as our example illustrates.

```
10100010   register
00001000   OR 8    Command=Poke56577,Peek(56577)OR8

_________
10101010   RESULT
```

Try entering the following commands to get a better understanding.

```
Poke 56579,15              DDR Setup
Poke 56577,3               lights LEDs to PBO and PB1
Poke 56577,Peek(56577)OR8   Sets bit 4 on without disturbing
                            the status of the other bits
```

When we enter our last command the status on the interface has the two LEDs lit. This status remains unchanged as we set bit four on, as indicated by the lit LEDs.

ELECTRONIC LOGIC

We have electronic circuit ICs that perform logic operations with voltages. Fi re 2-9 shows four common logic circuits. Each line input on the IC behaves as a bit. As before a binary "1" is +5 volts and a binary "0" is 0 volts. As you can see the logic results are identical to the computer's internal logic functions.

There are other interesting bit manipulations, I advise you to experiment on your own by purchasing a couple of logic ICs, setting them up on your experimenter's board and testing their operations. Use a VOM or logic probe to check the inputs and test the outputs. A small battery

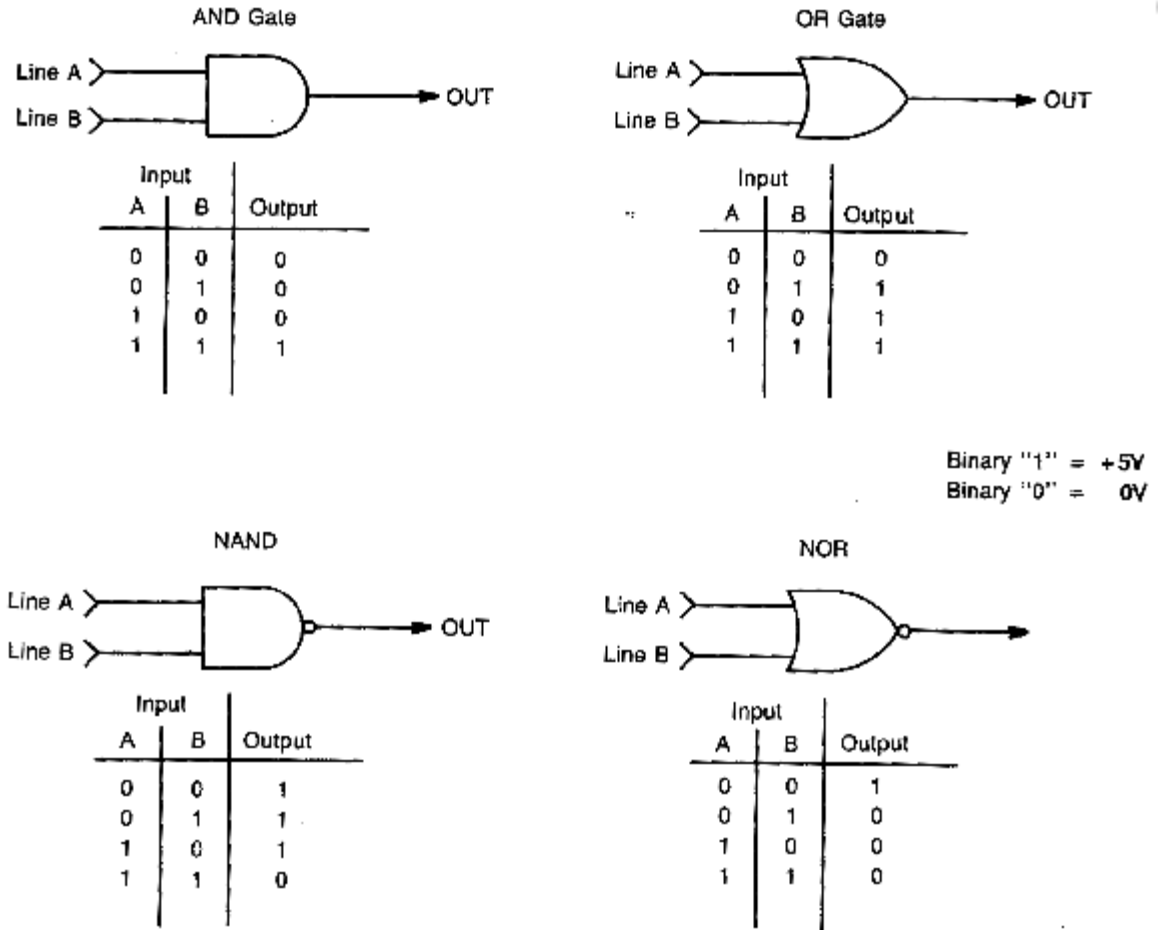


Fig. 2-9. Logic gates.

pack or power supply can supply the +5 volts for the binary 1's, ground the inputs for binary 0's. You can find logic IC's in Radio Shack stores.

Parts List

Quantity	**Item/Description**	**Part Number**
		RS = Radio Shack
8	Subminiature Red LED	RS# 276-0268
1	Experimenters Breadboard	RS# 276-175 or RS# 276-174
1	12/24 Card Connector	568-50-24A-30

Mouser Electronics
11433 Woodside Ave.
Santee, CA 92071
(619) 449-2222

3

Speech Synthesizer

In this chapter we will begin to apply what we learned in the last chapter with the practical application of synthesizing speech.

SPEECH SYNTHESIS

Speech synthesizers (or processors) appear in two main formats. One approach (format #1) uses digitally recorded speech stored in a ROM chip. The second approach (format #2) uses phonemes of English to construct words and sentences (a phoneme is a speech sound).

The main advantage in Format #1 is a excellent speech reproduction and fidelity. Its main disadvantage is a limited vocabulary of English that's been preprogrammed into the chip.

Format #2's strength is an unlimited user defined vocabulary. Its disadvantage is that the speech fidelity isn't as good as with the preprogrammed speech ROM. Even so, the speech fidelity of format #2 is quite acceptable in all but the most critical circumstances. We are taking this second approach to speech synthesis.

The speech synthesizer we will build, plugs into, and is powered by the user port. The cost is less than $25.00, and for that price it includes its own audio amplifier, filter, volume control, and speaker. Since it has an unlimited vocabulary, you can program any word you desire. You have the option to either modify existing programs to include speech, or, of course, to write new programs with speech.

THE SPEECH CHIP

General Instruments Company manufactures the 28-pin speech synthesizer chip (5P0256-A12) that is distributed by Radio Shack. This chip can generate 59 allophones (speech sounds) and five pauses (no sound) of various lengths (see allophones, Table 3-1). By adding(concatenating) allophones together, you can construct words and sentences. This may sound rather difficult at this point but it is not, the program does most of the work.

Table 3-1. Allophones.

Decimal Address	Allophone	Sample Word	Duration	Decimal Address	Allophone	Sample Word	Duration
0	PA1	Pause	10MS	32	AW	Out	370MS
1	PA2	Pause	30MS	33	DD2	Don't	160MS
2	PA3	Pause	50MS	34	GG3	Pig	140MS
3	PA4	Pause	100MS	35	VV	Venom	190MS
4	PA5	Pause	200MS	36	GG1	Gotten	80MS
5	OY	Toy	420MS	37	SH	Sharp	160MS
6	AY	Buy	260MS	38	ZH	Azure	190MS
7	EH	End	70MS	39	RR2	Train	120MS
8	KK3	Cat	120MS	40	FF	Forward	150MS
9	PP	Power	210MS	41	KK2	Sky	190MS
10	JH	Judge	140MS	42	KK1	Came	160MS
11	NN1	Pin	140MS	43	ZZ	Zolu	210MS
12	IH	Sit	70MS	44	NG	Anchor	220MS
13	TT2	To	140MS	45	LL	Lamb	110MS
14	RR1	Rural	170MS	46	WW	Wood	180MS
15	AX	Succeed	70MS	47	XR	Pair	360MS
16	MM	My	180MS	48	WH	Whine	200MS
17	TT1	Tart	100MS	49	YY1	Yes	130MS
18	DH1	They	290MS	50	CH	Chump	190MS
19	IY	Tee	250MS	51	ER1	Tire	160MS
20	EY	Beige	280MS	52	ER2	Tire	300MS
21	DD1	Should	70MS	53	OW	Beau	240MS
22	UW1	To	100MS	54	DH2	They	240MS
23	AO	Aught	100MS	55	SS	Best	90MS
24	AA	Home	100MS	56	NN2	Not	190MS
25	YY2	Yes	180MS	57	HH2	Noe	180MS
26	AE	Pat	120MS	58	OR	Pore	330MS
27	HH1	Him	130MS	59	AR	Arm	290MS
28	BB1	Boy	80MS	60	YR	Clear	350MS
29	TH	They	180MS	61	GG2	Guide	40MS
30	UH	Book	100MS	62	EL	Paddle	190MS
31	UW2	Food	260MS	63	BB2	Boy	50MS

A LITTLE ON LINGUISTICS

An allophone is the computer equivalent to English phonemes (speech sounds). There are two main points you should keep in mind when you are programming new words. First, in English there isn't a one to one correspondence between letters and sounds. This point is amply demonstrated by the younger members of our society who are learning to read and write. They are likely to spell cat as Kat and phone as *fone*, imitating in writing the way the words are pronounced. This is a very interesting point, because in order to program words to sound correct, you must spell the words phonetically. More about this later, let's continue on to the second point. Placement of a speech sound in a word can change the pronunciation. As an example, take a look at the two D letters in the word depend. The D's are pronounced differently. If we were to program this word using our table of allophones, the allophone DD2 would sound correct in the first position (Depend) and the allophone DD1 sounds correct in the second position (depend). We will return to programming technique later on. A booklet with more information on linguistics, allophones, and usage is included with the speech synthesizer chip.

CIRCUIT CONSTRUCTION

The circuit is comprised of two sections (see Fig. 3-1), separated by a dotted line. Section A on the left is the basic circuit: Section B contains the amplifier, low-pass filter, volume control and speaker added to the basic circuit.

The two sections A and B make up a stand-alone unit that only requires power and control signals from the user port to function. In contrast to section A of the circuit, which requires the use of the Sound Interface Device (SID) chip and a monitor or TV speaker.

By utilizing the SID chip in the C-64 or C-128 computer, you can eliminate section B, the audio amplifier, filter, volume control, and speaker (see Fig. 3-2), reducing the amount of parts required by more than half, simplifying the circuit, and saving a couple dollars. However, if you're using a Vic-20 computer you will have to build the entire circuit (see Fig. 3-3).

The C-64 and C-128 can use either section A, or the entire circuit. To use just section A, we eliminate section B and take the output of the circuit (at pin 24) and input the signal to the SID chip. We accomplish this with a wire to the "audio in" pin of the composite video connector (see pin 5 of the C-64 and C-128). Pin 24 is the digital output of the speech synthesizer chip. You can purchase the correct din plug for your computer or use a short wire pushed into the correct pin socket connected by a jumper wire to pin 24. You can use our experimenters breadboard for this circuit. See Fig. 3-2. Plug in your components as diagrammed and you're ready to begin programming.

For the Vic-20 I constructed the entire circuit on a modified experimenters card (see Fig. 3-3). The card is modified by cutting the end terminals on both sides leaving the center 12 positions. Use a 12I24 card connector and solder the lugs on the connector to the fingers on the board. If a 12I24 card connector isn't readily available you can modify a 22-position card connector into a 12-position connector by cutting 10 positions off as I have done. Only 10 connections are needed for this project. I did, however, solder all the connections to improve the mechanical strength of the unit. Pin 24 is connected into the B section circuit through a low-pass audio filter to a 10 k volume control pot. User either a trimmer pot that you can set once and forget about or eliminate the pot completely. The volume of sound with the pot removed isn't so great as to be objectionable. You'll probably use the speech synthesizer with the pot fully closed anyway.

Power for the entire unit is provided from the top side of the user port. The bottom side (Port B) accesses and controls the speech processor. If in wiring you get confused tracing the leads from the user port to the speech chip, I suggest holding the card connector (or experimenters board) to the diagram of the user port. This will help match where each wire connects. The diagram of the user port can be used this way because it shows how the user port appears when looking directly into it from the back. When completed the card connector plugs into the user port.

The manufacturer of the speech chip recommends using a 3.12 MHz crystal at pins 27 and 28. I recommend using a 3.57 MHz colorburst crystal instead. The reason is cost and availability. The 3.57 MHz colorburst crystal is approximately 1/4 the cost of the 3.12 MHz crystal, and is more readily available. This change will increase the timbre of the speech slightly, but has no other effect on circuit operation.

THE PROGRAM

Type in the program (see Fig. 3-4). Assign a value to "PB" in line 60 depending on which computer you are using.

For the Vic-20: PB = 37136
For the Com-64: PB = 56577
For the C-128 : PB = 56577

If you are using a C-64 or C-128 with section A only, type in this additional line:

```
55 S=54272:F0RL=0to24:POKES+L,0:NEXT:POKES+24,15
```

When the program is run, the computer should say hello. Adjust the trimmer pot if you built the entire circuit, and have included the pot in the circuit.

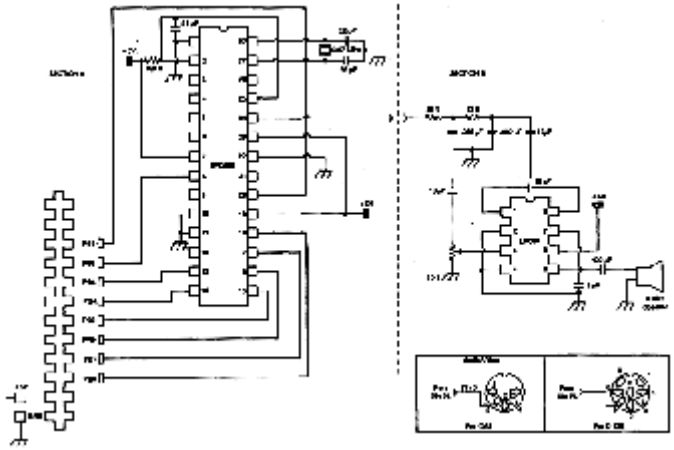
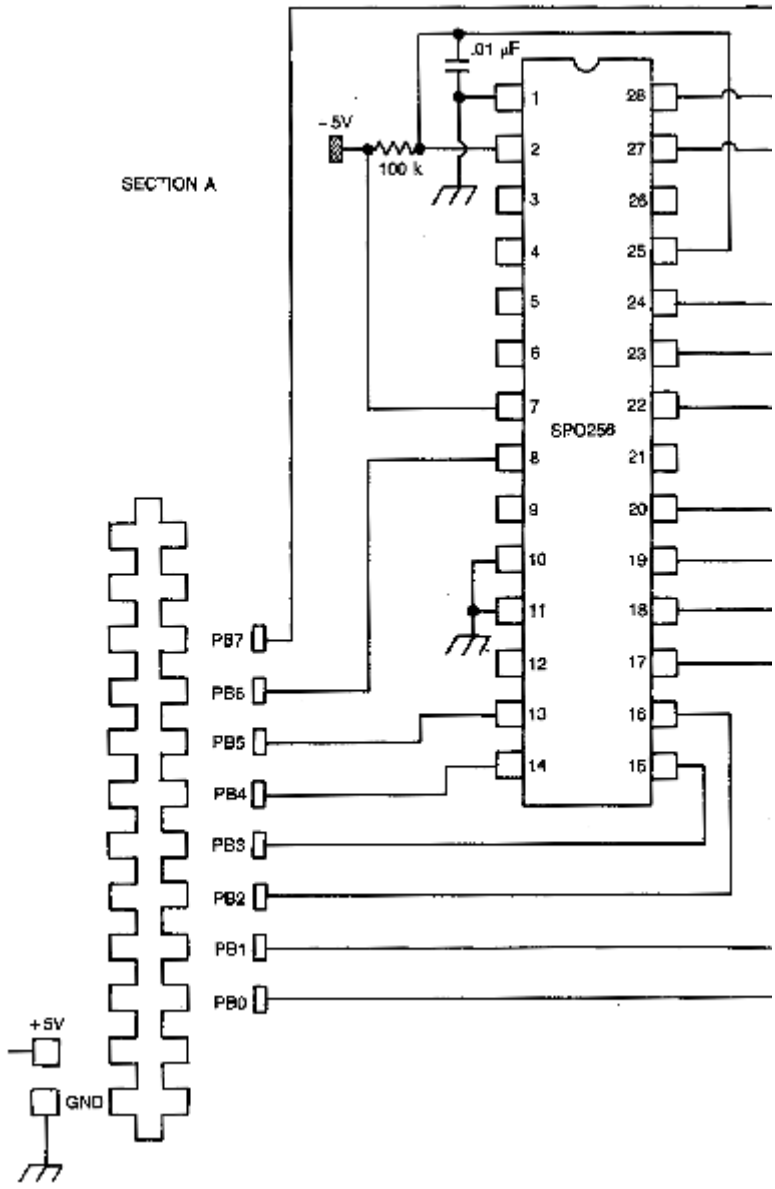
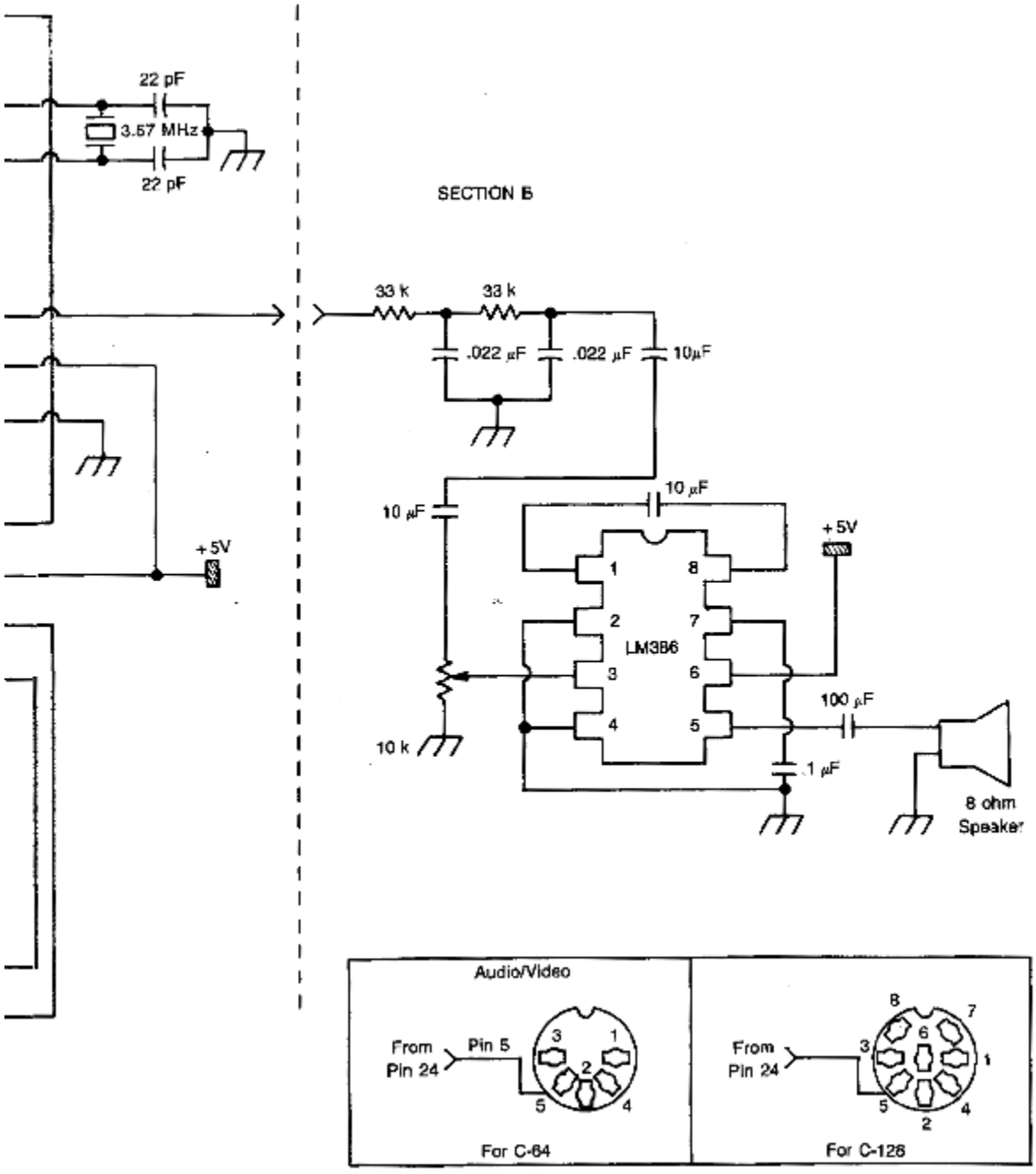


Fig. 3-1. Speech synthesizer sections A and B.




22 pF
3.57 MHz
22 pF
SECTION B
33 k
33 k
.022 µF
.022 µF
10µF
10 µF
10 µF
+5V
+5V
1
8
2
7
LM386
3
6
4
5
100 µF
.1 µF
10 k
8 ohm
Speaker
Audio/Video
From
Pin 24
Pin 5
3
1
2
5
4
For C-64
8
7
6
From
Pin 24
3
1
5
4
2
For C-128

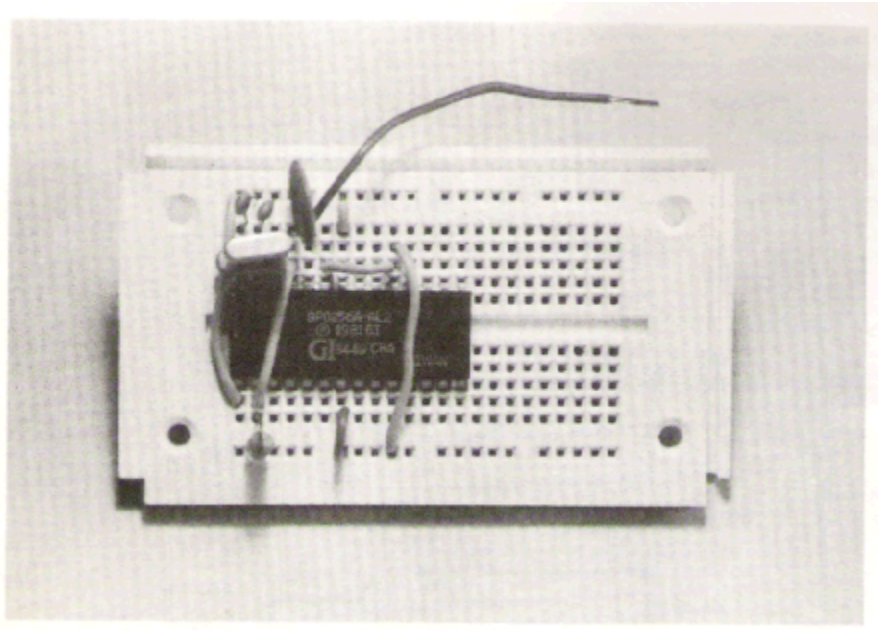


Fig. 3-2. Section A.

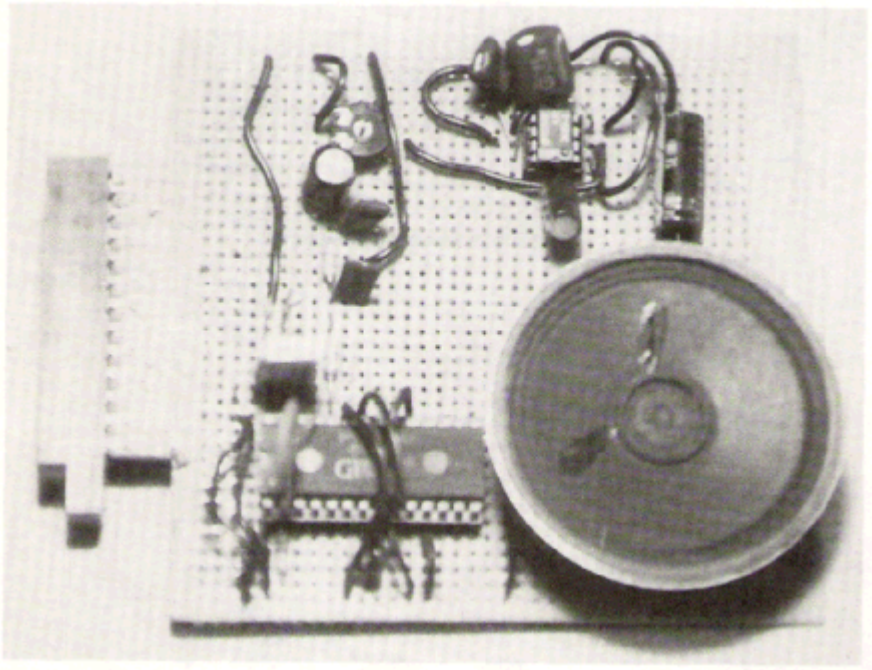


Fig. 3-3. Sections A and B on PC board.

```
10 REM ****** SPEECH PROCESSOR *****
15 REM BY JOHN IOVINE DATED 4-16-86
20 REM ****************************
25 REM ** VIC-20          PB=37136 **
30 REM ** C-64 & C-128    PB=56577 **
35 REM ****************************
50 REM SET UP DDR AND TABLE
60 PB=56577
65 POKE PB+2,191
70 DIMM$(63)
75 FORI=0TO63
80 READM$(I)
85 NEXTI
90 REM *** DATA TABLE ***
91 DATAPA1,PA2,PA3,PA4,PA5,OY,AY,EH,KK3
92 DATAPP,JH,NN1,IH,TT2,RR1,AX,MM,TT1
93 DATADH1,IY,EY,DD1,UW1,AO,AA,YY2,AE
94 DATAHH1,BB1,TH,UH,UW2,AW,DD2,GG3,VV
95 DATAGG1,SH,ZH,RR2,FF,KK2,KK1,ZZ,NG
96 DATALL,WW,XR,WH,YY1,CH,ER1,ER2,OW
97 DATADH2,SS,NN2,HH2,OR,AR,YR,GG2,EL,BB2
99 REM *** END OF DATA TABLE ***
150 REM *** SPEECH MODULE ***
151 REM HELLO
152 DATA HH1,EH,LL,AX,OW,PA1
153 LETG=G+1:IFG=7THEN157
154 READ A$
155 GOSUB10000
156 GOTO153:REM RETURN TO COUNTING LINE
157 G=0:REM RESET G
158 REM CONTINUE MAIN PROGRAM
159 PRINT"{CLR}"
160 PRINT"PROGRAM WOULD CONTINUE HERE"
161 PRINT"TYPE RUN THEN PRESS RETURN"
162 PRINT"TO HEAR AGAIN"
163 END
10000 FOR I=0TO63
10005 IF M$(I)=A$THEN10050
10010 NEXT I
10015 PRINT"ERROR IN DATA STATEMENT":POKE56577,0:END
10050 IF PEEK(PB)=128THEN10050
10055 POKE PB,I
10060 POKE PB,128
10065 RETURN
```

Fig. 3-4. Speech synthesizer program.

If the computer fails to speak, you have either a typing error in the program or a wiring error in the circuit. Check over the program to see that you entered it correctly. Recheck your wiring. If everything checks out, verify circuit operation by checking for clock pulses at pins 27, 28, and 24. If you show pulses the problem is in the audio section.

Although it isn't necessary to understand how the program operates to use it. Here is a brief description:

STEP 1 Lines 60 to 100:	Sets up Data Direction Register and allophone table
STEP 2 Lines 150 to 157:	Speech Module-reads speech in program
STEP 3 Lines 10000 to 10065:	Subroutine sends instructions to speech chip and returns

Until you gain some experience and feel comfortable designing your own speech program. STEP 2, lines 150 to 157 is the model to use to program speech in your basic programs. We will do a line by line analysis and an example will ensure a good understanding of the procedure.

Line 151	Is a REM statement labeling the word or phrase contained in the following data statements. This is useful in the event you wish to correct, change or eliminate words. By clear labeling you can locate the word quickly.
Line 152	Data statement; containing allophones for the word hello.
Line 153	Counting line; enabling the computer to read the proper number of allophones in the data statement then jump to the end of the speech module upon completion. G=(number of allophones)+1. Therefore if our word uses 6 allophones then G=7. If a sentence uses 31 allophones, then G=32.
Line 154	Reads allophones in data statement.
Line 155	Takes data read; jumps to subroutine line 10000. There computer compares to table, decodes and provides necessary electrical pulses to the speech chip; returns.
Line 156	Return program to counting; incrementing G; reading the next allophone. Process is repeated until G equals its assigned value.
Line 157	Line number called in line 153, C then 157 > -Resets G to 0, enabling G to be used again in other modules of the program.

Example

The booklet provided with the chip has a dictionary with over 200 words, with their proper allophone data. These words can be put into your program at once. Some of the words included are:

- numbers 0 to 1 million
- days of the week

- months
- letters
- common words

For our example we will construct a sentence concatenating 4 words and entering it in our program.

```
200 Rem See You Next Tuesday
201 Data SS,SS,IY,PA1
202 Data YYl,UWl,PA1
203 Data NNl,EH,KKl,SS,TT2,PA1
204 Data PA1,TT2,UW2,ZZ,PA2,DD2,EY,PAl
```

The REM statement describes what is contained in the following data statements. You can use or start with any line number you'd like, just remember to be consistent.

```
205 Let G=G+1: If G=22 Then 225
```

Count the allophones in the above data statements. You should count 21 allophones. Since G=(# of allophones)+1 therefore G=22. Note line number 225 call out <Then 225 > in this line, it marks the end of the speech module. You can easily predict this number since it is always 4 lines down from this line.

```
210 Read A$
```

Reads allophone in data statement.

```
215 GoSub 10000
```

Program goes to subroutine at line 10000.

```
220 GOTO 205
```

When program returns from subroutine this line returns program to line 205 the counting line. G is incremented, next allophone is read, until G equals its assigned value.

```
225 G=0
```

This is the line called when G reaches its assigned value. This line resets G to 0 so it can be used again for other speech modules.

The allophone table correlates each allophone with its approximate sound. This table is essential for programming words that aren't in the provided dictionary. Please be aware that there are a few typographical

errors in the dictionary. Such as in the following words:

Hello-HH,EH,LL,AX,OW,AW,ER1

AND

Computer-KK1,AX,MM,PP1,YY1,UW1,TT2,ER

In the word hello; the first allophone hh doesn't exist in the table. You should use HH1, or the word will sound like *ello*. In the word computer; the last allophone *er* doesn't exist. You must use ER1 or the word will sound like *compute* not computer.

If you use a word from the dictionary that doesn't sound correct, first check the allophones to see if there is a typo. Always end a word or phrase with one of the pauses PA1 to PA5. This is necessary to stop the computer from enunciating the last allophone.

BASIC CRUNCH

BASIC can run the speech processor, but it is a little slow. One of the easiest ways to bring BASIC up to speed is to use multiple statement lines and eliminate all unnecessary REMs. Effective programming has been known to help also. Experiment by crunching the program as much as possible. I would do this one step at a time or you stand a good chance of crashing.

Machine language is very quick and ideal to use with the circuit. If you're a machine language programmer here is a great opportunity to test your mettle. I advise running an ML wedge and implementing a new BASIC command such as "say" or "speak" that would completely eliminate all BASIC programming.

A good in between program could be implemented using a binary jump search routine for the data comparison after the data read, as this is the most time consuming portion of our program.

Conclusion

You now have the tools you need to program speech. To utilize the basic program into an existing program or to help organize a new program, think of the program as existing in three distinct modules; the data table, speech module, and speech routine. Set up the data table before it will be used by the program. Put the subroutine for speech near the end of the program. The speech modules are placed anywhere in between, where you want the computer to speak.

Examine the speech routine section of the program, with the knowledge and information given in the last chapter you should be able to figure out how this program is operating. If you have any problem you may want to place your LED interface at the user port and run the speech synthesizer program to observe the controlling bits, it'll definitely help.

What has been written in this installment is the bare essentials. Feel free to experiment and develop your own program.

For those of you who are more adventurous, Radio Shack sells a companion chip that's an ASCII text to speech converter. Practical applications for the chip are a text reader or verbal modem.

Parts List

Quantity	**Item Description**	**Part Number**	**Cost**
1	SP0256-AL2	RS# 276-1784	$12.95
1	LM386	RS# 276-1731	1.09
1	100k ¼ watt	RS# 271-1311	.39
2	33k ¼ watt	RS# 271-1341	.39
3	.1 µF Cap	RS# 272-135	2 @ ·49
2	.022 µF Cap	RS# 272-1066	4.69
2	22pF Cap		
2	100 µF Cap	RS# 272-1016	2 @ .79
1	10k Trimmer Pot	RS# 271-335	.49
1	8 ohm speaker	RS# 40-245	1.89
1	1 µF Cap	RS# 272-996	.79
1	10 µF Cap	RS# 272-999	.99
1	Experimenters Board	RS# 276-168	1.95
1	Card Connector 12/24 or Module	RS# 276-1551	2.99
1	3.57 MHz Crystal	RS# 272-1310	1.69

All parts are available from Radio Shack.

4

Analog to Digital Conversion

We are going to examine analog to digital conversion with a serial input to the user port. Then we will add a 60 Hz interrupt routine to have our project work continuously and transparently in the background of BASIC. This may sound a little complicated right now, but we shall take it one step at a time.

With these tools at your disposal you will be able to utilize your computer to monitor the real world environment. The real world environment is where we live, to the computer it's everything external to its circuit board. To whet your appetite, this is a list of projects we will cover in this chapter.

- Biofeedback monitor
- Transducers-light and heat
- Toxic gas detector

Before we begin, examine Fig. 4-1 to refresh our memory of the basic definition of binary signals. A binary "1" is equal to approximately 5 volts, a binary "0", approximately 0 volts. There are basically two types of interfacing schemes commonly used in personal computers today, serial and parallel.

Previously, we used parallel interfacing without explicitly stating so. Since you already have a working knowledge of parallel interfacing we'll begin with this, and then move on to serial interfacing.

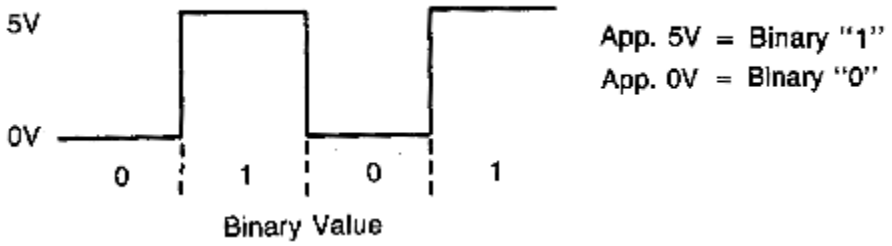


Fig. 4-1. Binary voltages "1" and "0".

Parallel interfacing transmits or receives eight data bits (Fig. 4-2) simultaneously on eight parallellines called a data bus. As we have seen when using Port B of the user port, we have the added advantage of being able to configure a combination of input/ output lines on our 8-bit parallel port.

Figure 4-2 details our PB lines off the user port. To read the binary number you add the decimal values represented by each bit. The binary

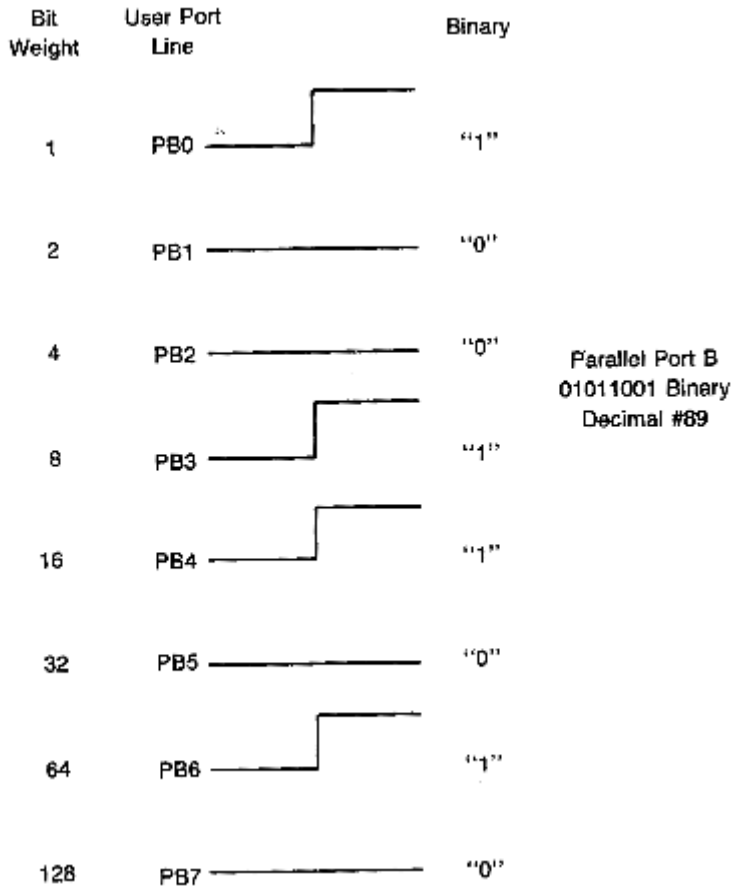


Fig. 4-2 Parallel 8-bit voltage to binary conversion.

number in this example is 10011010. Translating this binary number you would obtain a decimal value of 89. (1 + 0 + 0 + 8 + 16 + 0 + 64 + 0)

Figure 4-3 shows how the same information can be transmitted or received over a serial line. A serial bus need only consist of two lines. As its name implies the information is transmitted serially one bit at a time. The first bit transmitted or received is bit 7. The clocking line correlates the precise moment to receive or transmit data on the line.

Commodore computers have a built in serial register and clocking line that can receive or transmit data in such a fashion. This greatly simplifies our programming task.

ANALOG EVENTS

What is an analog event? This may appear to be an easy question to answer. We deal with analog events everyday. Such as time; temperature, speed. To define briefly, an analog event ,is one in which the reading or measurement is infinitely variable between any two points. Let's examine one example, the voltages existing between 1 volt and 2 volts. The possible number of voltage readings between these two points is infinite, it can have virtually any value, such as 1.1 volts or 1.00000001 V. or 1.0000000000000000001 V. As you can see, voltages can vary by infinitesimal amounts. The same is true for temperature, time, gravity, and a number of other natural phenomenon.

DIGITAL EVENTS

Digital events occur in discrete predefined steps. A simple example is an electric light switch that has two predetermined states, on or off. A rising voltage digitally plotted against time, would not trace as a straight line (analog event), but would jump in increments in a staircase fashion. See Fig. 4-4.

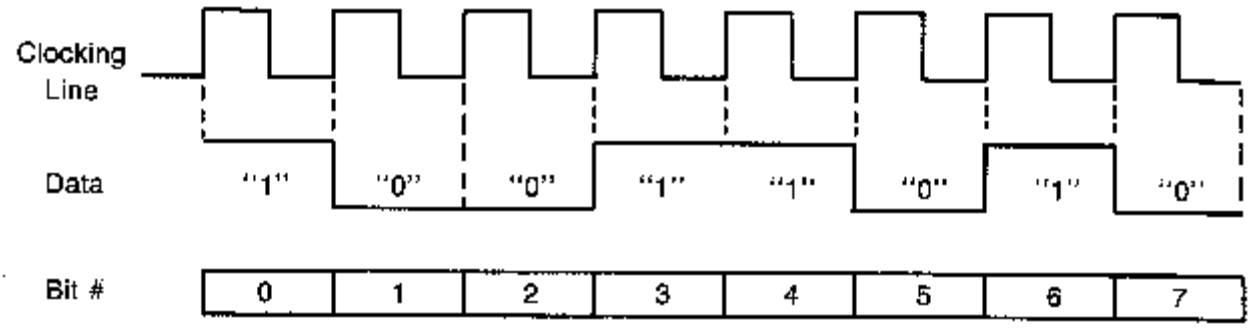


Fig. 4-3. Serial 8-bit binary number.

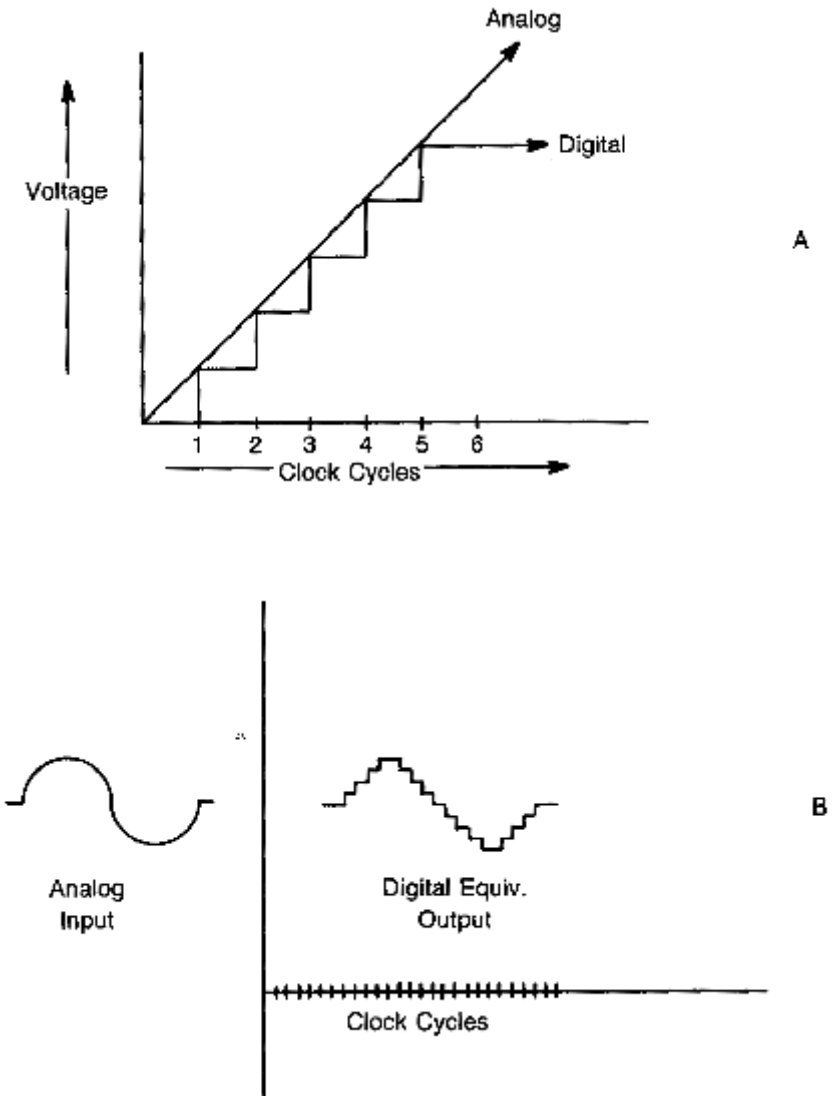


Fig. 4-4. Analog to digital and sine wave conversion.

SERIAL A/D CONVERTER CHIP

We will use an off the shelf serial analog to digital converter, available at Radio Shack.

An analog to digital converter does exactly what its name implies. It reads an analog voltage then converts it to the proportional digital (binary) value for use by the computer. In our case this digital value is transmitted serially into the computer (see Fig. 4-3).

Radio Shack sells a serial A/D (analog to digital or ADC) converter chip manufactured by Texas Instruments for $6.95 (see parts list) (see Fig. 4-9). This is an 8-pin chip that is extremely easy to interface to our user port. Some of the chips capabilities are as follows; max 40,000 samples

per second, internal clock and 8-bit conversion resolution. (See AID chip drawing and pinout description.)

We will utilize this chip extensively. To interface, we must solder additional lines on our card connector. These lines are the +5V, serial line and clocking line. See Fig. 4-5 for the Commodore 64 and C-128 (SP-2) is the serial and (CNT-2) clocking line we'll use. For the Vic-20, the serial line is CB2 and clocking line (CB1), (see Fig. 4-6).

Construct the circuit on your experimenters breadboard. A 10 k potentiometer is inserted between the +5 volt line and ground (pins 1 and 4 (see Fig. 4-5 or Fig. 4-6 and Fig. 4-8). The wiper of the pot is connected to pin 2, analog input of the A/D chip. This is a testing pot, (see Fig. 4-7) for you to test the circuit and the program.

TEST PROGRAM

Type in the respective program (Fig. 4-10 for C-128 and C-64 or Fig. 4-11 for Vic-20) for your computer and run. Vary the control knob on the pot and observe the results on the screen. The numbers represent the

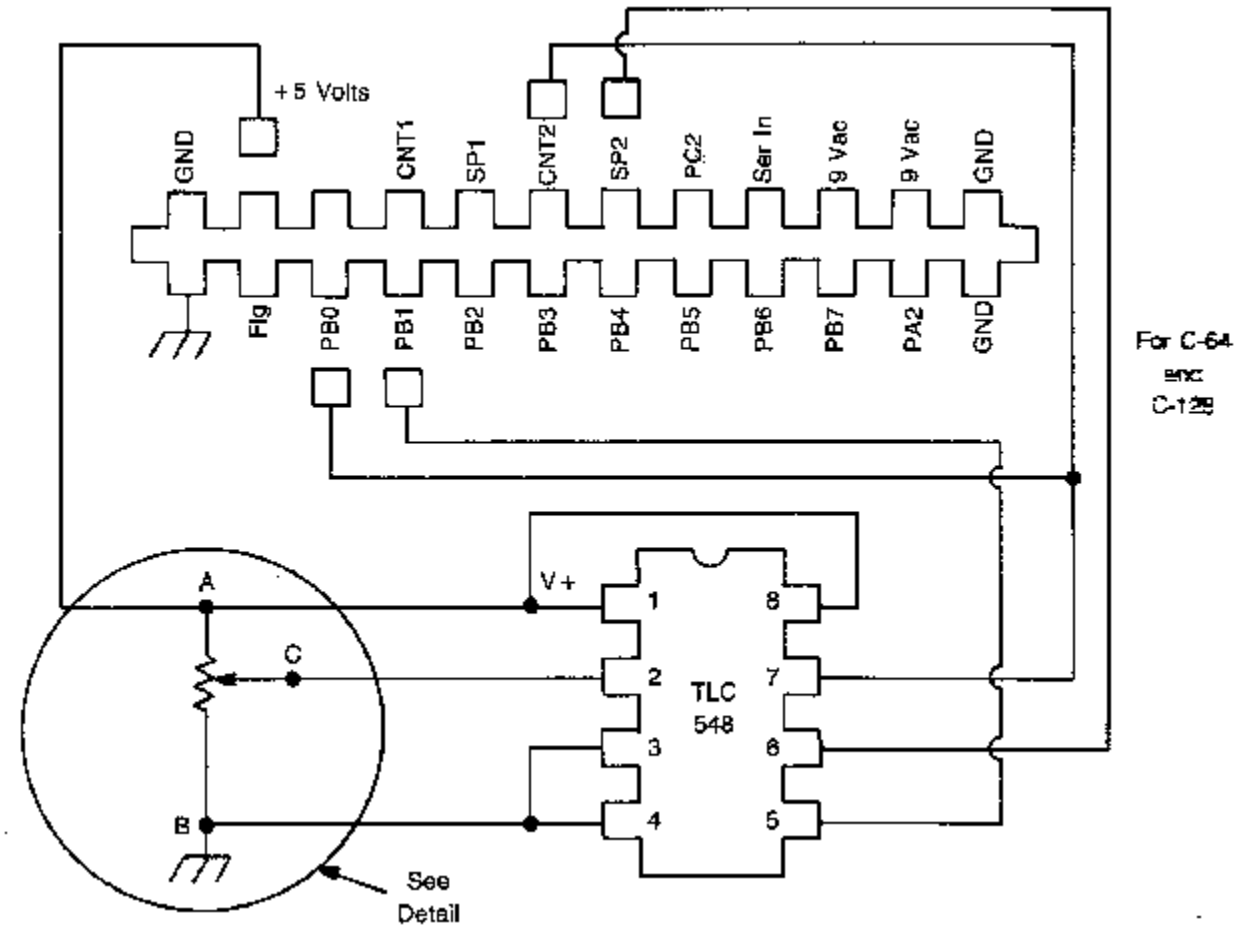


Fig. 4-5. Test circuit, C-128 and C-64.

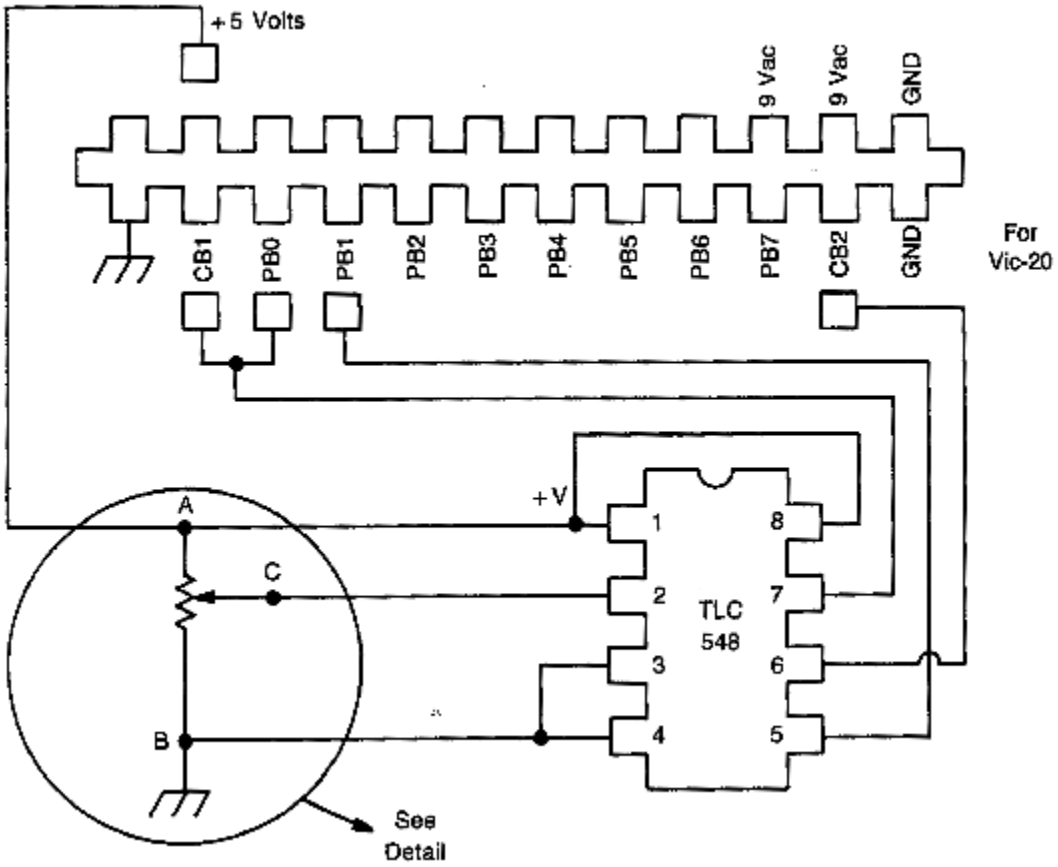
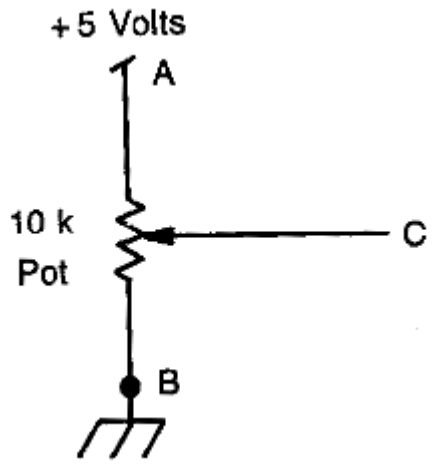


Fig. 4-6. Test circuit, Vic-20.



This section is where to place out transducer or circuit. To test, we insert a 10 k-ohm pot between the +5 volts and ground. The wiper is connected to Pin 2 (Analog In), by varying pot we corresponding by vary the voltage on Pin 2. The range is from 0 volts to 5 volts.

Fig. 4-7. Detail of variable resistor.

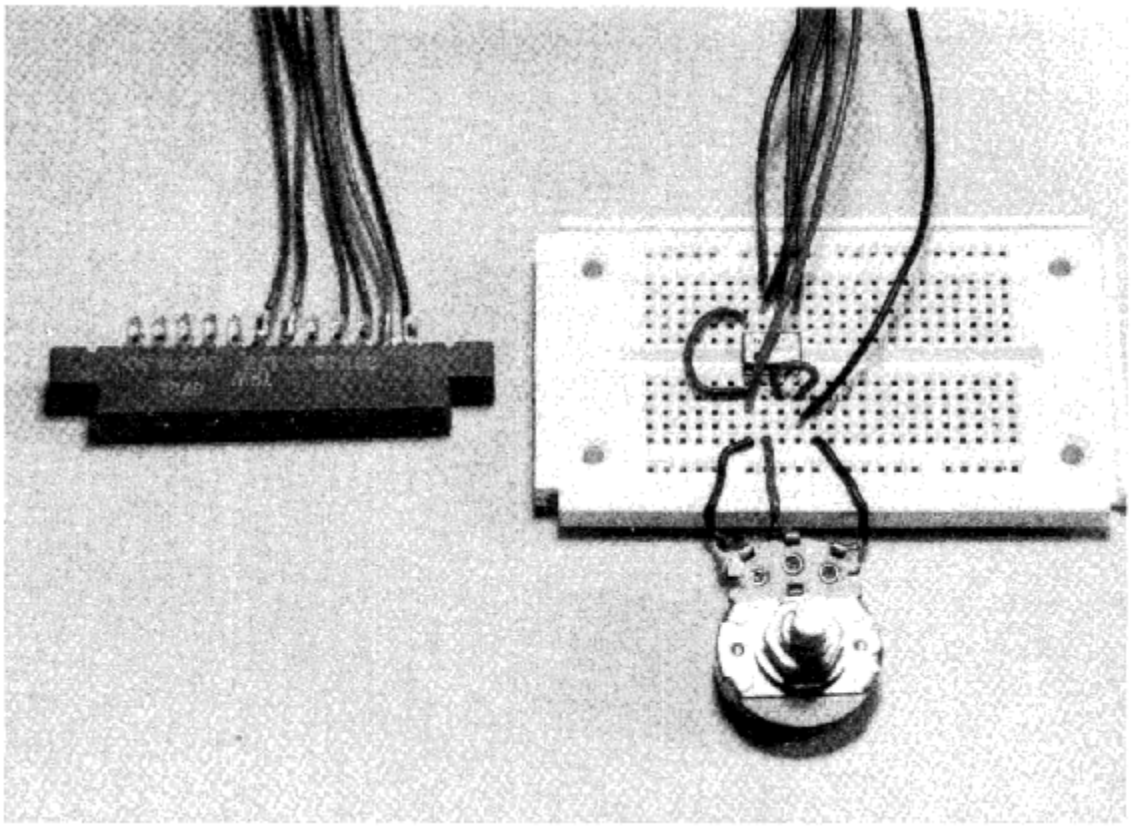


Fig. 4-8. Test circuit.

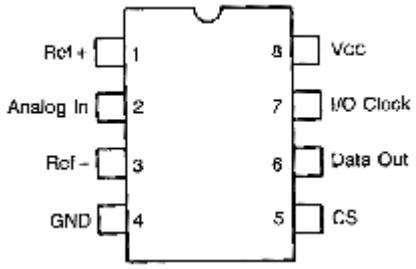
digital equivalent of the voltage present on pin two. If you have a volt meter handy you can connect the meter between pin 2 (analog in) and ground to observe the correlation of volts to the digital read out.

RESOLUTION

The serial register in Commodore computers and the A/D chip is one byte (8 bits) long. The largest number one byte can contain is (binary 11111111) decimal 255. Since we are reading the computers register to see what the serial AID chip is transmitting about our circuit, the readings can vary from a minimum of 0 to a maximum of 255.

We know that this number transmitted represents the digital equivalent of the voltage present on pin 2. The relationship between them is this, our reference voltages (ref + minus ref -) divided by 255 (resolution) equals volts per binary step (see Fig. 4-4). In circuits, Fig. 4-5 and 4-6, our reference voltages are +5 volts and ground 0 volts. So 5/255 - .0196078431 volts per incremented step. Each time the voltage varies by this amount our reading of the serial port would vary 1 point. It follows then, if the computer is reading 100, we can take this number multiply it by our volts per step and calculate what the voltage on pin 2 is. Let's do it, assume that our BASIC program is running and we're reading decimal 100 from the chip. We calculate the voltage thusly:100 x .0196078431 = 1.96078431V or approximately 2 volts.

Serial A/D Converter IC



Radio Shack
PN# 276
Cost $6.95

PIN	Description
1	Ref+ Pin is usually kept at supply voltage Vcc must be at least 1 volt greater than Ref– Min 2.5V Nom. 5V Max Vcc + .1
2	Analog In - Voltage to be digitized. Voltage should operate between Ref+ and Ref–
3	Ref– Min –.1 Nom or GND Max 2.5V
4	GND Ground Pin
5	Chip Select - High to Low transition begins outputting data on Pin 6
6	Data Out Pin - Transmits binary number in sequence with I/O Clock pulses
7	Input/Output Clock - Reads timing pulses for data out pin.
8	Vcc + Power Supply Min 3V Nom 5V Max 6V

*Note - Radio Shack does not supply Pin-Out description with their data sheets.
Fig. 4-9. Pin out serial A/D chip.

```
1 REM SERIAL A/D FOR C-128 AND C-64
10 POKE 56579,255
15 POKE 56577,0
20 POKE56589,127
25 FORX=0TO7
30 POKE56577,0:POKE56577,1
35 NEXTX
40 IF(PEEK(56589)AND8)=0THEN40
45 X=PEEK(56588)
50 PRINTX;
55 POKE56577,2
60 GOTO25
```

Fig. 4-10. Test Program, C-128 and C-64.

```
10 POKE37138,255
20 POKE37150,127:REM INTERUPT FLAG ENABL
30 POKE37147,12:REM AUXILIARY CONTROL REG
35 POKE37136,2
40 FORX=0TO7
50 POKE37136,1:POKE37136,0
60 NEXT
70 X=(PEEK(37149)AND4):REM SERIAL FG
80 X=PEEK(37146)
90 PRINT X;
95 POKE37136,2
100 GOTO40
```

Fig. 4-11. Test Program, Vic-20.

This chip has an 8-bit resolution. There are other chips on the market that have greater and lesser resolutions. These chips are classified by their bit resolutions such as 4-bit resolution, 12-bit resolution, and 16-bit resolution. It isn't practical for us to use these chips right now, but be aware that they are available for your use.

PROGRAMMING

The BASIC test program we used is slow and cumbersome, later we will use a machine language program that works with the 60 Hz interrupt. In the C-64 and C-128 BASIC program, we are using 2 addition registers aside from the ones we discussed in Chapter 1. They are the 56588 serial register and the 56589 interrupt control register. In the former we peek the register to see what number our A/D chip transmitted, the latter we mask all interrupts.

The CRA control register, located at 56590, controls whether the serial line will be an input or output. This register has the proper configuration we need on power up so it isn't necessary to change it.

We use PBO line to provide the clocking pulse to both the CNT line and the A/D chip. PB1 provides the high to low pulse every 8 clock cycles to start the chip transmitting its latest conversion.

The Vic program operates in a similar manner. To understand the serial register, interrupt register, and CRA, detailed information is provided in the Programmers Reference Guides.

TRANSDUCERS

By substituting different transducers for our testing pot, we can have the computer sense and measure light, heat, toxic gas, and galvanic skin resistance. The first of the transducers we will work with are variable resistor types (see Fig. 4-12) meaning as the sensor detects, the resistance of the sensor will change. This change in resistance changes the voltage

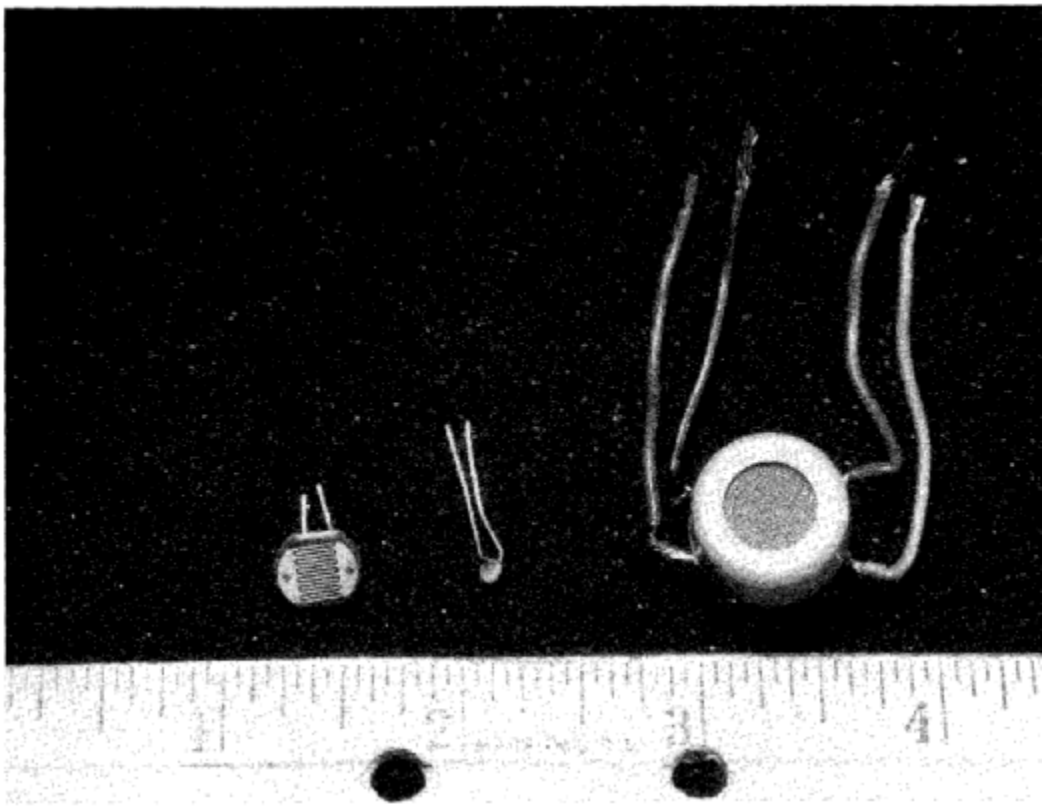


Fig. 4-12. Three transducers.

drop across the transducer and is picked up as a varying voltage on pin 2 of our A/D chip. The voltage on pin 2 will be displayed as before with the transducer resistance tracking like varying the pot did before.

LIGHT

Cadmium sulfide (CdS) photocells, (Radio Shack PN# 276-1657) respond to the intensity of light that falls on them. Their resistance is greatest in complete darkness, and decreases in proportion to the light made available. Examine Fig. 4-13 circuit C1. This shows the simplest method of connecting the cell into the circuit.

The disadvantage in this particular application is that we are utilizing just one half (128-255) of our possible range 0-255. We easily correct this situation in Fig. 4-13 circuit C2 by adding two resistors that make up a voltage divider. This changes our ref- from 0 volts to 2.5 volts. Our volts per step also changes (ref+ minus ref- = ref Voltage) 5V - 2.5V = 2.5V. Using the new ref voltage we get 2.5/255=0.00980392157 volts per step. With circuit C2 we are reading voltages between 2.5 volts (ref-) and 5 volts (ref+). This gives us full scale operation with the photocell (see Fig. 4-14).

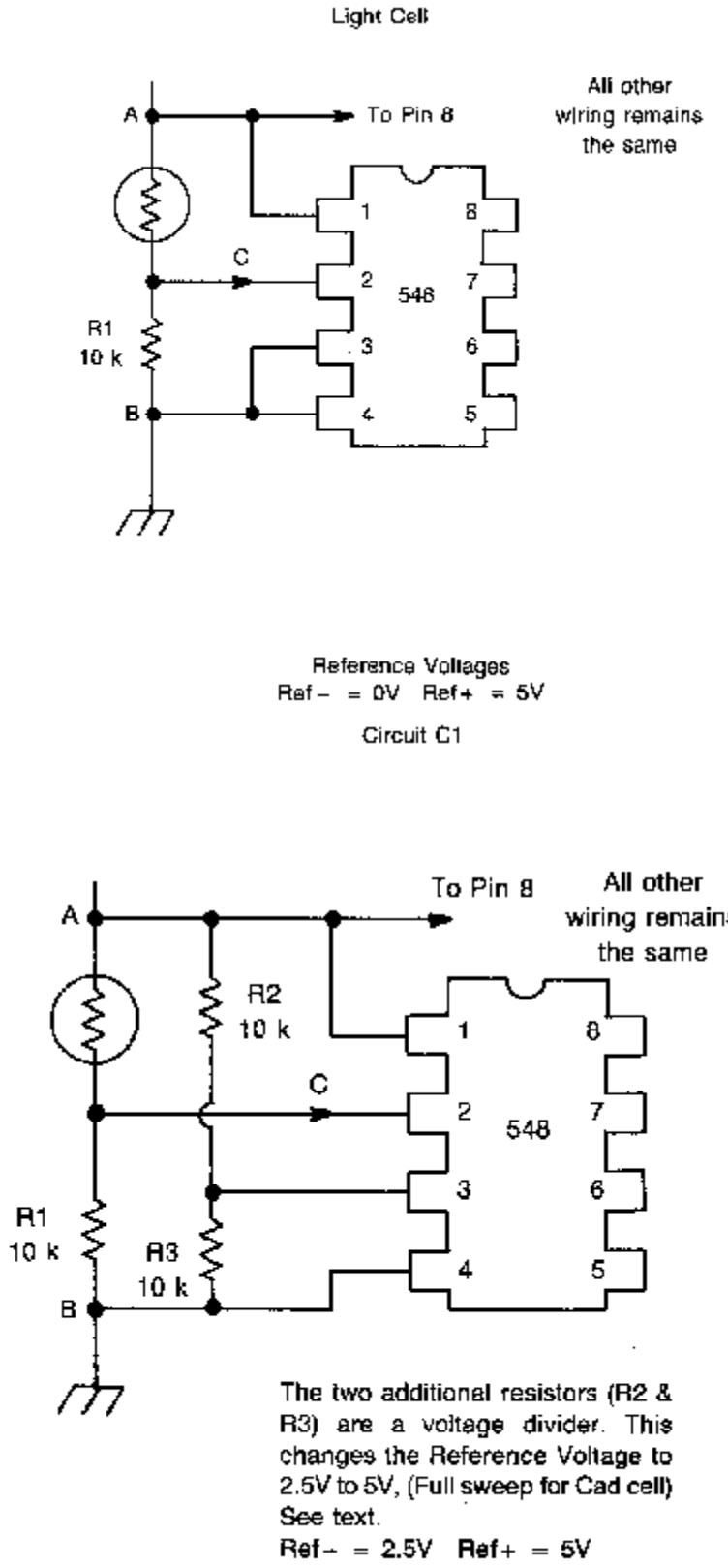


Fig. 4-13. Light cell C1 and C2.

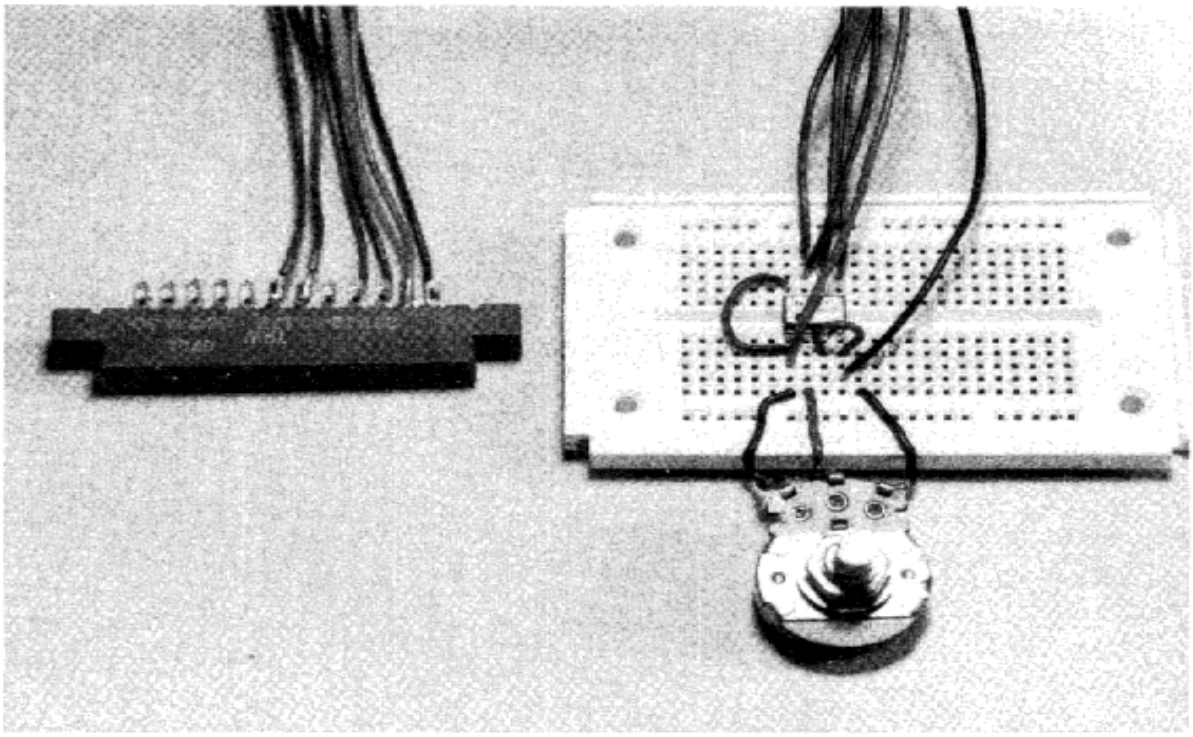


Fig. 4-14. Light cell circuit C1.

APPLICATIONS

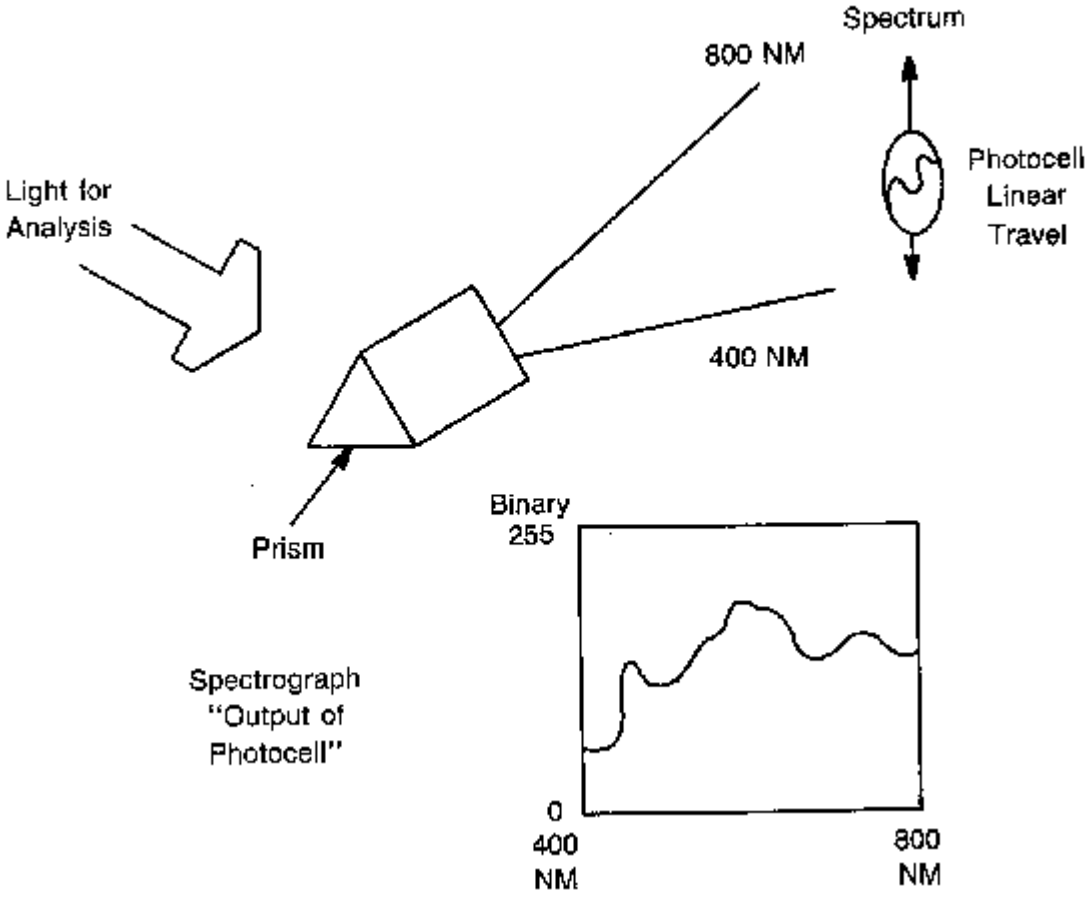
Now that we have a method of measuring light intensity, what are some applications for this device? If you are a photographer and do your own printing you could use this as an exposure meter for your enlarger. In the high tech end of applications, a spectrophotometer is possible (see Fig. 4-15). Spectrographic analysis is a method used by scientists to determine what elements are in an unknown compound. This technology was used to determine the composition of the Sun and stars.

TEMPERATURE

To measure temperature simply replace the photocell with the heat transducer (thermistor) in Fig. 4-13 circuit C2 (see Fig. 4-16). The thermistor is an NTC (negative temperature coefficient) type, that decreases in resistance as temperature increases (see Fig. 4-16). Resistance at 25°C (°F) is 10,000 ohms. Maximum operating temperature is 150°C. (302°F). (Digi-Key PN # KC006N-ND)

The resistor R1 that is in series with the transducer is good for sensing ambient room temperature and above. To change the scale and improve its response in the 0° to 120° F range replace R1 with a 47 k resistor.

Note at this time, although we are changing the reading range of temperatures by changing the resistor R1, the volts per incremented step and the voltage reading range on pin 2 remains the same. The only way to adjust this is by changing the voltage divider resistors. If you should decide to change the voltage divider make sure to remain within the range detailed in the spec sheet and pin out description.



* Note - Cds photocell output isn't linear throughout spectrum.

Fig. 4-15. Spectrographic analysis of light.

APPLICATIONS

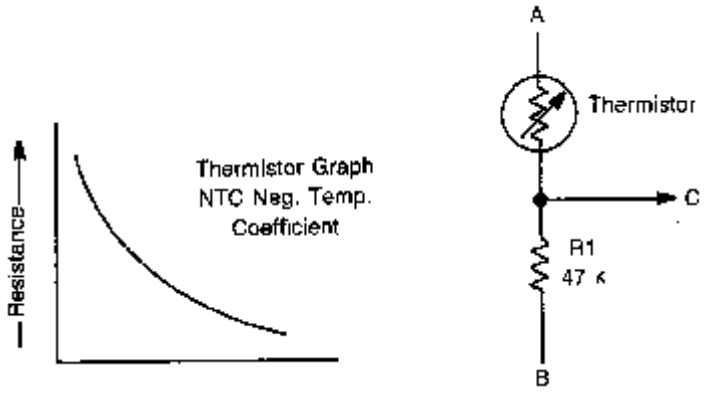
Calibration is necessary before using this sensor for any critical operation. One method of calibration is to submerge the sensor first in cold ice water then in boiling hot water marking each readout. The first number recorded is the equivalent of 32oF the second 212oF. The most obvious applications are an electronic thermometer and thermostat control.

TOXIC GAS SENSOR

The toxic gas sensor responds to a large number of airborne toxic compounds. Its operation is similar to the thermister in that as the sensor detects compounds the resistance of the device decreases.

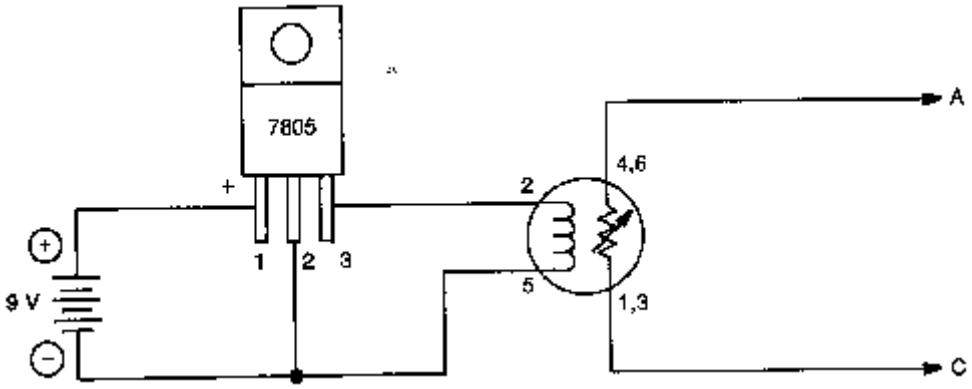
Examine Fig. 4-17. Pins 2 and 5 connected to a heater coil inside the transducer. The heater coil requires 5 volts at approximately 115 mA. This current is beyond what the user port can supply. This mandated the addition of a battery power supply with a 7805 5V voltage regulator.

Pins 4 and 6 are internally connected, as are pins 1 and 3. When you make your solder connectors to the sensor you need only connect to one pin of each pair (see Fig. 4-17 and Fig. 4-18).

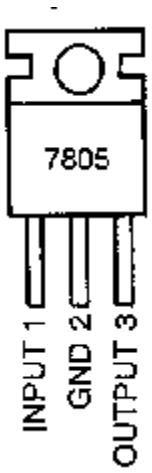


Same as Cad cell use circuit C2
Replace Cad cell with thermistor
Toxic Gas Sensor

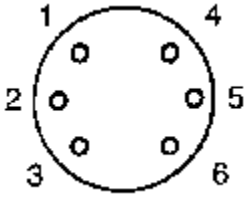
Fig. 4-16. Thermister circuit and graph.



7805 5V
Voltage
Regulator



Gas Sensor



Pins 2 & 5 heater
Pins 1 & 3 internally connected
Pins 4 & 6 internally connected

Fig. 4-17. Toxic gas sensor.

Fig. 4-18. Back view of toxic gas sensor.

Polarity isn't important for either the heater coil or sensor any way you connect the wires the unit will function properly. You may notice that the sensor feels quite warm when operating, don t be alarmed, this is normal and is a result of the internal heating coil.

Change R1 in Fig. 4-13 circuit C2 to a 47 k resi5tor and connect the circuit as shown in Fig. 4-17. Since the sensor has been in storage prior to you receiving it, it will require an initial 2 minute warm up period. This warm up period decreases with use. After the warm up period you can test the sensor with a number of household items. I first used a butane gas lighter, by releasing gas by the sensor (unlit), the sensor reacted immediately jumping from a base line of 0 to 255. By breathing on the unit it will detect the carbon dioxide. You can test and experiment with other items such as cleaning fluids.

APPLICATIONS

You can use the toxic gas sensor for an automatic ventilator control or gas leak detector and alarm.

BIOFEEDBACK

The biofeedback device (Fig. 4-19 and Fig. 4-21) has two uses. One as a lie detector, and as a stress level measurement device. The device operates by detecting changes in the galvanic skin resistance of the person connected to the device.

A persons' galvanic skin resistance at any particular time is an indicator of his or her state of arousal (emotional stress and tension level). This is called the *base line conductance*. The base line conductance will vary

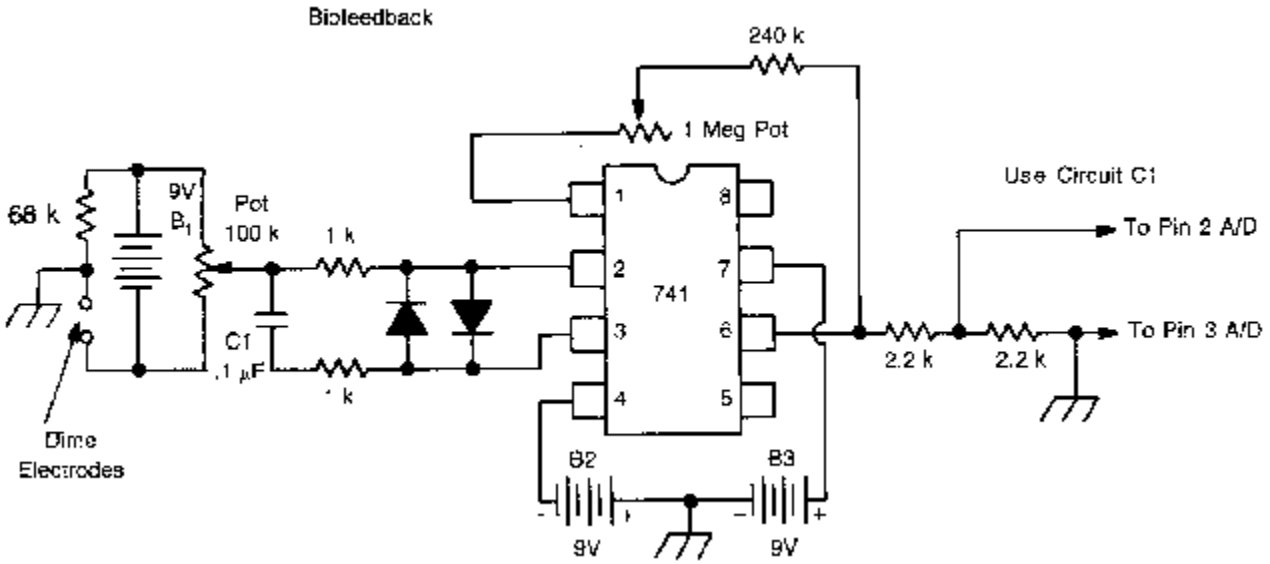


Fig. 4-19. Biofeedback circuit.

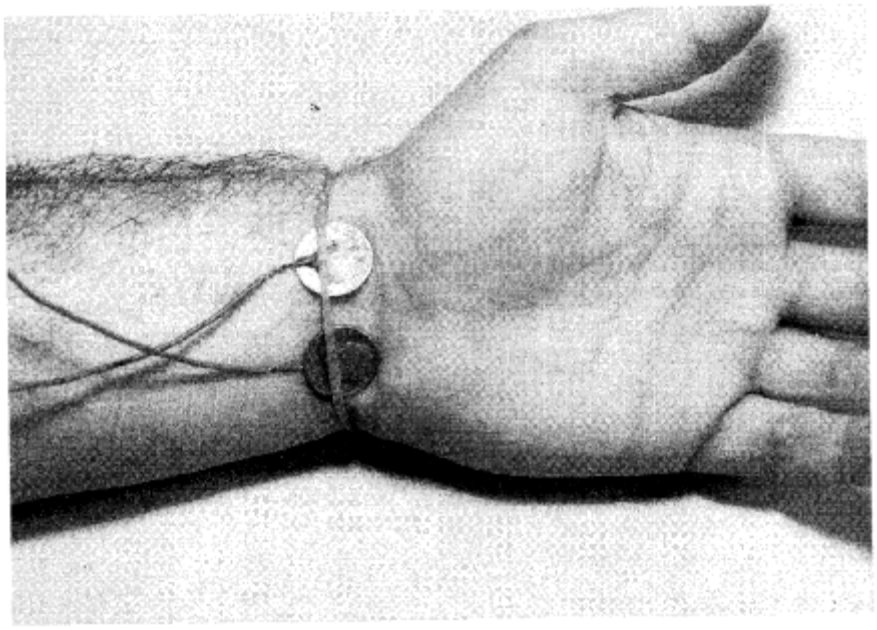


Fig. 4-20. Dime electrodes on wrist.

slightly as you use the biofeedback device making it necessary to adjust the device occasionally.
The electrodes are made by soldering a wire to a dime (see Fig.4-20).
To use, place a rubber band that fits snugly around the subjects wrist, and place the dime electrodes underneath the rubber band.

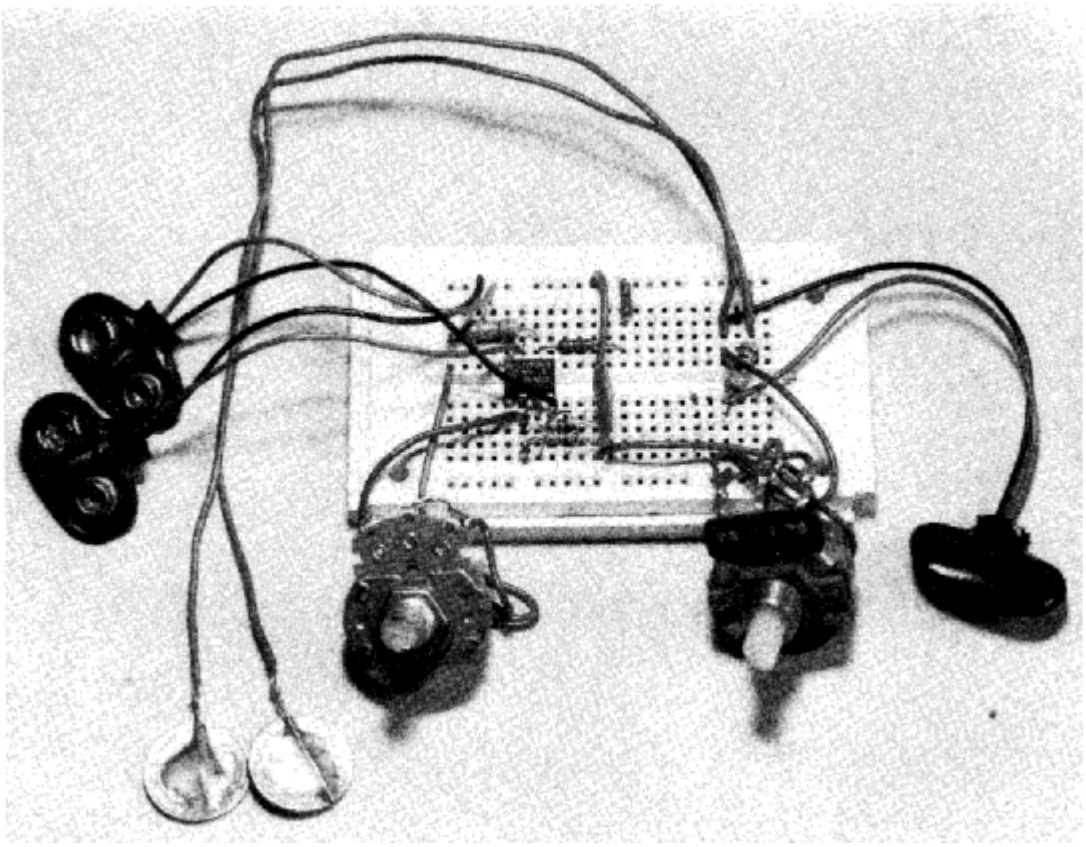


Fig. 4-21. Biofeedback circuit.

Set both pots at mid-position when beginning, attach both electrodes, use the R1 pot to adjust the reading. When adjusting R1 you notice a point when a small movement on the pot causes the reading to jump up or down a good amount. This is the trigger point. Depending upon what application (lie detector / stress level monitor) you have in mind at the time determines where to adjust the R1 pot. The R2 pot adjusts the gain of the 741 op-amp. Normally you won't need to adjust this.

To use as a lie detector adjust R1 till your reading is a little below 255. At this point press the electrodes further against the skin, the reading on the monitor should jump down. (Remember to attach the electrodes before you begin adjusting R1.) When you release the pressure on the electrodes the reading should rise to approximately what it was before. If this test works you're ready to begin. If riot, recheck all your wiring.

When you ask a question that evokes an emotional response the readings will increase. A simple test to perform with a deck of cards is to have your subject pick a card. And you try to ascertain which card was picked using the biofeedback device. A card player may respond to an Ace or Joker card even though it was not the card he picked. A highly emotional subject may respond in anticipation of you showing the target card. Anything that evokes a strong emotional response can be detected by this device, it could be the nature of a question regardless of the answer that causes the response. Please keep this in mind.

To use as a biofeedback device to reduce stress, adjust R1 till your almost reading 0. Now sit back and relax, imagine yourself to be in any place or situation you find soothing. As you respond the readings will start to rise. It is interesting to note that you can remain in a high state of awareness and be totally and completely relaxed. With practice your ability to relax quickly will develop and use of the machine will become unnecessary. You may also reset the device to approximately 0 after you topped it out and tried to make it rise again, bringing yourself to new levels of relaxation.

The reason you don't set the device to 0 or 255 when adjusting R1 is that you could overcompensate. This could make reading the changes in skin resistance impossible if all the changes are happening below 0 or above 255. Another point to note is that as your body relaxes its resistance increases. Under stress the resistance will decrease. Until you become familiar using the device, it can be frustrating to set the pots for a good reading, give yourself a little time to learn.

APPLICATIONS

The biofeedback device, as stated before can be used as a lie detector and stress management device. More than this, it should be considered an exercise in physiological measurement. You are not limited to this device, other devices such as EKG's and EEG's can be interfaced to the computer also.

60 Hz INTERRUPT VECTOR

Commodore computers use one of the 6526 timers to issue an interrupt every 1/6o of a second. The interrupt routine that follows scans the keyboard to see if a key has been pressed, updates the real time clock and performs a number of housekeeping functions. Our interest is not in the routine, but in utilizing the interrupt procedure for our own benefit.

A simple explanation of what happens when an interrupt is generated is as follows. When the microprocessor receives an interrupt signal the program instruction that is currently being performed is finished. The address of the next instruction is stored, then the program is directed into the interrupt subroutine program. Upon completion of the interrupt routine the address that was stored at the beginning of the interrupt routine is pulled and our program continues at this address, which is exactly where it left off.

This process happens continuously and transparently in the background of BASIC. This is how, when you are running a program you can stop it in the middle of operations by pressing the RUN/ STOP key before it is finished. The interrupt routine scans the keyboard 60 times a second, when it sees that you pressed the Run/Stop key, it knows to stop the program and return control. In fact everytime you press a key and see it appear on your monitor you're looking at the interrupt routine

at work. Our reason for bringing this up is to utilize this routine by adding our own Serial AID program to it. This is called implementing a wedge. It's called that because we are wedging or adding our own short program to the interrupt routine. By doing so our program will be executed 60 times a second and is transparent to any program running in BASIC. We can accomplish this by changing a vector in the interrupt routine, (A vector is an address that directs the program to its next instruction.) to point to our program before continuing to the standard interrupt routine.

DEMO INTERRUPT

To gain an appreciation of what we are doing I've written a demo program. This program will transfer one of the MPUs registers, the Y-Reg, into the user port. For the C-64 see Fig. 4-22, for the C-128 see Fig. 4-23, and Vic-20, Fig. 4-24. By connecting our LED interface from Part I we can examine the operation of the register as BASIC is running. This program must be written in machine language so I've written a BASIC loader for it. Save the program before you run because it erases itself from BASIC. The program reads and displays the Y-Reg 60 times a second. Observe what happens as you type in a program, run a program or load from a disk.

```
10 REM J. IOVINE  3-1-87
20 FORJ=40710TO40751:READX:POKEJ,X:NEXT
30 SYS40718:POKE56,PEEK(56)+1:NEW
40 DATA140,1,221,234,108,44,159,234,173,20,3,141,44
50 DATA159,173,21,3,141,45,159,120,169,6,141,20,3
60 DATA169,159,141,21,3,169,255,141,3,221,88,96,49,234,234,0
```

Fig. 4-22. Interrupt C-64.

```
10 REM JOHN IOVINE
20 FORJ=4864TO4905:READX:POKEJ,X:NEXT
30 SYS4870:NEW
40 DATA140,1,221,108,36,19,173,20,3,141,36,19,173,21
50 DATA3,141,37,19,120,169,0,141,20,3,169,19,141,21,3
60 DATA169,255,141,3,221,88,96,255,21,255
```

Fig. 4-23. Interrupt C-128.

```
10 REM VIC IRQ DEMO
15 REM J IOVINE
20 FORJ=7430TO7468
30 SYS7436:NEW
40 DATA230,33,66,55,2,83,30,255,243,77,8,241,148,127,117
50 DATA247,177,105,36,252,11,119,154,223,245,128,45,251
60 DATA33,127,96,246,177,241,128,4,86,70,212
```

Fig. 4-24. Interrupt Vic-20.

A/D INTERRUPT

These programs read the A/ D chip 60 times a second and place the information in memory location 255. All you need to do is peek the location for the current value, the BASIC program we used before will not be needed when using this machine language version.

For the C-64 see Fig. 4-25, for the C-128 see Fig. 4-26, and Vic-20 see Fig. 4-27. After you have typed in and saved the program, run it, and type in this line.

```
10 X=Peek(255):Print X:GoTol0
```

```
10 REM JOHN IOVINE
20 FORJ=40710TO40785:READX:POKEJ,X:NEXT
30 SYS40743:POKE56,PEEK(56)+1:NEW
40 DATA160,8,169,0,141,1,221,169,1,141,1,221,136
50 DATA192,0,208,241,173,13,221,173,12,221,133,255
60 DATA169,2,141,1,221,108,79,159,173,20,3,141,79
70 DATA159,173,21,3,141,80,159,120,169,6,141,20,3
80 DATA169,159,141,21,3,169,255,141,3,221,169,0
90 DATA141,1,221,169,127,141,12,221,88,96,0,255,74
```

Fig. 4-25. A/D interrupt C-64.

```
100 REM JOHN IOVINE
120 FORJ=4864TO4937:READX:POKEJ,X:NEXT
130 SYS4897:NEW
150 DATA160,8,169,0,141,1,221,169,1,141,1,221,136
160 DATA192,0,208,241,173,13,221,173,12,221,133,255
170 DATA169,2,141,1,221,108,160,19,173,20,3,141,160
180 DATA19,173,21,3,141,161,19,120,169,0,141,20,3
190 DATA169,19,141,21,3,169,255,141,3,221,169,0,141
200 DATA1,221,169,127,141,13,221,88,96,0
```

Fig. 4-26. A/D interrupt C-128.

```
10 REM VIC SER IRQ.BAS
15 REM J IOVINE
20 FORJ=7430TO7510:READX:POKEJ,X:NEXT
30 SYS7463:NEW
40 DATA160,8,169,0,141,16,145,169,1,141,16,145
50 DATA136,192,0,208,241,173,29,145,173,26,145
60 DATA133,255,169,2,141,16,145,108,84,29,173,20,3
70 DATA141,84,29,173,21,3,141,85,29,120,169,6,141
80 DATA20,3,169,29,141,21,3,169,255,141,18,145,169
90 DATA0,141,16,145,169,127,141,30,145,165,12,141
100 DATA27,145,88,96,234,234,234
```

Fig. 4-27. A/D interrupt Vic-20.

This one line will print serial A/D conversion. The program is not affected by the RUN/STOP key, but a RUN/STOP and RESTORE will reset the vector. To reinitiate the program, Sys (the number in program). Naturally, after the A/D interrupt is initiated you can peek this location anytime or anywhere in a program to read the latest conversion from the chip.

5

Digital Audio Recording and Playback

By interfacing numerous transducers to the user port, we gave the computer the ability to sense the environment. To restate: the analog information (signal) from the transducer was converted to its binary equivalent by the A/D chip then transmitted serially to the MPU via the user port. This time we will add another step. We will store the binary information in the computer's memory then take binary information from the computer's memory and reconvert it to its analog voltage equivalent.

You may wonder what good is such a procedure? One application we'll accomplish is that we can record an audio signal into the computer's memory and then play it back, kind of like an electronic tape recorder. But to look at this procedure from a broader viewpoints, once an analog signal is converted to a binary equivalent there is less of a chance of losing information through signal degradation and distortion when transmitting the signal.

And what good is that? Have you ever thought of the technology behind NASA's triumphant photographs of Mars, Saturn, and Jupiter? The procedure used to obtain those pictures is similar to what we are going to do.

Let's first look at what NASA does. Aboard NASA space probes, the image information is read from the on-board camera, pixel by pixel. Each pixel is converted into binary information. This binary information is transmitted via radio to Earth. Earthbound receivers receive and store

the information. The information is then reconverted from binary back to its analog equivalent pixel image. The pixels are reconstructed to form the original picture. Again, the reason for the conversion to binary before transmission is to prevent signal distortion.

Why aren't binary signals distorted? Well they are, they are just not as susceptible to distortion as analog signals are. Remember binary numbers and signals are made up of bits. Bits that are either 1 (+5V) or 0 (0V), so even with a great deal of distortion, let's say enough distortion to completely obliterate an analog signal, you can still determine during transition if a serial binary signal is a bit 1 or 0. And when you build up the 8 bits per byte, you will have recovered a close approximation to your original signal.

The project we'll construct to demonstrate our digital to analog (DAC or D/A) conversion is a digital audio recording and playback unit. The procedure is similar to the one described for NASA picture taking. First our sound information is converted to its binary equivalent. This information is stored in our computers memory. After the sound is completely recorded, we then reconvert the binary information back to its analog equivalent sound.

APPLICATIONS

At this point, you may say to yourself, so what! The bottom line is I could accomplish the same thing with a $10.00 tape recorder without going through all this analog to binary-binary back to analog jazz.

What makes this technique useful and unique is our ability to manipulate the binary information before we play it back. By doing so, we can get various special effects like echo, reverb repeat, frequency, and timing changes. These digital effects are being used by many rock and roll artists.

On the scientific end, you could work on algorithms for speech recognition by computers, or you may want to add a human voice to some of your programs. How about a voice in tutorial programs for foreign language or for educating preschoolers. These are new doors for entrepreneurs. The advantage this has over our speech synthesizer is a higher quality of sound. The quality of sound becomes critical in a number of important areas. For instance, when teaching a foreign language, inflection and subtle pronunciation becomes very important. The disadvantage to using digitized sound all the time however, is the tremendous amount of memory required.

SOUND SAMPLING

To understand how we measure the sound sample to obtain its binary equivalent, look at Fig. 5-1. This presents the waveform of a typical voice trace as would appear on an oscilloscope. Figure 5-2 shows a magnified section of the same waveform. On the left hand side of Fig. 5-2's (Y-axis)

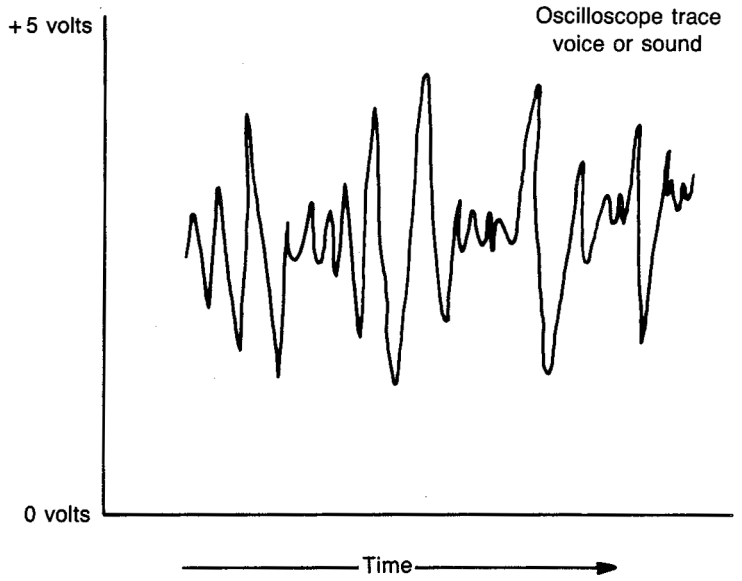


Fig. 5-1. Oscilloscope trace, voice or sound.

vertical axis, notice the voltage level and the binary equivalent of those voltage levels. On the horizontal axis (X-axis), each division represents one sampling cycle.

When we are recording, our serial A/D chip reads the voltage of the waveform at that particular instant, and transmits the binary number to the computer. The computer stores this number in memory and returns to the chip to get the next number. When it receives the next number, it stores that number in the next memory location. This continues for as long as we are recording. As you can see, it is following and recording the basic shape of the original waveform into memory.

When we playback, the computer reads the first binary number in memory and pokes it into the volume register of the computer sound interface device (SID) chip. The volume register of the SID chip outputs a voltage proportionate to the number we poke into this register. Although under usual circumstances the volume control by itself does not produce any sound, but we are varying the output voltage so fast that it does generate sound in synchronization to our recorded signal.

In summary, we can say our computer reads the sound encoded binary number then outputs the analog equivalent voltage through the SID chip. The computer retrieves the next number and goes through the same procedure until it is finished playing back the sample. Examine Fig. 5-2 again, notice that the chip will be outputting a close approximation of the original waveform.

Sampling cycle time is very important. It determines the fidelity and maximum frequency the computer can record. For the C-64 and C-128

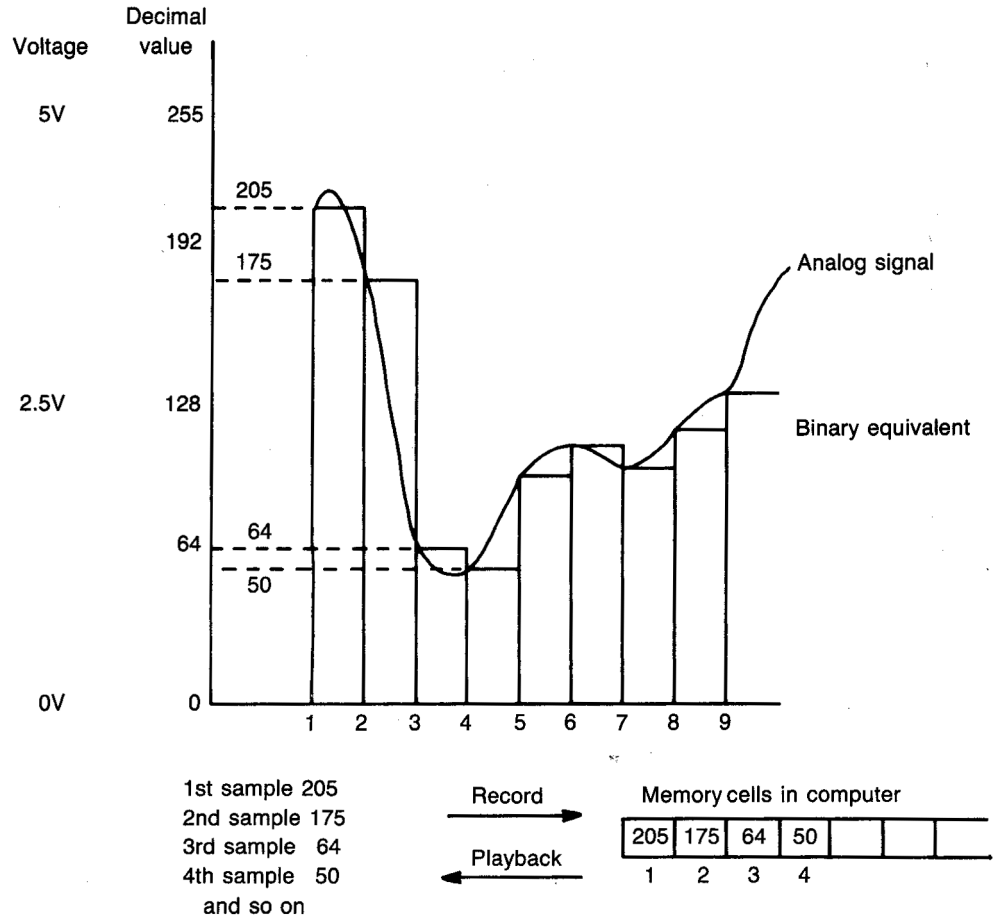


Fig. 5-2. Sound trace with A/D conversion.

(in slow mode), we will be sampling at approximately 5,000 samples a second. The 128 will sample at 10,000 samples a second in fast mode. We use one byte of memory for each sample we store, so we will use up our available memory pretty fast. You have about 10 seconds of record time at 5,000 rate, and 5 seconds at 10,000.

At 5,000 samples a second, the computer can record a maximum frequency of 2,500 Hz. This is easy to see by looking at Fig. 5-2 again. If our waveform jumped up and down between two sample points, the computer would not see it. It follows then, at 10,000 samples/second, we can record a maximum frequency, 5,000 Hz.

The slower rate is adequate for recording voice. At the higher rate, we can observe an interesting phenomenon called polyphonics. What this means is that the sampling rate is high enough to record more than a single sound. As is the case with music, both voice and instruments are recordable and played back. You can experiment with polyphonics with the 128 in fast mode.

DIGITAL OSCILLOSCOPE

After you have accomplished sound recording and playback, let's go a little further and display our sound sample information by converting the 128 computer into a digital oscilloscope.

By loading a small program after we're done recording, we can create a digital oscilloscope using the 128 graphic screen. We can use the scope to analyze our voice or music prints that we have placed in the computer memory. And as long as you stay within the input voltage requirements of our serial A/D chip (0 to +5V), you can perform a waveform analysis of any signal you'd like.

It is interesting to note that the image formed on screen is an exact real time rendition of Figs. 5-1 and 5-2. Our scope shows 320 sample cycles per screen. That's equal to about .064 seconds of sound in the slow mode, and about .032 seconds of sound in the fast mode. The program will continue displaying the entire waveform by automatically clearing the screen and plotting the next 320 points in memory.

CIRCUIT DESCRIPTION

The circuit is simple and straightforward (see Fig. 5-3). Most of the components are plugged into our breadboard (see Figs. 5-4 and 5-5).

We will again utilize our serial analog to digital chip from the last chapter to input the audio information into the computer memory. You can use audio information from any source you like, such as radio, tape deck etc. This article, however, will detail using a standard microphone for you to record your own voice.

The second chip in this project is the SID chip that is built inside your computer. This we are using as our DAC chip (the digital to analog chip). The SID chip reads the binary number and outputs the equivalent analog voltage. This voltage is our sound information, and will play through your monitor or TV speaker. This simplifies our circuit by requiring us to use only one amplifier at the input stage. (You could also connect an amplifier to the audio out wire and a ground wire to playback through it rather than the monitor.)

I decided that it would be more conducive for the project to purchase a small battery powered amplifier complete from Radio Shack. The amplifier cost is $11.95, which brings the total for this project to approximately $20.00. (This does not include the serial analog to digital chip that is assumed to have been purchased for the last chapter.) This allows us to concentrate on the main theme of what we are to accomplish, without being distracted or digressing into explanations of amplifier design. This also prevents our schematic from becoming unnecessarily and overtly complex.

The microphone plugs into the input jack of the amplifier. A similar plug (see parts list), that two wires are soldered to, plugs into the external

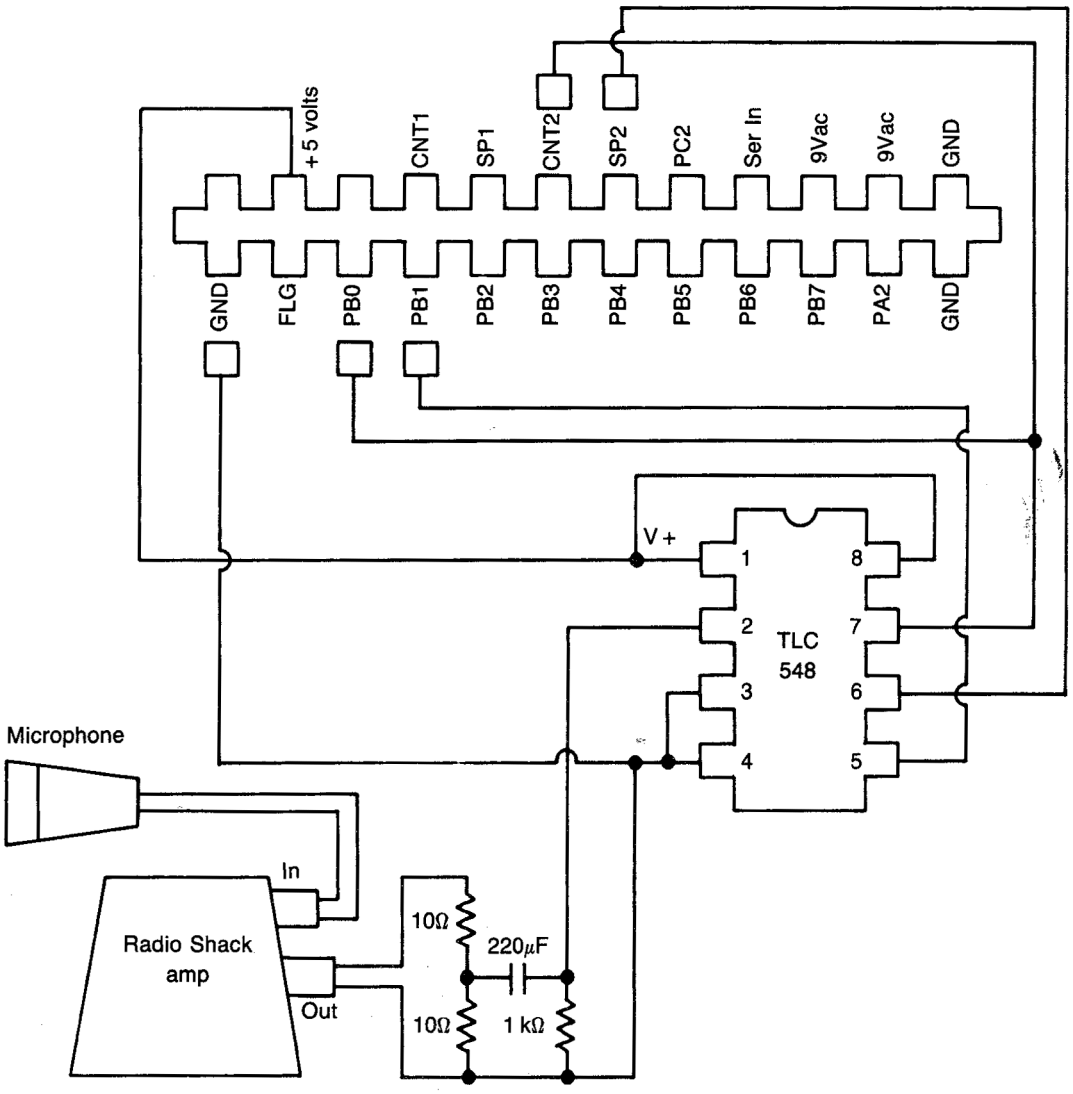


Fig. 5-3. Schematic of circuit.

speaker jack on the amp. These output wires from the amp go to the input of our serial A/D chip. Although this project is inexpensive to build, it is not short on performance. I'm sure you will be quite surprised by the accuracy and fidelity of the reproduction.

PROGRAMS

First, let me apologize to the Vic-20 users. Because of the high sample rate needed and therefore large memory required, it is not practical to attempt this project.

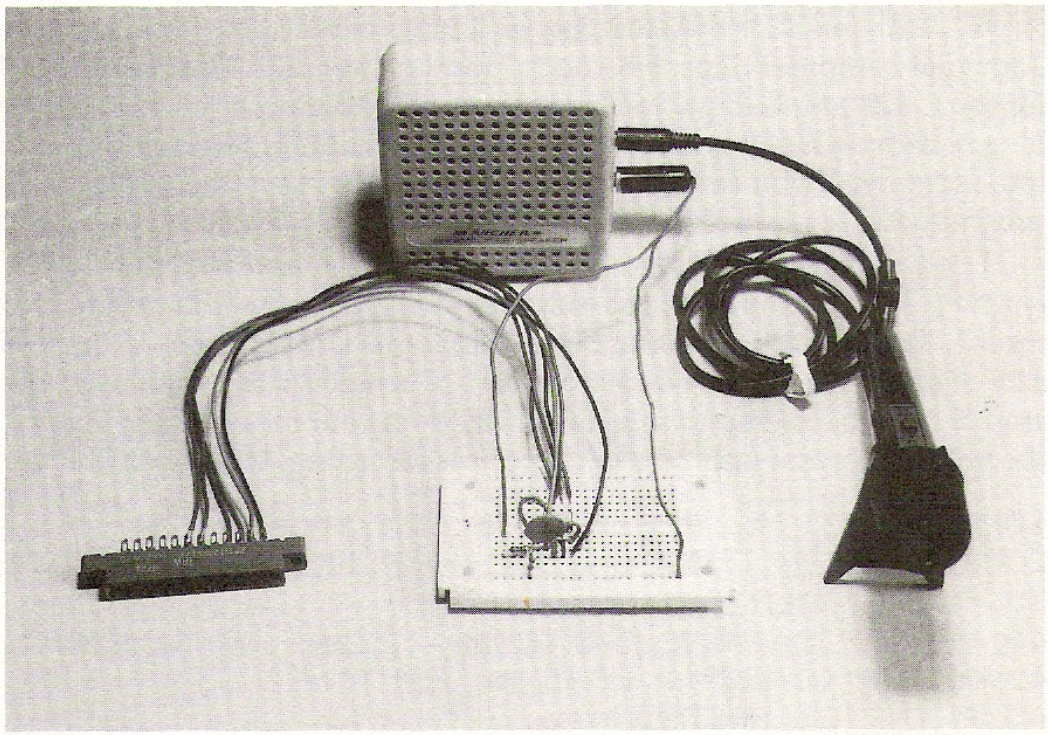


Fig. 5-4. Sound digitizer circuit.

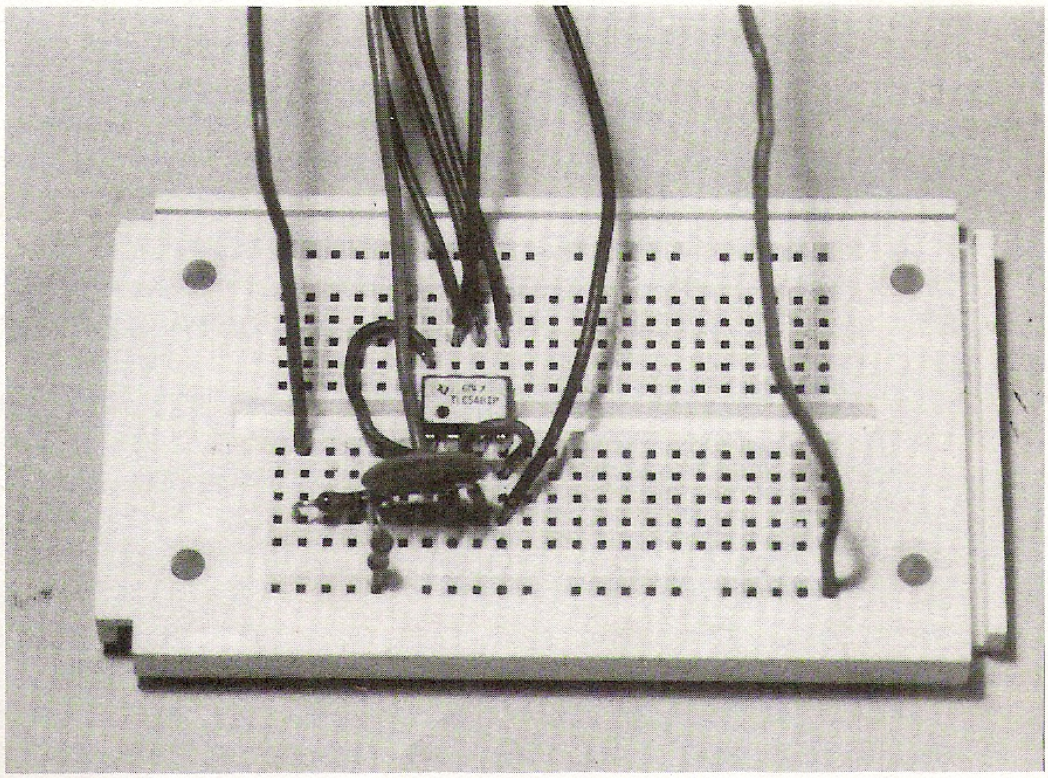


Fig. 5-5. Close up of circuit board.

Type in and save the basic loader and controller programs for your particular computer (Figs. 5-6 and 5-7 for the C-64, and Figs. 5-8 and 5-9 for the C-128). Make sure you save the basic loader program before running it, because it erases itself after it pokes the machine language program into memory. If you're using the C-128 computer, save the Digital Scope program also.

Load and run the Loader program, then load and run the Control program. Turn on the amp, full volume, then Sys number in program to record. To playback, Sys number in program. On playback, if there is a lot of static, it is due to overmodulation. Turn down the volume on the amp or hold the microphone further away from your mouth. Keep varying the volume control until you get a perfect recording. Once you find the

```
10 REM DIGITAL RECORDER & PLAYBACK C-64
20 POKE56579,255:POKE253,00:POKE254,20:POKE4986,00
35 PRINT"{CLR}"
40 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}MENU"
42 PRINT"{C/DN}1) LOAD ML ROUTINE"
45 PRINT"2) RECORD"
50 PRINT"3) PLAYBACK"
52 PRINT"4) QUIT"
55 PRINT"{C/DN}{C/DN}INPUT CHOICE 1 -- 4"
60 INPUT X
65 ON X GOTO 86,70,80,85
70 PRINT"{CLR} RECORDING":SYS4864:GOTO20
75 PRINT
80 PRINT"{CLR} PLAYBACK":SYS4937:GOTO20
85 PRINT"{CLR}":END
86 PRINT"{CLR} LOADING ML SUBROUTINE...."
87 FOR X=4864TO4986:READA:POKEX,A:T=T+A:NEXT
88 IFT<>16853 THEN PRINT"ERROR IN DATA STATEMENTS":END
89 GOTO10
95 DATA 120,160,008,169,000,141,001,221,169,001
105 DATA 141,001,221,136,192,000,208,241,173,013
115 DATA 221,173,012,221,160,002,140,001,221,172
125 DATA 122,019,145,253,200,140,122,019,192,255
135 DATA 208,022,160,000,140,122,019,230,254,165
145 DATA 254,201,148,208,009,169,020,133,254,088
155 DATA 096,234,234,234,162,002,202,208,253,076
165 DATA 001,019,096,120,172,122,019,177,253,141
175 DATA 024,212,200,140,122,019,192,255,208,022
185 DATA 160,000,140,122,019,230,254,165,254,201
195 DATA 148,208,009,169,020,133,254,088,096,234
205 DATA 234,234,162,033,202,208,253,076,073,019
215 DATA 096,234,000
```

Fig. 5-6. Basic control for C-64.

```
5 REM 128 BASIC LOADER DIGITAL RECORDER & PLAYBACK
10 REM JOHN IOVINE 4-16-87
20 FORJ=4864TO5003:READX:POKEJ,X:NEXT
25 LOAD"128 BAS. CONT",8
30 DATA 120,160,008,169,000,141,001,221,169,001
40 DATA 141,001,221,136,192,000,208,241,173,013
50 DATA 221,173,012,221,160,002,140,001,221,172
60 DATA 138,019,141,001,255,145,253,169,000,141
70 DATA 000,255,200,140,138,019,192,255,208,020
80 DATA 160,000,140,138,019,230,254,165,254,201
90 DATA 244,208,007,169,066,133,254,088,096,234
100 DATA 162,001,202,208,253,076,001,019,096,120
110 DATA 172,138,019,141,001,255,177,253,142,000
120 DATA 255,024,074,074,074,074,141,024,212,141
130 DATA 001,221,200,140,138,019,192,255,208,019
140 DATA 160,000,140,138,019,230,254,165,254,201
150 DATA 244,208,006,169,066,133,254,088,096,162
160 DATA 033,202,208,253,076,079,019,096,000,000
```

Fig. 5-7. Machine language for C-64.

```
 2 REM BASIC CONTROL DIGITAL RECORDER AND PLAYBACK
 4 POKE56579,255:POKE253,0:POKE254,66:POKE5002,0
 8 PRINTTAB(7)"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}
   {C/DN}MAIN MENU"
12 PRINT:PRINTTAB(7) "DIGITAL RECORDER AND PLAYER"
14 PRINT:PRINTTAB(7) " 1) LOAD ML PROGRAM"
16 PRINTTAB(7) "2) RECORD"
18 PRINTTAB(7) "3) PLAYBACK"
20 PRINTTAB(7) "4) FAST"
22 PRINTTAB(7) "5) SLOW"
24 PRINTTAB(7) "6) QUIT"
28 PRINTTAB(7)"ENTER NUMBER 1-6 THEN PRESS RETURN":INPUTJ
32 ONJGOTO100,110,120,130,140,150
100 PRINT"{CLR}":PRINTTAB(21)"{C/DN}LOADING":LOAD"SID
    PLAYBK",8
110 PRINT"{CLR}":PRINTTAB(12)"{C/DN}RECORDING":SYS4864:GOTO8
120 PRINT"{CLR}":PRINTTAB(12)"{C/DN}PLAYBACK":SYS4943:GOTO8
130 FAST:GOTO8
140 SLOW:GOTO8
150 END
```

Fig. 5-8. Machine language for C-128.

```
10 REM DIGITAL SCOPE ** J. IOVINE
20 GRAPHIC1,1
30 BA=16896
40 FORG=1TO320
50 BANK 0:H=(PEEK(BA+G)+90)
60 BANK 1:DRAW ,G,H
70 NEXT
80 IFBA<62464THENBA=BA+320
85 IFBA=>62464THENEND
90 GRAPHIC1,1:GOTO40
```

Fig. 5-9. Basic control for C-128.

right setting, it's really easy to get good recordings. If you're operating the 128 computer, you can go try the fast mode at this point, and notice the higher fidelity of sound.

If you like, try recording music to hear polyphonics sound. Vary between the slow and fast modes, and you'll see how important sampling time cycles are. Remember, if you're recording music the microphone itself will add some static and distortion. If you should make direct connections to the digitizing circuit, WATCH THOSE INPUT VOLTAGES.

For the 128 users, after you have a good sound sample in memory, load and run the Digital Scope program Fig. 5-8. You' may have to sit through a few screens before you start to see some activity. This would depend on how long it took you to speak after you Sys to record. Remember, you have over 40 k of memory to look at, and each screen holds only 320 bytes of information.

I would also advise to record at the 10,000 sample rate. This keeps the waveform more concise when plotting. If you want to do waveform

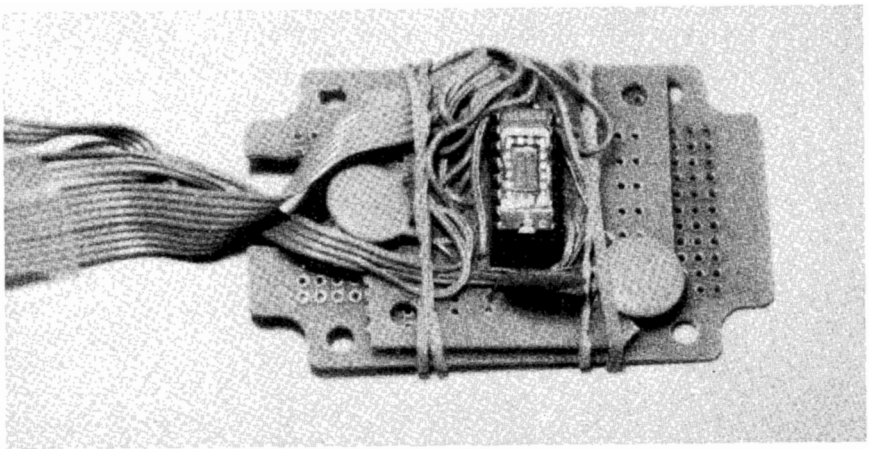


Fig. 5-10. Digital camera, C-128.

analysis with the digital oscilloscope, tie in your signal to the input of the serial chip and record. Again, watch the voltages.

The Digital Scope program as it stands, plots points. When doing waveform analysis of complex waves, the dots break up into a disconnect pattern. To alleviate this problem, change lines 50 and 60, and add line 55 as follows.

```
50 bank O:h=(peek(ba+g)+05
55 h1=(peek(ba+g+1)+05)
60 Bank l:Draw ,g,h to g+1,h1
```

This change allows the program to draw lines connecting the dots, and will help a great deal when studying waveforms.

APPLICATIONS

It should be obvious that there is a lot more you can do with this project. This includes digital special effects, waveform analysis, and loading and saving the binary sound information on disk to be used to put sound or words in your programs.

Parts List

Quantity	Item/Description	Part Number	Cost
1	Microphone	Radio Shack PN# 33-1054	4.99
1	Audio Amp	Radio Shack PN# 277-1008	11.99
1	1/8" phono plug	Radio Shack PN# 274-286	1.39
1	.1 µF cap	Radio Shack PN# 272-135	.49
1	1 k resistor 1/4 watt		.39
	10-ohm resistor 1/4 watt		.39

ADVANCED SOUND DIGITIZER PROGRAM

This is an advanced program for the sound digitizer that will provide maximum performance for minimal investment and minimum fuss, plus kick in a couple of special effects to boot. The principles of sound digitizing, as described in the previous chapter, remains the same. For those of you who are interested in the basic principles of sound digitizing, see the previous chapter.

I have made substantial improvements in the program. As with most things, we have advantages and disadvantages by taking any particular route, and this is no exception. One disadvantage to using the SID chip is that the SID chip can only output sound with a 4-bit resolution. But

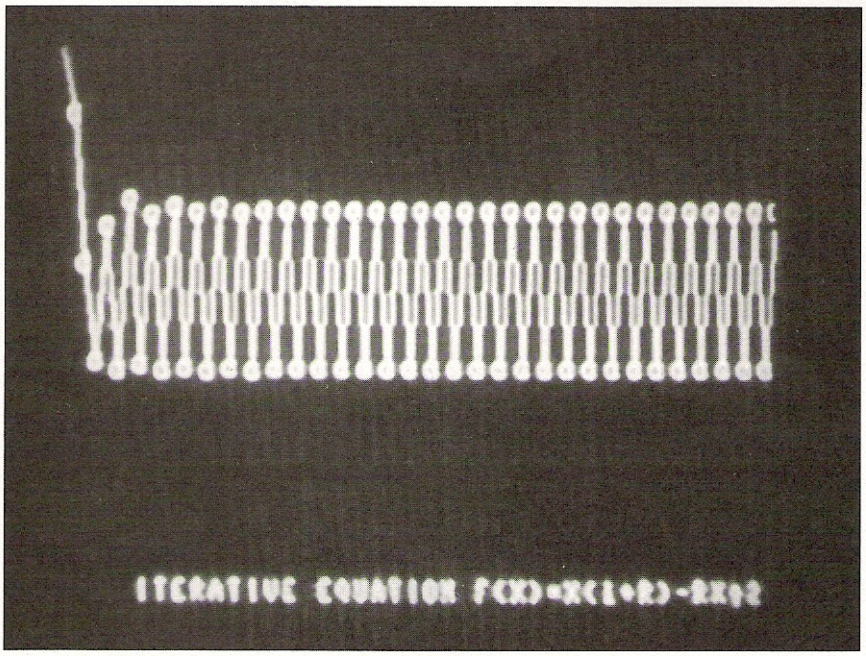


Fig. 5-11. Digital chaos plot (see page 141).

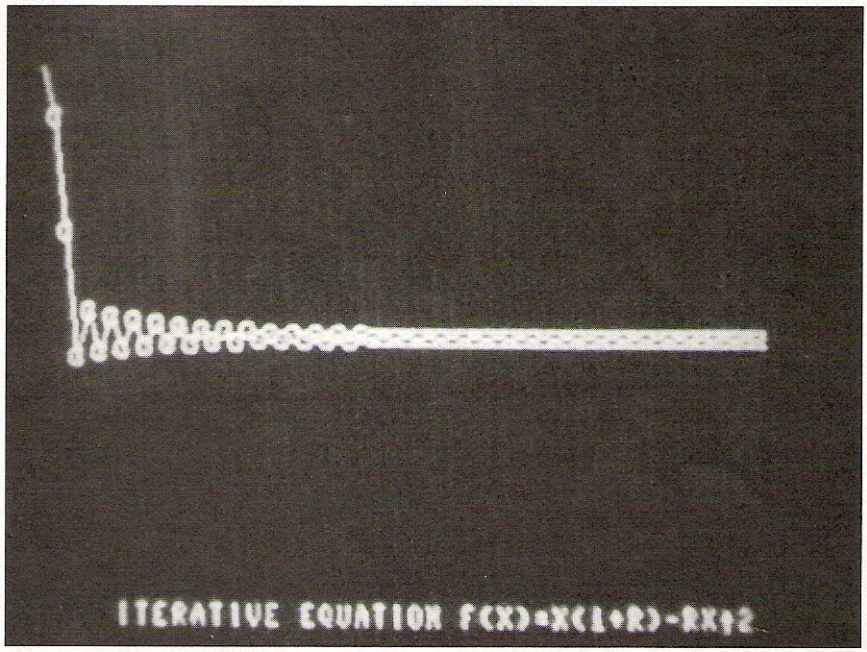


Fig. 5-12. Chaos plot (see page 143).

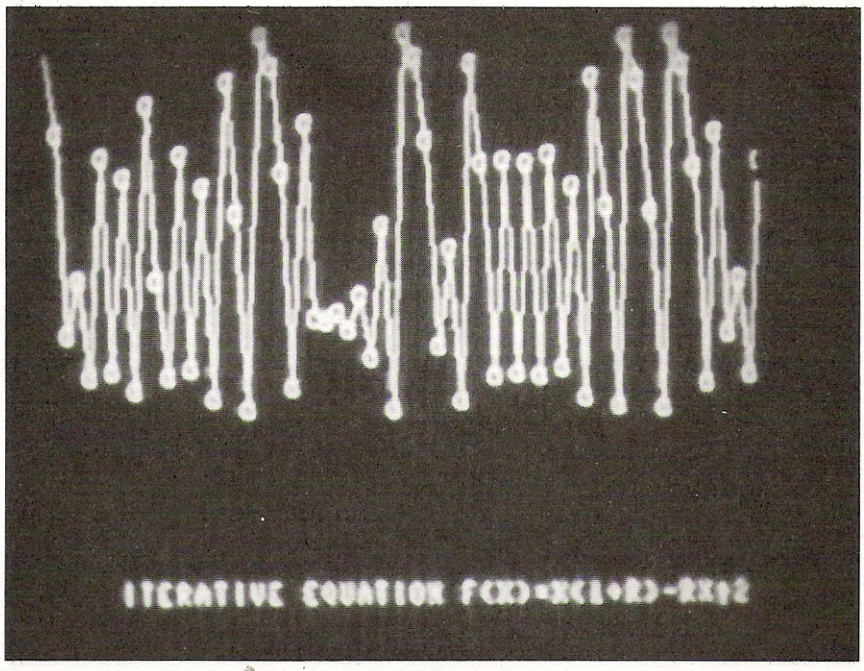


Fig. 5-13. Chaos plot (see page 143).

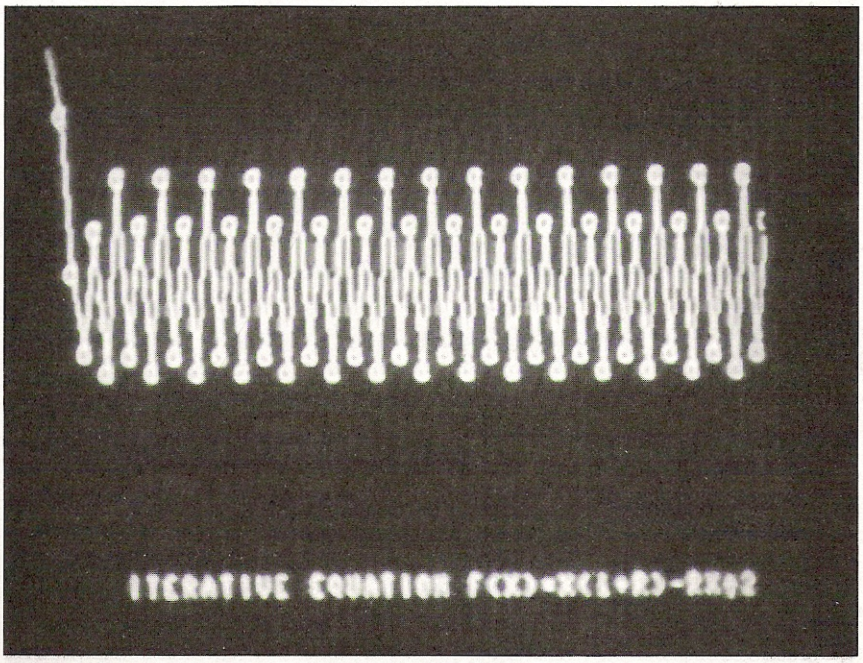


Fig. 5-14. Chaos plot (see page 143).

```
1 REM *****  SAVE THIS PROGRAM AS "128 1.0" ******
2 REM BASIC CONTROL DIGITAL RECORDER AND PLAYBACK
4 POKE56579,255:POKE253,0:POKE254,66:POKE5120,0:POKE5121,16
6 BANK15:   PRINTTAB(7)"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}MAIN MENU:"
8 PRINT:PRINTTAB(7) "DIGITAL RECORDER AND PLAYER"
10 PRINT:PRINT"1) LOAD ML PROG     7) SAVE SOUND SAMPLE"
12 PRINT"2) RECORD            8) CHANGE PITCH"
14 PRINT"3) PLAYBACK          9) DIRECTORY"
16 PRINT"4) FAST              10) QUIT"
18 PRINT"5) SLOW"
20 PRINT"6) LOAD SOUND SAMPLE"
22 PRINT:PRINTTAB(2)"ENTER NUMBER 1-10 THEN PRESS RETURN":INPUTJ
24 ONJGOTO26,28,30,34,36,40,50,68,80,38
26 PRINT"{CLR}":PRINTTAB(21)"{C/DN}LOADING":LOAD"SID 1.0",8:GOTO6
28 PRINT"{CLR}":PRINTTAB(12)"{C/DN}RECORDING":SYS4864:GOTO6
30 PRINT"{CLR}":PRINTTAB(12)"{C/DN}PLAYBACK":SYS4941:PRINT"{CLR}{C/DN}{C/DN}{C/DN}
32 FORT=1TO150:NEXT:POKE212,88:POKE208,0:GOTO6
34 FAST:GOTO6
36 SLOW:GOTO6
38 PRINT"{CLR}":END
40 PRINT"{CLR}"
42 PRINT:PRINT"DO NOT ADD PREFIX SND. TO FILENAME"
44 PRINT"PROGRAM WILL ADD PREFIX AUTOMATICALLY"
46 PRINT"ENTER NAME OF SOUND SAMPLE :":INPUT A$
48 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}LOADING..SND."A$:LOAD "SND."+A$,8,1:GOTO6
50 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}INFO ON SAVING OF SOUND SAMPLE"
52 PRINT:PRINT"WHEN SAVING SOUND SAMPLE PROGRAM WILL
54 PRINT"END. TO RESTART ENTER RUN AND RETURN"
56 PRINT"YOUR SOUND WILL BE SAVED TO THE DISK."
58 PRINT"{C/DN}{C/DN}{C/DN}ENTER NAME OF SOUND SAMPLE"
60 INPUT B$
62 A$="BSAVE"+CHR$(34)+"SND."+B$+CHR$(34)+",D0,P16896TOP62464"
64 BANK0:PRINT"{CLR}"A$
66 POKE842,19:POKE843,13:POKE208,2:END
68 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}DEFAULT SETTING IS 38":G=PEEK(5019)
70 PRINT:PRINT"CURRENT SETTING IS ";G
72 PRINT:PRINT:PRINT"A SMALLER NUMBER WILL INCREASE PITCH"
74 PRINT"A LARGER NUMBER WILL DECREASE PITCH"
76 PRINT:PRINT:INPUT"ENTER A NUMBER";A
78 POKE5019,A:POKE5154,A:POKE5160,A:GOTO6
80 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}DIRECTORY WILL LIST SOUND SAMPLES ONLY"
82 PRINT:PRINT:DIRECTORY "SND.*"
84 PRINT:PRINT"PRESS ANY KEY TO RETURN TO MENU"
86 GETKEY K$ :GOTO6

5 REM **** SAVE THIS PROGRAM AS "SID 1.0" ****
10 REM BASIC ML LOADER
12 REM FOR SID PLAYBACK
14 BANK15:  FORX=4864TO5168
15 READA:POKEX,A:T=T+A:NEXT
17 IFT<>35199THEN PRINT"ERROR IN DATA STATEMENTS":END
18 LOAD"128 1.0",8
20 DATA 120,032,004,020,024,074,074,074,074,172
30 DATA 000,020,141,001,255,145,253,169,000,141
40 DATA 000,255,032,004,020,172,000,020,141,001
50 DATA 255,041,240,017,253,145,253,169,000,141
60 DATA 000,255,200,140,000,020,192,255,208,019
70 DATA 160,000,140,000,020,230,254,165,254,201
80 DATA 244,208,006,169,066,133,254,088,096,162
90 DATA 001,202,208,253,076,001,019,120,172,000
```

Fig. 5-15. C-128 Advance program.

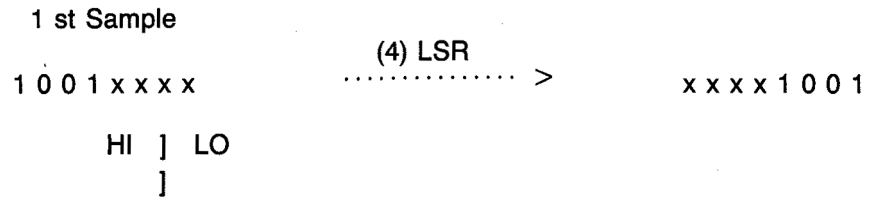
```
100 DATA 020,141,001,255,177,253,142,000,255,041
110 DATA 015,024,141,024,212,032,033,020,172,000
120 DATA 020,141,001,255,177,253,142,000,255,174
130 DATA 001,020,024,074,074,074,074,141,024,212
140 DATA 200,140,000,020,192,255,208,026,206,001
150 DATA 020,160,000,140,000,020,224,000,240,022
160 DATA 230,254,165,254,201,244,208,006,169,066
170 DATA 133,254,088,096,160,036,136,208,253,076
180 DATA 078,019,162,016,142,001,020,088,165,212
190 DATA 201,017,240,004,120,076,140,019,120,198
200 DATA 254,202,208,251,162,016,169,000,172,000
210 DATA 020,141,001,255,177,253,142,000,255,041
220 DATA 015,024,141,024,212,032,033,020,172,000
230 DATA 020,141,001,255,177,253,142,000,255,024
240 DATA 074,074,074,074,141,024,212,200,140,000
250 DATA 020,192,255,208,060,206,001,020,160,000
260 DATA 140,000,020,230,254,174,001,020,224,000
270 DATA 208,192,076,162,019,191,000,008,160,008
280 DATA 160,008,169,000,141,001,221,169,001,141
290 DATA 001,221,136,192,000,208,241,173,013,221
300 DATA 173,012,221,160,002,140,001,221,096,162
310 DATA 036,202,208,253,096,160,036,136,208,253
320 DATA 076,188,019,000,255
```

Fig. 5-15. Continued.

this in turn gives another advantage for using the SID. Since the SID requires 4-bits we can pack two 4-bit nybbles of information into each byte of memory. And what that boils down to is a doubling of our record or playback time.

The register on the SID chip that we are using to generate sound is the 4-bit volume control register. Normally, this register doesn't produce any sound by itself, but we are varying the output voltage so quickly, 5,000 to 10,000 times per second, that it does generate sound. And since our output voltage is an approximate value of what our input voltage was, we get back our original sound.

The ML program performs a couple of bit manipulations to increase our record and playback time. As I stated previously, the SID chip can only use 4 bits of information in our application. This provides us with the opportunity to double our record and playback times by packing two 4-bit nybbles of information into each memory byte. The program accomplishes this by first reading the 8-bit value left in the serial register from the circuit, then performs 4 (LSR) *Logic Shift Right*, which moves the hi-nybble to the lo-nybble position.



The next sample is pulled from the serial register. This byte is first ANDed with decimal 240 (hex FO), which effectively erases the lo-nybble while preserving the hi-nybble value.

```
                2nd Sample

                1 0 1 0 x x x x
                1 1 1 1 0 0 0 0    AND 240
                1 0 1 0 0 0 0 0    Result
```

Then this result is ORed with the first sample. This combines both 4-bit nybbles.

```
     1st Sample    0 0 0 0 1 0 0 1
     2nd Sample    1 0 1 0 0 0 0 0          OR
                   1 0 1 0 1 0 0 1          Result
```

This result is stored into memory. This entire process is repeated and stored into memory sequentially, until our allotted memory is full.

The playback works in a similar manner; first pulling the byte from memory and storing the lo-nybble into the SID volume register, then shifting the hi-nybble into lo-nybble position and transferring it into the SID.

Type in and save both programs under their proper names. This is essential, since the programs chain to one another. When you run the basic program, you are presented with a menu.

Item 1 of the menu loads in, then runs the BASIC loader for the ML program and returns to the menu. This is the first thing you should do when starting the program. The ML program is the driving force that operates the circuit and performs all the digitizing functions.

Item 2 selects the recording function. Upon entering 2 and pressing the return key, the computer will immediately begin recording.

Item 3 selects the playback function. By pressing the "R" key as the computer is in playback, your sound sample will enter into a half a second repeating loop until you release the key. It may take a second or so before the computer "sees" you've pressed the "R" key, so be a little patient.

Item 4 puts the 128 computer into the fast mode. You will achieve your best recordings in this mode.

Item 5 places the 128 in the slow mode. This will give the longest recording time.

Item 6 prompts you for a filename of a sound sample, then loads that sound sample you have recorded to your disk into memory for playback. Do not add the prefix SND. to the filename, as the program will do that automatically (see Item 7).

Item 7 prompts you for a filename to save a sound sample in memory to disk. The program adds a prefix SND. to your filename before saving for easier identification of sound sample files you have stored on your disk. The save function uses a dynamic keyboard technique to BSAVE the file. In order to accomplish this the program will end after every save. Just enter run after the computer is finished, saving your sound sample to reenter the program.

Item 8 adjusts the pitch of the playback. This function has no effect on the record function.

Item 9 views the directory of the disk currently in the drive. The directory is selective and will only display the sound sample files on the disk.

Item 10 ends the program.

When you're ready to record, turn the audio amp on to full, press #2, then return. When the program finishes recording, it will return to the main menu. Press #3 and return for playback. If there is a lot of static, it is probably due to overmodulation. Turn down the volume on the amp or hold the microphone further away from your mouth. After you have a satisfactory recording, press the "R" during playback to hear the digital repeat. The computer will continue playing the same half-second trackover and over, until you release the "R" key. You will get better sound recording by staying in the fast mode. The rest of the menu is really self explanatory. Have fun.

```
2 REM ***** 64 DIGITAL RECORDER ******
4 POKE56579,255:POKE253,0:POKE254,22:POKE5120,0:POKE5121,16
6 PRINTTAB(7)"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}MAIN MENU:"
8 PRINT:PRINTTAB(7) "DIGITAL RECORDER AND PLAYER"
10 PRINT:PRINT"1) LOAD ML ROUTINE  7) DIRECTORY"
12 PRINT"2) RECORD           8) QUIT"
14 PRINT"3) PLAYBACK "
16 PRINT"4) LOAD SOUND SAMPLE"
18 PRINT"5) SAVE SOUND SAMPLE"
20 PRINT"6) CHANGE PITCH"
22 PRINT:PRINTTAB(2)"ENTER NUMBER 1-8  THEN PRESS RETURN":INPUTJ
24 ONJGOTO68,26,28,34,38,46,58,32
26 PRINT"{CLR}":PRINTTAB(12)"{C/DN}RECORDING":SYS4864:GOTO6
28 PRINT"{CLR}":PRINTTAB(12)"PLAYBACK":SYS4941:PRINT"{CLR}RETURNING TO MENU"
30 FORT=1TO150:NEXT:POKE197,64:POKE198,0:GOTO6
32 PRINT"{CLR}":END
34 PRINT"ENTER NAME OF SOUND SAMPLE :":INPUT A$
36 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}LOADING.."A$:LOAD A$,8,1:GOTO6
38 PRINT"{CLR}{C/DN}{C/DN}ENTER NAME OF SOUND SAMPLE"
40 INPUT B$ :PRINT"{CLR}{C/DN}SAVING SAMPLE ";B$
42 SYS57812B$,8:POKE173,22:POKE172,0:POKE780,172
44 POKE782,148:POKE781,0:SYS65496:GOTO6
46 PRINT"{CLR}{C/DN}{C/DN}{C/DN}{C/DN}DEFAULT SETTING IS 38":G=PEEK(5019)
48 PRINT:PRINT"CURRENT SETTING IS ";G
50 PRINT:PRINT:PRINT"A SMALLER NUMBER WILL INCREASE PITCH"
52 PRINT"A LARGER NUMBER WILL DECREASE PITCH"
```

Fig. 5-16. C-64 Advance program.

```
54 PRINT:PRINT:INPUT"ENTER A NUMBER";A
56 POKE5019,A:POKE5154,A:POKE5160,A:GOTO6
58 SYS57812"$",8:POKE43,1:POKE44,192:POKE768,174:POKE769,167:SYS47003,1
60 POKE782,192:SYS65493:SYS42291:LIST:POKE44,8:POKE768,139:POKE769,227
62 PRINT:PRINT"PRESS ANY KEY TO RETURN TO MENU"
64 GETK$:IF K$="" THEN64
66 GOTO6
68 PRINT"{CLR}{C/DN}{C/DN}LOADING ML ROUTINE..."
70 FOR X=4864TO5168:READA:POKEX,A:T=T+A:NEXT
72 IF T<>38560 THEN PRINT"ERROR IN DATA STATEMENTS":END
74 GOTO6
76 DATA 120,032,004,020,024,074,074,074
78 DATA 074,172,000,020,234,234,234,145
80 DATA 253,169,000,234,234,234,032,004
82 DATA 020,172,000,020,234,234,234,041
84 DATA 240,017,253,145,253,169,000,234
86 DATA 234,234,200,140,000,020,192,255
88 DATA 208,019,160,000,140,000,020,230
90 DATA 254,165,254,201,148,208,006,169
92 DATA 022,133,254,088,096,162,001,202
94 DATA 208,253,076,001,019,120,172,000
96 DATA 020,234,234,234,177,253,234,234
98 DATA 234,041,015,024,141,024,212,032
100 DATA 033,020,172,000,020,234,234,234
102 DATA 177,253,234,234,234,174,001,020
104 DATA 024,074,074,074,074,141,024,212
106 DATA 200,140,000,020,192,255,208,026
108 DATA 206,001,020,160,000,140,000,020
110 DATA 224,000,240,022,230,254,165,254
112 DATA 201,148,208,006,169,022,133,254
114 DATA 088,096,160,036,136,208,253,076
116 DATA 078,019,162,016,142,001,020,088
118 DATA 165,197,201,017,240,004,120,076
120 DATA 140,019,120,198,254,202,208,251
122 DATA 162,016,169,000,172,000,020,234
124 DATA 234,234,177,253,234,234,234,041
126 DATA 015,024,141,024,212,032,033,020
128 DATA 172,000,020,234,234,234,177,253
130 DATA 234,234,234,024,074,074,074,074
132 DATA 141,024,212,200,140,000,020,192
134 DATA 255,208,060,206,001,020,160,000
136 DATA 140,000,020,230,254,174,001,020
138 DATA 224,000,208,192,076,162,019,191
140 DATA 000,002,160,008,160,008,169,000
142 DATA 141,001,221,169,001,141,001,221
144 DATA 136,192,000,208,241,173,013,221
146 DATA 173,012,221,160,002,140,001,221
148 DATA 096,162,036,202,208,253,096,160
150 DATA 036,136,208,253,076,188,019,000
152 DATA 255
```

Fig. 5-16. Continued.

6

Subliminal Communication

Let's begin an exploration into the world of subliminal phenomena. I'm sure a lot of you have an idea or familiarity of what subliminals are. For the uninitiated, subliminals are information (usually audio/visual) presented in such a manner as to not be consciously perceived. Therefore the person listening or viewing the subliminal is not consciously aware of it being present. Subliminal techniques are targeted to motivate a persons behavior or thought.

What we will construct is a video switch that works in conjunction with a VCR and your computer. Essentially what we will do is create a message screen on the computer's monitor and flash this message screen subliminally onto the VCR monitor's (usually a TV set). The VCR can either be playing a tape or, by using the built in TV tuner, receiving broadcast television. The subliminal switch will work in either configuration.

Using subliminal technique you can explore the prospects of self programming the human biocomputer, your brain. You may want to try this technique to shed a few pounds of extra weight, or to help you relax. I'll go into greater detail on the mechanics of the message screens later on.

HISTORY

As far back as the the turn of the century in 1894 W.R. Dunham M.D., wrote commentaries on subliminal communication. I believe that it was

the claim of a New Jersey theater owner in the 1950's who flashed refreshment subliminals over Kim Novak in the movie Picnic and reported a 58 percent increase in the sales of Coca-Cola, which brought subliminal communication out into public view. More recently subliminals are found in advertising, popular music and theater.

ORWELLIAN MIND CONTROL

The reason subliminal techniques are feared, is that subliminals effectively bypass our normal conscious mind. For instance, let's suppose someone wants to sell you a widget. After you listen to their sales pitch you make a conscious decision whether you want to purchase the widget, you analyze if it will perform as claimed or worth the money it cost, no problem here. But if we are unconsciously bombarded with subliminals that tell us this widget will make us wealthy, sexy, popular, intelligent, our conscious decision making process is short circuited. If our subconscious mind becomes convinced of the subliminal affirmations, we find ourselves wanting to buy this widget. We may think it's our own idea that we need or want it.

Various advertisements and their progenitor agencies have been accused of making free use of subliminals to generate a greater profit per advertising dollar. I will not try to justify this Orwellian concept of mind control by media but I have supplied a bibliography to this article for those who wish to pursue this interesting topic further.

AUDIO

Another area that is currently making a large splash in subliminal techniques are self-help audio cassettes. These tapes have subliminal messages masked into the background of music. The tapes are designed to help the listener to stop smoking, lose weight; relax, gain self confidence, etc. I don't know how effective the tapes are at helping people accomplish their goals and I am not advocating their use. But let me call to your attention the fact that this type of subliminal is being used in industry, a case in point, some large department stores use subliminals to help reduce customer and employee theft, and naturally, to increase sales. You may have heard music being played in stores and malls, what you can't hear is if there are any subliminals encoded into the music. Many popular rock groups insert subliminal messages into music.

VISUAL

Visual information can be encoded with two basic methods. The first is the subliminal cut. Here an image or phrase is inserted in between two corresponding film images as in a movie film; more of a cut and paste operation. When the movie is shown the subliminal images pass too quickly for them to register consciously, but our subconscious picks them

up. This is the method that we will be using, although we are working with video tape and/ or broadcast TV, the basics are the same. This is the tachistoscopic method.

The second method is more advanced and much harder to detect. Here an image or phrase is overlaid onto the film image. The phrase is held at a slightly lower illumination level than the overall picture. Again this image or phrase is not consciously picked up. This would be the method of choice, it s technologically more advanced, harder to detect, and effective.

SUBLIMINALS AND THE LAW

Currently there are no laws prohibiting the use of subliminals. There are no laws that anyone subjected to subliminals must be informed. The reason I am making a point of this is that I believed that such laws were in place. Upon doing research for this article I discovered that there are not.

The FCC has a regulation concerning deceptive advertising on television, but it relies on the FTC to make the determination on what's deceptive. The bottom line is that the use of subliminals are not checked, except for a cursory look see for the most basic tachistoscopic images.

CIRCUIT CONSTRUCTION

Check your VCR for video and audio output jacks. Most VCRs have separate video and audio inputs, and outputs. If your VCR doesn't, stop, you can't use this circuit. (You cannot use the RF out that is connected to the TV antenna leads.)

The circuit is quite simple and inexpensive. We are using a 4066 quad bilateral switch to block and steer our video image to the monitor. The program takes the video signal from the computer and displays it onto the screen for 1/60 of a second every three or four seconds. Whatever you put on your computer screen, will be flashed to your subconscious mind. All the remaining time the standard picture from the VCR will be playing.

Look at Fig. 6-1. Our two control lines PBO and PB1 are connected to the electronic switches. The line controls control the switches on and off operation. When we output a binary "1" on the line that switch will turn on allowing that video signal to be transmitted. It's important that only one switch be turned on at a time or you will display a rather messy picture.

The entire circuit is constructed, (see Fig. 6-3) in a small circuit box (see parts list). The circuit board that comes with the box may be a little difficult to construct the circuit on. I purchased another board that made the construction much easier and fit it into the box (see parts list). First drill all the holes required for the switches and phono-jacks. Cut a slot in the bottom of the box large enough to fit the user port connector terminals inside. I used crazy glue to mount the card connector to the box. Solder your wires to the card connector before you mount it.

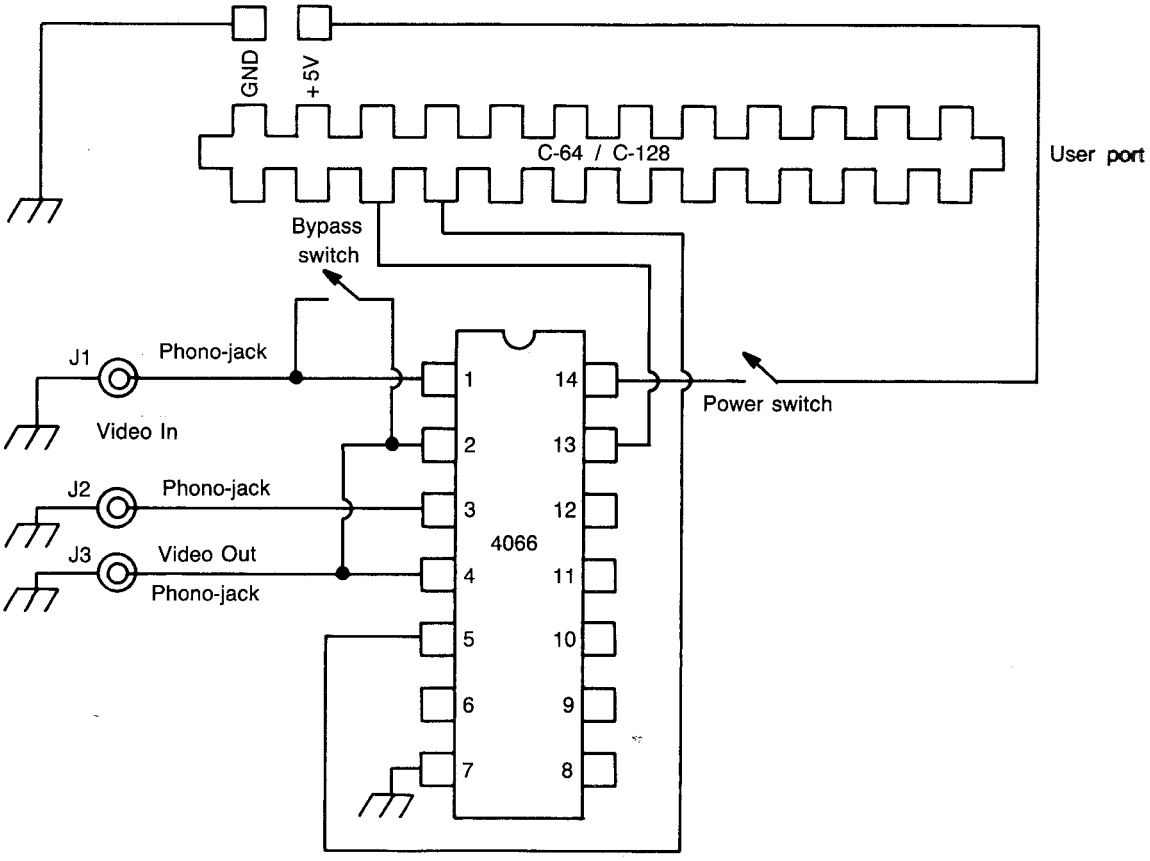


Fig. 6-1. Subliminal circuit.

HOOKUP

Look at the drawing (Fig. 6-2) for the hookup. The RF modulator (Radio Shack PN# 15-1273) accepts video and audio inputs. Use standard phono cables to connect the switch to the VCR and RF modulator. You will have to make a short cable for the computer to the switch. I tried using an 8-pin din plug to connect to the video-out of my C-128, it didn't fit. I had to insert two wires stripped about %z" into the appropriate socket holes and tape them to the computer. To the other end of the wires I soldered a standard phono plug (see parts list).

CIRCUIT OPERATION

Before installing the circuit in the user port, make sure both switches are in the off position. After installing, turn the computer on and configure the port with a poke 56579, 255. This, as you should know, turns our port into output bits. Now turn on the subliminal circuit power switch. One reason the switch is included is to power-up the user port although

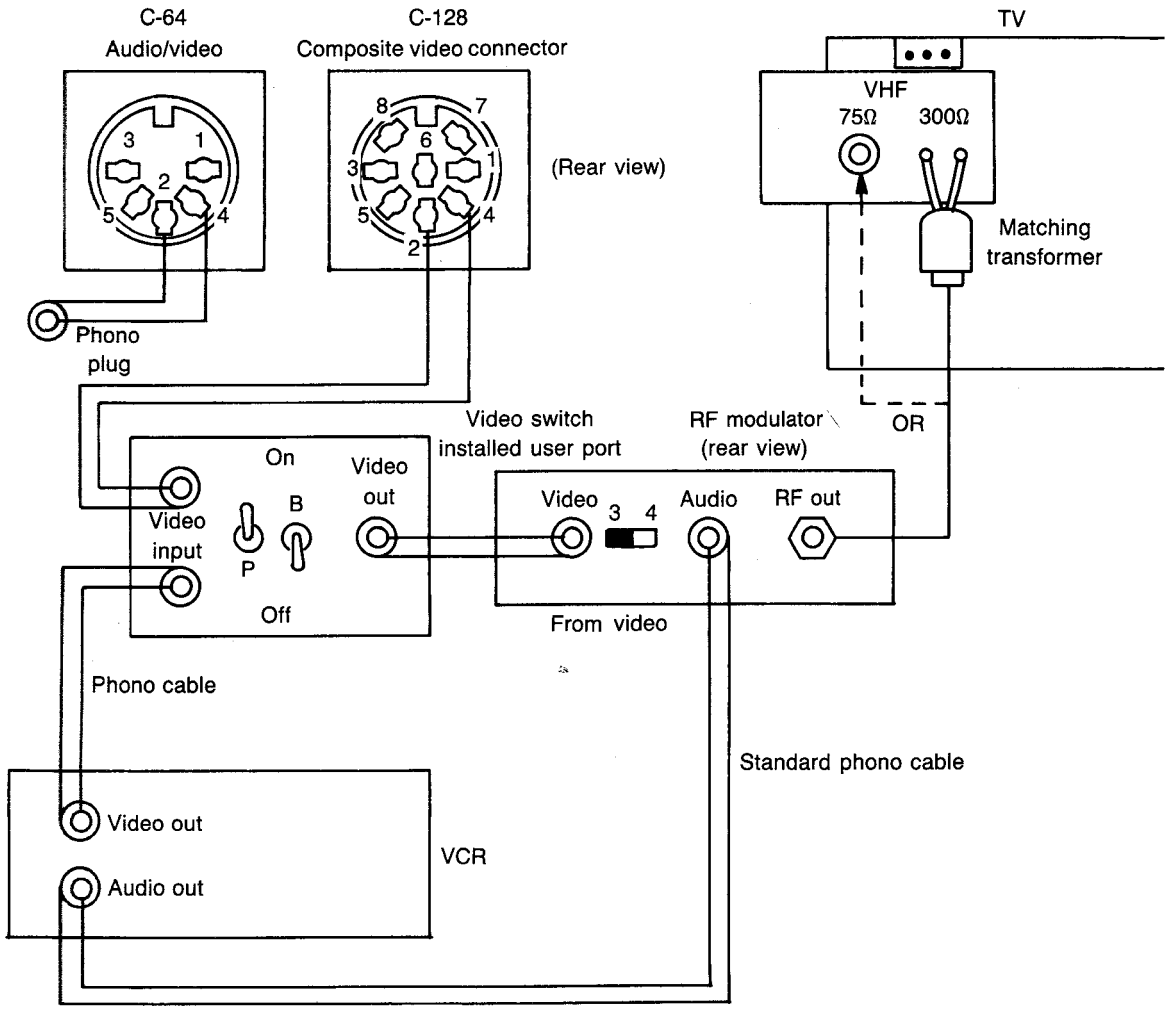


Fig. 6-2. Layout of hookup.

configured as an input device, outputs enough current through its pull up resistors to turn the subliminal circuit switches on.

Poke 56577,1 turns channel one on.
Poke 56577,2 turns channel two on.

By using the two pokes you should be able to switch screens between the computer video and the VCR video. If you encounter a problem at this point see troubleshooting.

Another reason the power switch is included is for you to be able to operate your computer without turning on the subliminal circuit. This is where the bypass switch also comes into effect. Rather than constantly

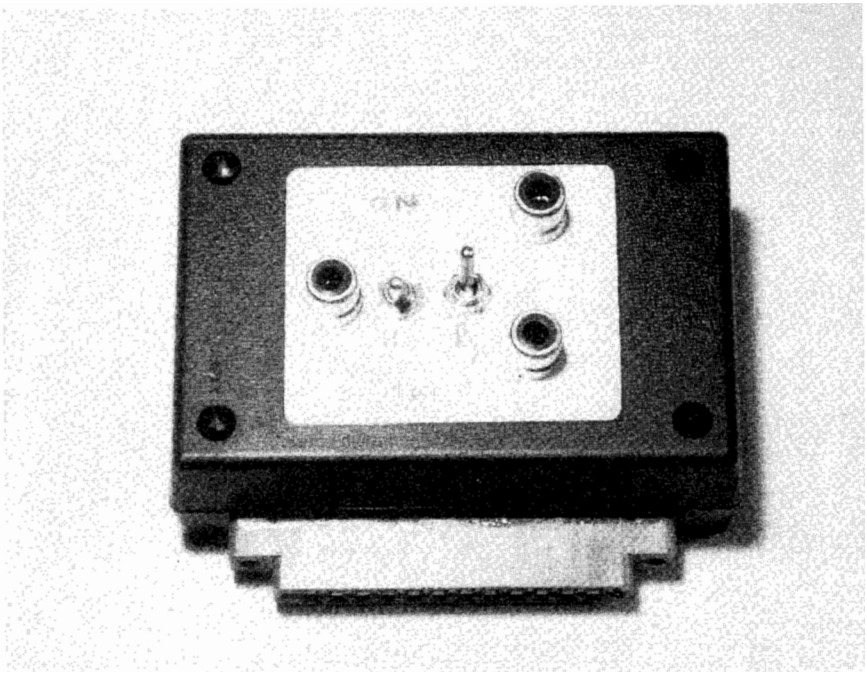


Fig. 6-3. Completed circuit.

```
10 REM BASIC PROGRAM FOR C-64 & C-128
15 REM SUBLIMINAL SWITCH
20 POKE56579,255: REM SET UP USER PORT
25 POKE56577,1: REM PUT CHANNEL ONE ON MONITOR
30 REM **    FOR C-128 ADD GRAPHICO,1 COMMAND **
35 PRINT"PRINT YOUR MESSAGE OR GRAPHIC SCREEN"
100 POKE56577,2:POKE56577,1
105 FORT=1TO9999:NEXTT:GOTO100
```

Fig. 6-4. Basic controller, C-128 and C-64.

switching cables, the bypass switch allows you to bypass the circuit and feed directly into your RF modulator. To bypass simply turn the switch on, for subliminal operation keep the switch in off position. (Note when you bypass make sure your power switch is off also.)

I have included two programs for each computer. One program is written entirely in BASIC. This program is to show how the system works, (see Fig. 6-4) in using it you'll see a noticeable flicker when the screens change. That problem is eliminated in the second program (Fig. 6-5 for the C-128 and Fig. 6-6 for the C-64) which contains a short ML program that does the screen cut.

The ML program switches the computer video onto the monitor for 1/60 second every three or four seconds. With the program up and run-

```
6 REM SUBLIMINAL SWITCH C-128
8 REM WITH ML SUBROUTINE
10 DATA 120,169,255,141,003,221,169,002,141,001
12 DATA 221,160,008,162,202,202,208,253,136,208
14 DATA 248,206,001,221,088,096,062
16 FORI=4864TO4890:READA:POKEI,A
18 B=B+A:NEXT
20 REM IF B<>3902 THEN PRINT "ERROR IN DATA STATEMENT
24 POKE56579,255
26 REM POKE56577,1 AND ,2 TO CHANGE SCREENS
28 GRAPHIC0,1
30 PRINT"{C/DN}{C/DN}{C/DN}{C/RT}{C/RT}{C/RT}PUT YOUR MESSAGE
   HERE"
32 PRINT"PRINT ANY GRAPHICS YOU'D LIKE"
34 SYS4864
36 FORT=1TO9999:NEXTT:GOTO34
```

Fig. 6-5. Machine language for C-128.

```
6 REM SUBLIMINAL SWITCH C-64
8 REM WITH ML SUBROUTINE
10 DATA 120,169,255,141,003,221,169,002,141,001
12 DATA 221,160,008,162,202,202,208,253,136,208
14 DATA 248,206,001,221,088,096,062
16 FORI=49152TO49178:READA:POKEI,A
18 B=B+A:NEXT
20 REM IF B<>3902 THEN PRINT "ERROR IN DATA STATEMENT
24 POKE56579,255
26 REM POKE56577,1 AND ,2 TO CHANGE SCREENS
30 PRINT"{C/DN}{C/DN}{C/DN}{C/RT}{C/RT}{C/RT}PUT YOUR MESSAGE
   HERE"
32 PRINT"PRINT ANY GRAPHICS YOU'D LIKE"
34 SYS49152
36 FORT=1TO9999:NEXTT:GOTO34
```

Fig. 6-6. Machine language for the C-64.

ning, if you find yourself looking at the computer video, switch the VCR and computer cables. At any time you can stop the program, and by using the appropriate poke command get back to your computer screen. Of course this command will be given in the dark.

MESSAGE SCREENS

You have as much latitude as you want. All message screens should try to convey the message in a positive way. As an example suppose you wanted to use this technique to lose a couple of pounds. You should not

use negative messages like "I'm fat", rather use a message that states "Not hungry" or "I like to exercise".

Whatever you have printed to the screen will be flashed via the circuit switch. If you have a program that prints to the screen in large letters that would be beneficial, or you can design your own using Commodore graphics.

For C-64 users the video out screen is the same as the RF out screen. For the C-128 users who are using an 80 column RGB (or Monochrome connected to the RGB) your video out is the graphics screen. I suggest to use the Graphics 3 screen to print type.

TROUBLESHOOTING

You would think that such a simple circuit wouldn't require any troubleshooting, and for the most part it doesn't. But there are a few points to keep in mind. First and foremost keep the ground wires straight. If you inadvertently criss-cross these wires that portion of the video won't work. If this happens on the video out, the entire circuit will not work. The ground wire is connected to the outside of the jacks and also the outside of the plug connector you'll be using from the computer. I advise that you buy standard phono cables for the rest of the hookup rather than making the cables.

BIBLIOGRAPHY

Subliminal Seduction By Wilson Bryan Key
Signet Books

Subliminal Communication
By Eldon Taylor
JAR Books
Box 7116
Salt Lake City, Utah 84107

Applications of Subliminal Video and Audio Stimuli in Therapeutic, Educational, Industrial, and Commercial Settings.
Eighth Annual Northeast Bioengineering Conference, Massachusetts Institute of Technology, Cambridge
(1980)

Parts List

Quantity	Item/Description	Part Number RS = Radio Shack		Cost
2	Submini switch	RS# 275-645	ea/	1.79
4	Phono-jacks	RS# 274-246	pkg/4	1.99
2	Phono plugs	RS# 274-339	pkg/2	1.49

Quantity	**Item/ Description**	**Part Number**	**Cost**
1	Box w/ PC board	RS# 270-291	3.99
1	4066 Quad bi-switch	RS# 276-2466	1.19
1	IC Board (optional)	RS# 276-159	1.49
1	6 foot audio/video cable	RS# 15-1537	6.95
1	RF Modulator	RS# 15-1273	26.95
1	Card Connector	568-50-24A-30	3.49

Mouser Electronics
11433 Woodside Ave.
Santee, CA 92071
(619) 449-2222

7

Appliance Controller

In the previous chapters the concentration has been on interfacing +5 volt TTL (Transistor-Transistor Logic) devices to the user port, with which the computer could sense, display, speak or make some form of decision. The limitation however was that the computer could not directly impact on the real world environment.

REAL WORLD ENVIRONMENT

The "real world environment" is where we live. We have equipped the computer with sensors that can partially inform the computer of our environment. Now it is time to equip the computer with control devices that allow it to impact on the real world environment.

In this chapter we will employ the user port to control high power electrical devices. By using the TTL voltage (+5V) off one PB line of the user port, the computer will be able to turn on or off electric power. The circuits described control either standard 115V ac electricity from your home or dc electricity. Naturally, by controlling the electric current to a device we are therefore controlling the device.

You can use these circuits to control any number of household appliances. In addition, by utilizing the sensors from Chapter 4 we can program the computer to impact directly based on sensor readings. This could be something as simple as turning on a light when the computer senses dusk or senses someone walking into the room.

For our example however, we will interface the toxic gas sensor from Chapter 4, to make an automatic ventilation control system. The operation of the system is simple. When the computer senses a toxic gas it will turn on an electric fan and keep it on until the gas concentration returns to a safe level.

INDUCTIVE AND RESISTIVE LOADS

Any device we are powering or controlling is called a load. Whatever the electrical device is will fall into one of two main electrical categories, inductive or resistive loads.

It is pretty easy to distinguish an inductive load from a resistive. An inductive device has coils or electrical wire windings in it such as motors, transformers, relays, solenoids. A resistive device doesn't have any coils or windings as in electric lights (not fluorescent), coffee makers, heaters.

The distinction between loads is important because, when electric power is turned off an inductive device, an electric pulse (transient) is generated. This electric pulse must be absorbed by a capacitor-diode combination in our circuit. Failure to do so may damage our circuit or computer. Fortunately, adding the capacitor-diode is very easy to do.

Dc LOADS

Examine Fig. 7-1. This circuit can control dc or ac loads up to 120 volts at 2 amps. Tn this particular circuit we are not concerned whether the main load is resistive or inductive it could be either. But notice the diode and capacitor connected across the relay. The relay itself is an inductive load connected directly to our computer. The diode and capacitor are necessary to prevent any damage to the computer. You can if you wish use a LED in place of the diode (make sure polarity of LED is correct) and see the LED flash every time you turn the circuit off with your computer. This flash represents the voltage pulse being absorbed. The reason you can replace the diode with the LED is that the LED is a diode. LED is an acronym for Light Emitting Diode.

Ac LOADS

Figures 7-2 and 7-3 circuits are exclusively for ac loads, resistive and inductive respectively. The six pin chip MOC 3010 Fig. 7-9 is an optocoupled triac. This device senses when the PB line connected to it is outputtin:g a binary "1" (+5V) with an internal LED. T'he internal LED triggers a photosensitive internal triac that in turn triggers the external triac, that powers our load. See Fig. 7-4 pin out.

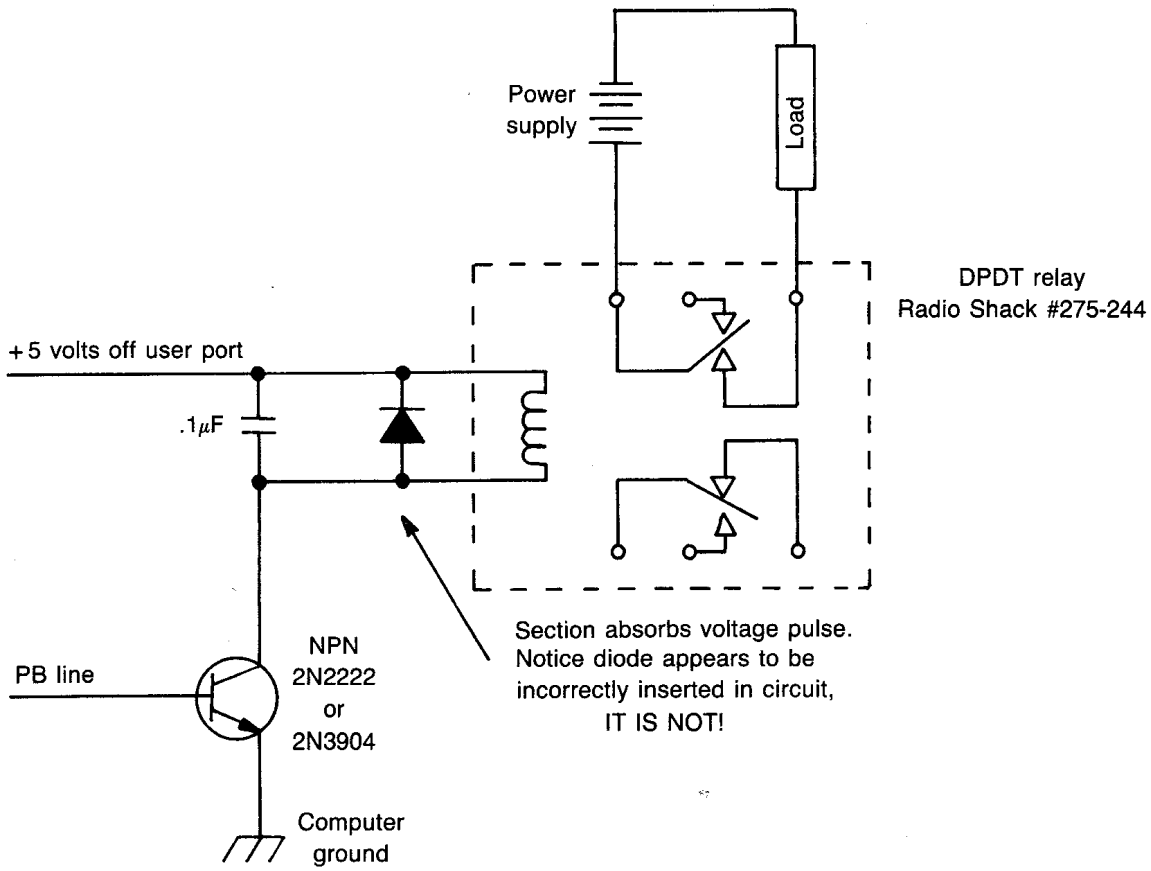


Fig. 7-1. Dc 120V controller.

CIRCUIT CONSTRUCTION

Be careful when building these circuits. The power available from your household electric is more than enough to reduce your computer to a cinder or to give yourself a nasty shock.

I advise constructing the inductive load circuits since they can be used for both types of devices. This will alleviate any potential problems in the future. I am however, including the schematics for resistive loads that you can use for either comparison with the inductive circuits or if you wish, as dedicated resistive load controllers.

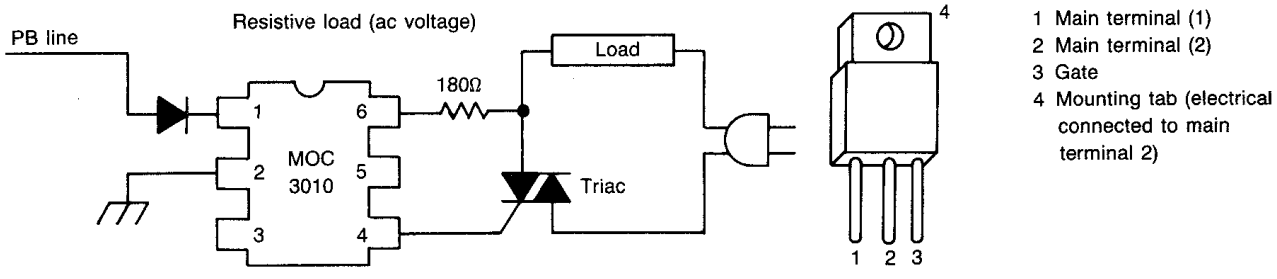


Fig. 7-2. Ac resistive load 120Vac.

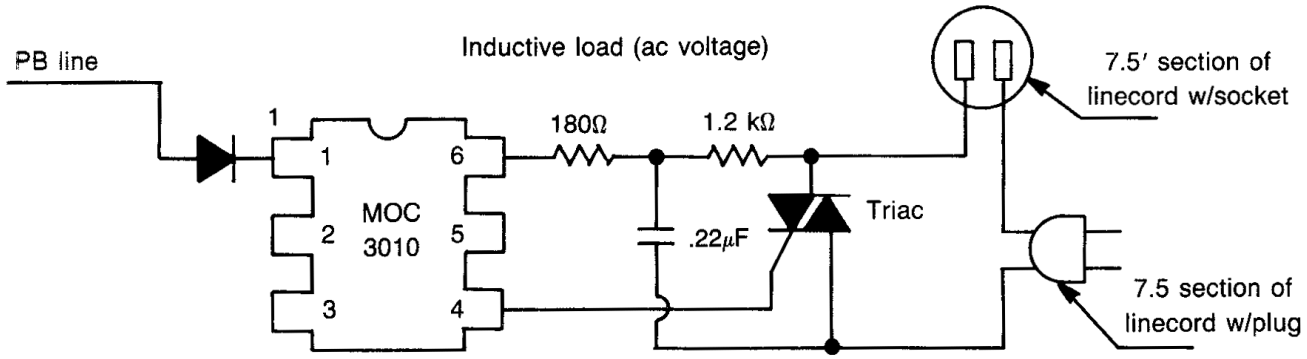


Fig. 7-3. Ac inductive load 120Vac.

Since I believe that most readers will be interested in controlling ac appliances or devices in their home, Fig. 7-3 is the circuit we will build. If you go on to the other circuits on your own, remember to take the same amount of care in building them.

Since it is important that this unit is put together properly, I am doing step by step instructions for the construction of this project. I don't want anybody accidentally electrocuting themselves, so please be careful.

We can't use our prototype breadboard for these projects. The voltages and currents are greater than what can be safely handled on the breadboard. Instead we will use a small plastic experimenters box available from Radio Shack. The box comes with a printed circuit board (PCB) that fits inside nicely. All the screws and hardware necessary to secure the PC board into the box and put the box together are also included.

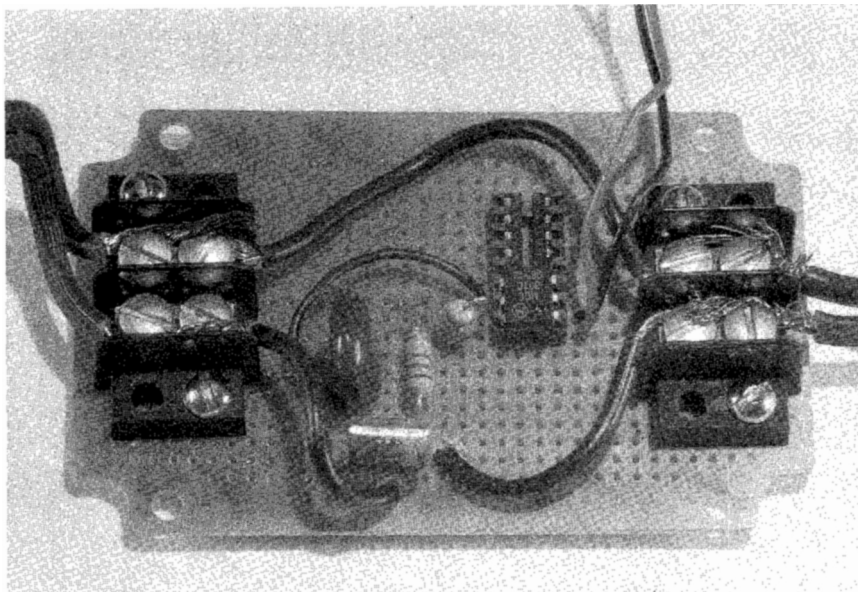


Fig. 7-4. Close up circuit.

All the components must be soldered to the PC board. We will use a 15-foot extension cord that we cut in half. The plug half of the line cord will bring power from the outlet to the circuit. The socket half of the line cord will lead from the box to the device you wish to power/control. See Fig. 7-3 and Fig. 7-4.

To begin, drill the holes in the plastic top of the experimenters box. You will need a hole on each side to accommodate the line cord going in and out. Four holes on the top surface for the push-button terminal strip (see Fig. 7-5). Use the terminal strip to mark the holes before drilling. You will have to ream the holes in the terminal strip to accommodate the 6-32 machine screws. Add one small hole in front of the terminal strip for the LED indicator light.

After you're finished drilling the box, get the PC board. Lay out the 2 terminal barrier strips as they are in Fig. 7-4, mark and drill the holes for the screws.

Assemble the barrier strips to the PC board with machine screws and hex nuts. Assemble the push-button terminal strip to the top of the box. Glue the LED indicator light into the hole. Solder a 22-gauge red wire from the red terminal to the LED. Check the polarity on the LED to make sure you solder the wire to the (anode) proper LED terminal. Then solder another red wire from the opposite side of the LED. Solder 22-gauge black wire to the black terminal. Make the lead length about 6 to 7 inches long off the LED and black terminal so that you will have sufficient wire to make the final assembly easy.

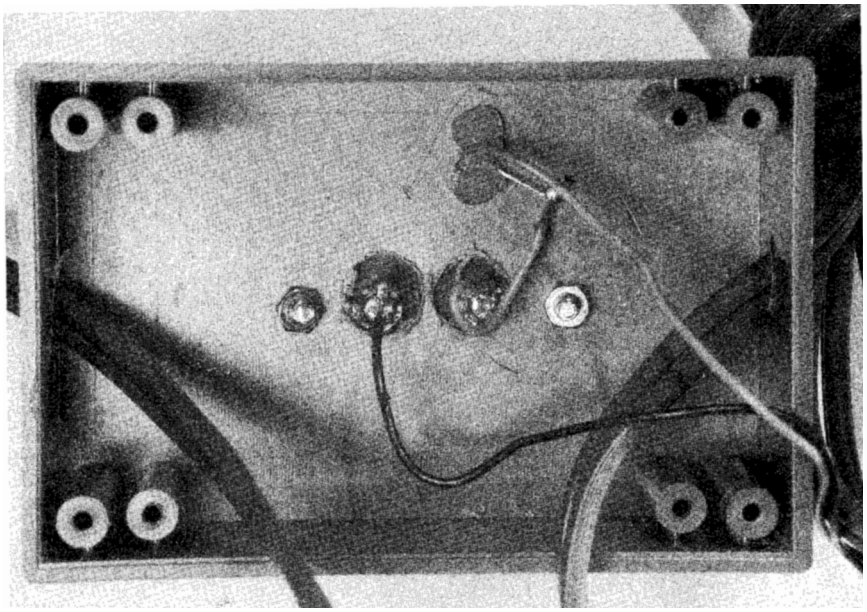


Fig. 7-5. Top view of box.

In Fig. 7-4 you can see that I used a 16-pin IC socket, for the MOC-3010. I used that socket because I had it lying around, an 8-pin socket would be fine. I advise you to use. an IC socket for soldering the unit together then placing the MOC unit in after you're finished. This will prevent you from overheating the IC with your soldering iron.

Lay out your parts on the PC board and begin soldering them point to point. Look at the picture diagram of the triac. The face up picture is how the triac looks straight on. Notice the lead numbers at the bottom, and compare them to the lead numbers on the schematic. Take extra care that you connect these leads properly.

Attach the line cords to the barrier strips. Notice that the wires that are carrying the main load current to and from the triac are heavier gauge wire than we usually use. Use 16- or 18-gauge wire for these connections.

Solder the red wire from the LED to pin 1 of the optocoupler, the black wire from the terminal to pin 2. Recheck all of your wiring at this point. Make sure you don't have any accidental solder bridges. If you're satisfied, assemble the PC board into the box and put the bottom plate on (see Fig. 7-6). Make sure none of your wiring on the bottom is touching the bottom plate. If it is, correct it.

TEST

Plug the line cord into your home electric socket, the device you want to power in the other end of the extension. Attach PBO line to the red push-button terminal. Attach a ground wire from the computer to the black terminal. Turn on the computer.

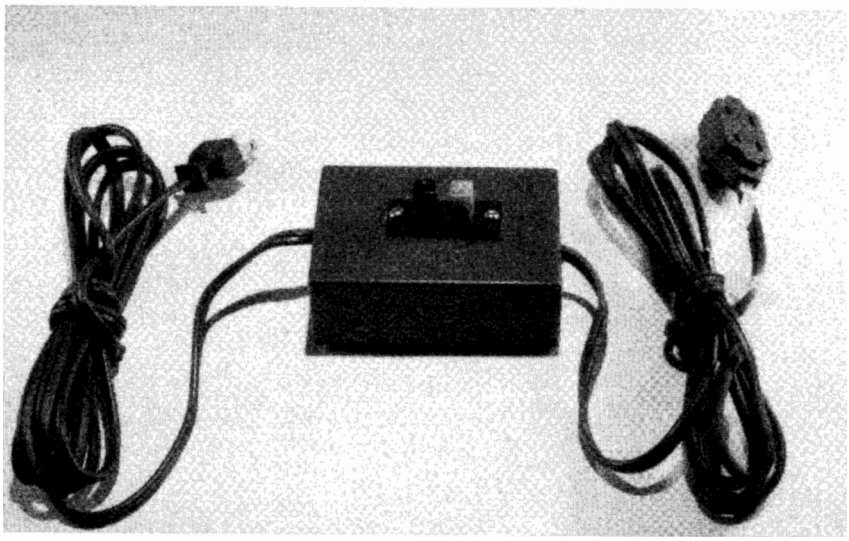


Fig. 7-6. Finished project.

C-64 and 128		Vic-20
Poke 56579,255	DDR set up	Poke 37138,255
Poke 56577,1	turn device on	Poke 37136,1
Poke 56577,0	turn device off	Poke 37136,0

This should allow you to turn on and off a device with the above pokes. If it doesn't, you have a wiring error, disconnect the plug from your home socket before opening the box to find the error.

The triac is rated at 200 volts at 6 amps which means that it is capable of handling 1200 watts. In order to pass that much current would require heat sinks which we haven't put in. I advise to keep the maximum power under 500 watts.

PROGRAM

Now we shall incorporate the toxic gas sensor from Chapter 4. My reason for doing this is twofold. First as an exercise in computer control and second as an exercise in logic instructions. It is essential that we use logic instructions in the program, so that we can read and then react through the user port. At the same time, being able to maintain or change the status of individual bits without disturbing the status of any other bits. If we fail to do this we could lose the integrity of our sensor readings and we would be constantly stopping and starting the power to the device, which in this case happens to be an electric fan.

So for all you people who glossed over the logic instructions in Chapter 2, it's time to go back and read it over.

```
2 REM SERIAL ANALOG TO DIGITAL CONVERSION
4 REM WITH OUTPUT PB2 FOR ELEC. CONTROL
6 REM FOR VIC-20 COMPUTER
10 POKE37138,255
20 POKE37150,127:REM INTERUPT FLAG ENABL
30 POKE37147,12:REM AUXILIARY CONTROL REG
35 POKE37136,2
40 FORX=0TO7
50 POKE37136,PEEK(37136)AND252:POKE37136,PEEK(37136)OR1
60 NEXT X
70 X=(PEEK(37149)AND4):REM SERIAL FG
80 X=PEEK(37146)
90 PRINT X;
92 IFX>50THENPOKE37136,PEEK(37136)OR4
94 IFX<50THENPOKE37136,PEEK(37136)AND251
95 POKE37136,PEEK(37136)OR2
100 GOTO40
```

Fig. 7-7. Program Vic-20.

```
2 REM  ** SERIAL ANALOG TO DIGITAL CONVERSION **
3 REM  ** WITH OUTPUT PB2 FOR ELEC. CONTROL   **
4 REM  ** JOHN IOVINE                 5-22-87  **
5 POKE56579,255
7 POKE56577,0
10 POKE56589,127
12 FORX=0TO7
14 POKE56577,PEEK(56577)AND252 :POKE56577,PEEK(56577)OR1
15 NEXTX
20 IF(PEEK(56589)AND8)=0THEN20
30 X=PEEK(56588)
40 PRINTX;
42 IFX>50THENPOKE56577,PEEK(56577)OR4
43 IFX<50THENPOKE56577,PEEK(56577)AND251
45 POKE56577,PEEK(56577)OR2
50 GOTO12
```

Fig. 7-8. Program C-64, C-128.

Construct the toxic gas sensor as described in Chapter 4. Attach PB2 to the red terminal on our electric control box and a ground wire to the black terminal. Enter the following program for the Vic-20, see Fig. 7-7 or Fig. 7-8 for the C-128 and C-64.

When the sensor detects gas it will automatically turn on the fan, and keep the fan on until the gas concentration returns to a safe level.

For the doubting Thomas's out there, who question the validity and necessity of the logic instructions, enter the program as they originally appeared in Chapter 4. Add the two program lines for decision making (If / Then) and see for yourself how inadequate simple poke commands operate the device.

SMART CONTROL

In most cases I would have finished at the last paragraph. I would like to make one more point on basic computer control circuits. To make this point I wish to draw an analogy.

Let's say that you've just returned from your local newspaper stand with the latest edition of your favorite magazine. You sit in your favorite easy chair, reach over to turn on the lamp to read by and behold, no light. "Darn", you say to yourself. You look down to the socket, check to make sure the lamp is plugged in, it is. You look over to the clock on the wall that's on the same fuse as the lamp. The clock is ticking away so you know you have juice going to the lamp. You flick the lamp switch a couple of times to make sure the switch isn't stuck. Now you take the lampshade off the lamp, and sure enough that black spot on the bulb lets you know that it's burn out. You replace the bulb, the lamp works fine, and you finally get to read the magazine you justly deserve.

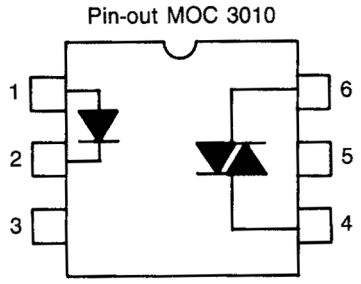


Fig. 7-9. MOC 3010 pin out.

Now what just happened in this incident? To you it may mean nothing, but it is a good example of a smart control, the person knew after he turned on the lamp that the light wasn't lit. Then went through various steps to locate and correct the problem. But what about the computer? Had it been the computer's job to turn on the lamp would it have known whether the light was on, probably not. To build a smart control we must give the computer some procedure or device (feedback) to check to see if the action it took was successful. For the light example we might use a photocell or a photoresistor for a feedback signal. If the feedback gave a negative response the computer could, if we want, go through testing and corrective procedures to find and possibly correct the fault. Naturally we wouldn't go through the time, trouble, or expense for a simple light. But in other circumstances such as with robotics, security systems, nuclear reactor controls, in-flight navigation systems, etc., you would. You would want feedback and redundancy built into every system.

Keep this information in mind so if you find someday that you have a need for a smart controller somewhere, your computer can handle it.

Parts List

Quantity	Item/ Description	Part Number RS = Radio Shack	Cost
1	Push Button Terminal Strip	RS# 274-315	$ .99
2	Terminal Barrier Strips (2/pk)	RS# 274-656	$1.29
6	32 Machine Screws	RS# 64-3012	$ .99
6	32 Hex Nuts	RS# 64-3019	$ .99
1	1.2 k resistor (2/pk)	RS# 271-024	$ .19
1	180 ohm resistor (2/pk)	RS# 271-014	$ .19
1	Subminiature Red LED (2/pk)	RS# 276-026B	$ .79
1	Triac 6 amp 200 Volts	RS# 271-1001	$1.29
1	.22 □F Cap (2/pk)	RS# 272-1070	$ .89
1	15 Foot Extension Cord	RS# 61-2748	$2.39
1	Experimenters Box w/PC Board	RS# 270-284	$3.79
1	Opto-coupler MOC.-3010	RS# 276-134	$1.00

8

Monitor Projects

Here are a few short, simple monitor projects. For those of you who own a C-128 and would like to use an inexpensive composite monochrome monitor, and have the 80 columns inherent with the RGB, we'll construct an adapter. The adapter uses two lines off the RGB socket, (see Fig. 8-5) at the back of the C-128 and feeds the lines directly into your composite monitor to give you 80 column screen. See Fig. 8-1.

The adapter is a stand-alone project, but if you wish you can add a simple switch that will allow you to manually switch from the RGB to Composite (Graphics Screens) whenever you like. You have to use the video out rather than the RF out (see Fig. 8-2). I have not found any standard plug that fits into this socket, so if you wish to use this circuit simply insert wires into the appropriate socket holes and tape it into place.

80 COLUMN TO TV

Here's another circuit you might want to entertain using. See Fig. 8-3. This circuit uses the RGB adapter to an RF Modulator (Radio Shack PN# 15-1273) to a standard TV set. The resolution isn't perfect, but it is quite usable when you sharpen it up with the following program lines.

```
10 Rem Use this program for 80 Col TV
20 POKEDEC("D600"),26 : POKEDEC("D601 "),2
30 POKEDEC("D600"),25 : POKEDEC("D601 "),0
```

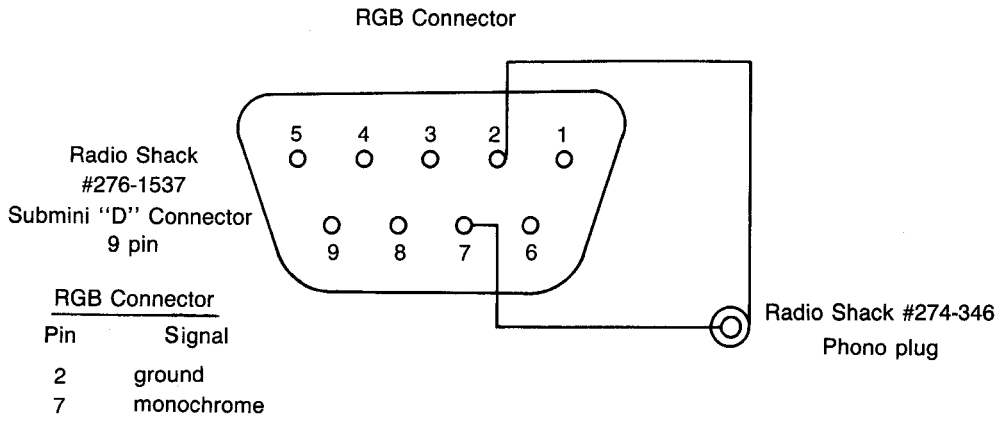


Fig. 8-1. Adapter RGB to phono.

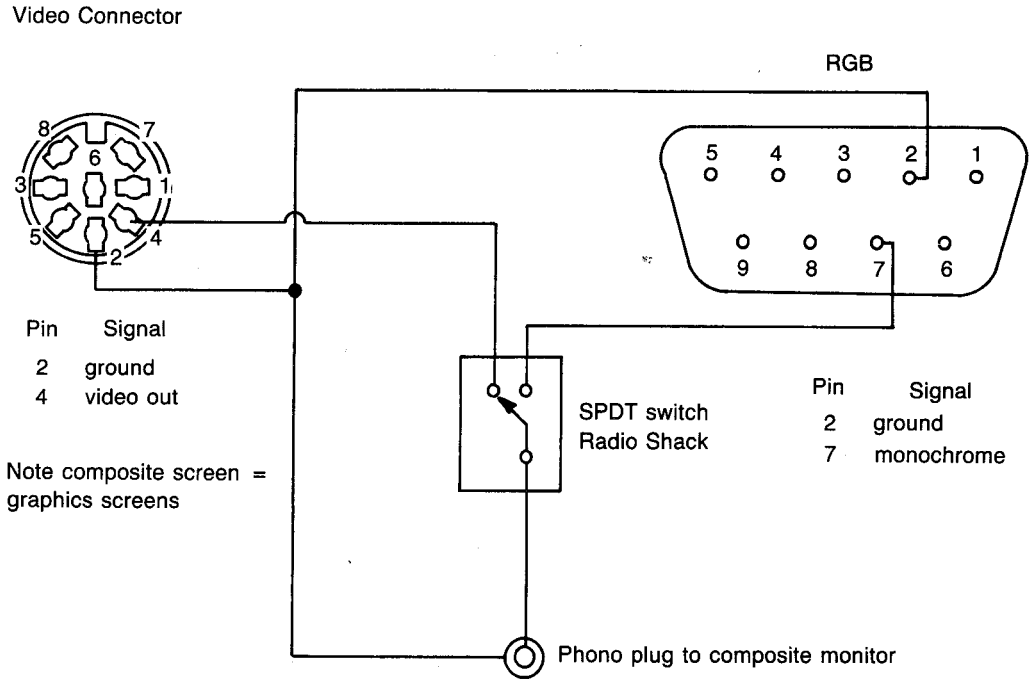


Fig. 8-2. Composite out wiring and switch.

I tested this circuit with a color TV, it's possible that a B/W TV may work better. Again you may have to adjust the TV to obtain the best picture possible.

CIRCUIT CONSTRUCTION

The circuit (Fig. 8-1) is practically self explanatory. Solder short leads from the RGB pin plug to the phono socket. Use a video/audio cable from

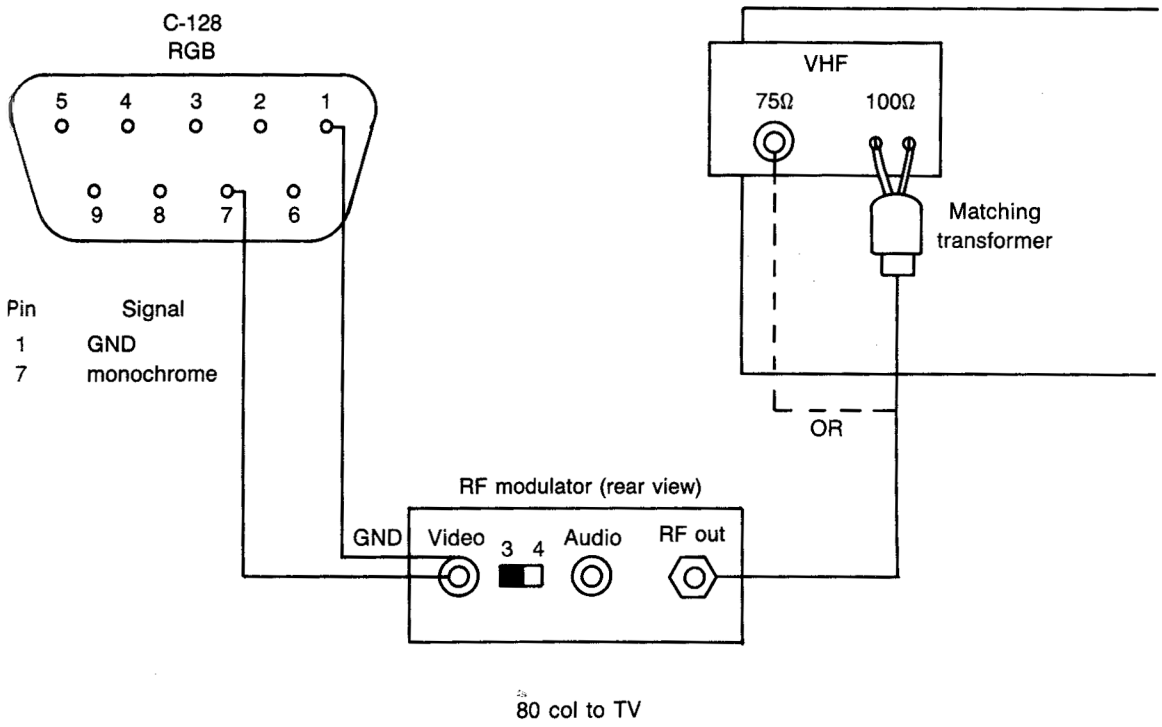


Fig. 8-3. RGB to TV

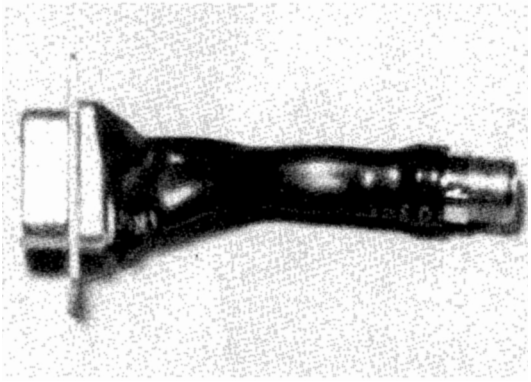


Fig. 8-4. RGB to composite monitor adapter

the phono socket to your monitor. You may have to adjust the intensity of your monitor, but you should find the setup quite satisfactory. You can cover the adapter wires with heat shrink tubing or a molded plastic cover, both of which are available at Radio Shack (see Fig. 8-4).

SCREEN SAVER C-128

This is a utility program designed to relieve potential damage to your video-monitor. How many times have you left your computer unattended

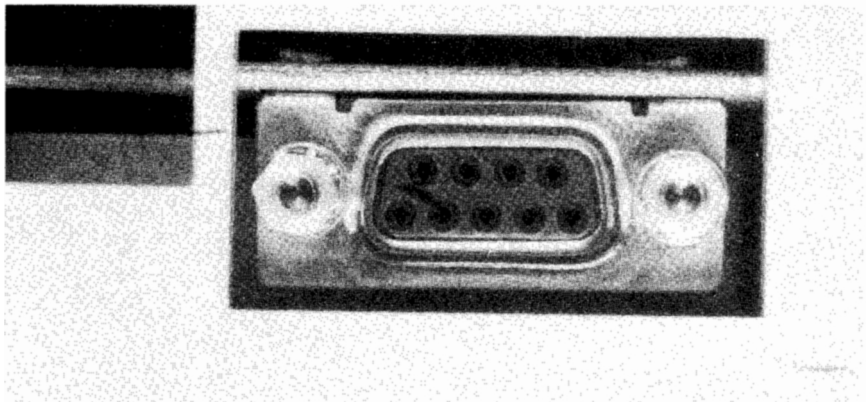


Fig. 8-5. RGB socket on C-128.

for awhile, returning to find a static image left on the screen. If you have, then you should be aware that you are slowly etching its image permanently on your monitor screen. Well this short memory resident program will prevent that from happening.

The program automatically turns off the monitor after three minutes of unattended use. Now the program doesn't actually turn off the monitor's power, it blanks the screen to prevent any damage. Turning the monitor back on is as simple as pressing a key, any key. This will snap the image back as it was before the blanking was evoked (see Fig. 8-6).

Although the screen saver program operates off the 60 Hz interrupt routine, it differs in one important aspect from standard interrupt routines. Standard routines are usually reset after pressing a RUN/STOP/RESTORE key combination, becoming inoperative until a Sys command reinvokes it. This routine however is unaffected by the RUN/STOP/RESTORE key combination, it goes merrily along, as if nothing happened.

```
 2 REM 80-COLUMN SCREEN SAVER C-128
 4 REM   JOHN IOVINE  1-16-88
 6 FORL=4864TO4993:READA:POKEL,A:B=B+A:NEXT
 8 IF B<>12347 THEN PRINT"ERROR IN DATA STATEMENTS":END
 10 SYS4864:SYS4961:NEW
 12 DATA 120,169,013,141,020,003,169,019,141,021,003,088,096,165,213
 14 DATA 201,088,208,043,238,115,019,169,255,205,115,019,208,030,169
 16 DATA 000,141,115,019,238,116,019,169,041,205,116,019,208,015,169
 18 DATA 000,162,025,032,117,019,162,026,032,117,019,142,114,019,076
 20 DATA 101,250,173,114,019,201,026,208,014,169,064,162,025,032,117
 22 DATA 019,169,240,162,026,032,117,019,169,000,141,114,019,141,115
 24 DATA 019,141,116,019,076,059,019,169,108,141,000,010,169,019,141
 26 DATA 001,010,096,032,000,019,076,003,064,000,000,000,142,000,214
 28 DATA 044,000,214,016,251,141,001,214,096,234
```

Fig. 8-6. C-128 screen-saver program.

When the program is run, it will appear as if nothing has happened. And nothing will happen unless the computer keys aren't pressed for three minutes. The program is operating though, in the background, counting the seconds between key presses. The computer's count is reset to zero after each keypress. When the program does time out (reaches three minutes), the screen is blanked, until a key is pressed.

The program is set at three minutes, you can change the timing by pokeing 4902. The program will wait approximately 4.25 seconds for each digit above zero. So if you poke a 10 at this address (poke 4902,10) the program will wait 42.5 seconds before blanking the screen.

The program resides at memory location 4864 to 4993. If you're using a program that also uses these addresses, there is going to be a conflict. But for most of your programming you shouldn't find any problem.

9

Digital Camera

In this chapter, we will construct a digital camera for the computer. With it, we can explore various leading edge topics such as machine vision, pattern recognition, and neural networks.

This is a low cost digital camera, do not confuse this with a digitizer for video cameras. A digitizer is a completely different animal. It takes a signal from a video source, converts the signal to binary values, then displays the information on a monitor.

Our project is a digital camera that our computer reads and displays on it's monitor. The cost for this project is under $65.00. The maximum resolution of our camera chip is 128 pixels by 64 pixels. Some features some grey scaling, false coloration, and user adjustable timing.

One unique feature of this camera is that we are not using any support circuitry. The computer is handling all the timing, addressing, and refreshing. This makes our circuit quite simple. The power for the chip, however, is provided by two batteries and a couple of resistors. Let's begin by examining the main functional component of the project, namely the digital camera chip.

D-CAM CHIP

The chip that we are utilizing for our camera chip is a modified dynamic RAM memory chip, It is a lesser known fact that the memory cells of these chips are photosensitive. To use these cells for image processing, they must be accessed in a certain manner.

Information in the form of binary bits are stored in arrays of memory cells in the dynamic RAM. The memory cells are arranged in a matrix array (see Fig. 9-1) or honeycomb structure. Using standard graphing techniques, the Y-axis would represent the rows and the X columns. Each box in the graph represents one memory cell whose location is identified by its row and column numbers. Addressing to any cell in the chip is accomplished by multiplexing the address lines. (See D-Cam chip pin out address lines A0-A6, Fig. 9-2.) Multiplexing simply means that we are using the same lines for both the row and column addresses. This is achieved using the RAS (Row Address Strobe) pin and the CAS (Column Address Strobe) pin as follows. The row address is placed on the address pins and the RAS pin is asserted. Next the column address is placed on the address pins and the CAS pin is asserted. Depending on the status (binary "1" or "0" of) the WE (Write Enable) pin, the operation will be either a read form or write to the cell utilizing the Data in or Data out pin respectively.

Each cell in the matrix is like a tiny capacitor. When a binary "0" is written to a cell no voltage (OV) is stored on the cell. When a binary "1" is written to a cell within the chip, that cell is charged to +5V level and held. No capacitor is perfect however, and the capacitor within our RAM chip is no exception. The charge on the cell will eventually leak away. Dynamic memory circuits handle this problem by issuing what is

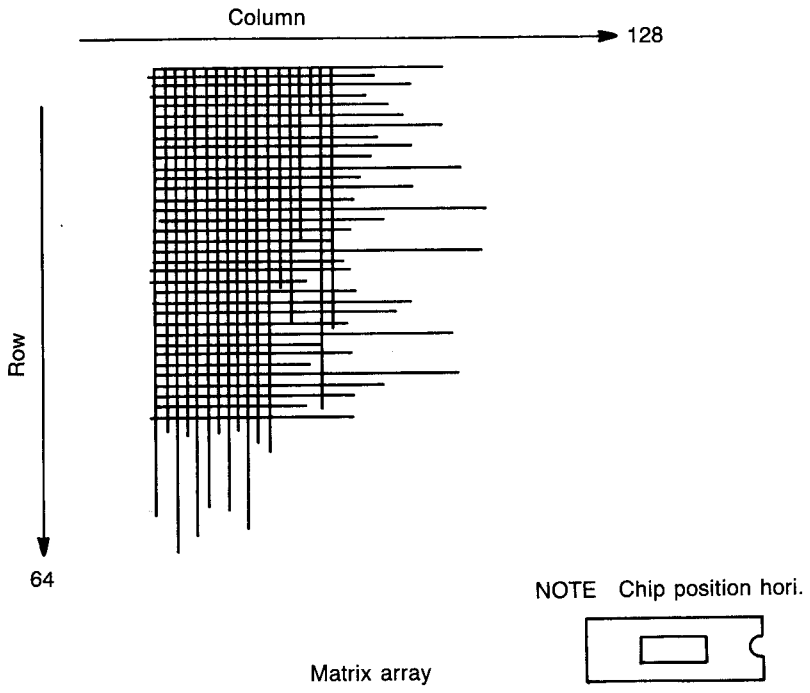


Fig. 9-1. Matrix array.

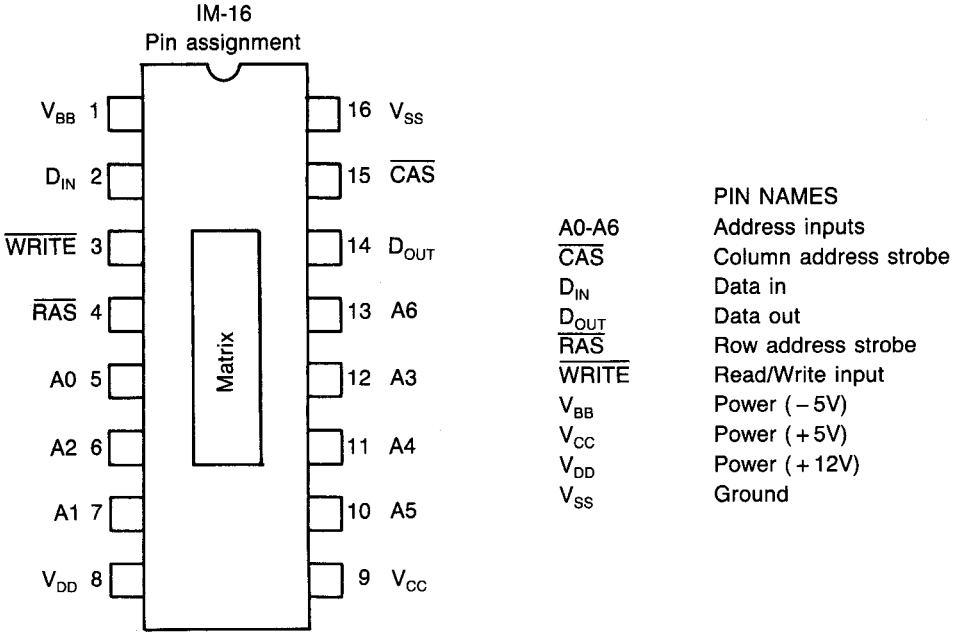


Fig. 9-2. IM-16 chip pin out.

known as a refresh cycle that recharges all the binary "1" cells to an appropriate +5V level. The amount of time allowed between refresh cycles to retain the integrity of the data will vary with the type of chip, but it is safe to say that it usually lies between 2 and 4 milliseconds (thousandths of a second).

Refreshing is a little easier than what it may first appear. A read or write operation to any cell will refresh the entire row that the cell lies in. Also, many chips have an RAS refresh mode that allows one to address the row only to refresh that entire row. Refreshing, I'm afraid, is still a hassle that must be contended with.

Our chip contains two banks of 128 by 64 pixels. Unfortunately, these banks are separated by a dead space area. To keep our picture continuous, we are only using one of the two banks available to us.

PHOTOELECTRIC EFFECT

When a memory cell of the D-RAM chip is holding a binary 1 or is charged to +5V, light falling upon the cell will increase the rate of discharge in proportion to light intensity and duration. With this information, we can see that by filling the entire memory matrix with binary 1's and exposing it to light, the areas with higher illumination will lose their charge and areas with little or no illumination will retain their charge. If this illumination happens to be an image projected onto the

array by a lens you would now have a binary image stored in the chip that the computer could read.

This is exactly what we are doing. Our program first writes a binary "1" to all the memory cells, waits, then reads back the information and displays it on the monitor. Any cell thats charge fell below a certain threshold would be read as a binary "0". The binary Os would denote areas of high illumination. These cells would be displayed as white pixels on the monitor. The binary "1"'s show areas of little or no illumination, and would be displayed as black pixels.

MATRIX UNSCRAMBLE

With all our program has to do, addressing, writing to then reading from the individual cells and displaying the information, there is one xnore task it has to accomplish for us. Our dynamic camera chip began its life as a memory chip for computers, not as a light imaging component. Therefore, the matrix of memory cells do not lie in order as shown in Fig. 9-1: Our program must also unscramble the matrix and put it in order, so that we can process images from it. Although that in itself isn't too much of a problem, it does decrease the overall speed of the program.

LIMITATIONS OF THE SYSTEM

As stated before, we are not using any support circuits for our camera. This keeps our unit inexpensive and simple, but it does impose limitations. The most important limiting factor is speed. The microprocessor in our computer operates at approximately a million clock cycles per second. A millionth of a second is equal to one microsecond. Depending upon the instruction the computer is executing, it will usually take a couple of clock cycles per instruction in machine language to complete.

The dynamic RAM chip operates in nanoseconds, billionths of a second. The chip is also very critical about timing. Naturally, with our computer operating in microseconds and doing all the work, some of the data can become corrupted. This corruption takes the form of image resolution degradation and unstableness. Of course, we will attempt to keep this to a minimum.

LOW RESOLUTION SCREEN

The low resolution screen we will start with is 40 pixels by 2(16) pixels. The reason I gave the row quantity as 2(16) is that the program divides each test screen byte into two vertically stacked pixels. This makes our overall vertical resolution 32. Therefore, our effective resolution is 40 x 32, which equals 1,280 pixels per screen.

I plan to jump into the bit map hi-resolution screen with a 128 by 64 pixel screen. You may be thinking to yourself "fine; but why are we

bothering with the low resolution in the first place." The reason is that experiments in edge detection, character and pattern recognition, and neural networks will be much easier to accomplish with the low resolution (1,280 pixel) screen.

It should be obvious that manipulating 1,280 pixels (or 640 bytes) of lo-resolution screen information is much easier and quicker than 8,192 bits of information on the hi-resolution screen. Additionally the lo-resolution text screen addresses proceed in an orderly fashion, starting at address 1024 in the upper left hand corner thru address 2023 (bottom right). In contrast to the bit map screen where pixel to screen location criss-crossing over the entire screen. Since the addresses on the lo-resolution screen precede in an orderly fashion, this makes programming subscanning programs for edge detection, pattern recognition, and neural networks that much easier.

Did you notice that the entire lo-resolution screen is composed of only 1,000 bytes, yet we are reading 1,280 pixels. The 1,280 pixels is our effective resolution: remember we are splitting our text bytes into two pixels. This means that we are using just 40 by 16 bytes of screen memory.

EXTENDED FIELD OF VIEW

The photosensitive area of our chip is rather tiny, and with the lo-resolution screen we are compounding the problem by using only 1/6 (40 x 32) of the pixels available to us in either one of the 128 x 64 pixel banks. This could make our image processing difficult, because it would be hard to fit a complete projected image onto that tiny section of the matrix. To alleviate the problem somewhat, I decided to extend the field of view (FOV) of the camera. I accomplished this by accessing every other pixel, horizontally and vertically on the camera chip. So although our resolution is still 40 by 32 pixels on the screen, we are reading the image off the chip as if it were 80 by 64 pixels. The skipped pixels make the edges of the object a little choppy, but as you can see from the photo it's not too bad. I also centered the FOV on the matrix rather than leaving it to either side. This makes aiming the camera easier.

BLACK AND WHITE CAMERA

The black and white camera (B/W) operates at three (3) to four (4) frames per second. This is substantially faster than the gray scale camera and affords a real time image.

Besides being an excellent camera in itself, it is also useful for aiming and adjusting the camera before going into the gray scale or hi-resolution mode. In fact, I advise not to enter the gray scale camera any other way, especially when you are still a beginner. Because of the additional time the gray camera requires, images will smear across the screen if the cam-era is moved during image processing. This can make aiming the camera somewhat frustrating. It therefore behooves you to have the camera at

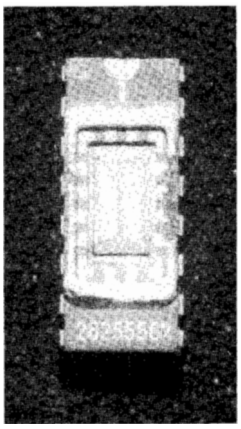


Fig. 9-3. IM-16 chip.

least partially aimed, and adjusted beforehand. As you gain experience using the camera, you may find that preadjusting the camera in the BI W mode is no longer necessary before entering the gray or hi-resolution modes.

As stated, the program splits each text byte into two pixels. Let's take a closer look at this and see how it operates. To divide each text byte I implemented four programmed characters. One white, one black, one top white/bottom black, and finally, one top black/bottom white. Depending on the scan number and feedback from the camera, the computer will choose the appropriate byte and store/display it on the screen.

GRAY SCALE

The gray scale camera gives us four shades of gray with one background color. How the camera interprets the binary information from the camera chip for the gray scale is based on the individual timing cycles of each gray scale scan. To achieve four gray scales, we are using four separate scans, each with its own timing cycle.

If you remember, we stated that after a memory cell was charged to +5 volts, light falling on the cell would increase the rate of discharge in proportion to intensity and duration.

If the light intensity is such that a cell is discharged below the binary "1" threshold, let's say in the first scan/cycle, that memory cell (pixel) is read as a binary "0" and displayed as a white pixel on our monitor. Further, let's say that another memory cell lying next to it (2nd pixel) discharged just above the threshold. Remember during any read or write to a cell, all memory cells lying in the row are refreshed, so at the same time our computer is reading the information, all the memory cells are refreshed. This means our 2nd pixel is recharged to a full 5 volts. The cells that fell below the binary "1" threshold are refreshed at 0 volts. So the computer reads that 2nd pixel as a binary "1", displays a black pixel and continues. When the 2nd scan begins for the 1st gray scale, all the

pixels that were partially discharged like the 2nd pixel are at +5V, therefore the timing of the program must wait a little longer than the first scan time to read any new information.

So what this boils down to is that the timing cycles are not additive. On our second scan, we cannot add a little time to the first scan and expect to read anything new. Our second scan must start from scratch (as far as timing is concerned) and last longer than the preceding one. To continue, let's say the computer is now running the second scan, and has waited 50 percent longer than the first scan, now that 2nd pixel (memory cell) has fallen below the binary "1" threshold. The computer reads that memory cell as a binary "0" and displays it as a light gray pixel. This procedure is followed for all shades of gray. After the four shades of gray have been scanned, the program resets and starts over.

One point I would like to mention on the display procedure is that once a pixel has changed, subsequent scans will not alter the pixel any further, until reset. This must be included in the program or the screen would constantly go black. The computer would read all previously changed pixels and currently changed pixels as the same, and would therefore display all of them at current gray level that would progressively advance to black.

I hope I didn't bore you with the above on timing, but this information is critical when you begin adjusting the timing on the gray scale camera. I've provided on-the-fly timing changes, as well as a menu option on the main program. At the time I was writing the program, I was undecided whether I should allow the user to adjust and control the timing of the scans. I could have taken the easy way out by plugging in what I thought was a general default value. But I realized that all lighting conditions couldn't be met with one timing. Since flexibility breeds innovation and experimentation, and inflexibility obsolescence, I opted to have the timing user adjustable.

256 SHADES OF GRAY

Although we are using only four shades of gray in our program, you should be aware that it is possible to generate 256 shades of gray. Before I describe the procedure to do this, let's first examine our four gray-scale generation. We are using the extended background mode as described in the Programmer's Reference Guide. Each gray scale scan is associated with one of the background color registers. This is how we also provided coloration, but we will come back to this later. In the default mode, we begin scanning with white, then light gray, medium gray, and dark gray, all with a black background.

To generate a 256 gray scale you must employ a technique known as dot dithering. How dot dithering works is similar to the procedure we use to split our text byte into two vertical stacked pixels. They both employ programmable characters. Each character the computer generates

is generated on an 8 x 8 dot matrix. That equals 64 dots per character. Which dots are turned on or off generate the character pattern displayed. By turning off the standard character generator and programming our own characters, we can generate 64 shades of gray. We do this by progressively turning on the dots one at a time for each character, starting with one dot turned on in the center and progressively adding dots for each new character, until we end up with all the dots turned on for character #64. (If this section on programmable characters is confusing, see the Programmer's Reference Guide.)

To continue to our 256 shades, we employ the extended background mode with our 64 dot dithered, programmed characters. Now we have four backgrounds white, light gray, medium gray, and dark gray, with the 64 dot dithered gray scale for each background. This comes to 4 x 64 = 256 gray scale. Although it is possible to do this, it would increase the processing time tremendously. A more realistic attempt would be a 16 or 32 gray scale.

COLORATION

Coloration is simple, once we have the gray scale in place. A color is assigned to each extended background register instead of the default gray scale. This option is provided in the main program. By choosing this option on the menu, each color you enter in the submenu for a particular scan will be displayed. It is interesting to note that the coloration can be implemented with the fast-scan black and white camera. The first and last colors picked in the coloration menu will be displayed with the black and white camera.

This is the kind of technology used by astronomers and movie producers. If you have ever seen an astronomical photograph that had colors assigned to each B/ W density, for improved image resolution, this is how they accomplished it. The photograph runs thru an image enhancer that assigns a color to each density. The machines sensitivity is much greater than the human eye in determining B/W density. Similar techniques are used in coloration of old black and white movies.

The CCD technology used in video camera, eye in the sky satellites, text readers, image enhancers, and a host of other applications is very closely related to our D-Cam chip and what we are doing with it.

TIMING

Timing changes can be implemented on-the-fly during camera operation, or from the main program. The > key will increase timing, the > key will decrease the timing. The timing is changed by one millisecond for each screen scan that the key is held. If you are operating in the gray camera mode, each one of the timing cycles will be affected.

There is a kink in on-the-fly timing changes that you should be aware of. If you decrease the timing beyond 0 milliseconds, the timing will roll

over to 255 milliseconds. This will show itself as a tremendous increase in scan time. On the other hand, if you increase the timing past 255, you will roll over to 0. At any time, if you should get stuck or lost in the timing, pressing the "R" key will return you to the main menu. There you can check, adjust or correct the timing by choosing the timing option for the camera you're currently operating. The menu option has the added advantage of reading the current scan times, which enables you to see where you are before modifying.

Since the program uses every microsecond available for processing, the keyboard is only checked once per screen scan. So you will have to hold down the key until the computer sees it, this may take one or two screens scans. In the fast scan B/W mode, the menu will appear almost instantaneously. Alas, in the gray cam, a couple of screen scans take longer.

CONSTRUCTION

The most critical aspect of constructing the camera is the lens. The lens must be at the proper distance to be focused on the matrix of the digital camera chip. If you use the same components I have, then all the measurements have been taken care of. But in the future, if you would like to use a better or different lens or case, you will have to redesign the camera a little, and place the lens at the proper focal length.

LENSES

The lens we are using for the camera is a surplus lens available from Edmund Scientific Co. (see parts list). I chose this surplus lens because it is very inexpensive for the quality (see Fig. 9-4). It is much easier to mount than a standard lens, as you can see from the picture, and it has its own housing, which means we don't have to build one. We can easily mount this lens on our camera housing with little or no hassle. The lens has an adjustable iris (f-stop) that controls the light entering the camera, a valuable aid for various lighting conditions. This feature by itself is worth the cost of the lens. It expands the operational latitude of the camera.

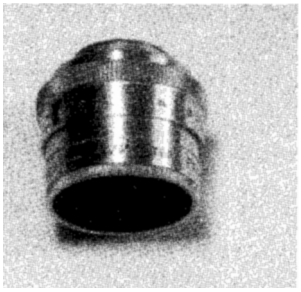


Fig. 9-4. Lens.

The lens has a focal length of ½ inch. This means our matrix must lie ½ inch down from the bottom of the lens. Notice I said matrix, not the top of the chip.

Begin construction by drilling the holes in the case for the lens and switch. The lens hole (⅝" dia.) is centered on the face of the case. (See Fig. 9-5.) Try for the best fit possible. With a good close fit, you can actually screw the lens on, instead of gluing it on with epoxy. Do not install the lens at this time, just drill all the holes. If you plan on adding a small tri-pod as I have, drill an additional hole in the bottom of the case.

We are using two PC boards with this project (see Fig. 9-6); one that comes with the case, and an additional PC board that facilitates soldering and wiring the IC socket. Later, these two PC boards will be mounted together.

Using ribbon cable, begin soldering the card connector to the IC socket (see Fig. 9-8). Make sure the IC socket is centered on PC board RS# 276-159. (See Fig. 9-6 and Fig. 9-7 schematics.) Solder the wires from the IC socket to the joystick plug. Solder in the capacitors to the IC socket. Note that both ground wires from the power supply and the user port must be connected to the chip for operation.

Now begin construction of the power supply: (See Fig. 9-9 and Fig. 9-10.) Use a small piece of perf board to mount the four resistors, then make all the connections to the mounted resistors. It will be installed in the camera housing permanently, therefore neatness counts, especially when you're installing in a small space.

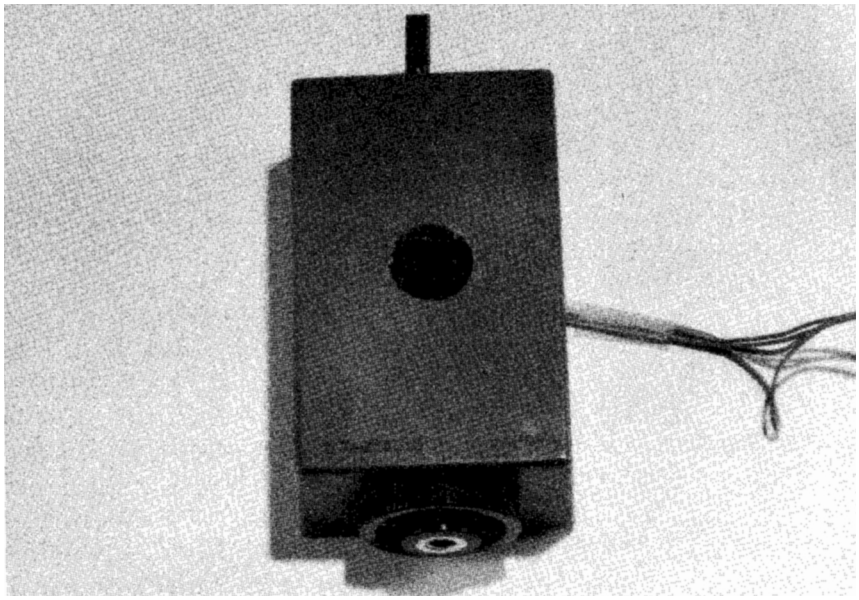


Fig. 9-5. Case with holes.

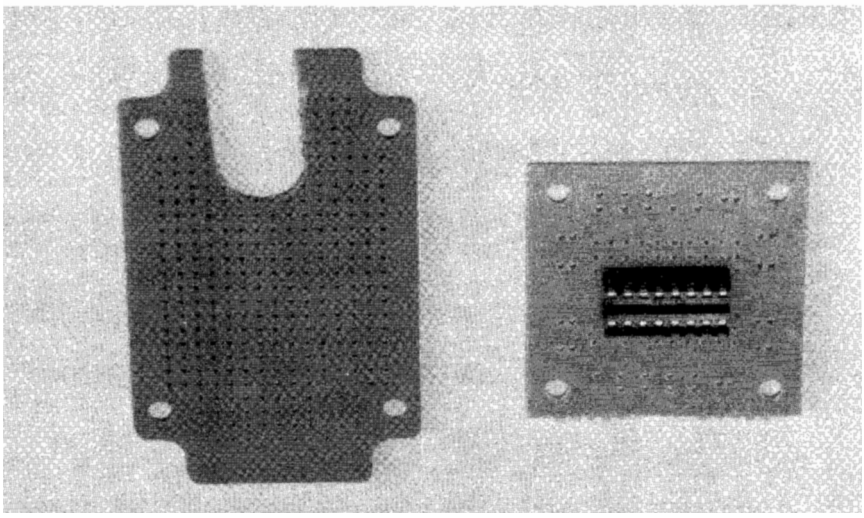


Fig. 9-6. 2 PC boards.

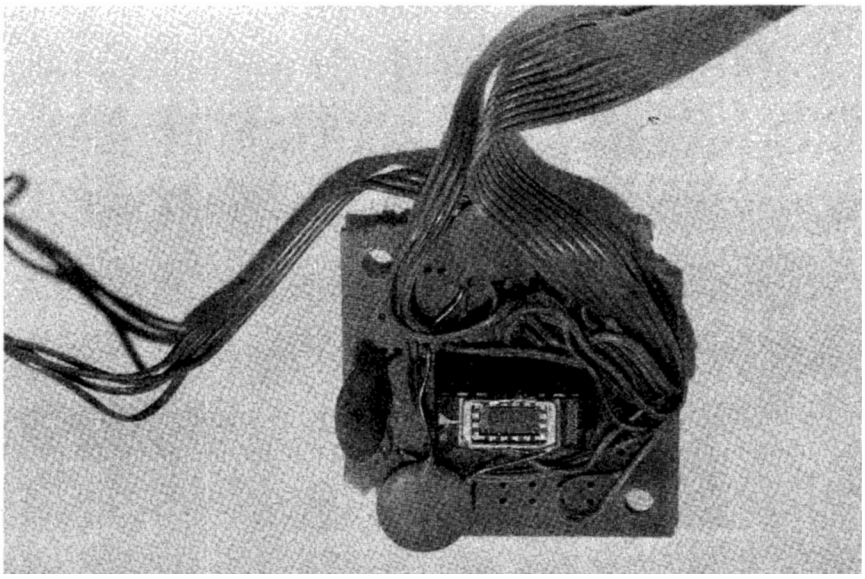


Fig. 9-7. Wired D-Cam chip.

The power supply is bipolar, meaning it supplies both positive voltage and negative voltage to the camera chip. Notice that the switch we use to turn the power on and off is a double pole. Do not substitute this type of switch, since both grounds for each battery must be turned on and off for proper operation. If you try to use a single pole switch and disconnect the main ground, electricity will still flow through the circuit, killing the battery and possibly destroying the camera chip.

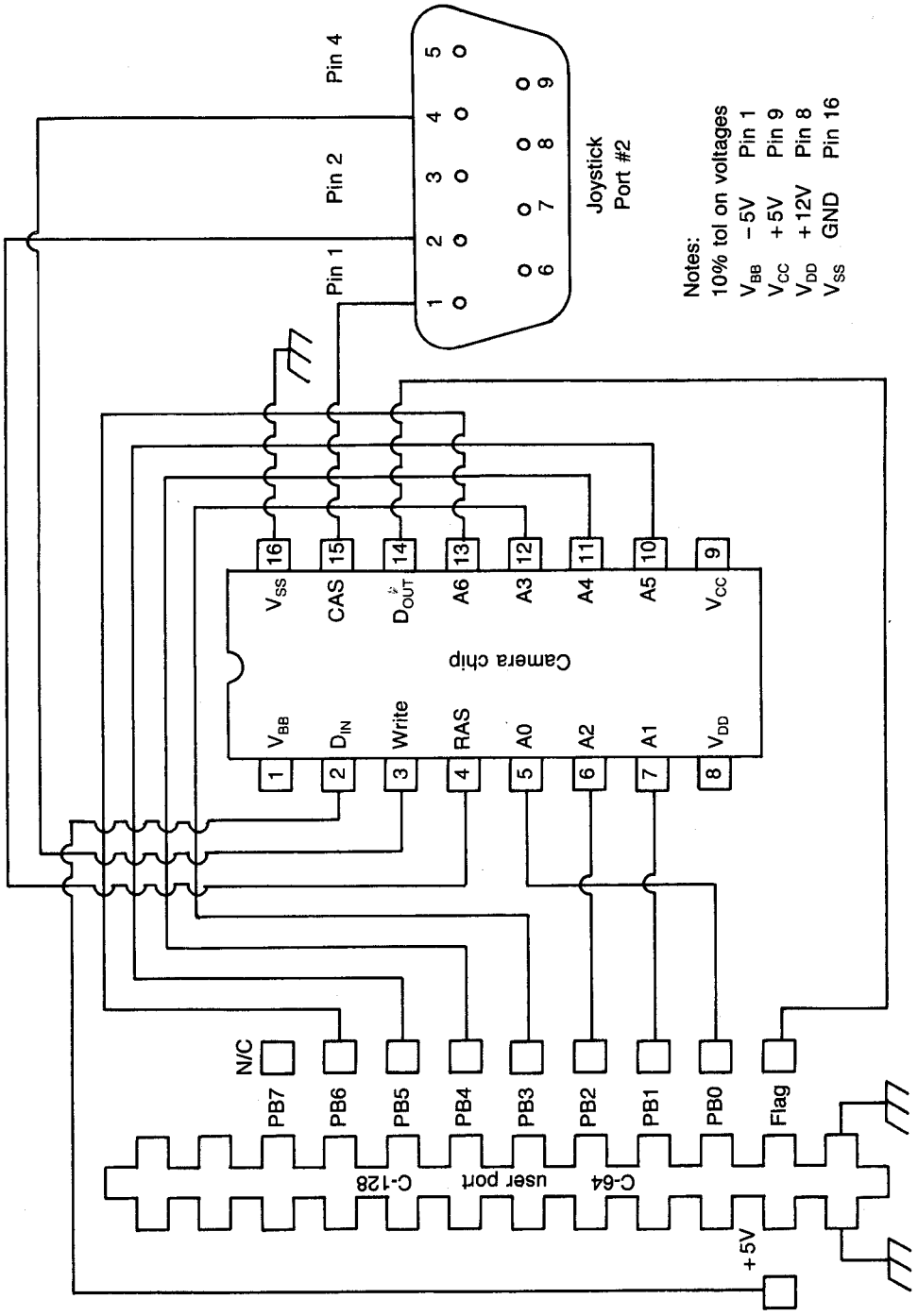
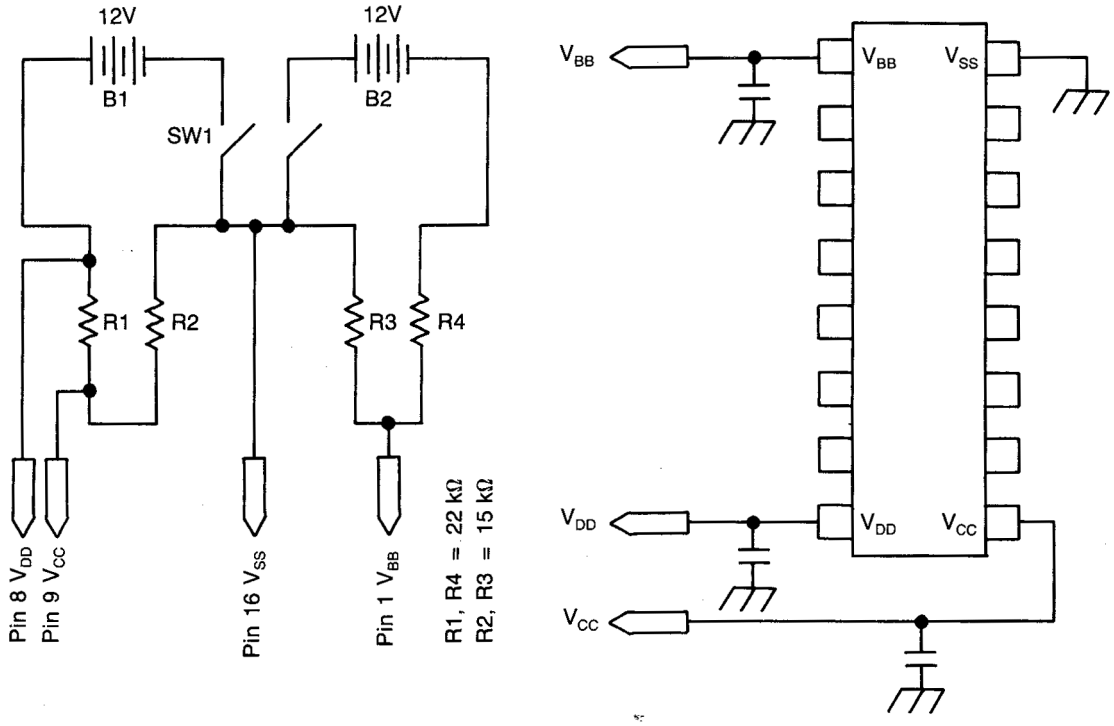


Fig. 9-8. Digital camera.



Notes
Decoupling capacitors should be located on PC board holding IC camera chip not on P.S. Board

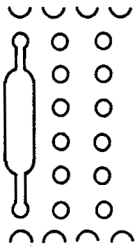


Fig. 9-9. Power supply.

You can check switch operation and the power supply with an inexpensive VOM from Radio Shack. If you have been following this book and building the projects, it's time you get one if you haven't already. Radio Shack sells an inexpensive VOM for $7.95, catalog #28-4012.

When you wire the power supply to the IC socket, use a minimum of 6 inches of wire inbetween. This will make changing the batteries easier when they wear out.

PREASSEMBLY TEST

When you have gotten this far, you're ready to check out the camera. Do this first before installation, in case you need to correct any wiring

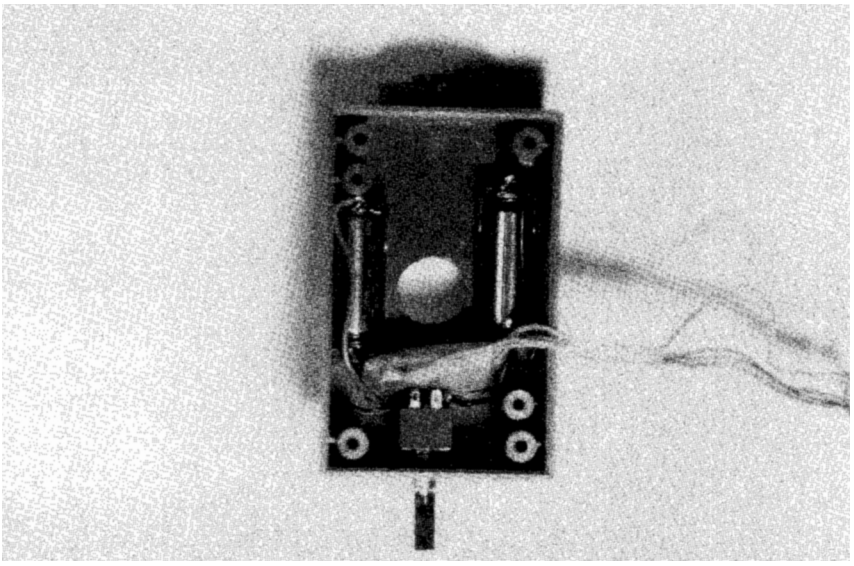


Fig. 9-10. Case with power supply.

error. Get the second IC socket that came in the package and install the camera chip onto it. Then install this little subassembly onto the wired PC board. Yes, that s correct, we are using two IC sockets this brings the camera chip to the correct height in the final assembly. Insert the joystick socket into joystick port #2. Insert the card connector into the user port, then turn on the computer. Load the main program, from the main menu load the B/W camera. When you are returned to the main menu, run the camera. At this point the camera screen should appear on the main screen. Turn on the power to the camera. Depending upon the amount of light available, the screen could be either black, white, or some combination. If black, make some light available to the chip; the entire screen should go white. If it's white to begin with, cover the chip with your hand to block the light; the screen should go completely black. If the chip passes this test, you can congratulate yourself, you're almost finished. If it didn't turn everything off, start checking the power supply wires. Check batteries to make sure they are fresh. Finally, check the wiring from the user and joystick ports to the IC.

FINAL ASSEMBLY

We start with installing the power supply. Wrap some scotch tape or electrical tape around the perf board holding the resistors, this is to prevent accidentally shorting anything out. Glue or epoxy the battery holders on both sides of the lens hole. (See Fig. 9-10.) install the switch in the top hole with the perf board underneath it. Take your time with the installation. You only have to do this once, so don't force any components in. You have ample space.

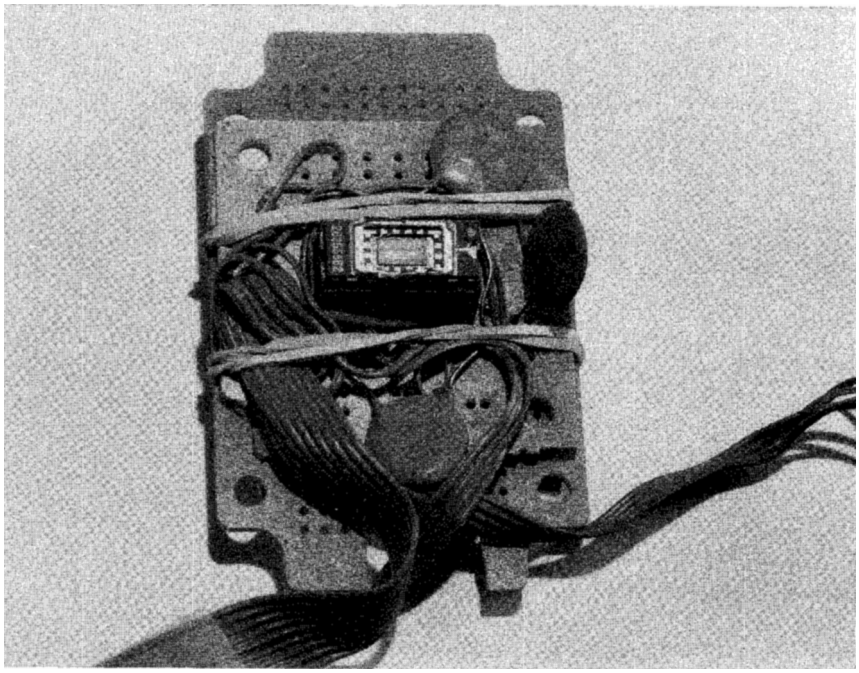


Fig. 9-11. D-Cam chip on 2 PC boards.

Now get the PC board that came with the housing, and cut a channel in one end of it for the wires to pass through (see Fig. 9-6): Place the IC wired PC board on top of this board and center it, making sure that you don't install the IC wired PC on the copper-clad side of the second PC board (see Fig. 9-11). This could short out the unit. With the one board centered on the other, glue or epoxy them together. You will notice on my prototype, I used two rubber bands to secure the boards together. You can do this, if you wish.

Now mount the entire board assembly into the housing, using the two screws that came with the housing for the boards (Fig. 9-12). Finally, mount the lens. If you succeeded in making a good fit, you can screw it in. If not, glue or epoxy it (see Fig. 9-13).

LIGHTING

When you start using the camera, start with simple lighting conditions and objects. In other words, start in a dimly lit room, with a light on a simple white object, such as the cup I have used for illustration (see Figs. 9-14, 9-21, 9-22, 9-23, 9-24, and 9-25). If you arbitrarily start aiming the camera everywhere, you won't be able to see the forest from the trees. You need to gain some experience adjusting the timing and f-stops of the camera.

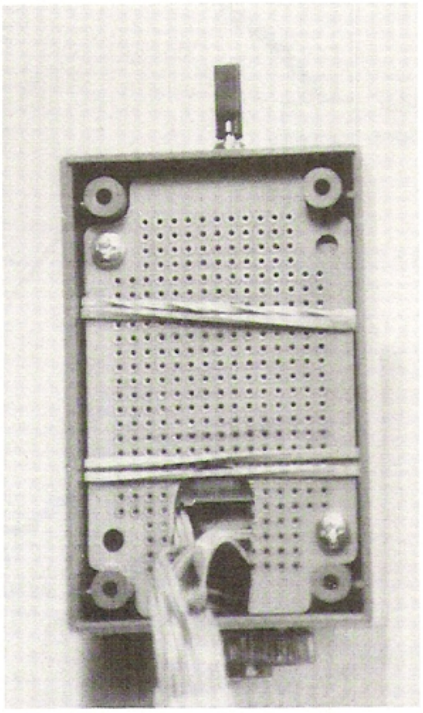


Fig. 9-12. Back view of camera.

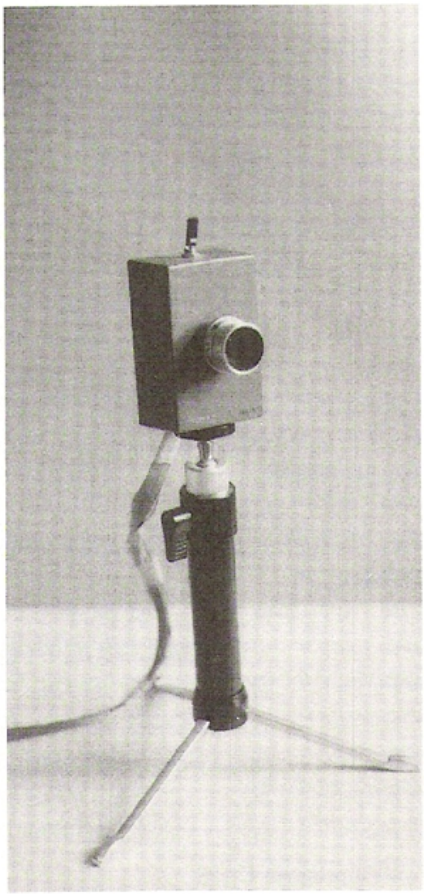


Fig. 9-13. Finished project.

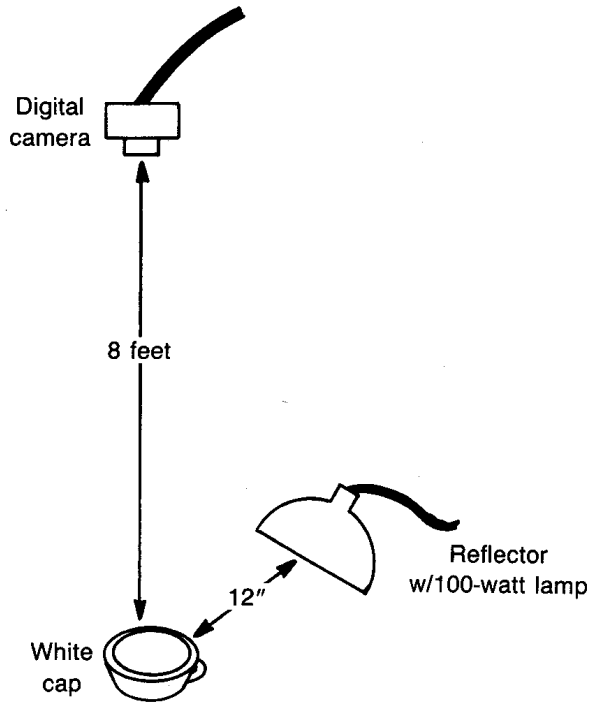


Fig. 9-14. Lighting setup.

```
5 REM 128 MAIN PROG   BY JOHN IOVINE
10 PRINTCHR$(142):REM SWITCH UPPER CASE
20 POKE54,48:POKE58,48:CLR:REM PROTECT CAP
70 FORJ=12288TO12288+32: REM PLACE CHARACTER DATA IN RAM
80 READ A:POKEJ,A:NEXT
120 DATA,0,0,0,0,0,0,0,0:REM @
130 DATA 255,255,255,255,0,0,0,0
140 DATA 0,0,0,0,255,255,255,255
150 DATA 255,255,255,255,255,255,255,255
160 B=0
162 POKE53281,1:POKE53282,15:POKE53283,12:POKE53284,11
165 POKE53265,PEEK(53265)OR64
170 REM MENU
180 PRINT"{CLR}"
190 PRINT:PRINT:PRINT:PRINT
200 PRINT"                    DIGITAL CAMERA MENU"
210 PRINT:PRINT:PRINT
215 PRINT" 1) INSTRUCTIONS"
220 PRINT" 2) LOAD B/W FAST SCAN CAMERA"
225 PRINT" 3) LOAD G/S GRAY SCALE CAMERA"
230 PRINT" 4)    SET/RESET TIMING B/W MODE"
235 PRINT" 5)    SET/RESET TIMING G/S MODE"
240 PRINT" 6)    COLORATION  G/S"
```

Fig. 9-15. Main program C-128.

```
250 PRINT" 7) START CAMERA"
255 PRINT" 8) QUIT"
260 PRINT:PRINT"INPUT OPTION NUMBER 1-8"
261 FORT=0TO175:NEXTT
262 POKE212,88:POKE208,0
265 INPUTX:IFX<0 THEN280:IFX>7THEN280
270 ONX GOTO290,300,400,500,600,700,800,900
280 PRINT"ERROR, PLEASE ENTER NUMBER BETWEEN 1-7":GOTO265
290 GOTO910
300 CLR
301 LOAD"128 B/W CAM",8
400 CLR
401 LOAD"128 GRAY CAM",8
500 PRINT"{CLR}"
501 PRINT:PRINT:PRINT
502 PRINT" RESET / SET TIMING"
503 PRINT" FAST SCAN B/W CAMERA"
504 PRINT:PRINT"NUMBERS REPRESENT DELAY IN MILLISECONDS"
505 PRINT"BETWEEN SCREEN SCANS"
506 PRINT:PRINT"DEFAULT DELAY IS 16 MILLISECONDS"
507 G=PEEK(5461)
508 PRINT"CURRENT VALUE IS ";G
509 INPUT"ENTER DELAY";G
510 POKE5461,G
511 PRINT:PRINT"THANK YOU"
512 FORT=1TO500:NEXTT
513 GOTO180
520 G=PEEK(5461)
600 PRINT"{CLR}"
601 PRINT:PRINT:PRINT
602 PRINT" SET / RESET TIMING"
603 PRINT:PRINT"NUMBERS REPRESENT DELAY IN MILLISECONDS"
604 PRINT"          BETWEEN GRAY SCANS"
605 PRINT:PRINT
606 J=PEEK(5456):K=PEEK(5772):L=PEEK(5804)
607 PRINT"1 ST SCAN IS NON-ADJUSTABLE "
608 PRINT:PRINT"2 ND SCAN DEFAULT DELAY IS  7"
609 PRINT"CURRENT DELAY IS ";J
610 INPUT"ENTER DELAY";J
611 PRINT:PRINT"3 RD SCAN DEFAULT DELAY IS  28"
612 PRINT"CURRENT DELAY IS ";K
613 INPUT"ENTER DELAY";K
614 PRINT:PRINT"4 TH SCAN DEFAULT DELAY IS  72"
615 PRINT"CURRENT DELAY IS ";L
616 INPUT"ENTER DELAY";L
```

```
617 POKE5456,J:POKE5772,K:POKE5804,L
618 PRINT"   THANK YOU  "
619 FORT=1TO500:NEXTT
620 GOTO180
700 PRINT"{CLR}"
701 PRINT:PRINT
702 PRINT"  COLOR CODES"
703 PRINT
704 PRINT"0 BLACK","8 ORANGE"
705 PRINT"1 WHITE","9 BROWN"
706 PRINT"2 RED","10 LIGHT RED"
707 PRINT"3 CYAN","11 DARK GRAY"
708 PRINT"4 PURPLE","12 MEDIUM GRAY"
709 PRINT"5 GREEN","13 LIGHT GREEN"
710 PRINT"6 BLUE","14 LIGHT BLUE"
711 PRINT"7 YELLOW","15 LIGHT GRAY"
712 PRINT:PRINT
714 PRINT"DEFAULT COLOR FOR 1 ST SCAN IS  1"
715 INPUT"ENTER COLOR CODE # ";C
716 PRINT
717 PRINT"DEFAULT COLOR FOR 2ND SCAN IS 15"
718 INPUT"ENTER COLOR CODE # ";D
719 PRINT
720 PRINT"DEFAULT COLOR FOR 3RD SCAN IS 12"
721 INPUT"ENTER COLOR CODE # ";E
722 PRINT
723 PRINT"DEFAULT COLOR FOR 4TH SCAN IS 11"
724 INPUT"ENTER COLOR CODE # ";F
725 PRINT
726 PRINT"DEFAULT COLOR FOR BKGRD IS  0"
727 INPUT"ENTER COLOR CODE # ";B
728 POKE53281,C:POKE53282,D:POKE53283,E:POKE53284,F
729 PRINT:PRINT"THANK YOU "
730 FORT=1TO500:NEXTT
731 GOTO180
800 PRINT"{CLR}":G=PEEK(2604):POKE2604,(PEEK(2604)AND240)+12
801 FORL=55296TO56295:POKEL,B:NEXT:SYS5120
804 POKE2604,G:GOTO1000
900 END
910 PRINT"{CLR}"
912 PRINT:PRINT:PRINT:PRINT
914 PRINT"   DIGITAL CAMERA INSTRUCTIONS"
916 PRINT"                          PG 1    "
918 PRINT:PRINT
920 PRINT"THE DIGITAL CAMERA PROGRAM PROVIDES"
```

Fig. 9-15. Continued.

```
922 PRINT"THREE COMMANDS THAT CAN BE UTILIZED"
924 PRINT"DURING CAMERA OPERATION. (ON-THE-FLY)"
926 PRINT"THESE COMMANDS ARE AS FOLLOWS:"
928 PRINT
930 PRINT"PRESS > KEY TO INCREASE TIMING"
932 PRINT"PRESS < KEY TO DECREASE TIMING"
934 PRINT"PRESS R KEY TO RETURN TO MENU"
936 PRINT
938 PRINT"THE TIMING KEYS OPERATE WITH BOTH"
940 PRINT"B/W AND GRAY CAMERAS."
942 PRINT"THE KEYS WILL INCREMENT OR DECREMENT"
944 PRINT"THE OVERALL SCAN TIMING BY ONE"
946 PRINT"MILLISECOND PER SCREEN SCAN THAT THE"
948 PRINT"KEY IS HELD."
950 PRINT"IN THE GRAY CAMERA MODE, THIS MEANS"
952 PRINT"THAT EACH OF THE THREE GRAY SCALES"
954 PRINT"TIMING MODES ARE SIMULTANEOUSLY CHANGED"
956 PRINT:PRINT"PRESS ANT KEY TO CONTINUE"
958 GETA$
960 IFA$=""THEN 958
962 PRINT"{CLR}"
964 PRINT:PRINT:PRINI
966 PRINT"    DIGITAL CAMERA INSTRUCTIONS"
968 PRINT"                          PG 2"
970 PRINT:PRINT
972 PRINT"BY PRESSING THE R KEY, THE PROGRAM"
974 PRINT"WILL RETURN TO THE BASIC MENU. HERE"
976 PRINT"FURTHER ADJUSTMENTS IN TIMING ARE"
978 PRINT"POSSIBLE, BY ALLOWING THE USER TO FIRST"
980 PRINT"READ THE TIMING SCANS BEFORE ADJUSTING"
982 PRINT"PSEUDO-COLORATION OF THE GRAY SCALES"
984 PRINT"IS IMPLEMENTED BY CHOOSING THE COLORATION"
986 PRINT"ITEM ON THE MENU. COLORATION OF THE B/W"
988 PRINT"CAMERA IS POSSIBLE. THE FIRST AND LAST "
990 PRINT"COLORS ENTERED ON THE COLORATION MENU"
992 PRINT"WILL BE DISPLAY WITH THE B/W CAMERA."
994 PRINT:PRINT"   END OF INSTRUCTIONS"
995 PRINT"PRESS ANY KEY TO RETURN TO MENU"
996 GETA$
997 IFA$=""THEN996
998 GOTO180
1000 GOTO180:REM VECTOR FOR PROGRAM INSERTATION
```

Fig. 9-15. Continued.

```
10 REM 128 B/W CAM
20 REM BY   JOHN IOVINE
22 PRINT"{CLR}"
23 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}
   {C/DN}PLEASE WAIT....LOAD
25 CS=0
30 FORL=5039TO5541:READA:POKEL,A
40 CS=CS+A:NEXTL
45 IFCS<>57222THENPRINT" DATA ERROR !!":END
50 PRINT:PRINT:PRINT"  LOAD SUCCESSFUL..."
55 PRINT"    RETURNING TO MAIN MENU"
56 FORT=1TO500:NEXTT
57 CLR
60 LOAD"128 MAIN PROG",8
70 DATA 000,004,003,007,008,012,011,015,016,020
80 DATA 019,023,024,028,027,031,032,036,035,039
90 DATA 040,044,043,047,048,052,051,055,056,060
100 DATA 059,063,064,000,000,000,000,000,000,000
110 DATA 076,012,079,015,080,016,084,020,083,019
120 DATA 087,023,088,024,092,028,091,027,095,031
130 DATA 096,032,100,036,099,035,103,039,104,040
140 DATA 108,044,107,043,111,047,112,048,115,051
150 DATA 000,173,002,220,141,080,019,169,255,141
160 DATA 003,221,120,169,011,141,002,220,169,011
170 DATA 141,000,220,141,000,220,160,000,169,011
180 DATA 141,000,220,140,001,221,169,009,141,000
190 DATA 220,200,192,127,208,238,169,011,141,000
200 DATA 220,234,234,234,234,234,234,160,000,162
210 DATA 000,169,011,141,000,220,185,175,019,141
220 DATA 001,221,169,009,141,000,220,169,001,141
230 DATA 000,220,189,215,019,009,128,141,001,221
240 DATA 206,000,220,169,008,141,000,220,169,009
250 DATA 141,000,220,232,224,040,208,225,162,000
260 DATA 200,192,033,208,202,169,000,133,250,133
270 DATA 252,169,004,133,251,076,068,021,234,160
280 DATA 000,162,000,169,011,141,000,220,189,175
290 DATA 019,141,001,221,169,009,141,000,220,185
300 DATA 215,019,141,001,221,169,008,141,000,220
310 DATA 173,013,221,201,016,208,096,076,096,021
320 DATA 234,234,200,192,040,208,223,160,000,232
330 DATA 224,033,240,106,169,011,141,000,220,189
340 DATA 175,019,141,001,221,169,009,141,000,220
350 DATA 185,215,019,141,001,221,169,008,141,000
360 DATA 220,173,013,221,201,016,208,052,076,108
370 DATA 021,234,169,001,141,000,220,200,192,040
```

Fig. 9-16. B/W camera program C-128.

```
380 DATA 208,219,024,165,250,105,040,133,250,169
390 DATA 000,105,000,133,252,165,251,101,252,133
400 DATA 251,234,234,160,000,234,234,234,232,234
410 DATA 076,132,020,169,003,145,250,076,169,020
420 DATA 177,250,201,003,240,007,169,002,145,250
430 DATA 076,221,020,169,003,145,250,076,221,020
440 DATA 169,000,133,250,133,252,169,004,133,251
450 DATA 173,080,019,141,002,220,088,076,128,021
460 DATA 234,234,120,169,011,141,002,220,076,052
470 DATA 020,096,141,000,220,160,000,169,011,141
480 DATA 000,220,162,202,202,208,253,200,173,013
490 DATA 221,192,025,208,243,234,234,141,000,220
500 DATA 076,127,020,169,000,145,250,169,001,141
510 DATA 000,220,076,169,020,177,250,201,003,208
520 DATA 007,169,001,145,250,076,220,020,169,000
530 DATA 145,250,076,220,020,165,212,201,088,240
540 DATA 012,201,044,240,011,201,047,240,013,201
550 DATA 017,240,015,076,055,021,238,085,021,076
560 DATA 055,021,206,085,021,076,055,021,076,064
570 DATA 021,255,019
```

Fig. 9-16. Continued.

```
10 REM  C-128 GRAY CAMERA
15 REM  BY     JOHN IOVINE
16 CS=0
17 PRINT"{CLR}"
18 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}
   {C/DN}{C/DN}{C/DN}PLEASE
20 FORL=5039TO5858:READA:POKEL,A
25 CS=CS+A:NEXTL
27 IFCS<>94302THEN PRINT "DATA ERROR !!":END
30 PRINT:PRINT:PRINT"LOAD SUCCESSFUL..."
35 PRINT"RETURNING TO MAIN MENU"
40 FORT=1TO500:NEXTT
50 CLR
51 LOAD"128 MAIN PROG",8
60 DATA 000,004,003,007,008,012,011,015,016,020
70 DATA 019,023,024,028,027,031,032,036,035,039
80 DATA 040,044,043,047,048,052,051,055,056,060
90 DATA 059,063,064,000,000,000,000,000,000,000
100 DATA 076,012,079,015,080,016,084,020,083,019
110 DATA 087,023,088,024,092,028,091,027,095,031
120 DATA 096,032,100,036,099,035,103,039,104,040
130 DATA 108,044,107,043,111,047,112,048,115,051
```

Fig. 9-17. Gray scale camera program C-128.

```
140 DATA 000,173,002,220,141,080,019,169,255,141
150 DATA 003,221,120,169,011,141,002,220,141,000
160 DATA 220,160,000,169,011,141,000,220,140,001
170 DATA 221,169,009,141,000,220,200,192,127,208
180 DATA 238,169,000,133,250,133,252,169,004,133
190 DATA 251,234,160,000,162,000,169,011,141,000
200 DATA 220,189,175,019,141,001,221,169,009,141
210 DATA 000,220,169,001,141,000,220,185,215,019
220 DATA 141,001,221,206,000,220,169,008,141,000
230 DATA 220,238,000,220,200,192,040,208,229,160
240 DATA 000,232,238,117,022,173,117,022,201,002
250 DATA 208,200,169,000,141,117,022,142,116,022
260 DATA 234,234,234,160,000,174,116,022,202,202
270 DATA 169,011,141,000,220,189,175,019,141,001
280 DATA 221,173,013,221,169,009,141,000,220,185
290 DATA 215,019,141,001,221,206,000,220,238,000
300 DATA 220,173,013,221,201,016,208,071,076,041
310 DATA 021,234,200,192,040,208,228,160,000,232
320 DATA 224,033,240,075,169,011,141,000,220,189
330 DATA 175,019,141,001,221,173,013,221,169,009
340 DATA 141,000,220,185,215,019,141,001,221,206
350 DATA 000,220,238,000,220,173,013,221,201,016
360 DATA 208,024,076,048,021,234,200,192,040,208
370 DATA 220,076,068,021,234,234,234,234,234,169
380 DATA 003,145,250,076,170,020,177,250,201,003
390 DATA 240,004,169,002,145,250,076,224,020,169
400 DATA 000,133,250,133,252,169,004,133,251,173
410 DATA 080,019,141,002,220,088,165,212,201,088
420 DATA 240,003,076,183,022,120,169,011,141,002
430 DATA 220,076,051,020,096,076,183,022,169,002
440 DATA 145,250,076,170,020,177,250,201,002,240
450 DATA 007,169,001,145,250,076,224,020,169,000
460 DATA 145,250,076,224,020,142,116,022,160,000
470 DATA 162,202,202,208,253,200,192,007,208,246
480 DATA 174,116,022,160,000,076,106,021,200,192
490 DATA 015,208,239,142,000,220,160,000,076,122
500 DATA 020,234,234,160,000,202,169,011,141,000
510 DATA 220,189,175,019,141,001,221,173,013,221
520 DATA 169,009,141,000,220,185,215,019,141,001
530 DATA 221,206,000,220,238,000,220,173,013,221
540 DATA 201,016,208,003,076,060,022,200,192,040
550 DATA 208,229,160,000,232,169,011,141,000,220
560 DATA 189,175,019,141,001,221,173,013,221,169
570 DATA 009,141,000,220,185,215,019,141,001,221
580 DATA 206,000,220,238,000,220,173,013,221,201
```

Fig. 9-17. Continued.

```
590 DATA 016,208,003,076,076,022,200,192,040,208
600 DATA 229,234,234,238,112,022,173,112,022,201
610 DATA 001,240,009,201,002,240,026,201,003,240
620 DATA 043,234,169,128,141,097,022,169,129,141
630 DATA 090,022,169,130,141,070,022,141,083,022
640 DATA 076,128,022,169,192,141,097,022,169,193
650 DATA 141,090,022,169,194,141,070,022,141,083
660 DATA 022,076,160,022,169,064,141,097,022,169
670 DATA 065,141,090,022,169,066,141,070,022,141
680 DATA 083,022,169,000,141,112,022,024,165,250
690 DATA 105,040,133,250,169,000,105,000,133,252
700 DATA 165,251,101,252,133,251,160,000,232,076
710 DATA 055,020,234,177,250,201,003,240,003,076
720 DATA 150,021,169,066,145,250,076,150,021,177
730 DATA 250,201,003,240,007,201,066,240,010,076
740 DATA 199,021,169,065,145,250,076,199,021,169
750 DATA 064,145,250,076,199,021,234,000,255,000
760 DATA 255,000,255,000,255,000,000,000,000,034
770 DATA 000,000,000,000,000,000,000,000,000,000
780 DATA 000,142,116,022,160,000,162,202,202,208
790 DATA 253,200,192,028,208,246,174,116,022,160
800 DATA 000,076,106,021,000,255,000,255,000,255
810 DATA 000,255,000,142,116,022,160,000,162,202
820 DATA 202,208,253,200,192,072,208,246,174,116
830 DATA 022,160,000,076,106,021,165,212,201,044
840 DATA 240,012,201,047,240,020,201,017,208,001
850 DATA 096,076,028,021,238,080,021,238,140,022
860 DATA 238,172,022,076,028,021,206,080,021,206
870 DATA 140,022,206,172,022,076,028,021,234,234
```

Fig. 9-17. Continued.

PROGRAM OPERATION

Type in the respective program for your computer, (Figs. 9-15, 9-16, and 9-17 for the C-128, Figs. 9-18, 9-19, and 9-20 for the C-64). Take care in saving the programs under their proper names. This is essential, for the main program to load the camera a subprogram into memory, and return. You can, if you wish, change the programs' names to whatever you like, but remember to change the names inside each individual program. For now, the program names are as follows:

	C-64	**C-128**
1)	64 main prog	128 main prog
2)	64 b/w cam	128 b/w cam
3)	64 gray cam	128 gray cam

It is also important to type in the entire main program. If you don't, you may run into difficulties when loading the camera programs because the start of variable pointer will be overwritten, causing you no end of problems. Do not leave out any of the lines or shorten the program.

After you have typed and saved the programs, load and run the main program. At the menu prompt choose item 2 "load the b/w camera". The computer will then load the b/w program, return to the main program, and then start the camera, item 7. In the beginning, I advised you to use a simple subject to get acquainted with the digital camera. As a prop, use a white cup as I have for illustration Fig. 9-21. Copy the lighting arrangement in the diagram (Fig. 9-14). With this set up, you can vary the f-stop on the lens and/or the timing cycle of the program. Notice the effects each one has on the image, You should see that the f-stop has more impact, and should be used to adjust the camera to the basic lighting conditions. The timing can then be used for fine adjustments. After you're satisfied with the b/w camera picture, return to the main menu by pressing the "R" key. Once there, load the gray scale camera item 3, then start it running. Return to the main menu by pressing the "R" key again. Now, this will take longer to happen, because I stated before the keyboard is only checked once per screen scan. Once you're back at the main menu, choose the gray timing option 5. Change the timing to 50, 60, and 70. The program automatically returns to the main menu. Restart the camera. Notice the changes the timing has on the digital camera picture. Return to the main menu again, and choose the coloration option 6. The submenu lists all the color codes. You are prompted for the coloration of each scan. Choose whatever colors you like; the program will return automatically after your choices are entered. Restart the camera. If you find you don't like the colors, or wish to change them, simply return to the main menu as before and change them.

As you can see, these are just basic camera manipulations. Feel free to experiment with the timing and lighting as you see fit. If you get stuck anywhere, remember to return to the basic operations. A trick you might like to try is generating negative images. You can generate negative images by reversing the gray scale or the b/w. Remember that the first and last colors chosen on the coloration menu can be displayed with the black and white camera.

This camera can be used for many types of experiments: machine vision, pattern and character recognition, the front end of a neural network, and the front end of a vision implantation system for the blind.

HI-RESOLUTION DIGITAL CAMERA

We will increase the resolution to 6X. The photo illustrations accompanying this section show what the camera can do at this point in its development, imaging the covers of magazines, currency, and high-contrast portraits of people (see Figs. 9-26, 9-27, and 9-28).

```
5 REM C-64 MAIN PROG    JOHN IOVINE
10 PRINTCHR$(142):REM SWITCH UPPER CASE
30 POKE56334,PEEK(56334)AND254
70 FORJ=12288TO12288+32: REM PLACE CHARACTER DATA IN RAM
80 READ A:POKEJ,A:NEXT
100 POKE56334,PEEK(56334)OR1
120 DATA,0,0,0,0,0,0,0,0:REM @
130 DATA 255,255,255,255,0,0,0,0
140 DATA 0,0,0,0,255,255,255,255
150 DATA 255,255,255,255,255,255,255,255
160 B=0
162 POKE53281,1:POKE53282,15:POKE53283,12:POKE53284,11
165 POKE53265,PEEK(53265)OR64
170 REM MENU
180 PRINT"{CLR}"
190 PRINT:PRINT:PRINT:PRINT
200 PRINT"                    DIGITAL CAMERA MENU"
210 PRINT:PRINT:PRINT
215 PRINT " 1) INSTRUCTIONS
220 PRINT" 2) LOAD B/W FAST SCAN CAMERA"
225 PRINT" 3) LOAD G/S GRAY SCALE CAMERA"
230 PRINT" 4)    SET/RESET TIMING B/W MODE"
235 PRINT" 5)    SET/RESET TIMING G/S MODE"
240 PRINT" 6)    COLORATION  G/S"
250 PRINT" 7) START CAMERA"
255 PRINT" 8) QUIT"
260 PRINT:PRINT"INPUT OPTION NUMBER 1-8"
261 FOR T=0TO175:NEXTT
262 POKE197,64:POKE198,0
265 INPUTX:IFX<0 THEN280:IFX>7THEN280
270 ONX GOTO290,300,400,500,600,700,800,900
280 PRINT"ERROR, PLEASE ENTER NUMBER BETWEEN 1-7":GOTO265
290 GOTO910
300 CLR
301 LOAD"64 B/W CAM",8
400 CLR
401 LOAD"64 GRAY CAM",8
500 PRINT"{CLR}"
501 PRINT:PRINT:PRINT
502 PRINT"    RESET / SET TIMING"
503 PRINT"   FAST SCAN B/W CAMERA"
504 PRINT:PRINT"NUMBERS REPRESENT DELAY IN MILLISECONDS"
505 PRINT"           BETWEEN SCREEN SCANS"
506 PRINT:PRINT"DEFAULT DELAY IS 16 MILLISECONDS"
507 G=PEEK(49574)
```

Fig. 9-18. Main program C-64.

```
508 PRINT"CURRENT DELAY IS ";G
509 INPUT" ENTER DELAY";G
510 POKE49574,G
511 PRINT:PRINT"THANK YOU"
512 FORT=1TO500:NEXTT
513 GOTO180
600 PRINT"{CLR}"
601 PRINT:PRINT:PRINT
602 PRINT"   SET / RESET TIMING"
603 PRINT"NUMBERS REPRESENT DELAY IN MILLISECONDS"
604 PRINT"            BETWEEN GRAY SCANS"
605 PRINT:PRINT
606 J=PEEK(49569):K=PEEK(49885):L=PEEK(49917)
607 PRINT:PRINT" 1ST SCAN IS NON-ADJUSTABLE"
608 PRINT:PRINT" 2ND SCAN DEFAULT DELAY IS   7"
609 PRINT"CURRENT DELAY IS ";J
610 INPUT"ENTER DELAY";J
611 PRINT:PRINT" 3RD SCAN DEFAULT DELAY IS 28"
612 PRINT"CURRENT DELAY IS ";K
613 INPUT"ENTER DELAY";K
614 PRINT:PRINT" 4TH SCAN DEFAULT DELAY IS 72"
615 PRINT"CURRENT DELAY IS ";L
616 INPUT"ENTER DELAY";L
617 POKE49569,J:POKE49885,K:POKE49917,L
618 PRINT"   THANK YOU"
619 FORT=1TO500:NEXTT
620 GOTO180
700 PRINT"{CLR}"
701 PRINT:PRINT:PRINT
702 PRINT"    COLOR CODES"
703 PRINT
704 PRINT"0 BLACK"," 8 ORANGE"
705 PRINT"1 WHITE"," 9 BROWN"
706 PRINT"2 RED","10 LIGHT RED"
707 PRINT"3 CYAN","11 DARK GRAY"
708 PRINT"4 PURPLE","12 MEDIUM GRAY"
709 PRINT"5 GREEN","13 LIGHT GREEN"
710 PRINT"6 BLUE","14 LIGHT BLUE"
711 PRINT"7 YELLOW","15 LIGHT GRAY"
712 PRINT:PRINT
714 PRINT"DEFAULT COLOR FOR 1ST SCAN IS  1"
715 INPUT"ENTER COLOR CODE # ";C
716 PRINT
717 PRINT"DEFAULT COLOR FOR 2ND SCAN IS 15"
718 INPUT"ENTER COLOR CODE # ";D
```

Fig. 9-18. Continued.

```
719 PRINT
720 PRINT"DEFAULT COLOR FOR 3RD SCAN IS 12"
721 INPUT"ENTER COLOR CODE # ";E
722 PRINT
723 PRINT"DEFAULT COLOR FOR 4TH SCAN IS 11"
724 INPUT"ENTER COLOR CODE # ";F
725 PRINT
726 PRINT"DEFAULT COLOR FOR BKGRD IS  0"
727 INPUT"ENTER COLOR CODE # ";B
728 POKE53281,C:POKE53282,D:POKE53283,E:POKE53284,F
729 PRINT:PRINT" THANK YOU"
730 FORT=1TO500:NEXTT
731 GOTO180
800 G=PEEK(53272):POKE53272,(PEEK(53272)AND240)+12
801 PRINT"{CLR}":FORL=55296TO56295:POKEL,B:NEXT:SYS49233
802 POKE53272,G:GOTO1000
900 END
910 PRINT"{CLR}"
912 PRINT:PRINT:PRINT:PRINT
914 PRINT"  DIGITAL CAMERA INSTRUCTIONS"
916 PRINT"                         PG1"
918 PRINT:PRINT
920 PRINT"THE DIGITAL CAMERA PROGRAM PROVIDES"
922 PRINT"THREE COMMANDS THAT CAN BE UTILIZED"
924 PRINT"DURING CAMERA OPERATION.(ON-THE-FLY)"
926 PRINT"THESE COMMANDS ARE AS FOLLOWS:"
928 PRINT
930 PRINT"PRESS > KEY TO INCREASE TIMING"
932 PRINT"PRESS < KEY TO DECREASE TIMING"
934 PRINT"PRESS R KEY TO RETURN TO MENU"
936 PRINT
938 PRINT"THE TIMING KEYS OPERATE WITH BOTH"
940 PRINT"B/W AND GRAY CAMERAS."
942 PRINT"THE KEYS WILL INCREMENT OR DECREMENT"
944 PRINT"THE OVERALL SCAN TIMING BY ONE"
946 PRINT"MILLISECOND PER SCREEN SCAN THAT THE"
948 PRINT"KEY IS HELD."
950 PRINT"IN THE GRAY CAMERA MODE, THIS MEANS"
952 PRINT"THAT EACH OF THE THREE GRAY SCALE"
954 PRINT"TIMING MODES ARE SIMULTANEOUSLY CHANGED"
956 PRINT:PRINT"PRESS ANY KEY TO CONTINUE"
958 GETA$
960 IFA$=""THEN958
962 PRINT"{CLR}"
964 PRINT:PRINT:PRINT
```

Fig. 9-18. Continued.

```
966 PRINT"   DIGITAL CAMERA INSTRUCTIONS"
968 PRINT"                          PG 2"
970 PRINT:PRINT
972 PRINT"BY PRESSING THE R KEY, THE PROGRAM"
974 PRINT"WILL RETURN TO THE BASIC MENU. HERE"
976 PRINT"FURTHER ADJUSTMENTS IN TIMING ARE "
978 PRINT"POSSIBLE BY ALLOWING THE USER TO FIRST"
980 PRINT"READ THE TIMING SCANS BEFORE ADJUSTING"
982 PRINT"PSEUDO-COLORATION OF THE GRAY SCALES"
984 PRINT"IS IMPLEMENTED BY CHOOSING THE COLORATION"
986 PRINT"ITEM ON THE MENU. COLORATION OF THE B/W"
988 PRINT"CAMERA IS POSSIBLE, THE FIRST AND LAST"
990 PRINT"COLORS ENTERED ON THE COLORATION MENU"
992 PRINT"WILL BE DISPLAY WITH THE B/W CAMERA"
994 PRINT:PRINT"  END OF INSTRUCTIONS"
995 PRINT"PRESS ANY KEY TO RETURN TO MENU"
996 GETA$
997 IFA$=""THEN996
998 GOTO180
1000 GOTO180:REM VECTOR FOR PROGRAM INSERTATION
```

Fig. 9-18. Continued.

```
5 REM 64 B/W CAM
10 REM B/W FAST SCAN CAMERA
20 REM BY   JOHN IOVINE
22 PRINT"{CLR}"
23 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}
   {C/DN}PLEASE WAIT....LOAL
25 CS=0
30 FORL=49152TO49654:READA:POKEL,A
40 CS=CS+A:NEXTL
45 IF CS<>62021THENPRINT "DATA ERROR":END
50 PRINT:PRINT:PRINT"  LOAD SUCCESSFUL..."
55 PRINT"    RETURNING TO MAIN MENU"
56 FORT=1TO500:NEXTT
60 LOAD"64 MAIN PROG",8
70 DATA 000,004,003,007,008,012,011,015,016,020
80 DATA 019,023,024,028,027,031,032,036,035,039
90 DATA 040,044,043,047,048,052,051,055,056,060
100 DATA 059,063,064,000,000,000,000,000,000,000
110 DATA 076,012,079,015,080,016,084,020,083,019
120 DATA 087,023,088,024,092,028,091,027,095,031
130 DATA 096,032,100,036,099,035,103,039,104,040
140 DATA 108,044,107,043,111,047,112,048,115,051
```

Fig. 9-19. B/W camera program C-64.

```
150 DATA 000,173,002,220,141,037,192,169,255,141
160 DATA 003,221,120,169,011,141,002,220,169,011
170 DATA 141,000,220,141,000,220,160,000,169,011
180 DATA 141,000,220,140,001,221,169,009,141,000
190 DATA 220,200,192,127,208,238,169,011,141,000
200 DATA 220,234,234,234,234,234,234,160,000,162
210 DATA 000,169,011,141,000,220,185,000,192,141
220 DATA 001,221,169,009,141,000,220,169,001,141
230 DATA 000,220,189,040,192,234,234,141,001,221
240 DATA 206,000,220,169,008,141,000,220,169,009
250 DATA 141,000,220,232,224,040,208,225,162,000
260 DATA 134,252,200,192,033,208,200,169,200,133
270 DATA 250,169,004,133,251,076,149,193,234,160
280 DATA 000,162,000,169,011,141,000,220,189,000
290 DATA 192,141,001,221,169,009,141,000,220,185
300 DATA 040,192,141,001,221,169,008,141,000,220
310 DATA 173,013,221,201,016,208,096,076,177,193
320 DATA 234,234,200,192,040,208,223,160,000,232
330 DATA 224,033,240,106,169,011,141,000,220,189
340 DATA 000,192,141,001,221,169,009,141,000,220
350 DATA 185,040,192,141,001,221,169,008,141,000
360 DATA 220,173,013,221,201,016,208,052,076,189
370 DATA 193,234,169,001,141,000,220,200,192,040
380 DATA 208,219,024,165,250,105,040,133,250,169
390 DATA 000,105,000,133,252,165,251,101,252,133
400 DATA 251,234,234,160,000,234,234,234,232,234
410 DATA 076,213,192,169,003,145,250,076,250,192
420 DATA 177,250,201,003,240,007,169,002,145,250
430 DATA 076,045,193,169,003,145,250,076,045,193
440 DATA 169,200,133,250,234,234,169,004,133,251
450 DATA 173,037,192,141,002,220,088,076,209,193
460 DATA 234,234,120,169,011,141,002,220,076,133
470 DATA 192,096,141,000,220,160,000,140,001,221
480 DATA 169,011,141,000,220,162,202,202,208,253
490 DATA 200,192,016,208,243,234,234,173,013,221
500 DATA 076,208,192,169,000,145,250,169,001,141
510 DATA 000,220,076,250,192,177,250,201,003,208
520 DATA 007,169,001,145,250,076,045,193,169,000
530 DATA 145,250,076,045,193,165,197,201,064,240
540 DATA 012,201,044,240,011,201,047,240,013,201
550 DATA 017,240,015,076,136,193,238,166,193,076
560 DATA 136,193,206,166,193,076,136,193,076,145
570 DATA 193,255,019
```

Fig. 9-19. Continued.

```
10 REM  C-64 GRAY CAMERA
15 REM  BY    JOHN IOVINE
16 CS=0
17 PRINT"{CLR}"
18 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}{C/DN}
   {C/DN}{C/DN}{C/DN}PLEASE
20 FORL=49152TO49971:READA:POKEL,A
25 CS=CS+A:NEXTL
27 IFCS<>107792THEN PRINT "DATA ERROR !!":END
30 PRINT:PRINT:PRINT"LOAD SUCCESSFUL..."
35 PRINT"RETURNING TO MAIN MENU"
40 FORT=1TO500:NEXTT
50 CLR
51 LOAD"64 MAIN PROG",8
60 DATA 000,004,003,007,008,012,011,015,016,020
70 DATA 019,023,024,028,027,031,032,036,035,039
80 DATA 040,044,043,047,048,052,051,055,056,060
90 DATA 059,063,064,000,000,000,000,000,000,000
100 DATA 076,012,079,015,080,016,084,020,083,019
110 DATA 087,023,088,024,092,028,091,027,095,031
120 DATA 096,032,100,036,099,035,103,039,104,040
130 DATA 108,044,107,043,111,047,112,048,115,051
140 DATA 000,173,002,220,141,198,194,169,255,141
150 DATA 003,221,120,169,011,141,002,220,141,000
160 DATA 220,160,000,169,011,141,000,220,140,001
170 DATA 221,169,009,141,000,220,200,192,127,208
180 DATA 238,169,000,133,250,133,252,169,004,133
190 DATA 251,234,160,000,162,000,169,011,141,000
200 DATA 220,189,000,192,141,001,221,169,009,141
210 DATA 000,220,169,001,141,000,220,185,040,192
220 DATA 141,001,221,206,000,220,169,008,141,000
230 DATA 220,238,000,220,200,192,040,208,229,160
240 DATA 000,232,238,199,194,173,199,194,201,002
250 DATA 208,200,169,000,141,199,194,142,200,194
260 DATA 234,234,234,160,000,174,200,194,202,202
270 DATA 169,011,141,000,220,189,000,192,141,001
280 DATA 221,173,013,221,169,009,141,000,220,185
290 DATA 040,192,141,001,221,206,000,220,238,000
300 DATA 220,173,013,221,201,016,208,071,076,122
310 DATA 193,234,200,192,040,208,228,160,000,232
320 DATA 224,033,240,075,169,011,141,000,220,189
330 DATA 000,192,141,001,221,173,013,221,169,009
340 DATA 141,000,220,185,040,192,141,001,221,206
350 DATA 000,220,238,000,220,173,013,221,201,016
360 DATA 208,024,076,129,193,234,200,192,040,208
```

Fig. 9-20. Gray scale camera program C-64.

```
370 DATA 220,076,149,193,234,234,234,234,234,169
380 DATA 003,145,250,076,251,192,177,250,201,003
390 DATA 240,004,169,002,145,250,076,049,193,169
400 DATA 000,133,250,133,252,169,004,133,251,173
410 DATA 198,194,141,002,220,088,165,197,201,064
420 DATA 240,003,076,008,195,120,169,011,141,002
430 DATA 220,076,132,192,096,076,008,195,169,002
440 DATA 145,250,076,251,192,177,250,201,002,240
450 DATA 007,169,001,145,250,076,049,193,169,000
460 DATA 145,250,076,049,193,142,200,194,160,000
470 DATA 162,202,202,208,253,200,192,007,208,246
480 DATA 174,200,194,160,000,076,187,193,200,192
490 DATA 015,208,239,142,000,220,160,000,076,203
500 DATA 192,234,234,160,000,202,169,011,141,000
510 DATA 220,189,000,192,141,001,221,173,013,221
520 DATA 169,009,141,000,220,185,040,192,141,001
530 DATA 221,206,000,220,238,000,220,173,013,221
540 DATA 201,016,208,003,076,141,194,200,192,040
550 DATA 208,229,160,000,232,169,011,141,000,220
560 DATA 189,000,192,141,001,221,173,013,221,169
570 DATA 009,141,000,220,185,040,192,141,001,221
580 DATA 206,000,220,238,000,220,173,013,221,201
590 DATA 016,208,003,076,157,194,200,192,040,208
600 DATA 229,234,234,238,201,194,173,201,194,201
610 DATA 001,240,009,201,002,240,026,201,003,240
620 DATA 043,234,169,128,141,178,194,169,129,141
630 DATA 171,194,169,130,141,151,194,141,164,194
640 DATA 076,209,194,169,192,141,178,194,169,193
650 DATA 141,171,194,169,194,141,151,194,141,164
660 DATA 194,076,241,194,169,064,141,178,194,169
670 DATA 065,141,171,194,169,066,141,151,194,141
680 DATA 164,194,169,000,141,201,194,024,165,250
690 DATA 105,040,133,250,169,000,105,000,133,252
700 DATA 165,251,101,252,133,251,160,000,232,076
710 DATA 136,192,234,177,250,201,003,240,003,076
720 DATA 231,193,169,066,145,250,076,231,193,177
730 DATA 250,201,003,240,007,201,066,240,010,076
740 DATA 024,194,169,065,145,250,076,024,194,169
750 DATA 064,145,250,076,024,194,234,000,255,000
760 DATA 255,000,255,000,255,000,000,000,000,034
770 DATA 000,000,000,000,000,000,000,000,000,000
780 DATA 000,142,200,194,160,000,162,202,202,208
790 DATA 253,200,192,028,208,246,174,200,194,160
800 DATA 000,076,187,193,000,255,000,255,000,255
810 DATA 000,255,000,142,200,194,160,000,162,202
```

Fig. 9-20. Continued.

```
820 DATA 202,208,253,200,192,072,208,246,174,200
830 DATA 194,160,000,076,187,193,165,197,201,044
840 DATA 240,012,201,047,240,020,201,017,208,001
850 DATA 096,076,109,193,238,161,193,238,221,194
860 DATA 238,253,194,076,109,193,206,161,193,206
870 DATA 221,194,206,253,194,076,109,193,255,000
```

Fig. 9-20. Continued.

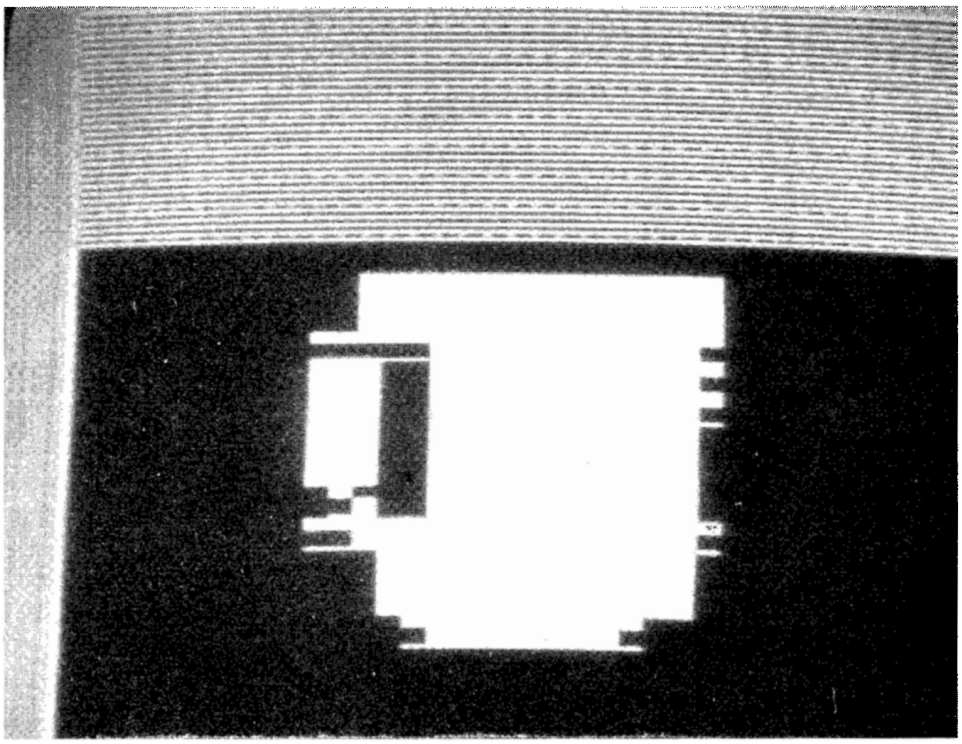


Fig. 9-21. D-Cam screen lo-res item, cup.

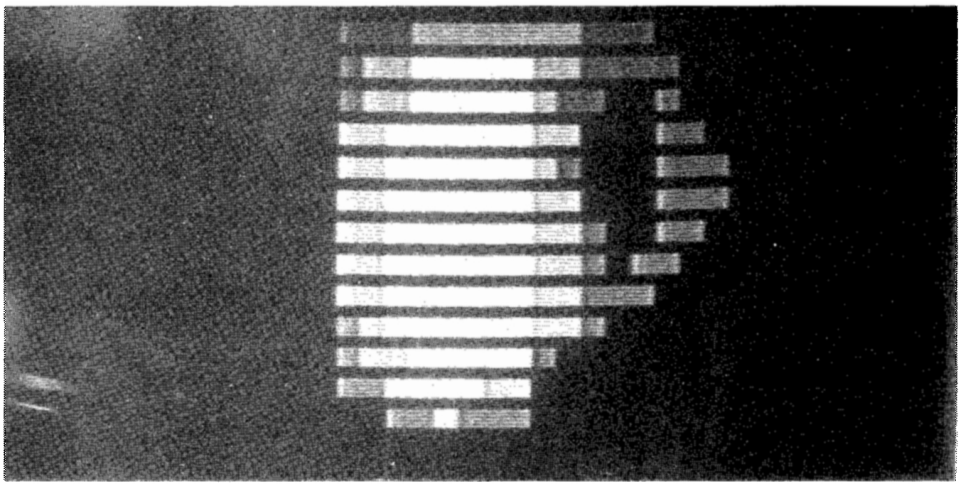


Fig. 9-22. D-Cam screen lo-res item, cup with gray scale.

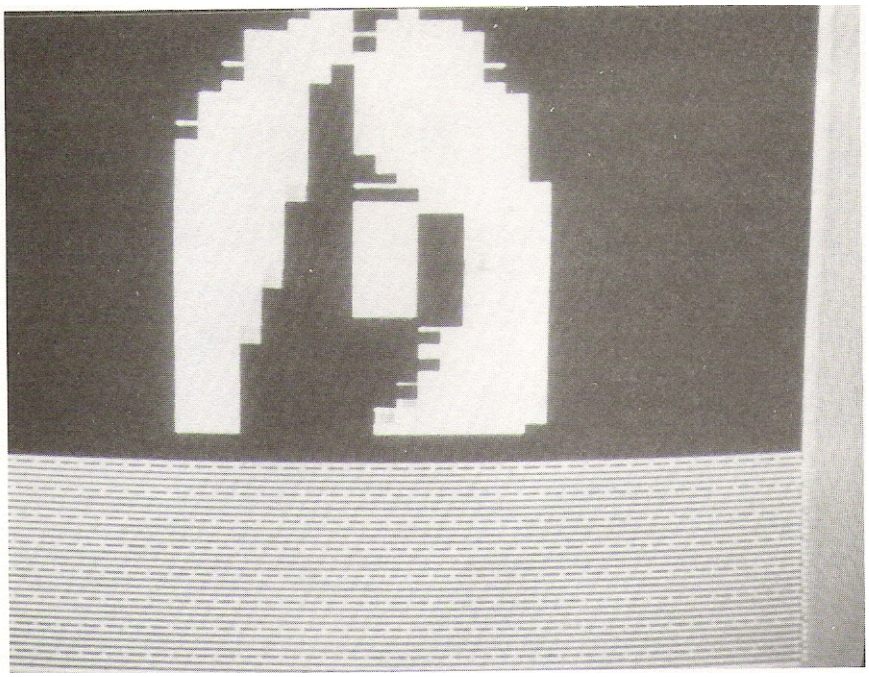


Fig. 9-23. D-Cam screen lo-res item, hand "OK" sign.

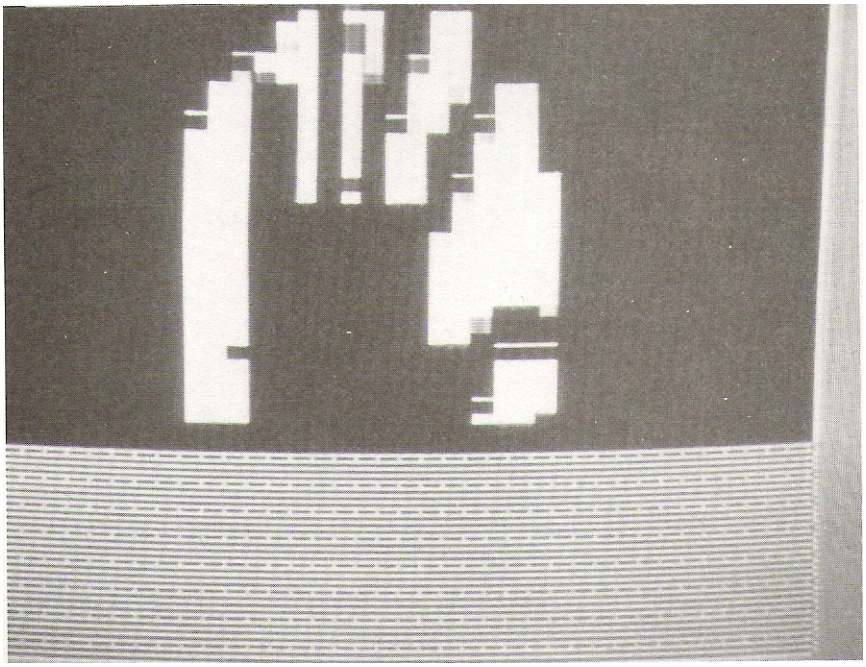


Fig. 9-24. D-Cam screen lo-res item, hand.

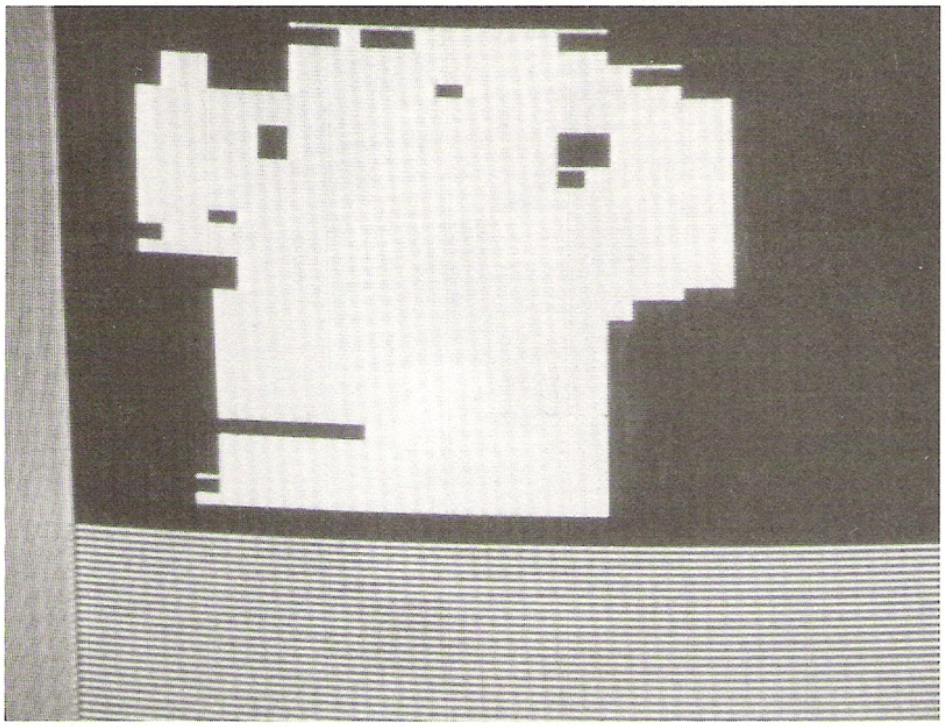


Fig. 9-25. D-Cam screen lo-res item, telephone.

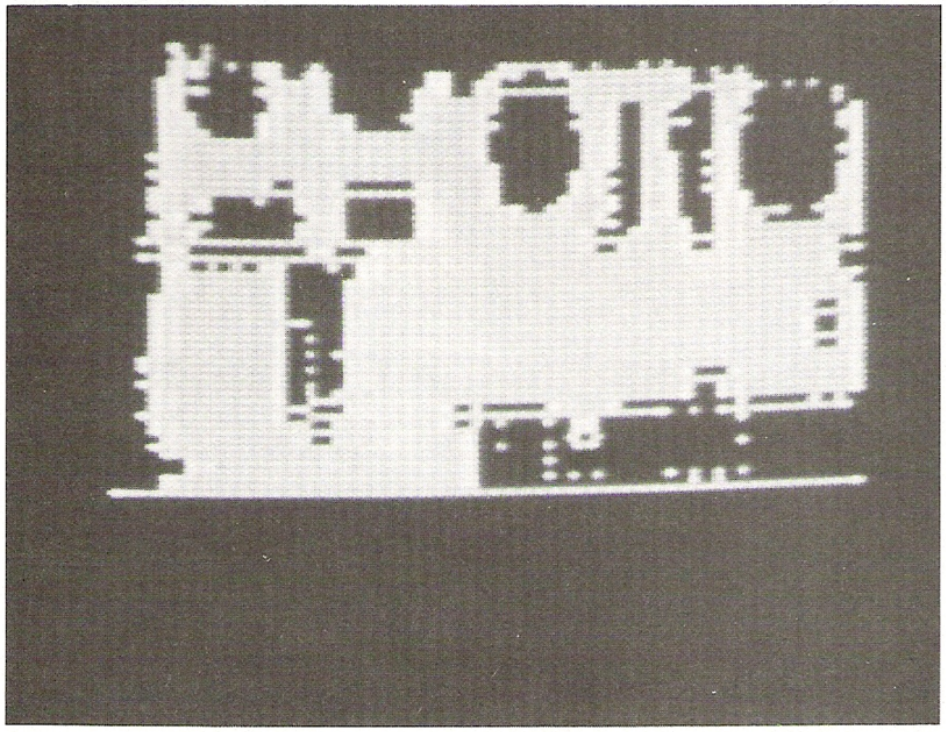


Fig. 9-26. D-Cam screen hi-res item, letters.

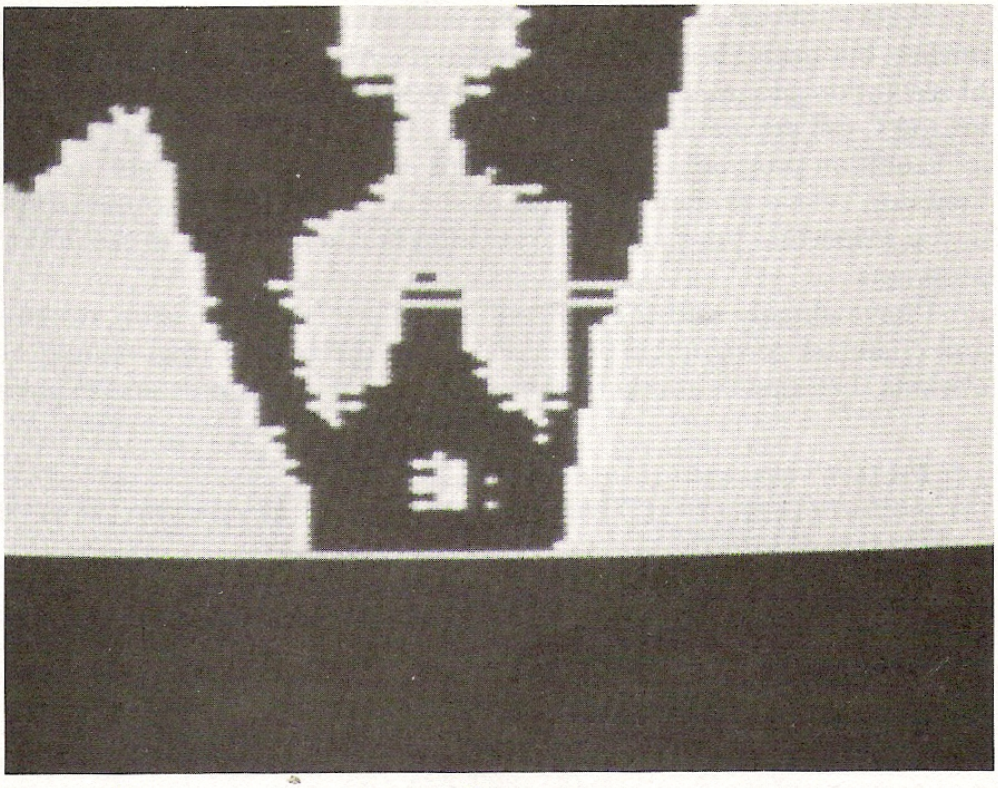


Fig. 9-27. D-Cam screen hi-res item, portrait.

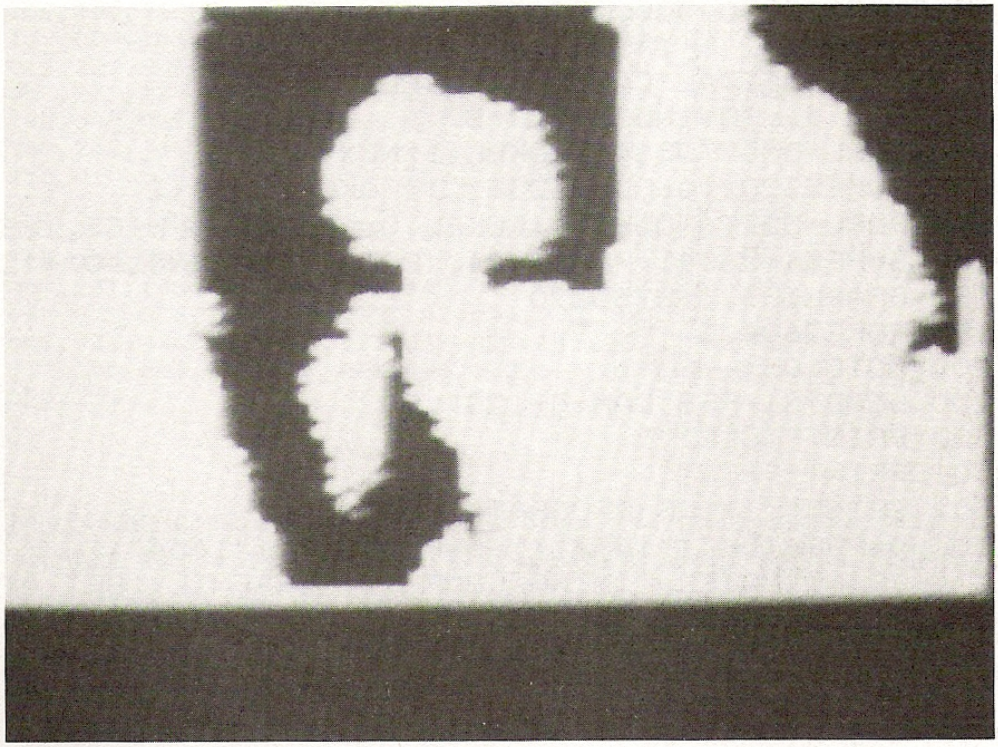


Fig. 9-28. D-Cam screen hi-res item, portrait.

A gray scale is of course possible, and would increase the resolution further, but hasn't been implemented in the hi-resolution programs. The hi-resolution programs (Figs. 9-29 and 9-30 for the C-128, and Fig. 9-31 for the C-64) are stand-alone programs. They are not connected to the lo-resolution programs. These programs are simpler to operate because they don't have as many features as the lo-resolution programs. Type in the respective program; the operation is similar to the lo-resolution programs.

I have already said that you could explore machine vision systems and character recognition programs. From the photos, that should be obvious. Let's push a little further now; stretch our minds and explore

```
10 PRINT"{CLR}"
20 PRINT:PRINT"{C/DN}{C/DN}{C/DN}{C/DN}MAIN MENU"
30 PRINT:PRINT
35 PRINT"  1) LOAD HI-RES CAMERA
36 PRINT"  2) CHANGE TIMING OF CAMERA
37 PRINT"  3) START CAMERA
38 PRINT"  4) QUIT"
39 FORT=1TO175:NEXT
40 POKE212,88:POKE208,0
41 INPUT"ENTER CHOICE (1-4)";X
42 IFX <1ORX>4THENPRINT"PLEASE ENTER NUM. BETWEEN 1 & 4":GOTO41
43 ONXGOTO250,300,152,350
100 REM 1452 & 1454
150 REM DIGITAL CAMERA CONTROL HI-RES
151 REM J. IOVINE 12-27-87
152 FORL=5632TO5640:POKEL,1:NEXT
155 POKE53281,0:POKE52382,1:POKE52383,12
160 FORL=55296TO56319:POKEL,0:NEXT:GRAPHIC3,1:SYS5120
180 G=PEEK(2604):POKE2604,(PEEK(2604)AND240)+12
185 GRAPHIC3,1:SYS5120
190 POKE2604,G
200 GOTO10
250 LOAD"HI-RES CAM C-128",8
300 PRINT"{CLR}"
301 PRINT:PRINT
302 PRINT"TIMING PROGRAM" :PRINT
303 D=PEEK(DEC("1452")):F=PEEK(DEC("1454"))
304 PRINT"OUTER TIMING LOOP IS CURRENTLY SET AT ";F :PRINT
306 PRINT"INNER LOOP TIMING IS CURRENTLY SET AT ";D :PRINT
307 INPUT"ENTER NEW INNER TIMING";D :PRINT
308 INPUT"ENTER NEW OUTER TIMING";F :PRINT
309 POKEDEC("1452"),D:POKEDEC("1454"),F
310 GOTO10
350 END
```

Fig. 9-29. Program hi-res camera C-128.

```
 IOVINE
4 REM HI-RES CAM C-128
6 PRINT"{CLR}":PRINT"LOADING PROGRAM...."
8 FORL=4864 TO 5423:READA:POKEL,A:B=B+A:NEXT
10 IF B<>54770 THEN PRINT"ERROR IN DATA STATEMENTS":END
12 PRINT"{C/DN}{C/DN}{C/DN}PROGRAM LOAD SUCESSFUL...RETURNING TO MAIN PROGRAM.."
14 FORI=1TO500:NEXTI
16 LOAD"128 HI-RES CAM",8
18 DATA 000,001,004,005,002,003,006,007,008,009,012,013,010,011,014
20 DATA 015,016,017,020,021,018,019,022,023,024,025,028,029,026,027
22 DATA 030,031,032,033,036,037,034,035,038,039,040,041,044,045,042
24 DATA 043,046,047,048,049,052,053,050,051,054,055,056,057,060,061
26 DATA 058,059,062,063,065,064,000,001,069,068,004,005,067,066,002
28 DATA 003,071,070,006,007,073,072,008,009,077,076,012,013,075,074
30 DATA 010,011,079,078,014,015,081,080,016,017,085,084,020,021,083
32 DATA 082,018,019,087,086,022,023,089,088,024,025,093,092,028,029
34 DATA 091,090,026,027,095,094,030,031,097,096,032,033,101,100,036
36 DATA 037,099,098,034,035,103,102,038,039,105,104,040,041,109,108
38 DATA 044,045,107,106,042,043,111,110,046,047,113,112,048,049,117
40 DATA 116,052,053,115,114,050,051,119,118,054,055,121,120,056,057
42 DATA 125,124,060,061,123,122,058,059,127,126,062,000,255,000,255
44 DATA 000,255,000,255,000,255,000,255,000,255,000,255,000,255,000
46 DATA 255,000,255,000,255,000,255,000,255,000,255,000,255,000,255
48 DATA 000,255,000,255,000,255,000,255,000,255,000,255,000,255,000
50 DATA 255,000,255,000,255,000,255,000,255,000,255,000,255,000,255
52 DATA 000,173,002,220,141,047,021,169,255,141,003,221,120,169,011
54 DATA 141,002,220,160,000,162,000,169,011,141,000,220,185,000,019
56 DATA 141,001,221,169,009,141,000,220,169,001,141,000,220,189,064
58 DATA 019,141,001,221,169,000,141,000,220,169,008,141,000,220,169
60 DATA 009,141,000,220,232,224,128,208,225,162,000,200,192,064,208
62 DATA 202,076,081,020,076,021,021,162,190,160,190,136,208,253,202
64 DATA 208,248,234,160,000,162,000,169,011,141,000,220,189,000,019
66 DATA 141,001,221,169,009,141,000,220,185,064,019,141,001,221,206
68 DATA 000,220,173,013,221,201,016,208,119,076,006,021,200,192,128
70 DATA 208,227,160,000,232,224,064,240,191,234,076,096,020,142,048
72 DATA 021,134,255,140,049,021,132,253,006,253,234,169,248,037,253
74 DATA 141,051,021,165,255,041,007,013,051,021,168,169,000,133,252
76 DATA 169,248,037,255,010,038,252,010,038,252,010,038,252,133,251
78 DATA 165,255,074,074,074,024,101,254,105,032,101,252,133,252,096
80 DATA 165,253,041,007,170,232,169,000,056,106,202,208,252,017,251
82 DATA 145,251,096,165,253,041,007,170,232,169,000,056,106,202,208
84 DATA 252,073,255,049,251,145,251,096,032,147,020,032,227,020,174
86 DATA 048,021,172,049,021,076,131,020,032,147,020,032,209,020,174
88 DATA 048,021,172,049,021,076,131,020,173,047,021,141,002,220,088
90 DATA 165,212,201,017,240,003,076,038,021,096,120,169,011,141,002
92 DATA 220,076,017,020,255
```

Fig. 9-30. Program hi-res C-128.

the possibility of interfacing to the brain. We could consider that the ultimate interfacing challenge. Our reason to accept such a challenge? To provide an artificial vision system for the visually handicapped.

ARTIFICIAL VISION

The question of providing sight for the blind is not a question of possibility. Over the past few decades, experiments providing electrical stimulation of the vision center of the human brain, caused totally blind people to see glowing phosphenes. The work that remains to be done

to complete a vision prosthesis, is a refinement in technique and technology.

To understand where we are, at the present time, let's first define a phosphene, the unit of light that has been generated. A phosphene to sighted individuals can be described as the after image left from a flash bulb. Perhaps you can remember from a party, when a friend took your picture, the after image left from the camera flash. That phosphene would probably be remembered as a glob of light that took a little while to disappear. The phosphenes generated by electrical stimulation are much smaller, more pixel like. It is interesting to note that this electrical stimulation is immediately recognized as visual information from blind research patients.

```
10 PRINT"{CLR}"
20 PRINT:PRINT"{C/DN}{C/DN}{C/DN}{C/DN}MAIN MENU"
30 PRINT:PRINT
35 PRINT"  1) LOAD HI-RES CAMERA
36 PRINT"  2) CHANGE TIMING OF CAMERA
37 PRINT"  3) START CAMERA
38 PRINT"  4) QUIT"
39 FORT=1TO255:NEXT
40 POKE197,64:POKE198,0
41 INPUT"ENTER CHOICE (1-4)";X
42 IFX <1ORX>4THENPRINT"PLEASE ENTER NUM. BETWEEN 1 & 4"
   :GOTO41
43 ONXGOTO360,300,152,350
150 REM DIGITAL CAMERA CONTROL HI-RES
151 REM J. IOVINE 3-14-88
152 G=PEEK(53272):H=PEEK(53265)
155 POKE53272,PEEK(53272)OR8
160 SYS49344
180 POKE53272,G:POKE53265,H
200 GOTO10
300 PRINT"{CLR}"
301 PRINT:PRINT
302 PRINT"TIMING PROGRAM" :PRINT
303 D=PEEK(49490):F=PEEK(49492)
304 PRINT"OUTER TIMING LOOP IS SET AT ";F :PRINT
306 PRINT"INNER TIMING LOOP IS SET AT ";D :PRINT
307 INPUT"ENTER NEW INNER TIMING";D :PRINT
308 INPUT"ENTER NEW OUTER TIMING";F :PRINT
309 POKE49490,D:POKE49492,F
310 GOTO10
350 END
```

Fig. 9-31. Program hi-res C-64.

```
354 PRINT"{CLR}{C/DN}{C/DN}DIGITAL CAMERA IS ALREADY IN
    MEMORY"
355 PRINT"DO NOT RELOAD OR ERROR IN DATA STATEMENT WILL
    RESULT"
356 PRINT"PRESS ANY KEY TO REURN TO MAIN MENU"
357 GET K$:IFK$=""THEN 357
358 GOTO10
360 IFPZ=1THEN354
363 PRINT"{CLR}":PRINT"{C/DN}{C/DN}LOADING HI-RES CAMERA"
    :REM C-64 HI-RES LOADER"
364 FORI=49152TO49715:READA:POKEI,A
365 B=B+A:NEXT:PZ=1
366 IFB<>60182 THEN PRINT"ERROR IN DATA STATEMENTS":END
367 PRINT"{C/DN}{C/DN}{C/DN}LOAD SUCESSFUL,..RETURNING TO
    MAIN MENU"
368 FORT=1TO999:NEXT:GOTO10
370 DATA 000,001,004,005,002,003,006,007,008,009
380 DATA 012,013,010,011,014,015,016,017,020,021
390 DATA 018,019,022,023,024,025,028,029,026,027
400 DATA 030,031,032,033,036,037,034,035,038,039
410 DATA 040,041,044,045,042,043,046,047,048,049
420 DATA 052,053,050,051,054,055,056,057,060,061
430 DATA 058,059,062,063,065,064,000,001,069,068
440 DATA 004,005,067,066,002,003,071,070,006,007
450 DATA 073,072,008,009,077,076,012,013,075,074
460 DATA 010,011,079,078,014,015,081,080,016,017
470 DATA 085,084,020,021,083,082,018,019,087,086
480 DATA 022,023,089,088,024,025,093,092,028,029
490 DATA 091,090,026,027,095,094,030,031,097,096
500 DATA 032,033,101,100,036,037,099,098,034,035
510 DATA 103,102,038,039,105,104,040,041,109,108
520 DATA 044,045,107,106,042,043,111,110,046,047
530 DATA 113,112,048,049,117,116,052,053,115,114
540 DATA 050,051,119,118,054,055,121,120,056,057
550 DATA 125,124,060,061,123,122,058,059,127,126
560 DATA 062,000,169,032,013,017,208,141,017,208
570 DATA 169,032,133,252,162,064,169,000,133,251
580 DATA 168,145,251,200,208,251,230,252,228,252
590 DATA 176,245,162,250,169,001,202,157,000,004
600 DATA 157,250,004,157,244,005,157,238,006,208
610 DATA 241,076,000,193,255,000,255,000,255,000
620 DATA 255,000,255,000,255,000,173,002,220,141
630 DATA 047,021,169,255,141,003,221,120,169,011
640 DATA 141,002,220,160,000,162,000,169,011,141
```

Fig. 9-31. Continued.

```
650 DATA 000,220,185,000,192,141,001,221,169,009
660 DATA 141,000,220,169,001,141,000,220,189,064
670 DATA 192,141,001,221,169,000,141,000,220,169
680 DATA 008,141,000,220,169,009,141,000,220,232
690 DATA 224,128,208,225,162,000,200,192,064,208
700 DATA 202,076,081,193,076,021,194,162,002,160
710 DATA 008,136,208,253,202,208,248,234,160,000
720 DATA 162,000,169,011,141,000,220,189,000,192
730 DATA 141,001,221,169,009,141,000,220,185,064
740 DATA 192,141,001,221,206,000,220,173,013,221
750 DATA 201,016,208,119,076,006,194,200,192,128
760 DATA 208,227,160,000,232,224,064,240,191,234
770 DATA 076,096,193,142,048,194,134,255,140,049
780 DATA 194,132,253,006,253,234,169,248,037,253
790 DATA 141,051,194,165,255,041,007,013,051,194
800 DATA 168,169,000,133,252,169,248,037,255,010
810 DATA 038,252,010,038,252,010,038,252,133,251
820 DATA 165,255,074,074,074,024,101,254,105,032
830 DATA 101,252,133,252,096,165,253,041,007,170
840 DATA 232,169,000,056,106,202,208,252,017,251
850 DATA 145,251,096,165,253,041,007,170,232,169
860 DATA 000,056,106,202,208,252,073,255,049,251
870 DATA 145,251,096,032,147,193,032,209,193,174
880 DATA 048,194,172,049,194,076,131,193,032,147
890 DATA 193,032,227,193,174,048,194,172,049,194
900 DATA 076,131,193,173,047,194,141,002,220,088
910 DATA 165,197,201,017,240,003,076,038,194,096
920 DATA 120,169,011,141,002,220,076,017,193,255
930 DATA 063,127,000,255
```

Fig. 9-31. Continued.

A LITTLE HISTORY

In 1955, J.D. Shaw was issued a patent (No. 2,721,316). This patent detailed a system to provide electrical stimulation to the vision centers to inform the blind of ambient light levels.

In 1968, G. Brindley and W. Lewin developed the first neural prosthesis to stimulate the occipital lobe.

In 1977, another prosthesis, the Dobelle, was created using a matrix of 64 computer controlled electrodes. The computer produced patterns recognizable by the blind patient. Information obtained from this experiment showed that there isn't a one-to-one correspondence between electrode placement and phosphenes generated.

It would appear that the next step would be a matrix decoder of the vision center. (Something similar to the matrix decoder I needed to perform for the D-Cam chip.)

Since these pioneering experiments, more information on visual processes have been acquired. Form, color, and spatial information are processed by the brain along three distinct pathways. Different areas in the visual centers of the brain appear to predominantly process the aforementioned visual information. Since the information I have regarding the experiments of 1977 are sketchy, it may be that the 64 electrodes were not implanted in the most favorable area of the vision center. This leads us to believe that considerable improvement could be obtained with this single advancement.

D-CAM

The type of digital camera we built could provide the front end processing unit for continued experiments. But real hope lies if it may also be used as the back end. If we could use the memory cells on the silicon wafer as the electrodes, our resolution would increase dramatically. The assumption is based on the following.

If you remember, our memory cells are loaded with a binary "1"'s, which are equal to +5V. This 5 volts, placed against the brain, may provide sufficient electrical stimulation to produce phosphenes. If this is true, then at one end we could be reading the information off of one chip, and painting the image into the implanted chip with binary "1"'s. Our chip isn't the state of the art, of course. Superior chips are to be had, but they cost many dollars. With these chips, we could get an increase in resolution equal to or exceeding broadcast television. Our chip is using 8,152 pixel elements in bank one, but compared to 64 pixels in the last mainstream experiment of 1977 would be a major improvement.

The currents used in the experiments however, did exceed the capacity of the pint-sized memory cells of the chip. But the current required may have been the result of the size electrodes used and/or electrode placement. Since the memory cells are much smaller, it is possible that less current may be used to stimulate the brain.

This ends our mental exercise. Type in the program. The menu is smaller and self explanatory. The timing is controlled by a nested loop. The first timing number is nested into the second. Have fun, good luck.

CONCLUSION

Many projects say "you will enjoy this for years to come" or "cutting, leading, or whatever edge of technology". I'm saying that this is a tool. With it you can explore leading edge topics like neural networks, and character and pattern recognition. But it is still only a tool, it is up to you to use it. I left an open vector for subprograms that you may want to add for pattern recognition and so forth. Slight modifications in the program will cause the image to stay the same when you return from the camera, but that isn t necessary for recognition or networks. That would be aesthetically pleasing to the user.

Parts List

Quantity	Item/Description	Part Number	Cost
		RS = Radio Shack	
1	Case	RS# 270-283	3.69
2	Battery holders	RS# 270-405	.49
1	Switch DPDT	RS# 275-663	2.49
1 pkg	Circuit board	RS# 276-159	1.49
2 pk			
1 pkg	IC Sockets	RS# 276-1998	.89
2 pk			
2	12 volt battery	RS# 23-144	.89
1	Ribbon cable	RS# 278-772	3.59
1	Joystick connector	RS# 276-1538	2.49
3 pkg	.1 µF Disc cap.	RS# 272-135	.59
2 pk			
1	IM-16 D-Cam	IM-16	32.00

Images Co.
P.O. Box 313
Jamaica, N.Y.11418

1	Lens	E41,146	11.50

Edmund Scientific
101 E Gloucester Pike
Barrington, N.J. 08007

1	Card connector	As had last Digi-Key	

10

Dynamic Equations

A limitation of standard geometry is its inability to describe many of nature's forms. Landscapes mountains, clouds, coastlines, all exceed the functions of standard geometry. Nature does not limit itself to the standard geometric forms such as cones, circles, straight lines and triangles.

Fractal equations graphed on computer screens can mimic natures forms. Many popular motion pictures use special effects containing fractal landscapes and planets. How do fractal equations generate images that mimic nature's forms? What are the unique aspects of these equations?

This is what we will start to explore. We will not provide the definitive answer in this one article, but will begin to build a foundation. A foundation that is understandable, comprehensive, and most important can be built upon. This I feel is better than throwing numerous equations and concepts at the reader, that approach would most likely obscure rather than illuminate the mathematical concepts we want to explore.

DYNAMIC EQUATIONS AND NATURE

Dynamic equations mimic nature during transitions from one state to another. Contrary to what you may have learned in physics, events do not proceed in an orderly, linear fashion. A case in point is turbulence that is created when fluids flowing in a pipe reach a critical point.

Imagine we have a perfectly smooth pipe and an even supply of water flowing through the pipe. When the flow is smooth or laminar, small

disturbances die out. But if we increase the pressure we reach a point where turbulence is created. Where is this turbulence created, we still have a smooth pipe and even supply of water, but now the flow is broken up into whorls and eddies. These disturbances dissipate energy and create additional drag on the system. The system has become chaotic.

You may not be impressed with the dynamics of water flowing through a pipe, since from an engineering standpoint the question always leads to getting rid of the turbulence. Let's digress a bit and see where in the real world these systems apply. Turbulent airflow over a wing destroys lift. Turbulence created by a moving submarine creates additional drag on the sub and probably makes detection easier. Dynamic equations can help us understand transition points in nature.

BEGINNING OF CHAOS

Dynamic equations are self modifying equations. Meaning that the answer obtained from the first pass through the equation is fed back into the beginning of the equation and repeated. We start with a seed value for X, calculate through the equation, then use the resulting X for the next iteration. It is most convenient to graph each iteration on our computer screen. This way we can see the results and effects of each iteration more clearly.

This type of equation I believe was first defined by P.F. Verhulst in 1845 for growth limitations. He asserted that a given niche can maintain a certain optimum maximal population, (let's call this number X) further that as the population approached X the growth rate factor (let's call this number R) would decrease. This produces a dynamic nonlinear equation with a variable growth rate. At high growth rate factors however the equation produces catastrophic consequences (chaos). Let's take a look.

POPULATION GROWTH MODEL

The Verhulst equation can be broken down into two main functional parts. The first part (1+r)x is the growth factor. We can see that in each iteration x is increased by itself (x times 1) and the growth rate factor (x times r).

To limit this growth at 1 (the optimum population size) the second half of the equation (-rx↑2) varies with the value of x to bring the value of x to 1. This second half of the equation works well for low values of R (small growth rate factors). But as we shall see as R is increased, the equation begins to oscillate, first between 2 points, then 4 points then 8,16, and quickly into chaos. When the equation enters chaos, what is meant is that we have come to the end of predictability. It is no longer possible to predict the results of the equation except by letting it run. Before we continue our discussion on chaos let's first graph our population growth model equation.

GRAPHING PROGRAMS

The photo illustrations accompanying this article are screen images from a C-128. I have included a bit map plotting routine for the C-64. This routine plots the identical image of the Chaos 2 program as the C-128, for the Chaos 1 program, however, it plots dots without the connecting lines. Therefore the image will not appear the same. Although the images aren't as clear or dramatic as the C-128 you can still see how the program plots the equation. And as this is the method for plotting Chaos 2 they both appear the same.

For the C-64 users, enter and run the C-64 plotting routine before entering either of the chaos programs. Do remember to save the plotting program before you run it as it erases itself from BASIC.

Enter Chaos program Fig.10-1 for the C-128. Figures 10-2 and 10-3 for the C-64. Figure 10-2 is a plotting subroutine for the C-64 that must be run before running the program of Fig. 10-3. When you run this program it will prompt you for the growth rate factor. Enter 1.9 for our first test. Observe the results on your screen. After the program is finished plotting, your screen should look like Fig. 10-4. Notice as the program is plotting, it approaches the value 1, overshoots slightly, compensates, undershoots, compensates etc.. These oscillations dampen until it reaches an approximate value of 1. Consider the value 1 as the attractor at this point (growth factor) in the equation. The beginning oscillations are transient values the equation will go through before it finally settles on its attractor.

```
5 REM C-128 CHAOS #1
10 GRAPHIC1,1:COLOR0,12:COLOR1, 2:COLOR4, 1
14 CHAR1,1 ,24,"   ITERATIVE EQUATION F(X)=X(1+R)-RX↑2    ",0
15 INPUT "INPUT GROWTH RATE FACTOR";R
20 X=.10:XX=.1
30 Y=5
40 X=(1+R)*X-(R*X↑2)
50 PRINT"THE VALUE OF X AT THIS ITERATION IS";X
80 BOX1,Y-2.5,X*100-2.5,Y+2.5,X*100+2.5
90 DRAW1,Y,X*100TOY-5,XX*100
100 XX=X
110 Y=Y+5 :IFY>320THEN120
114 GOTO40
120 PRINT:PRINT"DO YOU WANT TO DO ANOTHER?"
125 INPUT"Y OR N";A$
130 IFA$="Y"THEN10
135 IFA$="N"THENEND
140 GOTO120
150 END
```

Fig. 10-1. Chaos #1 C-128.

```
10 REM COM -64 PLOTTING ROUT
20 FOR L=49152TO49313
30 READY:POKEL,Y:NEXT
40 DATA 169,003,013,002,221,141,002,221,169,252
50 DATA 045,000,221,009,001,141,000,221,169,024
60 DATA 141,024,208,169,032,013,017,208,141,017
70 DATA 208,096,234,169,160,133,252,162,191,169
80 DATA 000,133,251,168,145,251,200,208,251,230
90 DATA 252,228,252,176,245,096,234,162,250,165
100 DATA 002,202,157,000,132,157,250,132,157,244
110 DATA 133,157,238,134,208,241,096,234,169,248
120 DATA 037,253,133,002,165,255,041,007,005,002
130 DATA 168,169,000,133,252,169,248,037,255,010
140 DATA 038,252,010,038,252,010,038,252,133,251
150 DATA 165,255,074,074,074,024,101,254,105,160
160 DATA 101,252,133,252,096,234,032,078,192,120
170 DATA 165,001,041,254,133,001,165,253,041,007
180 DATA 170,232,169,000,056,106,202,208,252,017
190 DATA 251,145,251,165,001,009,001,133,001,088
200 DATA 096,234
```

Fig. 10-2. Plotting routine C-64.

```
2 REM C-64 CHAOS 1
5 PRINT"{CLR}":PRINT:PRINT:PRINT:INPUT "INPUT GROWTH RATE
  FACTOR";R
6 G=PEEK(53272):B=PEEK(56576):K=PEEK(56578)
10 SYS49152:SYS49185:POKE2,1:SYS49209
20 X=.10:XX=.1:Y=1
40 X=(1+R)*X-(R*X↑2)
80 POKE253,YAND255:POKE254,Y/256
90 POKE255,X*100
95 SYS49278
100 XX=X
110 Y=Y+3 :IFY>320THEN116
114 GOTO40
116 POKE53265,PEEK(53265)AND223
117 POKE53272,G:POKE56576,B:POKE56578,K
118 PRINT:PRINT
120 PRINT" DO YOU WANT TO DO ANOTHER?"
125 INPUT"Y OR N";A$
130 IFA$="Y"THEN5
135 IFA$="N"THENEND
140 GOTO120
150 END
```

Fig. 10-3. Chaos #1 C-64.

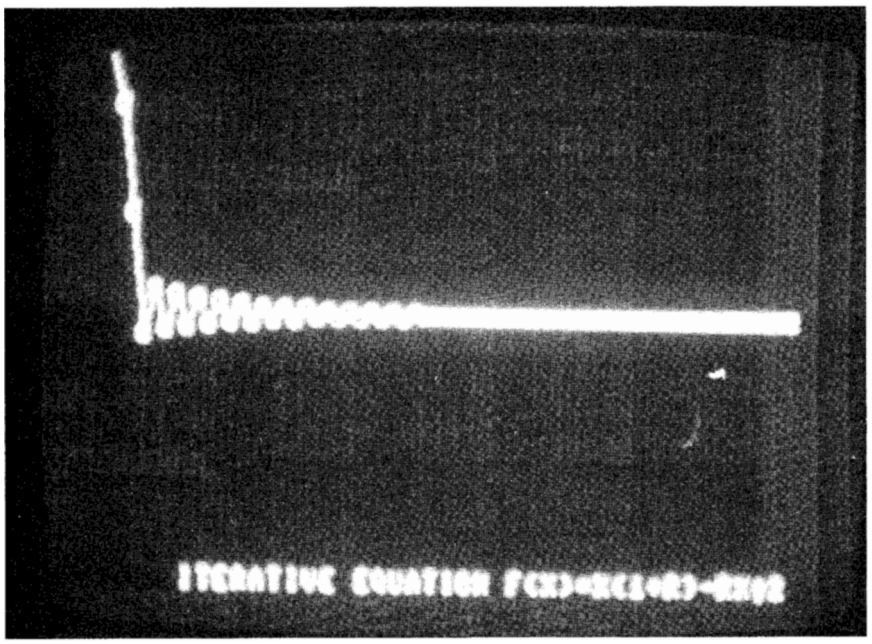


Fig. 10-4. Graph #1.

Run the program again and enter 2.4 at the prompt (Fig.10-5). Notice how the oscillations do not dampen, they continue indefinitely. It is no longer possible to reach the first attractor (optimum size of 1 for the population). The program at this point has two attractors. The values of X are printed on your screen and displayed graphically.

Run the program again and enter 2.5 at the prompt (Fig.10-6). The program oscillates at four points. Each point is an attractor. Run and enter 2.98 (Fig.10-7). The program has entered chaos. The value of X jumps all over now. It is no longer possible to predict the value of X at any iteration except by letting the equation run to the point in question.

ORDER OUT OF CHAOS

Enter and run program #2, Fig.10-8 for the C-128, and the programs of Figs.10-2 and 10-9. Let's analyze the equations in program 2 and look at the results of the program (Fig.10-10). First notice that we are using the same equation as in program 1 and that we have this equation repeated twice within the program. The first section of the program goes through 275 iterations to dampen the random oscillations (transients) as noted in program 1, test 1. After the transients have settled we then go to the second half of the program, using the same equation and plot the next 200 points. These points are the attractors. Then the program increments R by .0035 and repeats the process.

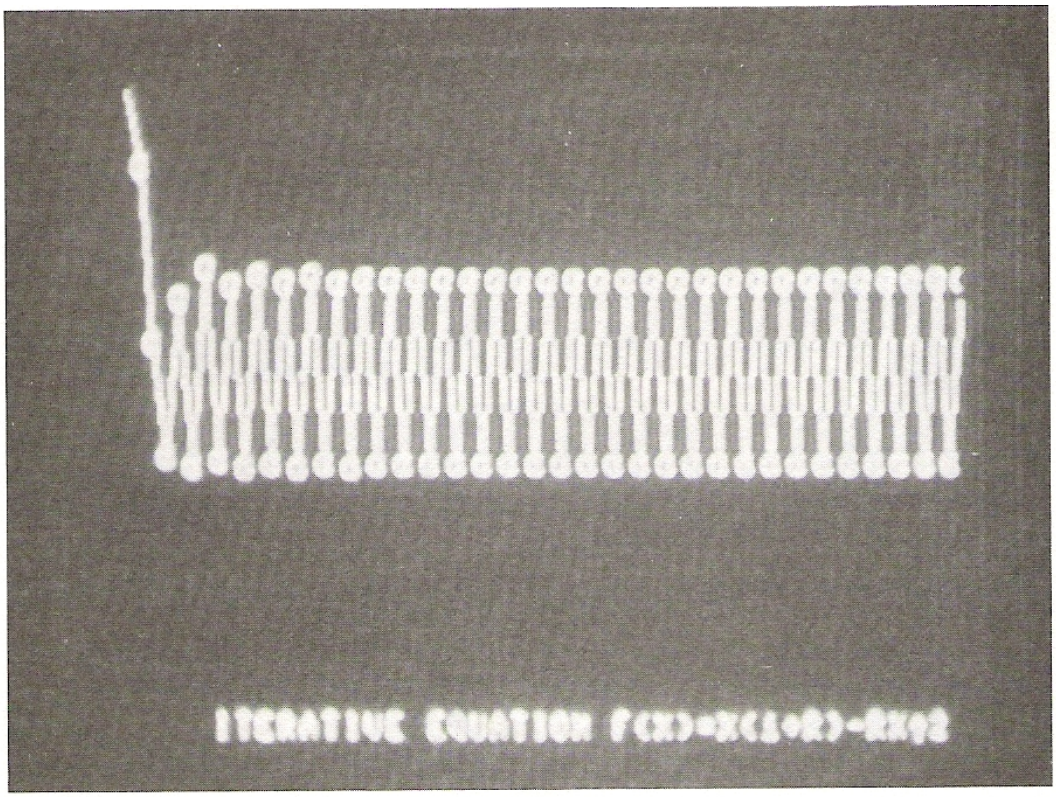


Fig. 10-5. Graph #2.

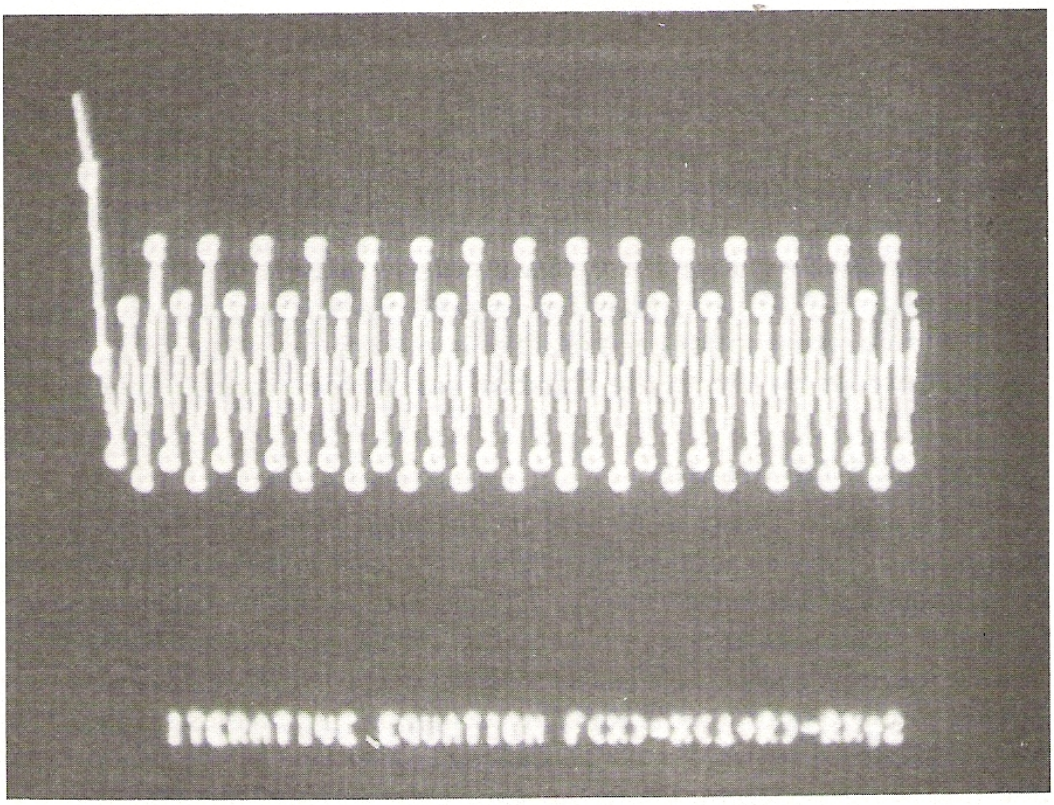


Fig. 10-6. Graph #3.

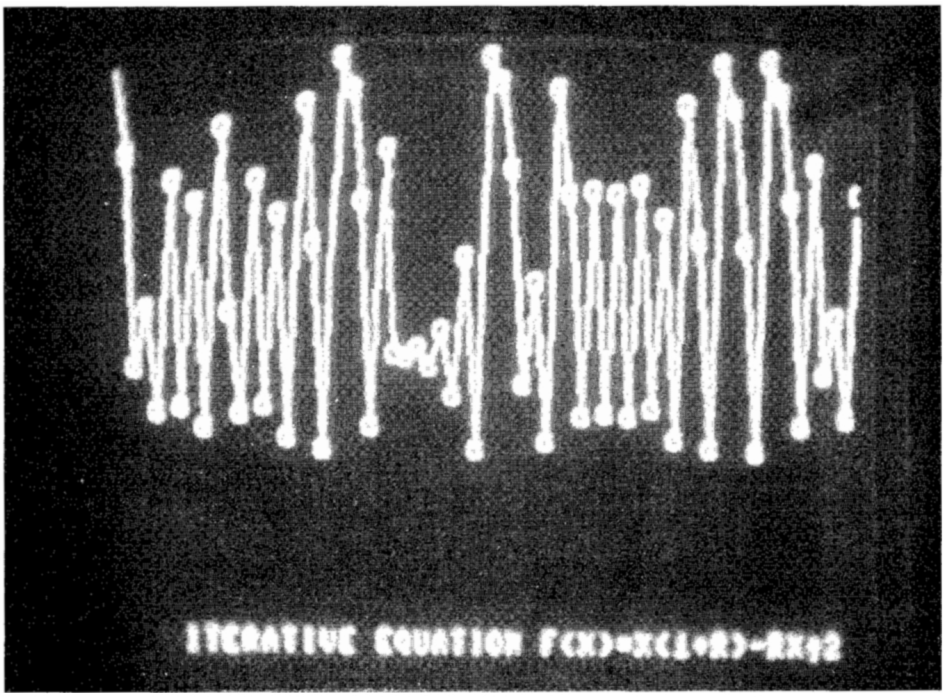


Fig. 10-7. Graph #4.

```
10 GRAPHIC1,1:COLOR0,12:COLOR1, 2:COLOR4, 1
12 Y=1
14 CHAR1,1 ,24,"   ITERATIVE EQUATION F(X)=X(1+R)-RX↑2   ",0
15 R=1.99
20 X=.25
30 FORI=1TO275
40 X=(1+R)*X-(R*X↑2)
50 NEXT
60 FORI=1TO200
70 X=(1+R)*X-(R*X↑2)
80 DRAW1,Y,110*X
90 NEXT
100 R=R+.0035
110 Y=Y+1
114 IFY>289THENEND
120 GOTO30
```

Fig. 10-8. Chaos #2 C-128.

We can see that when the program begins, it is plotting a single attractor. Analogous to test 1. The program continues incrementing R and plotting the attractors. This plots as a single line across the screen. When R reaches its first critical point it branches into 2 points or attractors, analogous to test 2. This is plotted as two separating lines. As R is

```
10 SYS49152:SYS49185
12 POKE2,1:Y=1:SYS49209
15 R=1.99
20 X=.25
30 FORI=1TO275
40 X=(1+R)*X-(R*X↑2)
50 NEXT
60 FORI=1TO200
70 X=(1+R)*X-(R*X↑2)
80 POKE253,YAND255:POKE254,Y/256
90 POKE255,X*110:SYS49278
100 R=R+.0035
110 Y=Y+1
114 IFY>289THENEND
120 GOTO30
```

Fig. 10-9. Chaos #2 C-64.

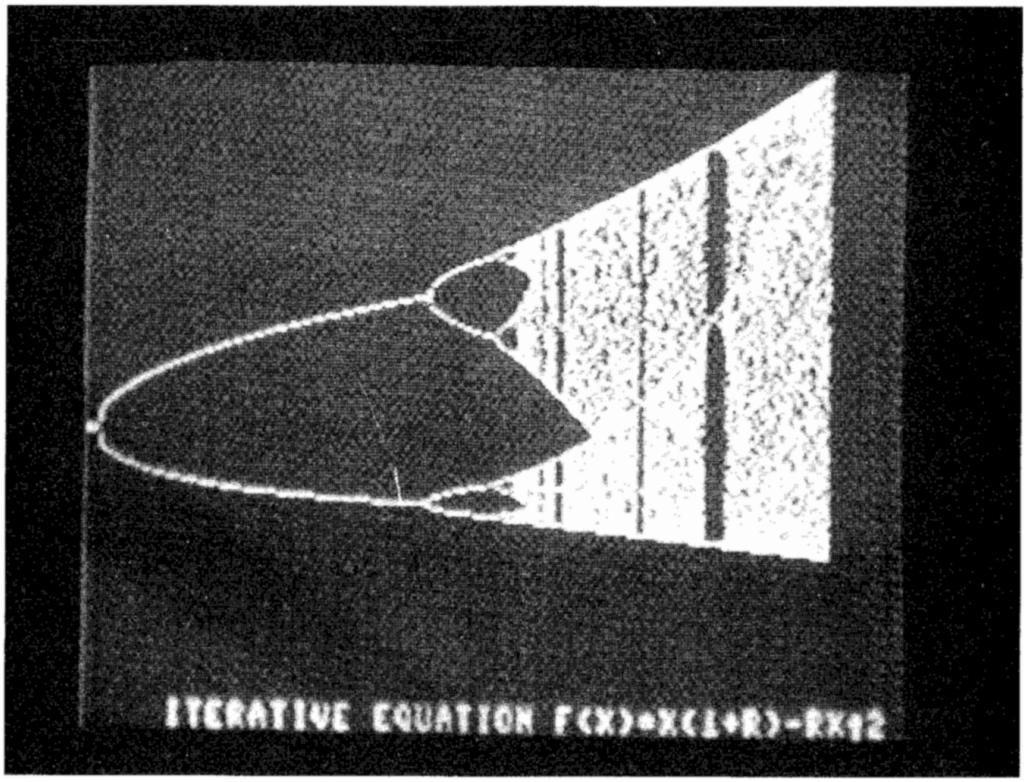


Fig.10-10. Graph chaos program #2.

increased the branching spreads further apart until we reach the next critical point when the equation begins oscillating between 4 points and thereby branches into 4 lines, analogous to test 3. Finally we enter chaos.

SELF-SIMILARITY

If we take an overview of the entire bifurcation (branching) interestingly, there appears to be a pattern to the chaotic dynamics of

our nonlinear equation. In addition, the overall pattern is repeated in the matrix of chaos. This paradoxical organization is our starting point of fractals.

Fractals, the word is synonymous with Benoit Mandelbrot, who developed the concept of fractals. I will return to the discussion of fractals later, for now I wish to continue with the dynamic equation.

We can magnify portions of our equation to observe the self-similarity. By setting the value of R between two points we wish to examine, then adjusting the step value of R to give a full-screen image. This is accomplished by dividing the difference of R (endpoints) by our resolution in the X line (320). This number is the step value used to increment R. Doing this, the entire field of view or screen image will begin at the first endpoint and end at the second. See example below. With our magnified portion we can see how the overall pattern repeats. See Fig. 10-11.

End points: *R*= 2.8125 and R= 2.8829
Subtraction: 2.8829 - 2.8125 = .0704
Step: .0704/320 = .00022

For the example given the above we would start R at 2.8125 in line 15 and change the step value to .00022 in line 100 of Chaos program 2.

```
15   R=2.8125
100  R = R + .00022
```

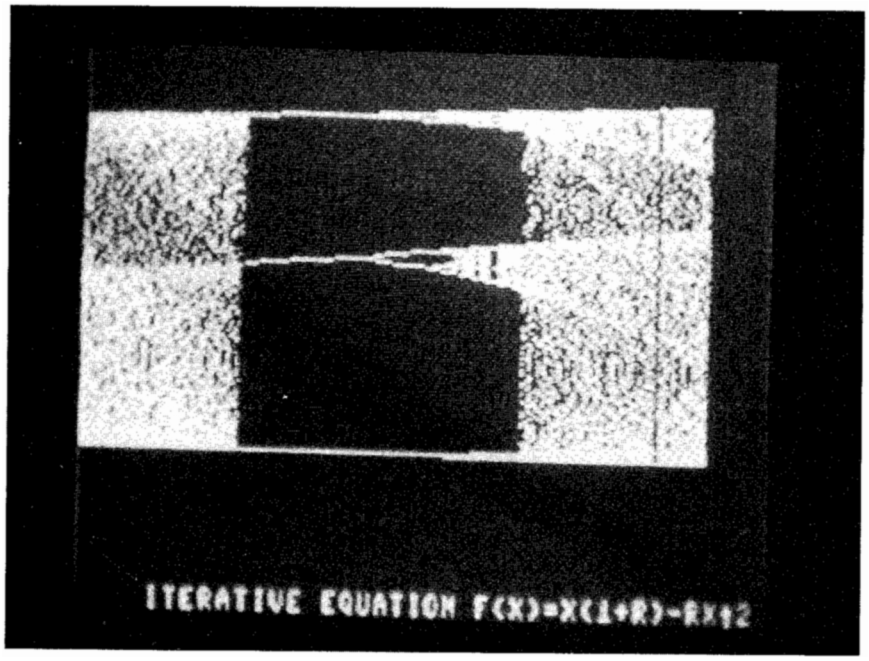


Fig.10-11. Magnified section graph program #2.

THE BUTTERFLY EFFECT

Edward Lorenz, a meteorologist, in the early 1960's created a interesting simulation program of Earth's atmosphere. This program used 12 basic equations, containing the variables for heat, pressure, wind direction, and other meteorlogic factors tied together in a nonlinear dynamic fashion.

The "toy weather" created by the computer varied and remained as unpredictable as the real world weather. (In a restrained chaotic fashion.) The weather printed from the computer at a rate of one day per minute.

In the winter of 1961 Lorenz wanted to examine a section of the weather printout. Rather than start the computations from the beginning, he fed the computer the midpoint calculations for that particular section. At first the machine tracked almost identical to the first printout, but within the space of a few months the weather was completely different from the original run. Lorenz checked his computer for a possible malfunction, the computer checked out fine, looking back at the numbers he fed into the machine he noticed that he used a rounded off number .506 instead of .506127. Lorenz originally felt that the small difference, one part in a thousand, would be equivalent to small puffs of wind in the atmosphere that would cancel out. Obviously this was not the case.

Small differences in the input could quickly become overwhelming differences in output. This is as we can note with the dynamic equation we have experimented with, small increments in R can provide catastrophic results. The definition of the butterfly effect is that "A butterfly stirring the air today in Peking can transform storm systems next month in N.Y.". Even today weather forecasts are speculative beyond two to three days due in fact to the "butterfly effect."

NATURE

Self-similarity in nature appears to be the rule, examples are abundant everywhere. Let's look at a few. Leaves on a tree are self similar. They all have the same shape and structure and are replicated thousands of times on each tree. Branches or branching, although not identical, follow the same basic rules. If you examine the branching of a bare tree, then compare it to the branched structure of the human bronchial system of the lung, you'll find the similarity amazing. DNA contained in the nucleus of cells is the same compound in all living creatures on Earth.

Let's not forget the atomic structure of matter, the versatility of matter and energy in our universe is composed of what appears to be a limited dynamic set of parts.

USEFULNESS

You may think that the dynamic equation for population growth is useless outside of being a mathematical exercise. Afterall do we ever find

a growth rate of 200% or 300%. The human population, no of course not, but insect populations, yes. The equation can accurately predict these populations.

Also, bear in mind that although the equation is striving to reach the optimum value of 1, the number 1 can represent any value such as one million or one billion. Remember the variable growth rate (R) is positive when X is less than 1 (X < 1) and negative when X is greater than one (X > 1). The positive value (R) represents population increases, the negative value (R) represents population decreases. One way to look at population decreases is that when the population has exceeded its optimum size of 1, the decrease (increased death rate) could be attributed to the depletion of food supplies, or from diseases in an overcrowded environment.

I have noted a practical example of the dynamic equation usefulness with the insight it provides in understanding weather and the butterfly effect.

Dynamic equations have much broader applications elsewhere. In physics, they have been used successfully for theoretical work in lasers, kinetics of chemical reactions, and hydrodynamics. Other fields include economics, electrical response of cardiac cells, feedback control of electronic circuits, and quantum mechanics.

FRACTALS

The dynamic equation is very similar to fractal equations. They operate the same way. The difference is that the value R in our equation is a real number, in a fractal equation it would be a complex number. A complex number consists of two numbers, a real and an imaginary number. I do not want to go any further than this. Because we have just scratched the surface of dynamic equations. Other areas of research and interest are Fibonacci branching and self organizing equations.

WHY NOW?

The question begs to be asked. If Verhulst worked on his equations over one hundred years ago why is it that just recently scientists are working on dynamic equations and fractals. The answer is that before high speed digital computers the results of the equations were obscured. No one could plot millions of complex equations to discover the underlying pattern to chaos. Although the roots of the mathematics are well established it is only recently scientists have been able to correlate this information, thanks to our computers.

Experiment with both programs to gain further insight into their workings. Some suggestions are, use Chaos 1 and plot a few low growth rate factors. Use Chaos 2 and magnify different sections of the matrix and see what you can find.

11

Mandelbrot Graphics

Mandelbrot graphics are named after an IBM research fellow Benoit Mandelbrot, who developed the field of fractal geometry. Mr. Mandelbrot coined the term fractal to describe this special type of geometry. Basically a fractal is a geometric object with fractional dimensions.

Computer graphics have enhanced our understanding of mathematics, and added a dimension of beauty. Patterns in chaotic, non-linear systems that were previously hidden have been brought to light with computers.

In Chapter 10 we investigated chaotic equations and plotted their screen images. Those equations were of the iterative form as are the equations we'll work with now. The main difference is that in these Mandelbrot equations we use complex numbers instead of real numbers.

COMPLEX NUMBERS

A complex number is made up of two parts. The two parts are called real and imaginary. The number 9 + 3i is a complex number, with the 9 being the real part and the 3i the imaginary. The i next to the 3 shows which part of the number is imaginary. Complex numbers can be represented graphically by the point P whose rectangular coordinates are (x,y). See Fig.11-1. In this form we see the X axis is the axis of reals, and the Y axis the axis of imaginaries. See the ending paragraphs,

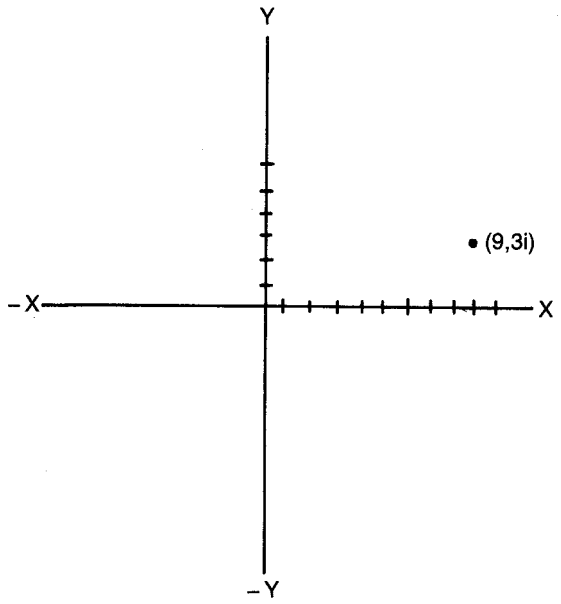


Fig.11-1. Complex number plot (9,31).

Properties of Complex Numbers, Algebraic Operations and Computer Choke for more on complex numbers.

PLOTTING

As stated, the equation is of the iterative variety, and is functionally equivalent to the chaotic equations we worked on last chapter.

```
Z z↑2 + c
```

Starting with a seed value for Z, we square z and add c then feed this value of z back into the equation, square it add c and so forth and so on (Zn+1). Remember z and c are both complex numbers.

In this iterative process some complex numbers become very large, exceeding the capacity of any computer, these numbers we treat as if they reach infinity. We shall see later how the computer programs handle this. Many complex numbers remain small after many iterations. Depending upon the number of iterations it took for the number to reach infinity determines what we color that particular coordinate. (See Fig.11-2.) We have added another axis, the z axis. The z axis is equivalent to the c counting of iterations. It shows us how fast a particular point escapes to infinity. As an example, if a point (which is a complex number coordinate) takes 4 or less iterations to reach infinity it is colored color #1. If the number of iterations it takes is greater than 4 but less than 6 the point is colored color #2.

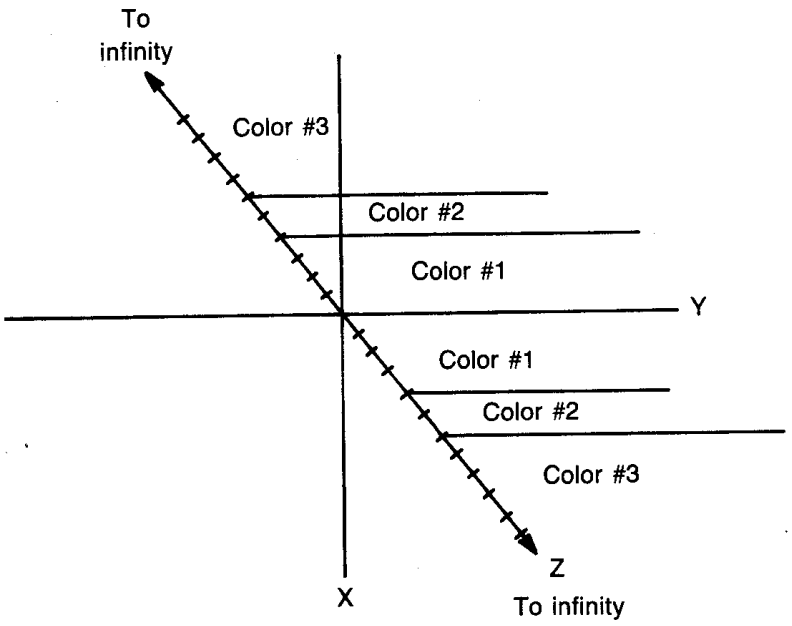


Fig. 11-2. Z-axis and color relationship.

Colors are assigned to complex numbers that reach infinity depending on the number of iterations (C=count) it took. For practical application see the Advanced Operations section.

FIRST MANDELBROT PICTURE

Our first Mandelbrot picture uses the standard coordinates -2.25 to .75 for the x dimension and -1.5 to 1.5 for the y dimension. The graphic screen resolution for the C-64 and C-128 is a maximum of 320 x 200. We will use the multi-color mode (160 x 200) however to get some color into our pictures. Our coordinates are divided by the resolution (-1) of our screen. Thus:

```
DX=(.75-(-2.25)) OR  DX=3.00/159(y res) = .0188679245
DY=(1.5-(-1.5))  OR  DY=3.00/199(y res) = .0150753769
```

These numbers DX and DY become our step values. Our first run through the equation the coordinates start at -2.25x and -1.5yi. The iterative process is repeated until either we reach infinity or we reach the maximum value that we assign to C to repeat the iterative equation.

Let's assume that for this first number we reached the threshold of infinity in 3 iterations. We take the number 3 and assign a color value to it, plot or draw the pixel to the screen. Using the standard coordinates for the pixel location - 2.25, -1.5i is the uppermost left of the screen.

Now we increment the Y value by the step value, so the next coordinate pair is -2.25x and -1.485yi. We repeat the process as de-

scribed, assign a color and draw the pixel. This pixel coordinate is one pixel down from the last. When we have stepped through the entire range of Y, we increment X by its step value and repeat for all the values of Y again.

The program I've written uses a variable CT as a maximum number to repeat the iterative process. CT is equal to 48 in the program (line 108), which means if the complex number doesn't reach infinity in 48 iterations it is assumed to be lying within the Mandelbrot set, and the next number is checked.

As you can see drawing the screen is a time consuming, and labor intensive process. It should come as no surprise that it takes approximately 8 hours to draw a picture in C-128 fast mode.

PROGRAM FEATURES

The Mandelbrot programs contain a number of useful features. See Fig.11-3 for the C-64 and Fig.11-4 for the C-128. Type in and save the respective program for your computer.

```
10 REM MANDELBROT PROGRAM C-128
12 IFX=0THENX=1:GRAPHIC3,1
14 REM DEFINE SPRITE
16 FORN=3584TO3646:READA:POKEN,A:NEXT
18 DATA255,255,255,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,
   128,0,1
20 DATA128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,128,
   0,1,128,0,1
22 DATA128,0,1,128,0,1,128,0,1,128,0,1,255,255,255,255,255,
   255
24 GRAPHIC5
26 REM MANDELBROT PROGRAM C-128
28 REM JOHN IOVINE
30 PRINT"{CLR}":PRINT"{C/DN}{C/DN}MANDELBROT PROGRAM C-128
   MAIN MENU":PRINT
32 PRINT" 1) VIEW DIRECTORY"
34 PRINT" 2) LOAD MANDELBROT PICTURE
36 PRINT" 3) VIEW MANDELBROT PICTURE
38 PRINT" 4) INPUT COORDINATES
40 PRINT" 5) CHANGE COLORS"
42 PRINT" 6) VIEW & CREATE MANDELBROT PICTURE
44 PRINT" 7) SAVE MANDELBROT PICTURE
46 PRINT" 8) QUIT
48 INPUT"MENU CHOICE 1 THUR 8";A
50 IF A <0THEN56
52 IFA>8THEN56
```

Fig-11-3. Mandelbrot program C-128.

```
54 ON A GOTO 248,228,260, 58,148,162,206,160
56 PRINT"ERROR, PLEASE ENTER NUMBER BETWEEN 1-8":GOTO48
58 PRINT"{CLR} {C/DN}{C/DN}ENTER NEW COORDINATES.."
60 INPUT"INPUT XL";XL:INPUT"INPUT XR";XR:INPUT"INPUT YT";YT:
   INPUT"INPUT YB";YB
62 PRINT"{CLR} ARE THESE THE COORDINATES YOU WANT"
64 PRINT"XL = ";XL:PRINT"XR = ";XR:PRINT"YT = ";YT:PRINT"YB
   = ";YB
66 INPUT"( Y/N OR M FOR MENU)";A$
68 IFA$="Y"THEN 76
70 IFA$="N" THEN 58
72 IF A$="M" THEN24
74 GOTO66
76 PRINT"{CLR}"
78 PRINT"{C/DN}{C/DN}ENTER 1 TO PROCEED TO DRAW"
80 PRINT"{C/DN}{C/DN}ENTER 2 TO RETURN TO MENU"
82 PRINT:PRINT
84 INPUT"INPUT 1 OR 2";A
86 IFA=1THEN94
88 IFA=2 THEN 24
90 PRINT"ANSWER 1 OR 2 ONLY":GOTO84
92 REM COORDINATES
94 PRINT"{CLR}{C/DN}{C/DN}{C/DN}DRAWING":DX=(XR-XL):DY=(YB-
   YT)
96 GRAPHIC1,1:FAST
98 POKE53280,0     :REM BORDER BLACK
100 POKE53281,3     :REM BACKGROUND GREY
102 SM=3:CT=48
104 FORXO=XLTOXRSTEP(DX)/159
106 FORY=YTTOYBSTEP(DY)/199
108 A=XO*XO-Y*Y+XO:B=2*XO*Y+Y:C=0
110 R=A*A-B*B+XO:I=2*A*B+Y:C=C+1
112 A=R:B=I
114 IFR<-10↑16THEN130
116 IFR<SMTHENIFC<CTTHEN110
118 X1=INT((XO-XL)/((DX)/159))
120 Y1=INT((Y-YT)/((DY)/199))
122 IFC<4 THENGOSUB140:GOTO130
124 IFC<6 THENGOSUB142:GOTO130
126 IFC<48THENGOSUB144
128 IFX1=>158THENSLOW:GOTO146
130 GETK$:IFK$<>""THENSLOW:GOSUB274:FAST
134 NEXT Y,XO
136 X=2*X1+2
138 DRAW1,X,Y1:RETURN
```

Fig.11-3. Continued.

```
140 X=2*X1+1 :GOSUB138:RETURN
142 GOSUB136:RETURN
144 GOSUB136:X=X-1:GOSUB138:RETURN
146 GOTO24
148 REM CHANGE COLORS
150 FORX=1TO255
152 FORI=7168TO8167:POKEI,X:NEXTI
154 GETK$:IFK$=""THEN154
156 IFK$="Q"THENPOKE49500,X:GOTO24
158 NEXTX
160 END
162 GRAPHIC1
164 SPRITE1,1,3
166 MOVSPR1,160,100
168 X=PEEK(49500)
170 FORI=7168TO8167:POKEI,X:NEXT
172 POKEVIC+21,1
174 GET A$
176 IFA$=""THEN174
178 IFA$="L"THENMOVSPR1,-1,+0:GOTO174
180 IFA$="R"THENMOVSPR1,+1,+0:GOTO174
182 IFA$="U"THENMOVSPR1,+0,-1:GOTO174
184 IFA$="D"THENMOVSPR1,+0,+1:GOTO174
186 IFA$="E"THEN190
188 GOTO174
190 HP=PEEK(53248):VP=PEEK(53249)
192 HP=HP-24:HP=HP/2:VP=VP-50
194 DX=(XR-XL):DY=(YB-YT)
196 XR=(XL)+((HP+24)*(DX/159)):XL=(XL)+((HP)*(DX/159))
198 YB=(YT)+((VP+21)*(DY/199)):YT=(YT)+((VP)*(DY/199))
200 SPRITE1,0
202 GRAPHIC5
204 GOTO62
206 REM SAVE COORDINATES & BINARY PIC
208 PRINT"{CLR}"
210 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}":INPUT"FILENAME";F$
212 OPEN1,8,2,"@0:"+F$+".NUM,S,W"
214 PRINT#1, XL:PRINT#1, XR
216 PRINT#1, YT:PRINT#1, YB
218 CLOSE1
220 F$=F$+".PIC"
222 Q$=CHR$(34)
224 PRINT"{CLR}BSAVE"Q$F$Q$",P7168TOP16383":POKE842.19:
    POKE843.13:POKE208.2:END
226 GOTO26
```

Fig.11-3. Continued.

```
228 REM LOAD COORDINATES & BINARY PIC
230 PRINT"{CLR}":PRINT"{C/DN}{C/DN}{C/DN}{C/RT}{C/RT}{C/RT}"
232 INPUT"FILENAME";F$
234 OPEN3,8,3,F$+".NUM,S,R"
236 INPUT#3,XL,XR,YT,YB
238 CLOSE3
240 LOADF$+".PIC",8,1
242 FORI=1024TO2023:POKEI,18:NEXT
244 GOTO26
246 END
248 REM DIRECTORY
250 PRINT"{CLR}"
252 DIRECTORY
254 PRINT:PRINT:PRINT"PRESS ANY KEY TO CONTINUE"
256 GET K$:IFK$=""THEN256
258 GOTO26
260 GRAPHIC3
262 X=PEEK(49500)
264 FORI=7168TO8167:POKEI,X:NEXT
266 GET K$
268 IFK$=""THEN266
270 GRAPHIC5
272 GOTO24
274 FORT=1TO350:NEXT:POKE842,0:POKE208,0
276 GETK$:IFK$=""THEN276
278 IFK$="C"THENRETURN
280 IFK$="E"THENRESTORE:GOTO24
```

Fig. 11-3. Continued.

```
10 REM MANDELBROT PROGRAM C-64
12 REM DEFINE SPRITE
14 FORN=896TO958:READA:POKEN,A:NEXT
16 DATA255,255,255,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,
   128,0,1
18 DATA128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1,128,0,
   1,128,0,1
20 DATA128,0,1,128,0,1,128,0,1,128,0,1,255,255,255,255,255,
   255
22 HP=160:VP=100:POKE2040,14
24 POKE49500,18
26 POKE53272,21:POKE53265,27:POKE53270,200
28 REM MANDELBROT PROGRAM C-64
30 REM JOHN IOVINE
```

Fig. 11-4. Mandelbrot program C-64.

```
32 PRINT"{CLR}":PRINT"{C/DN}{C/DN}MANDELBROT PROGRAM C-64
   MAIN MENU":PRINT
34 PRINT" 1) VIEW DIRECTORY"
36 PRINT" 2) LOAD MANDELBROT PICTURE
38 PRINT" 3) VIEW MANDELBROT PICTURE
40 PRINT" 4) INPUT COORDINATES
42 PRINT" 5) CHANGE COLORS"
44 PRINT" 6) VIEW & CREATE MANDELBROT PICTURE
46 PRINT" 7) SAVE MANDELBROT PICTURE
48 PRINT" 8) QUIT
50 INPUT"MENU CHOICE 1 THUR 8";A
52 IF A <0THEN58
54 IFA>8THEN58
56 ON A GOTO 268,246,282, 60,152,170,226,168
58 PRINT"ERROR, PLEASE ENTER NUMBER BETWEEN 1-8":GOTO50
60 PRINT"{CLR} {C/DN}{C/DN}ENTER NEW COORDINATES.."
62 INPUT"INPUT XL";XL:INPUT"INPUT XR";XR:INPUT"INPUT YT";YT:
   INPUT"INPUT YB";YB
64 PRINT"{CLR} ARE THESE THE COORDINATES YOU WANT"
66 PRINT"XL = ";XL:PRINT"XR = ";XR:PRINT"YT = ";YT:PRINT"YB
   = ";YB
68 INPUT"( Y/N OR M FOR MENU)";A$
70 IFA$="Y"THEN 78
72 IFA$="N" THEN 60
74 IF A$="M" THEN26
76 GOTO68
78 PRINT"{CLR}"
80 PRINT"{C/DN}{C/DN}ENTER 1 TO PROCEED TO DRAW"
82 PRINT"{C/DN}{C/DN}ENTER 2 TO RETURN TO MENU"
84 INPUT"INPUT 1 OR 2";A
86 IFA=1THEN94
88 IFA=2 THEN 26
90 PRINT"ANSWER 1 OR 2 ONLY":GOTO84
92 REM COORDINATES
94 DX=(XR-XL):DY=(YB-YT)
96 POKE53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
98 POKE53272,PEEK(53272)OR8
100 FORI=8192TO16191:POKEI,0:NEXT
102 FORI=1024TO2023:POKEI,16+2:NEXT
104 POKE53280,0     :REM BORDER BLACK
106 POKE53281,3     :REM BACKGROUND GREY
108 BA=8192:SM=3:CT=48
110 FORXO=XLTOXRSTEP(DX)/159
112 FORY=YTTOYBSTEP(DY)/199
```

Fig. 11-4. Continued.

```
114 A=XO*XO-Y*Y+XO:B=2*XO*Y+Y:C=O
116 R=A*A-B*B+XO:I=2*A*B+Y:C=C+1
118 A=R:B=I
120 IFR<-10↑16THEN134
122 IFR<SMTHENIFC<CTTHEN116
124 X1=INT((XO-XL)/((DX)/159))
126 Y1=INT((Y-YT)/((DY)/199))
128 IFC<4 THENGOSUB144:GOTO134
130 IFC<6 THENGOSUB146:GOTO134
132 IFC<48THENGOSUB148
134 IFX1=>158THEN150
136 NEXT Y,XO
138 X=2*X1
140 RO=INT(Y1/8):CH=INT(X/8):LN=Y1AND7:BI=7-(XAND7):BY=BA+RO
    *320+CH*8+LN
142 POKEBY,PEEK(BY)OR2↑BI:RETURN
144 X=2*X1-1 :GOSUB140:RETURN
146 GOSUB138:RETURN
148 GOSUB138:X=X-1:GOSUB140:RETURN
150 GOTO26
152 REM CHANGE COLORS
154 POKE53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
156 POKE53272,PEEK(53272)OR8
158 FORX=1TO255
160 FORI=1024TO2023:POKEI,X:NEXTI
162 GETK$:IFK$=""THEN162
164 IFK$="Q"THENPOKE49500,X:GOTO26
166 NEXTX
168 END
170 POKE650,128
172 VIC=53248:POKEVIC,HP :POKEVIC+1,VP :POKEVIC+39,13
174 POKE53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
176 POKE53272,PEEK(53272)OR8
178 X=PEEK(49500)
180 FORI=1024TO2023:POKEI,X:NEXT
182 POKEVIC+21,1
184 GET A$
186 IFA$=""THEN184
188 IFA$="L"THENHP=HP-1:GOTO214
190 IFA$="R"THENHP=HP+1:GOTO214
192 IFA$="U"THENVP=VP-1:GOTO220
194 IFA$="D"THENVP=VP+1:GOTO220
196 IFA$="E"THEN200
198 GOTO184
200 HP=HP-24:HP=HP/2:VP=VP-50
```

Fig.11-4. Continued.

```
202 DX=(XR-XL):DY=(YB-YT)
204 XR=(XL)+((HP+24)*(DX/159)):XL=(XL)+((HP)*(DX/159))
206 YB=(YT)+((VP+21)*(DY/199)):YT=(YT)+((VP)*(DY/199))
208 POKEVIC+21,0:POKEVIC+39,0
210 POKE53272,21:POKE53265,27:POKE53270,200
212 GOTO64
214 SF=(HP>255)
216 POKEVIC,HP+(SF*256)
218 POKEVIC+16,SF*(-1)
220 POKEVIC+1,VP
222 GOTO184
224 POKEVIC+1,VP
226 REM SAVE COORDINATES & BINARY PIC
228 PRINT"{CLR}"
230 PRINT"{C/DN}{C/DN}{C/DN}{C/DN}":INPUT"FILENAME";F$
232 OPEN1,8,2,"@0:"+F$+".NUM,S,W"
234 PRINT#1, XL:PRINT#1, XR
236 PRINT#1, YT:PRINT#1, YB
238 CLOSE1
240 SYS57812F$+".PIC",8:POKE173,8192/256:POKE172,8192-PEEK
    (173)*256:POKE780,172
242 POKE782,16191/256:POKE781,16191-PEEK(782)*256:SYS65496
244 GOTO28
246 REM LOAD COORDINATES & BINARY PIC
248 PRINT"{CLR}":PRINT"{C/DN}{C/DN}{C/DN}{C/RT}{C/RT}{C/RT}"
250 INPUT"FILENAME";F$
252 OPEN3,8,3,F$+".NUM,S,R"
254 INPUT#3,XL,XR,YT,YB
256 CLOSE3
258 POKE53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
260 POKE53272,PEEK(53272)OR8
262 FORI=1024TO2023:POKEI,18:NEXT
264 LOADF$+".PIC",8,1
266 END
268 REM DIRECTORY
270 PRINT"{CLR}"
272 SYS57812"$",8:POKE43,1:POKE44,192:POKE768,174:POKE769,
    167:SYS47003,1
274 POKE782,192:SYS65493:SYS42291:LIST:POKE44,8:POKE768,139:
    POKE769,227
276 PRINT:PRINT:PRINT"PRESS ANY KEY TO CONTINUE"
278 GET K$:IFK$=""THEN278
280 GOTO28
282 POKE53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
284 POKE53272,PEEK(53272)OR8
```

Fig. 11-4. Continued.

```
286 X=PEEK(49500)
288 FORI=1024TO2023:POKEI,X:NEXT
290 GET K$
292 IFK$=""THEN290
294 GOTO26
```

Fig.11-4. Continued.

Item 1 of the main menu is for looking at the disk directory, with it you can see picture files you have saved. Notice all the picture files have a ".PIC" appended to the file. This will help to remind you of picture file names you've forgotten.

Item 2 first prompts you for the picture name, loads the picture and its coordinates, then returns you to the main menu. The file name that you assign to your pictures will have ".PIC" appended to them. Do not include the ".PIC", when prompted for a file name, the program will do that for you automatically.

Item 3 put the computer into a view mode, to look at any Mandelbrot picture you have drawn, pressing any key in view retums you to the main menu.

Item 4 prompts you for coordinates to create a new picture. There are many articles and books on fractals and you may wish to try some coordinates you found elsewhere. Two excellent books are referred to at the end of this chapter.

Item 5 puts the computer into a view mode, then by pressing the space bar changes the colors of the Mandelbrot set. You have over 255 combinations of colors to choose from. Pressing the "Q" key returns you to the main menu.

Item 6, this is the most powerful feature of the program. When this item is selected the computer goes into the view mode and creates a movable window on the screen. You can move the window anywhere on the screen with the following commands:

U = for up
D = for down
L = for left
R = for right

Whatever area lies under the window can be magnified by approximately 10 x by pressing the "E" key. By pressing the "E" key the computer calculates the required coordinates, lists the coordinates, and prompts you if it should draw the coordinates, change coordinates or return to main menu. If you go ahead and draw, the section that lay under the window will now be enlarged to a full screen image.

It is interesting to note before I leave this item that anyone using this function say 8 or 9 times in progression will in fact enlarge the original screen image to the size of the United States. It's also probable that after

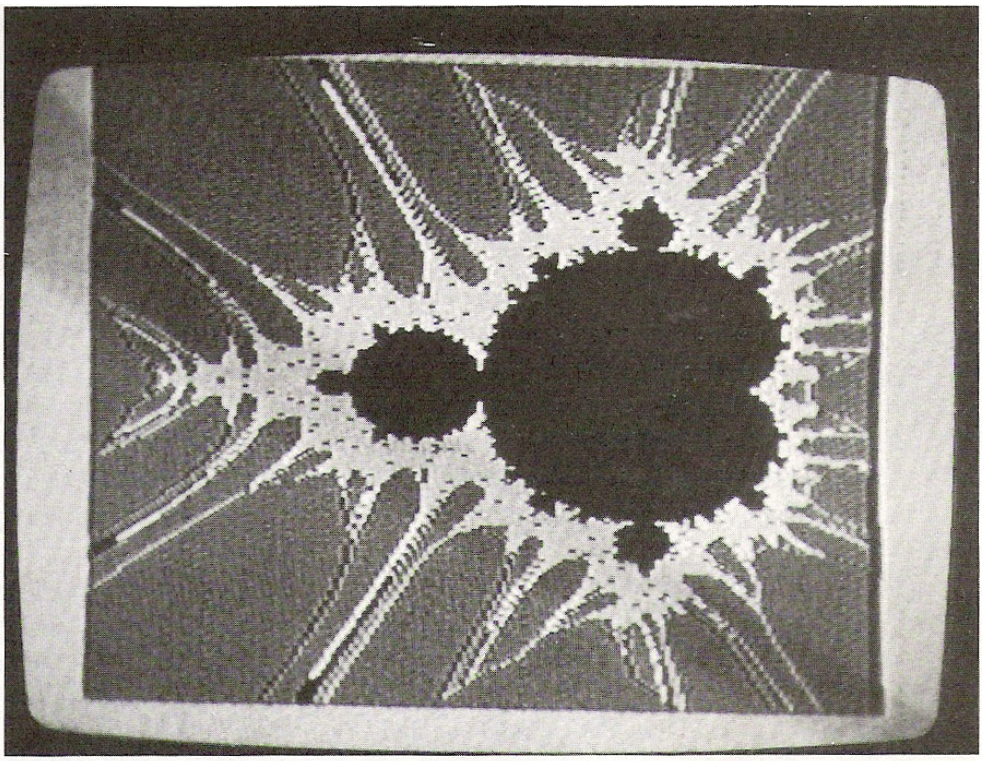


Fig. 11-5. Photo screen image Mandelbrot set.

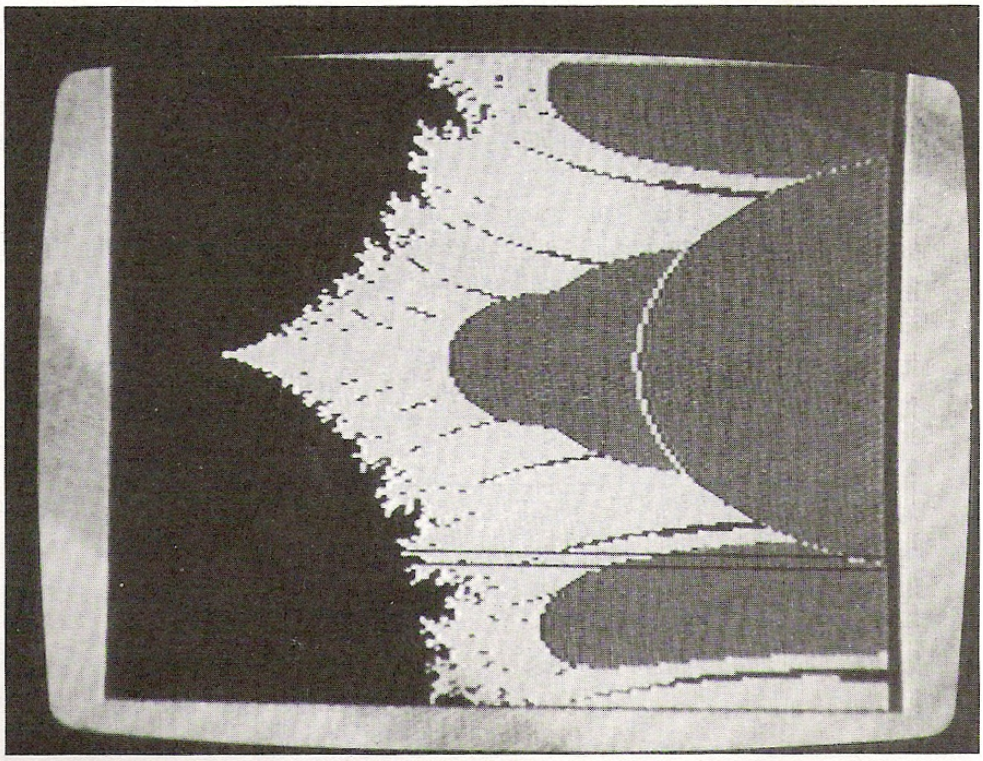


Fig. 11-6. Photo screen image magnification +10.

that many progressions you'd be exploring an area no one has ever seen before.

Item 7 saves the Mandelbrot picture and its respective coordinates to a disk file, it will prompt you for the file name.

PROGRAM OPERATIONS

After you've typed and saved the program, start the program and choose item four from the main menu. Enter the first coordinates in Table 11-1. Answer the prompts to allow program to draw this Mandelbrot picture. This can take about 12 hours so I'd advise you to start the program before you retire for the night and the program should be finished drawing by the morning. When the program finishes drawing a picture it automatically returns to the main menu. Choose item three to see what you have drawn. After the drawing is complete save it to the disk using item 7 at the menu prompt.

To create additional drawings, from this point you can either enter coordinates using item four, or you can load any .PIC file on the disk and explore it using item 6. Naturally any drawings you create with item 6 can also be saved and used as another starting point for your explorations.

ADVANCED OPERATIONS

This program can be changed to go further and create more interesting work. The two main limitations is the variable CT (line 108) and the banding or transition numbers (lines 128 through 132). I have kept both pretty low to facilitate drawing speed. More interesting pictures can be had by increasing CT and changing the banding numbers (see Fig.11-9).

To get an idea of what this means look at Fig.11-7 and Fig.11-8. These are actually the same picture. All that was changed are the transition numbers. For Fig.11-7 the default values used are 4, 6, and 48. For Fig. 11-8 the transition numbers were changed to 13, 27, and 47.

Table 11-1. Values Reflecting Fig.11-5 Through Fig. 11-9.

Figure	XL	XR	YT	YB
11-5	-2.25	.75	-1.5	1.5
11-6	.193396226	.636792453	.173366834	.203517588
11-7	.299365136	.366292868	.0578394484	.0104921592
11-8	same as Fig. 11-7			
11-9*	.299786065	.309888364	.0278607628	.0228643152

*The value of CT was incremented to 450 and the values of C changed to 45,190 and 450 see program.

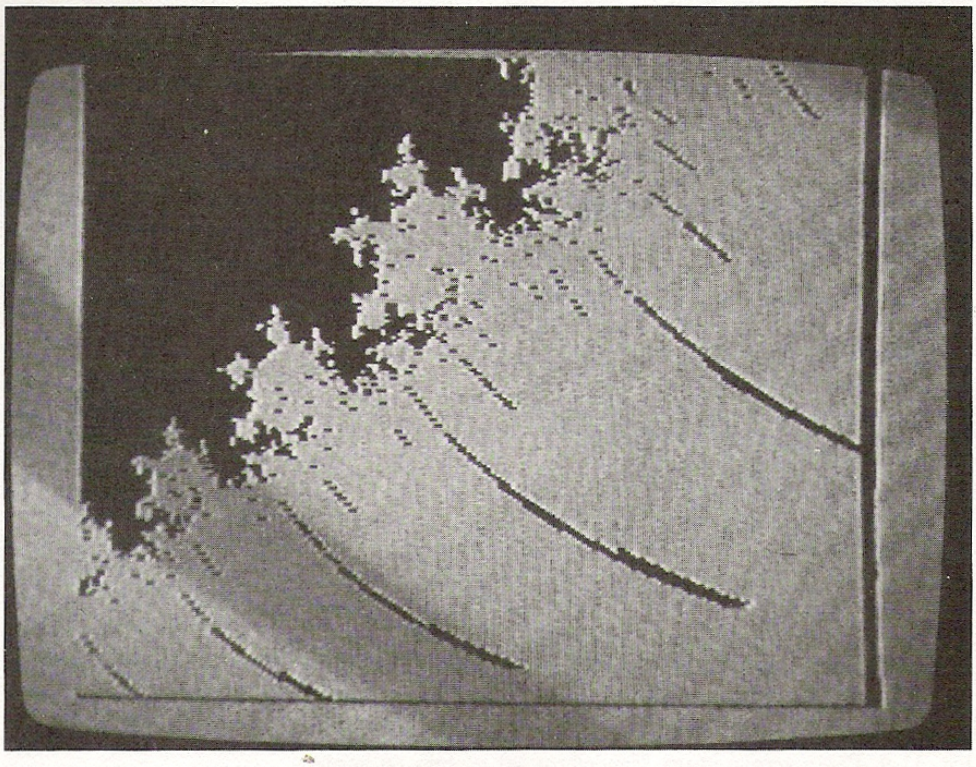


Fig. 11-7. Photo screen image magnification +100.

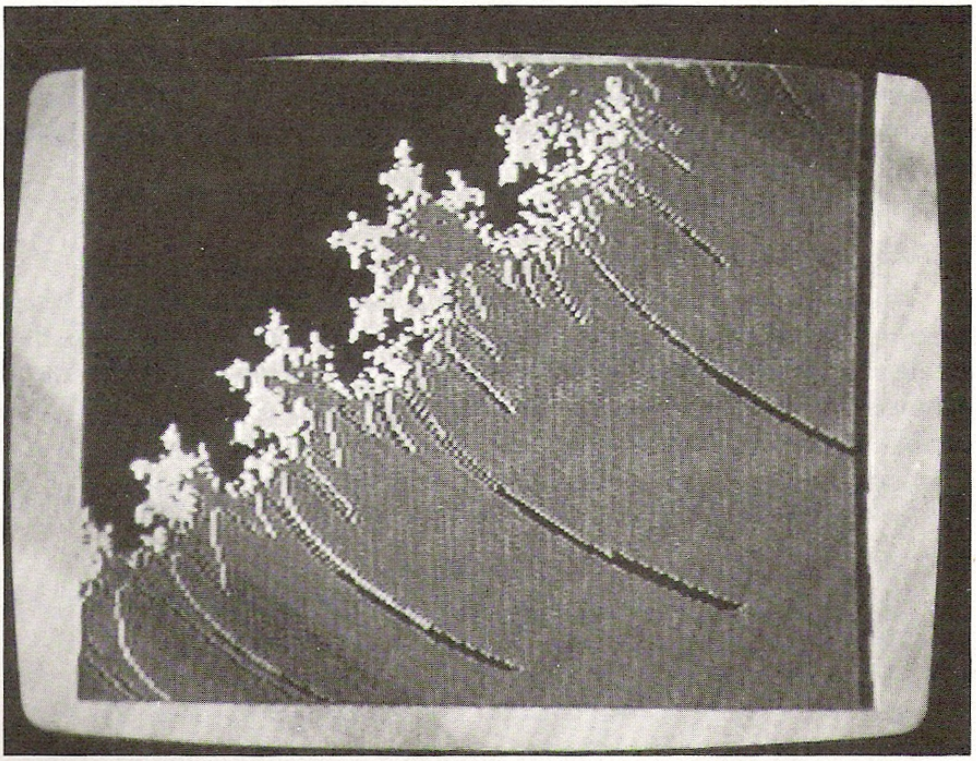


Fig. 11-8. Photo screen image magnification +100.

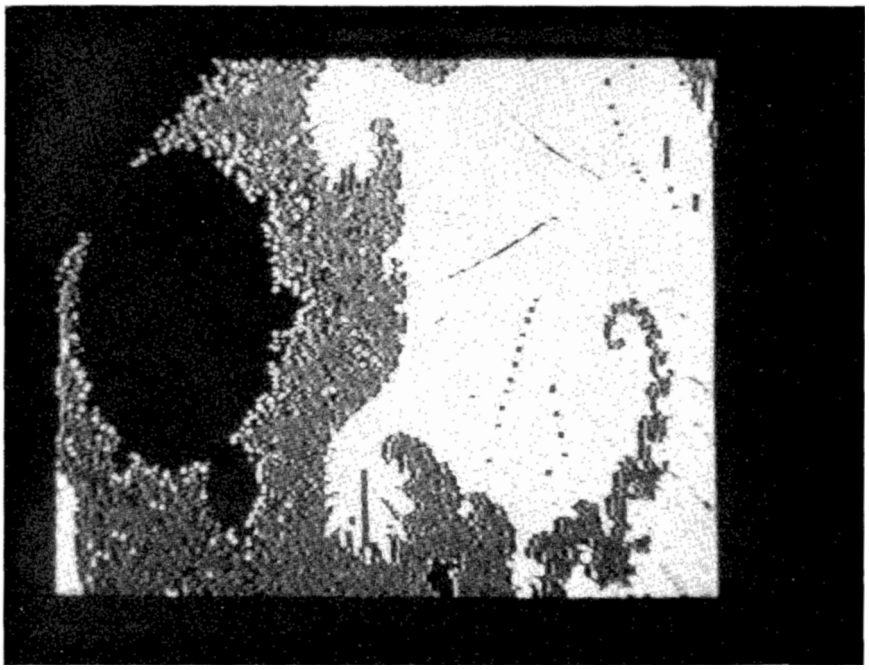


Fig.11-9. Photo screen image Mandelbrot set capture on border.

Many Mandelbrot pictures are created by using a ·GT value of 1000. Be aware this can greatly increase your processing time (see Fig.11-9). The banding numbers are another complete area of experimentation. As a rule of thumb the greater the CT value the better the final resolution of the picture. If you increase the CT value change the C values accordingly.

CLASSIC FRACTALS

What we have to play with are classic fractals. These geometric objects are already 9 years old. New and more powerful fractals, the kinds that create realistic landscapes, planets and plants are the latest toys in fractal geometry. What I'm trying to say in a roundabout manner is that there is still a lot more that lies ahead than what we've covered. Julia sets, IFS graphics, self-organizing equations, the fields are wide open.

PROPERTIES OF COMPLEX NUMBERS

There are two properties of complex numbers you need to know in order to perform algebraic operations.

1) When you square the symbol i it has the property of it 2 = -1.
2) The conjugate of a complex number x + yi is the complex number x - yi. Therefore the conjugate of the complex numbers 9 + 3i and 4 - 7i are 9 - 3i and 4 + 7i.

I will not go further into any other properties or reasons for the above properties; if you are interested you can purchase a mathematical textbook on the subject.

ALGEBRAIC OPERATIONS

In order for our program to work we must be able to perform mathematical operations. These are not difficult as the following examples will prove.

1) Addition. To add two complex numbers first add the real parts then add the imaginary parts.

Ex. 1 $(9 + 3i) + (4 - 7i) = (9 + 4) + (3 - 7)i = 13 - 4i.$
Ex. 2 $(7 + 2i) + (3 + 4i) = (7 + 3) + (2 + 4)i = 11 + 6i.$

2) Subtraction. To subtract two complex numbers first subtract the reals then subtract the imaginaries.

Ex. 3 $(9 + 3i) - (4 - 7i) = (9 - 4) + [3 -(-7)]i = 5 + 11i.$
Ex. 4 $(7 + 2i) - (3 + 4i) = (7 - 3) + (2 - 4)i = 4 - 2i.$

3) Multiplication. To multiply two complex numbers, multiply as if they are ordinary binomials. Then replace $i\uparrow 2$ by -1.

Ex. 5 $(9 + 3i)(4 - 7i) = 36 - 51i - 2i\uparrow 2 = 36 - 51i - 21(-1) = 47 - 51i$
Ex. 6 $(7 + 2i)(3 + 4i) = 21 + 34i + 8i\uparrow 2 = 21 + 34i + 8(-1) = 29 + 34i$

4) Division. To divide two complex numbers, first take the conjugate of the denominator and multiply both numerator and denominator.

Ex. 7

$$\frac{(9 + 3i)}{(4 - 7i)} = \frac{(9 + 3i)\ (4 + 7i)}{(4 - 7i)\ (4 + 7i)} = \frac{15 + 75}{16 + 49}\, i = \frac{15}{65} + \frac{75}{65}\, i$$

Note the form of the result is neither $\frac{15 + 75i}{65}$ nor $\frac{1}{65}(15 + 75i)$

COMPUTER CHOKE

I should have titled this paragraph "How not to make". If we attempt to plug a complex number into our program, the computer will choke on it. We must rewrite the complex number in a manner that the computer can work with.

Our basic iterative function Z z ↑ 2 + c where both z and c are complex

numbers. We must reduce z and c to their real and imaginary parts thus z=x+yi, and c=p+qi. Our iterative function now looks like this:

```
x1 = x*x-y*y+p     y1 = 2*x*y+q
```

See program lines 114 to 122 for complete breakdown.

REFERENCE MATERIAL

The Beauty of Fractals
H. Peitgen & P. Richter

The Fractal Geometry of Nature
Benoit Mandelbrot
W.H. Freeman & Co. N.Y.

12

Additional 6526 Functions

The CIA registers and memory mapped I/O were briefly introduced in Chapter 2. Glossed over in fact so as to not get boggled down in what would then have been esoteric concepts that may have discouraged further exploration. Well as you have gotten this far it's time to take a giant step back and review the CIA registers in greater detail. Although we have touched upon a great many of the CIA functions that are implemented and controlled by the registers we have done so without explicitly saying so. Some samples you will remember are the 60 Hz interrupt routine and the AID serial input projects. Learning how to use these addition registers will increase your versatility to program and interface your computer. But first let's review what we know thus far.

Commodore computers contain two CIA chips. Each chip contains two 8-bit I/O ports, two 16-bit counter/timers, a time of day clock (TOD), and an 8-bit serial port. The CIA has 16 registers that are addressed as successive memory locations by the CPU. Below are the memory locations and descriptions for CIA #2 registers for the C-64 and C-128.

Memory Mapped Addresses for CIA #2

Address	Reg #	Register Description
56576	0	Peripheral data register A
56577	1	Peripheral data register B

Address	Reg #	Register Description
56578	2	Data direction register A
56579	3	Data direction register B
56580	4	Timer A low byte
56581	5	Timer A high byte
56582	6	Timer B low byte
56583	7	Timer B high byte
56584	8	1/10 second register
56585	9	Seconds register
56586	10	Minutes register
56587	11	Hours register
56588	12	Serial Data register
56589	13	Interrupt control register (ICR)
56590	14	Control register A (CRA)
56591	15	Control register B (CRB)

** Note CIA #1 has the same registers starting at address 56320 **

CIA #1 is used intensively by the operating system of the computer. It is used for the 60 Hz interrupt routine that reads the keyboard, joystick port, and updates the real time clock. It is for these reasons we will concentrate on CIA #2.

We have dealt with the peripheral control and data direction registers in Chapter 2. You should be familiar with their operations. If not go back and reread the chapter. We will begin with the 6526 timers.

TIMERS

There are two 16-bit timers in the 6526. Labeled Timer A and Timer B. The 16 bits of the timer allow each timer to count to 65535. Since we are still dealing with an 8-bit machine, reading from or writing to requires us to use two (8-bit) bytes, identified as a low order byte and a high order byte. To load a timer to count 20,000 pulses, clock cycles, or events we would obtain the high byte by:

```
20,000/256 = 78.125    remove the fraction to obtain:
20,000/256 = 78
```

The number 78 is our high order byte. To calculate the low order byte, multiply the high byte by 256 and subtract the results from the original number thus:

```
78 x 256 = 19968
20,000 - 19,968 = 32
```

The number 32 is our low order byte.

The timers are controlled by their respective control registers #14 CRA and #15 CRB.

CONTROL REGISTERS

Each bit on the 8-bit register controls a function. We program this register by poking a one byte control word (number) into it. The first five bits for each register are identical, their function is as follows:

Bit	**Function**
0	1= starts timer 0 = stops timer
1	1= timer output appears on PB6 for timer A and on PB7 for timer B
2	1= toggles (inverts) output for PB6 or PB7 each timeout , 0= continuous output essentially remains high with a brief negative pulse each timeout
3	1= One shot mode, 0=continuous
4	1= forces loads timer
	For CRA only
5	1= counts pulses on CNT line 0= counts system clock pulses
6	1= serial port is output, 0 = serial port input
7	1= real time clock runs at 50 Hz 0= real time clock runs at 60 Hz

For CRB only

Bit 6	Bit 5	Function
0	0	Timer B counts system clock pulses (1 MHz)
0	1	Timer B counts positive transitions on CNT line
1	0	Timer B counts Timer A underflow pulses
1	1	Counts Timer A underflow pulses only while CNT is held high
7	1= Sets TOD alarm, 0= set TOD clock	

INTERRUPT CONTROL REGISTER (ICR)

The bits on the ICR signal if an event has occurred, the bits on the ICR are called flags. The ICR monitors events from 5 different sources. The ICR can be programmed to issue an interrupt whenever any of the flags becomes set. Naturally the ICR can also be programmed not to issue an interrupt. The individual bits function as follows:

Bit	Function
0	Timer A time out (underflow)
1	Timer B time out (underflow)
2	Set when TOD clock = Alarm
3	Set when serial register is full (input) or when serial register is empty (output)
4	Set by negative transition on flag pin
5 8z 6	NOT USED
7	Set when any of the other ICR bits are set

**Note peeking this register erases it, therefore you only have one peek per event. **

Setting the register for an interrupt is accomplished by writing a binary "1" to bit 7 along with a binary "1" to whatever bit you wish to generate the interrupt. For an example to generate an interrupt for timer B time out you would poke the number 128 + 02 into the register.

```
Poke56589,130   binary #  10000010
To disable
Poke 56589,2    binary #  00000010
```

The binary equivalents are poked into the registers, see C□apter 2 if you're confused.

In order to capitalize on interrupt routines that can be generated by the ICR you really must be able to program in ML. Therefore, we will disable the ICR for our applications. We can however, still read the ICR to see if an event has occurred. To disable all interrupts we:

```
Poke 56589,127  binary # 01111111
```

TIME OF DAY CLOCK (TOD)

Let's begin by setting the time of day clock and alarm. We will set the alarm to go off 5 seconds after we run the program. You could modify the clock alarm to be set to whatever timebase you'd like. For expediency I chose 5 seconds. When the alarm time is reached the computer will beep. The timing of the computer is controlled by a quartz crystal and is very accurate. See Fig. 12-1 for C-128 and Fig. 12-2 for C-64.

Let's do a small line analysis: In line 5 we perform a logical "AND 127" with control register B and poke it back into the register. What this does is effectively set bit 7 to 0. Look at the chart detailing the bit functions of control register B. To set the time this bit must be 0. Having accomplished this we then go on to poke a start time into the TOD registers. In line 30 we perform a logical "OR 128" with control register B and poke

```
1 REM FOR C-128
5 POKE56591,PEEK(56591)AND127:REM SET CLOCK
10 POKE56587,0:REM HOURS
15 POKE56586,0:REM MINUTES
20 POKE56585,0:REM SECONDS
25 POKE56584,0:REM 1/10 SECONDS
30 POKE56591,PEEK(56591)OR128:REM SET ALARM
35 POKE56587,0:REM HOURS
40 POKE56586,0:REM ALARM MINUTES
45 POKE56585,5:REM ALARM SECONDS
50 POKE56584,0:REM ALARM 1/10 SEC
90 X=PEEK(56589):REM RESET FLAG
100 WAIT 56589,4 :REM READ ICR FLAG
115 SOUND1,20960,60
120 PRINT"{CLR}"
130 PRINT:PRINT:PRINT:PRINT"ALARM TIME MET"
135 END
```

Fig. 12-1. Time of day C-128.

```
5 REM FOR C-64 OR 128
15 POKE56591,PEEK(56591)AND127:REM SET CLOCK
20 POKE56587,0:REM HOURS
25 POKE56586,0:REM MINUTES
30 POKE56585,0:REM SECONDS
35 POKE56584,0:REM 1/10 SECONDS
40 POKE56591,PEEK(56591)OR128:REM SET ALARM
45 POKE56587,0:REM ALARM HOURS
50 POKE56586,0:REM ALARM MINUTES
55 POKE56585,5:REM ALARM SECONDS
60 POKE56584,0:REM ALARM 1/10 SEC
90 X=PEEK(56589): REM RESET FLAG
100 WAIT 56589,4
105 S=54272
110 FORL=STOS+24:POKEL,0:NEXT
115 POKES+5,9:POKES+6,0
120 POKES+24,15
125 POKES+1,25:POKES,177
130 POKES+4,33
135 PRINT"{CLR}"
140 PRINT:PRINT:PRINT:PRINT"ALARM TIME MET"
145 FORT=1TO250:NEXT
150 POKES+24,0: END
```

Fig. 12-2. Time of day C-64.

it back into the register. This sets bit 7 to a binary "1" so that we can set the alarm time. Our alarm time is the same except for the TOD seconds register that we advance by 5. At this point we can take one of two methods to check when our alarm time has been reached. One is to peek the ICR register and loop back to the peek statement until our alarm time is reached. This is detailed below:

```
90 x=peek(56589): rem read ICR flag
100 If x=4 then 115: rem alarm time reached loop out
110 gotol00: rem alarm time not reached go back and read again
```

Notice, although we are not generating an interrupt, bit 2 will become set when the alarm time is reached. (See ICR bit functions above.) The second method available to us is to use a little known BASIC command: Wait.

The Wait command stops program execution until a certain bit pattern is recognized at a specified memory location. The command structure is as follows:

```
WAIT < location > , < mask >
```

We utilize this command: Wait 56589,4
If you remember, and you should, bit 3 has a binary weight of 4. The computer waits until this bit becomes set then continues program execution. See the Programmers Reference Guide for a more complete description of the Wait command.

FREQUENCY COUNTER

By setting the control register B (bits 5 and 6), we can have the computer count positive transitions on the CNT line. We will utilize this function along with the 60 Hz interrupt routine to make a frequency counter. See programs Fig. 12-3 for the C-128 and Fig. 12-4 for the C-64.

The timers in the CIA count down from a preexisting number in their registers to zero. So part of our program loads decimal 255 into the high and low byte timer registers. In the case we are using, each transition on the CNT line will decrement the low byte counter by one. Everytime the low byte counter rolls past zero the high byte counter is decremented by one and the low byte counter is reloaded with decimal 255 (hex $FF).

The 60 Hz interrupt routine issues an interrupt every .0166666667 of a second. We break into the interrupt routine to read the counters and place those numbers at memory locations 251 and 252. Our basic program reads the numbers and performs the necessary arithmetic to give the frequency.

We have the interrupt routine read the counters every sixth time around, or ten times a second this is the timebase (TB). The time base

```
10 REM FREQUENCY COUNTER C-128
20 FOR L=4864 TO 4941
30 READX:POKEL,X:NEXT
35 TB=6:REM TIME BASE
40 POKE254,TB:POKE250,TB
50 SYS4864:REM START TIMER
60 CNT=(PEEK(251)*256+PEEK(252))
70 FQ=(65535-CNT)/(.0166666667*TB)
75 FZ=INT(FQ):REM REMOVE FRACTION
80 PRINT "{CLR}"
85 PRINT"{C/DN}{C/DN}{C/RT}{C/RT}THE FREQUENCY IS "FZ" HZ"
90 GOTO60
100 DATA 008,120,173,020,003,141,080,019,173,021
110 DATA 003,141,081,019,169,032,141,020,003,169
120 DATA 019,141,021,003,169,003,141,013,221,040
130 DATA 088,096,072,198,254,240,004,104,108,080
140 DATA 019,165,250,133,254,169,000,141,014,221
150 DATA 173,005,221,133,251,173,004,221,133,252
160 DATA 169,255,141,005,221,141,004,221,169,033
170 DATA 141,014,221,104,108,080,019,000
```

Fig. 12-3. Frequency counter C-128.

```
10 REM FREQUENCY COUNTER C-64
20 FOR L=49152 TO 49229
30 READX:POKEL,X:NEXT
35 TB=6:REM TIME BASE
40 POKE254,TB:POKE250,TB
50 SYS49152:REM START TIMER
60 CNT=(PEEK(251)*256+PEEK(252))
70 FQ=(65535-CNT)/(.0166666667*TB)
75 FZ=INT(FQ):REM REMOVE FRACTION
80 PRINT "{CLR}"
85 PRINT"{C/DN}{C/DN}{C/RT}{C/RT}THE FREQUENCY IS "FZ" HZ"
90 GOTO60
100 DATA 008,120,173,020,003,141,080,192,173,021
110 DATA 003,141,081,192,169,032,141,020,003,169
120 DATA 192,141,021,003,169,003,141,013,221,040
130 DATA 088,096,072,198,254,240,004,104,108,080
140 DATA 192,165,250,133,254,169,000,141,014,221
150 DATA 173,005,221,133,251,173,004,221,133,252
160 DATA 169,255,141,005,221,141,004,221,169,033
170 DATA 141,014,221,104,108,080,192,000
```

Fig. 12-4. Frequency counter C-64.

is adjustable by changing one line in the program. You would adjust the timebase to read faster or slower frequencies. To read faster frequencies decrease the timebase. To read slower frequencies increase the timebase. As the program stands it is capable of reading frequencies between 10 Hz and 655340 Hz.

I have made a small frequency generator that can be connected to the user port (see Fig.12-5). It generates a frequency of 10,000 Hz. Because of the rather wide tolerances of the components used the frequency is only approximate. The circuit is a 555 timer generating square waves that are counted on pin 6 (CNT line). You can use this circuit to test the program. Keep in mind when you wish to use this program. Do not connect anything but a TTL source to the CNT line. A TTL source voltage lies between 0 volts and +5 volts. Anything else could damage your computer.

FREQUENCY GENERATOR

We can also use our computer to generate square wave frequencies. In this application we are setting the CRB to toggle the PB7 line on every timeout of our timer (see Fig.12-6). Therefore by setting up the proper numbers in the high and low byte registers we can generate frequencies between 10 Hz and 500,000 Hz.

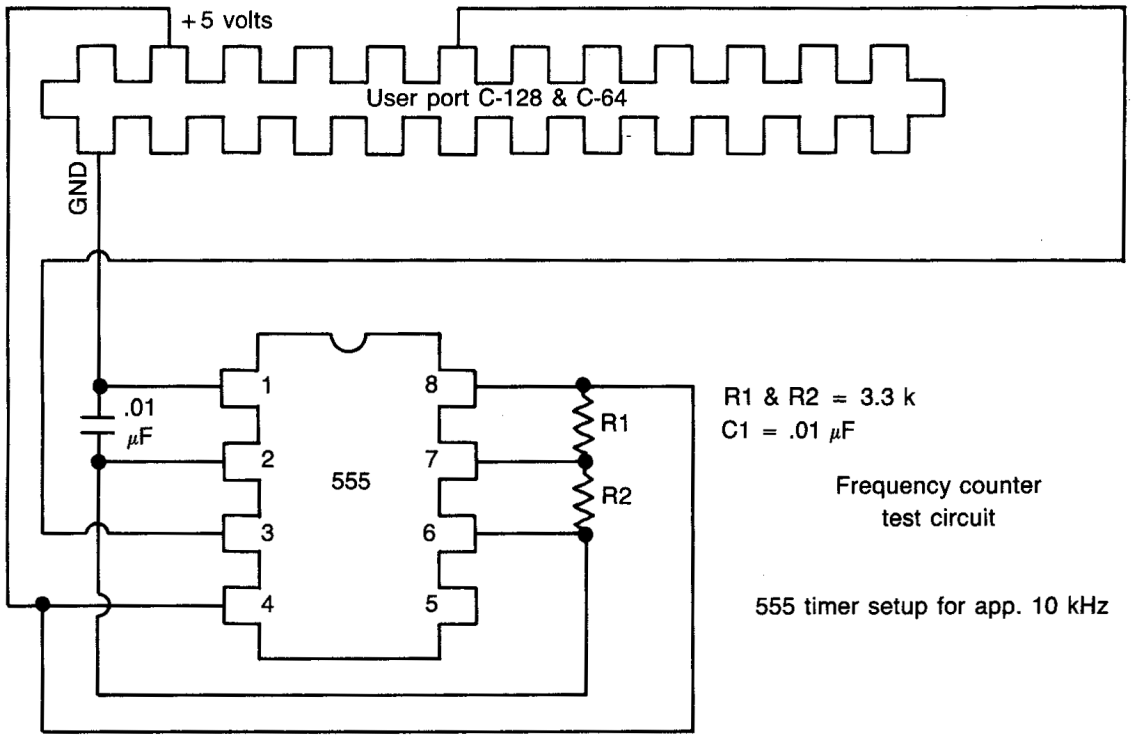
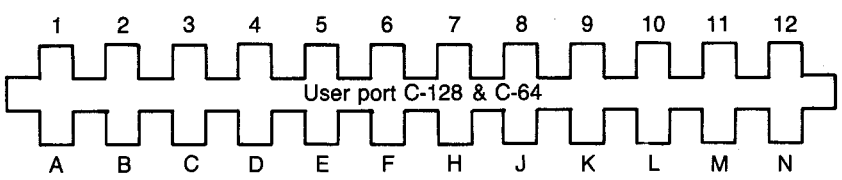


Fig.12-5. Test circuit.



Pin	Function	Pin	Function
1	GND	A	GND
2	+5 volts	B	FLAG Max 100 mA.
3	RESET	C	PBO
4	CNT 1	D	PB1
5	SP1	E	PB2
6	CNT 2	F	PB3
7	SP2	H	PB4
8	PC2	J	PB5
9	Serial in	K	PB6
10	9Vac	L	PB7
11	9Vac	M	PA2
12	GND	N	GND

Fig. 12-6. Pin out user port.

```
10 REM JOHN IOVINE
15 REM FREQ GENERATOR FOR C-128 OR C-64
20 REM CALCULATE HI AND LO BYTE FOR REQUESTED FREQ
25 INPUT"ENTER FREQ. TO GENERATE ";F
30 IF F<10 OR F>500000 THEN PRINT"OUT OF RANGE REENTER FREQ"
   :GOTO25
35 A=1000000/F    :REM FOR C-64 A=1022700/F
40 B=A/2:C=INT(B/256):D=C*256:E=INT(B-D)
50 REM MAIN ROUTINE
55 POKE56579,255 :REM SET UP DDR
60 POKE56589,127 :REM DISABLE INTERRUPTS
65 POKE56580,E   :REM TIMER LOW
70 POKE56581,C   :REM TIMER HIGH
75 POKE56590,7   :REM CRA SET AND RUN
```

Fig. 12-7. Frequency generator C-128 and C-64.

The program does all the math required for setting the registers(see Fig.12-7 for C-128 and C-64). It will prompt you for the required frequency and go about setting the registers and starting. It is interesting to note that the frequency generator once started will continue even if you load and run another program. Provided, of course that the new program doesn't use the timers or user port. If you have access to an oscilloscope you can observe the waveform generated by the computer. If you don't, use the LED interface from Chapter 1. The LED to PB7 will light, enter

a low frequency of 10 Hz because if you go too fast the LED will appear as if it is on constantly. This is an excellent low-cost tool for generating precise frequencies.

Examine the program and see how the CRB is set to accomplish this.

Index

A
A-D interrupt, 53-54
ac loads, 83, 84, 85
address bus, 1
addressing, 1
advanced sound digitizer program, 65, 68-72
algebraic operations, Mandelbrot graphics and, 165
algorithms, speech recognition, 56
allophone, 24
analog events, 36
analog to digital conversion, 34-54
  60 Hz interrupt vector, 51-54
  analog events for, 36
  biofeedback device using, 48-51
  digital events in, 36
  programming of, 42
  resolution in, 40
  serial A-D converter chip for, 37, 41
  sound trace with, 58
  test circuit for, 38, 39, 40
  test program for, 38-42
  toxic gas sensor using, 46-48
  transducers using, 42-46
  variable resistor for, 39
AND, 18, 20
appliance controller, 82-90
  ac loads, 83
  circuit construction, 84-87
  dc loads, 83
  inductive and resistive loads, 83
  parts list for, 90
  program for, 88-89
  real world environment and, 82
  smart control in, 89-90
  testing, 87, 88
artificial vision, 133-134, 136
audio subliminal communication, 74
automatic ventilator control, 48

B
bank switching, 2
base line conductance, 48
BASIC, 6
  speech synthesizer and, 32
binary counting program, 16
binary signals, 9-10, 34
  transmission of data with, 55-56
binary voltage, 35
biofeedback device, 34, 48-51
  circuit for, 49, 50
  electrodes for, 49
bit weights, 10, 11, 26
bits, OR to set, 18, 19
black and white digital camera, 100, 124-125
bootstrap programs, 2
Brindley, G., 136
buses, 1
butterfly effect, dynamic equations and, 148

C
cadmium sulfide (CdS) photocells, 43
calibration, thermistor, 46
central processing unit (CPU), 1, 7
chaos, 140
  graphic programs for, 141-143
  order out of, 143-146
climatic models, 148
clocking line, 36
coloration, digital camera, 103
column address strobe (CAS) pin, 97
complex interface adapter, 7
complex numbers, 150, 164-166
computer choke, 165
computer fundamentals, 1-6
control program, digital oscilloscope, 62
control registers, 6536 CIA chip, 169

D
D-CAM chip, 96-98, 137

matrix array, 97
pinout for, 98
data bus, 1
dc loads, 83
DDR register, 10
demo interrupt program, 52
digital audio record and play, 55-72
applications for, 56
digital oscilloscope, 59
sound sampling with, 56-58
digital camera, 96-138
256 shades of gray for, 102
artificial vision and, 133-134
black and white, 100
black and white program for, 124-125
case for, 105
circuitry for, 107
coloration, 103
construction of, 104
D-CAM chip in, 96-98, 137
extended field of view in, 100
final assembly for, 109
gray scale, 101
gray scale, program for, 117-119, 126-128
hi-resolution, 120
hi-resolution, program for,
hi-resolution screens, 130-131 131-136
lenses for, 104
lighting conditions for, 110
low resolution screen for, 99, 128-130
main program for, 112-117, 121-124
matrix array in D-CAM chip, 97
matrix unscramble for, 99
parts list for, 138
photoelectric effect and, 98-99
power supply for, 105-106, 108
preassembly test for, 108
program operation in, 119
system limitations, 99
timing, 103
digital events, 36
digital logic probe, 5
digital oscilloscope, 59-72
advanced sound digitizer program, 65, 68-72
circuit board for, 61
circuit description for, 59
circuit for, 61
control program for, 62
digital scope program for, 62
input voltages for, 64
loader program, 62
machine language for C-128, 63
machine language for C-64, 63
programs for, 60
schematic for, 60
traces on, 66-67
digital scope program, 62, 64, 65
digital to analog (DAC) chip, 59
disk drives, 3
Dobelle, 136
dynamic equations, 139-149
butterfly effect, 148
graphing programs for, 141-143
nature and, 139-140
order out of chaos, 143-146
population growth model for, 140
self-similarity in, 146-147
usefulness of, 148
dynamic RAM chips, 96

E

80-column to TV monitor, 91-93
electronic logic, 20
electronic thermometer, 46
exposure meter, 45

F

56588 serial register, 42
56589 interrupt control register, 42
fractal equations, 139, 149, 164, 166
frequency counter, 6536 CIA chip, 172, 173
frequency generator, 6536 CIA chip, 174, 175

G

gray scale digital camera, 101
256 shades of gray for, 102
program for, 117-119, 126-128

H

heat transducers, 45-46
hi-resolution digital camera, 120, 131-136
high power electrical devices, 82

I

inductive loads, 83, 85
input, user port, 11
input-output devices, 2
interfacing, 3
parallel, 34, 35
serial, 34
interrupt control register (ICR), 42
6536 CIA chip, 169
interrupt routines, 34
interrupt vector, 51-54
iterative equations, 150

K

kernel ROM, 2
keyboards, 2

L

LEDs, 83
lenses, digital camera, 104
Lewin, W., 136
lie detector, 50
light meter, 45
light transducers, 43-45
applications for, 45
light cell and circuit for, 44
linguistics, 24
loader program, digital oscilloscope, 62
loads
ac, 83
dc, 83
inductive and resistive, 83
logic, 18-19
electronic, 20
logic gates, 20
Lorenz, Edward, 148
low resolution screen, digital camera and, 99

M

machine language, 6, 63
Mandelbrot graphics, 150-166
advanced operations for, 162
algebraic operations and, 165
complex numbers and, 150
computer choke, 165
first picture using, 152-153
fractals and, 164
photo screen images of, 161, 163
plotting, 151
program for, 153-162
program operations for, 162
properties of complex numbers in, 164-165
reference material for, 166
memory, 1-2
memory mapped I-O, user port, 8
message screens, subliminal communication, 79-80
microprocessor unit (MPU), 1, 7
mind control, subliminal communication and, 74
modems, 3
monitor projects, 91-95
80-column to TV, 91-93
screen saver C-128, 93-95
monitors, 2

N
NAND gate, 20
negative temperature coefficient type thermistors, 45
NOR gate, 20

O
operating systems (OS), 2
OR, 18-20
output, user port, 12

P
parallel interfacing, 34, 35
parallel to binary voltage conversion, 35
PDR register, 11
PEEK, 18
phonemes, 22
photocells, 43, 96
photoelectric effect, 98-99
pixels, 55
plotting, Mandelbrot graphics and, 151
POKE, 18
binary codes, 11
population growth model, dynamic equations, 140
program disk, 176
programming, analog to digital conversion, 42
prototype breadboard, 3-5

R
random access memory (RAM), 1
read only memory (ROM), 2
read operations, 2
real world environment, 82
register locations, 10
resistive loads, 83, 84
resolution, analog to digital conversion and, 40
row address strobe (RAS) pin, 97

S
60 Hz interrupt vector, 51-54
6522 VIA chip, 7
6526 CIA chip, 7-8, 167-176
6536 CIA chip
control registers in, 169
frequency counter in, 172
frequency generator in, 174
interrupt control register in, 169
memory mapped addresses for, 167
time of day clock (TOD) in, 170
timers in, 168
screen saver C-128, 93-95
self-modifying equations, 140
self-similarity, 146-148
serial register, 36
serial A-D conversion chip, 37, 41
serial interfacing, 34
serial register, 42
Shaw, J.D., 136
smart control, appliance controller, 89-90
soldering iron, 3
solderless breadboard, 4
sound interface device (SID) chip, 24, 57, 59
sound sampling, digital audio record and play for, 56-58
sound trace, 57, 58
spectrographic analysis, 45, 46
speech recognition algorithms, 56
speech synthesizer, 22-33
BASIC crunch for, 32
circuit construction of, 24-25
digitally recorded speech ROM for, 22
linguistics for, 24
parts list for, 33
phonemes for, 22
power supply for, 25
program for, 25, 29-32
schematic, sections A and B, 26-27
Section A, 28
Section B, PC board mounted, 28
sound interface device (SID) chip for, 24
speech chip (SPO256-A12) for, 23
theory and operation of, 22
SPO256-A12 speech chip, 23
stress management device, 51
subliminal communication, 73-81
audio, 74
basic controller program for, 78
bibliography for, 80
circuit construction for, 75-76
circuit operation of, 76-79
history of, 73
hookup for, 76, 77
law regarding, 75
machine language, C-64 and C-128, 79
message screens for, 79-80
Orwellian mind control with, 74
troubleshooting, 80
visual, 74
switching, bank, 2

T
temperature transducers, 45-46
thermistors, 45-47
thermometer, electronic, 46
thermostat control, 46
time of day clock, 6536 CIA chip, 170, 171
timers, 6536 CIA chip, 168
timing, digital camera, 103
tools, 3
toxic gas detector, 34, 46-48
applications for, 48
schematic of, 47
transducers, 34, 42-46
truth tables, 20

U
user port, 7-21
advanced project for, 15
analog to digital conversion of serial input to, 34
basic project for, 14
binary relationships, 9
circuit construction, 12
connections in, 13
connector wiring for, 15
DDR register, 10
electronic logic in, 20
input, 11
LED substitution for, 14
logic and, 18-19
memory mapped I-O, 8
output, 12
peripheral data register (PDR), 11
schematic of, 8
setting bit with OR, 18, 19
wiring diagram, 13

V
variable resistor, analog to digital conversion usage of, 39
Verhulst, P.F., 140
versatile interface adapter, 7
visual subliminal communication, 74
visual transmissions, 55
voice synthesizer, 56
VOM, 5

W
wiring, prototype breadboard, 5
write operations, 2

ELECTRONIC PROJECTS FOR YOUR COMMODORE 64 AND 128

John Iovine

T1536000797962

Do amazing things with your Commodore—for under $65!

This handy sourcebook presents eleven inexpensive projects that will greatly enhance the performance and value of any Commodore PC. The information provided here will serve as the spark to ignite your own creativity. With the skills and concepts you'll learn while building these projects, you'll be able to explore further—to design and build interfaces that fulfill your own special needs!

Projects range in complexity from a simple LED interface for the user port to a full-featured digital camera. You'll cover such areas as analog-to-digital conversion, subliminal communication, appliance control, monitoring, and fractals. And all of these projects can be constructed for under $65, including:

- Speech synthesizer
- Speech recognition program
- Digital oscilloscope
- Digital audio record/playback unit
- Automatic ventilation control
- Toxic gas sensor
- Heat sensor
- Biofeedback
- Light sensor
- Electric fan

This book offers fascinating projects and a feast of circuits for the electronics hobbyist to build, experiment with, and put to use in many different applications. The information in this book can serve as a foundation for explorations into neural networks, character and pattern recognition, and more. You will be amazed at what you can do with your Commodore!

John Iovine is an avid electronics hobbyist and writer. He has had numerous articles published in *Radio-Electronics*® magazine and *Commodore Magazine*.

0589 $15.95
ISBN 0-8306-9383-1
90000
9 780830 693832

TAB

TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0850