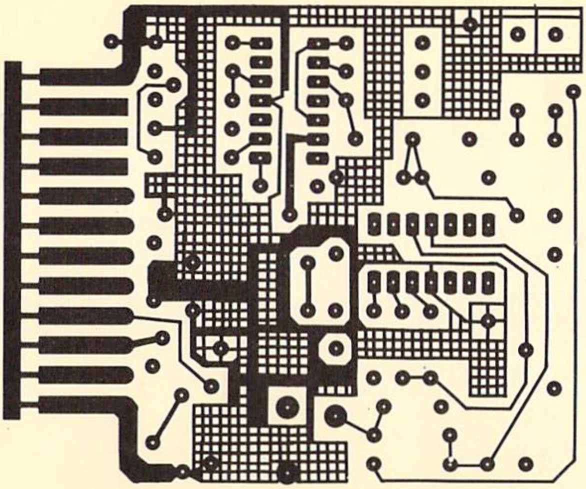
COMMODORE 64 INTERFACING BLUE BOOK

V. J. GEORGIOU



COMMODORE 64 INTERFACING BLUE BOOK

By
V. J. Georgiou, Ph. D.

MICROSIGNAL PRESS
GOLETA, CALIFORNIA

ISBN 0-912911-01-8
Published by MICROSIGNAL PRESS
P.O. Box 388, Goleta, CA 93116

PREFACE

The Commodore 64 computer is a milestone in computer development. It offers substantial computational power and great versatility at a price everyone can easily afford. Thus it has found a wide range of applications, from games to scientific programming.

There is, however, a vast class of applications which remains to a large degree unexplored: Interfacing the 64 to the real world so that it can sense and control physical processes.

This book and its predecessor, the "VIC-20 Interfacing Blue Book" address this problem from a practical point of view: Useful interfacing projects are defined, their theory of operation is explained and then information is given on how to build and program them.

Great emphasis has been placed in making the projects easy to understand and easy to build, with as few parts as possible. The latest Integrated Circuits are specified for best performance and simplicity. At the same time, an effort has been made to avoid projects that are trivial or purely instructional and do not have any practical use.

The wide variety of backgrounds of the users of the 64 makes it difficult to select a presentation level for this book. After much thought, an intuitive approach was adopted with the hope that anyone who can read schematics and has a minimal background in electronics will have no trouble following the book. The writing style is kept to the point and overexplanation is avoided.

If you have built electronic circuits before you should be able to successfully build at least some of the simpler projects. If you encounter problems, do not give up, look for help. The local user's club is a good place to start. Whether you can build something depends more on your determination to succeed and less on your qualifications when you start out.

The customary (for this type of book) section on basic electronics has been omitted. Your local library will have at least a few good texts on introductory electronics. Do not hesitate to consult them if you need help.

A necessary companion to this book is the "Commodore 64 Programmer's Reference Guide", published by Commodore Business Machines and distributed by Howard W. Sams & Co. It is also a good idea to write to the manufacturers of the IC's used in the projects you are interested in and get a copy of their specification sheets.

Your comments and questions on this book are welcome. You may write to me c/o Microsignal Press.

March 1984 V. J. Georgiou

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTORY INFORMATION

I1. Introduction 2
I2. The Ports on 64 4
I3. General Programming Notes 9
I4. Programming the USER PORT 12
I5. Construction Methods 14
I6. Sources of Materials 18

CHAPTER 2: IMPROVING THE 64

1. USER PORT Breadboarding 20
2. External Audio and Video 25
3. Universal Cassette Interface 29
4. Parallel Printer Interface 37
5. Expansion Port Buffer 40
6. Reset Switch 45
7. Light Pen 50
8. RS-232C Interface 55
9. Adding a User Port 60
10. More I/O Ports 63
11. 8K RAM with Data Retention 69
12. Adding Another 64K of RAM 72

CHAPTER 3: SENSOR INTERFACING

13. Computerized Thermometer 77
14. Light Sensor 83
15. Liquid Level Sensor 85
16. Force Sensor 87

CHAPTER 4: TELEPHONE INTERFACES

17. Telephone Ring Detector 90
18. Pulse Telephone Dialer 93
19. DTMF Telephone Dialer 96

20. Phone Line Status Monitor 100
21. DTMF Receiver 101
22. Voice Mail 106

CHAPTER 5: VOICE INTERFACES

23. Voice Output 114
24. Allophone Speech Synthesizer 123
25. Voice Input 133

CHAPTER 6: A/D AND D/A INTERFACES

26. Simple D/A Converter 141
27. Precision 8-Bit D/A Converter 143
28. 8-Channel 8-Bit A/D Converter 146
29. 12-Bit Data Acquisition System 150

CHAPTER 7: CONTROLLER INTERFACES

30. PWM Motor Controller 154
31. AC Power Controller 157

CHAPTER 8: INSTRUMENTATION CIRCUITS

32. Capacitance Meter 161
33. FFT Spectrum Analyzer 164
34. Logic Analyzer 171

CHAPTER 9: HOME SECURITY SYSTEMS

35. Pickproof Digital Lock 178
36. Home Security System 183

CHAPTER 1

INTRODUCTORY INFORMATION

INTRODUCTION

What does a computer do? It "runs" programs which process information. The information (or data) is entered into the computer, the program processes it and the result, which is also information, is output from the computer.

In the most familiar case, the input information is generated by a human and the output information is received by a human. For example, a program that computes interest obtains from the user data (interest rate, time in the bank and amount of capital) and produces data readable by the user (amount of interest generated).

There is, however, a class of applications where the data is generated without intervention of the user, by an instrument. For example, a temperature transducer converts room temperature into an electrical signal. A human cannot directly read such a signal. He must use a voltmeter to convert the signal to a visual image (deflection of a needle) which he can perceive with his eyes. Likewise, a computer cannot directly "perceive" this signal, even though the signal is electrical and the computer itself is an electronic device. There are two reasons for this incompatibility: First, the signal from a temperature transducer is an analog (continuous) signal. The computer operates only on binary digital (discrete) signals. Second, computer programs operate on data, not signals. The signal must be input into the computer in such a way that is recognized as data.

What we need here is a device which converts signals to data and vice versa. Such a device is called an "interface".

In general, the term "interfacing" refers to the design, construction and testing of "interfaces" which are devices that convert one type of signal to another or convert signals to data and vice versa.

Here are two examples of interfaces:

1. RS-232C INTERFACE. Allows the computer to communicate with other computers or peripherals using the RS-232 standard. The interface converts data into electrical signals compatible with the standard and vice versa.
2. VOICE SYNTHESIZER. Data is converted to speech.

An interface usually consists of some hardware (circuits) that do the signal conversion and some software. The software used in conjuction with an interface falls into two categories: Driver routines and application programs. A driver routine generates the digital signal(s) to drive the interface and then takes the raw data coming out of it and converts it to information that is put to use by the application program to perform a useful function.

Interfaces used to sense physical quantities such as temperature, light, pressure etc. (also called "real world" interfaces) tend to have similar hardware, consisting of the following parts:

1. Transducer. Senses a physical quantity and generates an electrical signal whose variations are related to the variations of the physical quantity. The relationship in most cases is not linear.
2. Signal Conditioner. Amplifies and filters the output of the transducer.
3. Analog-to-Digital (A/D) Converter. Converts the analog signal to a digital word.
4. Input Port. Reads into the computer the digital word generated by the A/D converter.

If the result of the computations is used to affect the outside world directly, for example, turn on a light or drive a motor, then the hardware of the interface will consist of similar parts but in reverse order:

1. Output Port. Converts data into a digital word.
2. Digital-to-Analog (D/A) Converter. Converts a digital word to an analog signal.
3. Signal Conditioner. Amplifies and filters the output of the D/A converter (this stage is often not required).
4. Transducer. Converts an electrical signal to a physical quantity such as heat, movement, light, sound etc.

THE PORTS ON 64

Simply put, a port is a connector on the computer via which data can be received or transmitted. The 64 has many ports, making it an excellent machine for interfacing purposes. From our standpoint, the most important ports are the USER PORT and the expansion port.

The voltage levels on these ports are TTL compatible. Input voltages should never exceed +5 volts. Negative (with respect to ground) voltages are not allowed on any port and they may damage the 64.

The USER PORT provides 8 bits of I/O, that is, each pin can be programmed to be either an input or an output. There are also three "handshake" lines which are used to accomplish the data transfer between the computer and the device connected to the port. In addition, there is a pin that supplies +5V at up to 100 mA for use by the peripheral device on the port, and two pins that supply 9VAC at 100 mA which can be used to generate other voltages that may be needed by the peripheral. The rest of the pins are for use by the RS-232C interface.

The pin labeled RESET will reset the computer when it is grounded momentarily. Do not use it unless you know exactly what you are doing. For more information on this pin and on resetting procedures, see project 6.

The data I/O lines are designated PB0-PB7, PB0 being the least significant bit. The handshake lines are:

1. PA2 : Can be either input or output. It is a data I/O line of another port (port A) on the same chip. The rest of the lines of port A are used internally by the 64 and are not available to the user.
2. FLAG2: Input only. When it goes from "1" to "0", it can set the flag flig-flop inside the 64. The flag is cleared every time it is read by the program.
3. PC2 : Output only. It goes low for one cycle (1 us) every time the USER PORT is written or read.

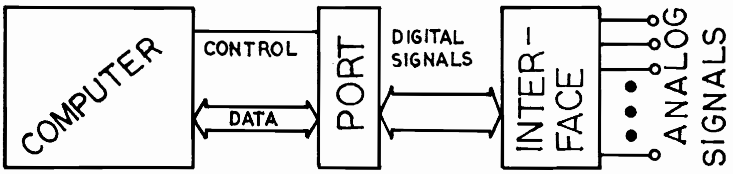


Figure 1.1. Interfacing Block Diagram

The USER PORT is very convenient for interfacing because it can be directly addressed from BASIC or machine language and requires no additional hardware.

The expansion port allows access to the inner sanctums of the 64. Thus, it is more difficult to use fully and one should be more careful in dealing with it. A wrong connection on the user port may result in a fried output chip but on the expansion connector it may be the end of your 64 as a computer.

We will be using the expansion port mainly to add I/O ports similar to the USER PORT. Thus our interfacing will be relatively straightforward, using the signals I/O1 and I/O2 which define the two I/O address blocks reserved in the 64, the data bus D0-D7, some address lines in the range A0-A7 and the R/W line.

The two I/O address blocks are 256 long each, which means we can have up to 256 I/O ports in each block – many more than we are ever likely to need.

To add an I/O port we need an address decoder that determines the location of the port within the address space of the 64, the port itself and steering logic that enables the input or output port depending on the state of the R/W signal. This is shown in general terms in the block diagram in Fig. 1.2. For more details on port construction, see projects 9 and 10.

The rest of the signals on the expansion port are to be used in specialized applications (like dual processor operation) which are outside the scope of this book.

The pinouts of all the ports of the 64 are given in Fig. 1.3 for reference. The pins used in each project and their designations are shown in the schematic for that project.

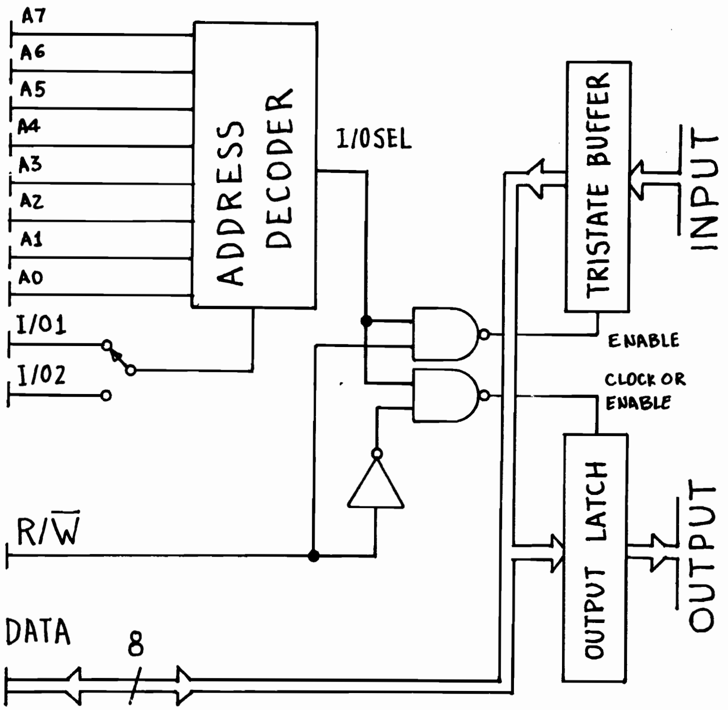


Figure 1.2. I/O Port Block Diagram

CONTROL PORT 1

Pin	Type
1	JOYA0
2	JOYA1
3	JOYA2
4	JOYA3
5	POT AY
6	BUTTON A/LP
7	+5V
8	GND
9	POT AX

CONTROL PORT 2

Pin	Type
1	JOYB0
2	JOYB1
3	JOYB2
4	JOYB3
5	POT BY
6	BUTTON B
7	+5V
8	GND
9	POT BX

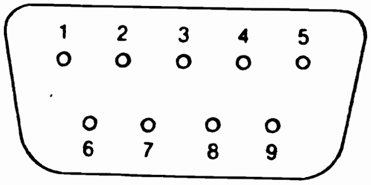
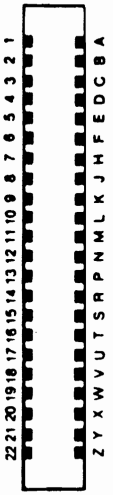


Figure 1.3. Port Pinouts

EXPANSION PORT

Pin	Type
1	GND
2	+5V
3	+5V
4	$\overline{IRQ}$
5	R/$\overline{W}$
6	Dot Clock
7	I/O 1
8	$\overline{GAME}$
9	$\overline{EXROM}$
10	I/O 2
11	$\overline{ROML}$

Pin	Type
A	GND
B	$\overline{ROMH}$
C	$\overline{RESET}$
D	$\overline{NMI}$
E	S 02
F	A15
H	A14
J	A13
K	A12
L	A11
M	A10

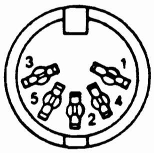


Pin	Type
12	BA
13	$\overline{DMA}$
14	D7
15	D6
16	D5
17	D4
18	D3
19	D2
20	D1
21	D0
22	GND

Pin	Type
N	A9
P	A8
R	A7
S	A6
T	A5
U	A4
V	A3
W	A2
X	A1
Y	A0
Z	GND

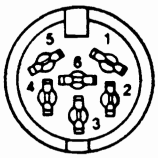
AUDIO/VIDEO PORT

Pin	Type	
1	LUMINANCE	
2	GND	
3	AUDIO OUT	
4	VIDEO OUT	
5	AUDIO IN	



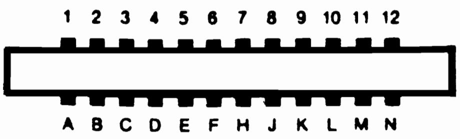
SERIAL I/O PORT

Pin	Type
1	SERIAL $\overline{SRQIN}$
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	$\overline{RESET}$



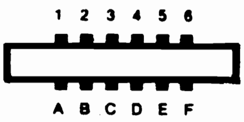
USER PORT

PIN	SIGNAL	PIN	SIGNAL
1	GND	A	GND
2	+5V	B	FLAG2
3	RESET	C	PB0
4	CNT1	D	PB1
5	SP1	E	PB2
6	CNT2	F	PB3
7	SP2	H	PB4
8	PC2	J	PB5
9	SER ATN	K	PB6
10	9 VAC	L	PB7
11	9VAC	M	PA2
12	GND	N	GND



CASETTE PORT

Pin	Type
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



GENERAL PROGRAMMING NOTES

The software used in conjuction with an interface falls in two categories: Driver routines and application programs. A driver routine generates the digital signal(s) to drive the interface and then takes the digital signals coming out of it and converts them into data that the application program can manipulate.

To illustrate this, let's take as example the simple thermometer project. The driver program generates a trigger pulse and measures the length of the output pulse of the 555 timer. This length is stored in two consecutive bytes. The information in these bytes is easy to manipulate (for example we can use BASIC to PEEK their values and display them on the screen) but this information by itself is not terribly useful. It is the job of the application program to turn the information gathered by the driver program into something useful. For example, in the case of the thermometer a useful output is a reading of the temperature in degrees C.

A driver program is usually written in assembly language because it must move data as fast as the hardware can generate them, and of course assembly language programming gives the fastest programs. On the other hand, application programs are rarely required to perform at microsecond speeds and they are written mostly in BASIC because it is much easier to program in BASIC than in assembly language.

Driver programs are normally associated with the hardware they drive and have little to do with the ultimate application. In the thermometer example, the driver program will be the same whether you measure the temperature of a room or control the temperature of an oven. The application programs will obviously be quite different in these cases.

For these reasons, in this book you will always find the driver program listing if a driver is needed. Also for every project there are application programming hints but not necessarily an application program. For simplicity, all drivers are position independent, meaning they can be placed anywhere in memory and they will run without modification.

In practice, the best place for a driver routine is a reserved area that cannot contain a BASIC program or variables. Fortunately, the 64 has such a reserved area and it is of generous proportions – a full 4096 bytes. It starts at location 49152 ($C000) up to location 53247 ($CFFF). There is sufficient space there for all machine language programs in this book to fit comfortably.

There is, however, the possibility that the manufacturer of a peripheral you are now using has already taken advantage of this area and it is not available for your use. If this is the case, you can still have your bought peripheral and your own one too. You simply borrow some memory space from BASIC by manipulating its top-of-BASIC pointer.

To reserve space at the top of the RAM used by BASIC, let us say 100 bytes, you must decrease the top-of-BASIC pointer (MEMSIZ) and the bottom-of-string pointer (FRETOP) by 100. These pointers are in locations 55-56 and 51-52 respectively. Here is a BASIC program that will reserve 100 bytes at the top of the BASIC RAM:

```
100 REM ** TOP-OF-RAM RESERVE **
101 REM
110 Q1=PEEK(55)+256*PEEK(56)
120 Q2=Q1-100
130 Q3=INT(Q2/256): Q4=Q2-Q3*256
140 POKE55,Q4: POKE56,Q3
150 POKE51,Q4: POKE52,Q3
```

Line 120 can be changed to accomodate programs whose length is different than 100 bytes. For example, for a 40 byte program use:

```
120 Q2=Q1-40
```

Assembly language listings in this book follow standard notation, having the object code first (in hexadecimal notation) followed by label (if any), instruction mnemonic, address or data field and comment as follows:

```
        A900 TRIGG LDA #0 ; GENERATE TRIGGER
       /          /     \             \
 Op Code  Label  /      Data            Comment
           Instruction Mnemonic
```

Only the object code is to be entered in the computer, using a monitor program like 64MON.

If a monitor program is not available, the object code must be converted to decimal notation first and then entered using the POKE command from BASIC. The conversion and POKEing can be done easily with the help of the BASIC program given below:

```
200 REM   ** MEMORY POKER **
201 REM
210 PRINT CHR$(19);CHR$(147)
220 PRINT"ENTER STARTING ADDRESS IN DECIMAL"
230 INPUT SA: I=0
240 PRINT"DATA BYTES ARE IN HEXADECIMAL"
250 PRINT"TO EXIT THE PROGRAM, PRESS RUN/";
260 PRINT"STOP AND RESTORE AT THE SAME TIME"
270 PRINT:PRINT
300 PRINT"ENTER BYTE #";I;:INPUT H$
310 IF LEN(H$)>2 GOTO 300
320 L=ASC(LEFT$(H$,1))
330 IFL<60THEN L=L-48
340 IFL>64THEN L=L-55
350 R=ASC(RIGHT$(H$,1))
360 IFR<60 THEN R=R-48
370 IFR>64 THEN R=R-55
380 POKE SA+I,L*16+R
390 I=I+1:GOTO300
```

In most projects in this book the machine language program is in the form of DATA statements in the application program. When the application program runs, the first thing it does is to place in memory the driver program. It reads the DATA statements and POKEs the data into memory. This is the simplest way to deal with short machine language drivers.

PROGRAMMING THE USER PORT

The USER PORT on the 64 is port B on CIA #2. A CIA (which stands for Complex Interface Adapter) lives up to its name by being a complex but very flexible peripheral chip. For our purposes, we will use only a very limited number of its capabilities, namely I/O port B and its handshake signals.

The I/O port has 8 lines, each of which can be an input or output. These lines are designated PB0-PB7. However, they cannot be at the same time both inputs and outputs. They must be assigned a function, either input or output before they can be used. This is done by writing into the DDR (Data Direction Register) an "1" for each line we want to be an output and a "0" for each line we want to be an input. For example, if the least significant bit of DDR is set to "0", then PB0 will be an input.

The port itself is located at address 56577 ($DD01) and the data direction register is located at address 56579 ($DD03). Here is an example of setting up the port. We want the lower four bits to be inputs and the upper four bits to be outputs. The DDR then must be set to 11110000. This in Hex is F0 and in decimal 240. So, to set up the DDR we POKE 56579,240. Now let's say we want to output on the two most significant bits an "1" and a "0" on the next two (we can do this because we have set the four most significant bits of DDR to "1" so the corresponding port lines are set to output). To do this, we need to POKE into the port the number 1100XXXX where XXXX are don't cares – they can be either "1" or "0" without affecting anything. Let us say they are "0", in which case our number is B0 in Hex or 192 in decimal. So we POKE 56577,192. If we use a voltmeter, we can verify that the four most significant bits are set as ordered ("1" is + 5 volts). We can also verify input operation with the following program:

```
10 PRINT PEEK(56577) AND 15: GOTO 10
```

The port has also three handshake lines that can be used to coordinate information transfer to and from the port. These are PA2, FLAG2 and PC2. PA2 is actually the third bit of port A on CIA #2. It is programmed the same way as the bits in port B we just explained. The port A address is 56576 ($D000) and its DDR is located at address 56578 ($D002). The rest of port A is used internally by the 64 and it is best not to disturb it when you want to use PA2. This is done as follows:

1. To set PA2 to input:
   POKE 59578, PEEK (59578) AND 251
2. To set PA2 to output:
   POKE 59578, PEEK (59578) OR 4
3. To output a "0" to PA2:
   POKE 59576, PEEK (59576) AND 251
4. To output an "1" to PA2:
   POKE 59576, PEEK (59576) OR 4

FLAG2 is an input line that sets the fifth bit of a CIA #2 register called ICR (Interrupt Control Register) when:

1. The fifth bit of the ICR mask register is set, AND
2. The input at FLAG2 changes from "1" to "0" (this is a transition sensitive input).

Reading the ICR clears it. The ICR and the ICR mask register are at the same address, 59589 ($D00D). The first is read-only and the second is write-only. To set the ICR mask register to detect the negative transition of the FLAG2 input,

```
POKE 59589,16: A=PEEK (59589)
```

To see if a transition has occured,

```
20 PRINT PEEK (59589) AND 16: GOTO 20
```

When a transition occurs, 16 is printed out.

PC2 is output only and goes low for one clock cycle (1 us) when port B is written or read. This pulse is automatically generated, no programming is required.

For an example of a handshake using FLAG2 and PC2 see the printer interface project.

CONSTRUCTION METHODS

All projects in this book require certain amount of electronic construction. You will need small hand tools, a soldering iron and solder and if you plan to do wirewrapping, you will need the special tools required for this construction technique.

To construct a circuit, you interconnect the components specified in the parts list according to the schematic diagram. The notation used in the schematics of this book is standard but you may not be familiar with the notation for a data bus, as shown in Fig. 1.4.

TYPES OF CONSTRUCTION

There are four recommended methods of constructing the circuits in this book: on a "superstrip", on a perf board, wirewrap and PC (Printed Circuit) board. Sockets are recommended for all IC's. They are not needed when a superstrip is used because it is itself a form of socket.

The superstrip is shown in Fig. 1.5. The IC's are inserted as shown and connections are made using #22 insulated solid wire. It is great for breadboarding and trying circuits out but it is not a good idea for permanent circuits because of the chance of somebody pulling a wire or component out of place.

The perf board method requires a piece of phenolic or epoxy perf board with holes spaced at 0.1". The components are inserted from one side and connections are made on the other using thin (#25 or thinner) insulated wire. The wire is soldered on the component leads. Teflon insulation is best because the wire pieces are small and tend to overheat during soldering melting some types of insulation. Teflon wires on the other hand are difficult to strip and expensive unless found surplus.

Wirewrapping requires a wrap tool, a wire stripper an unwrap tool and special wire, plus special wirewrapping

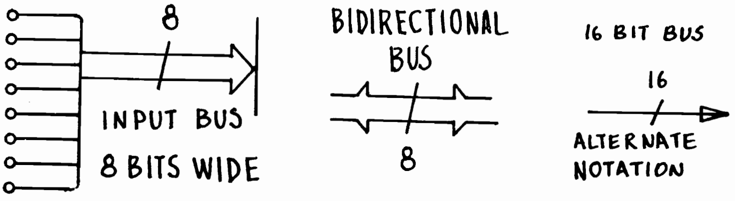


Figure 1.4. Bus Notations

sockets with long square leads. It results in a very reliable assembly and connections can be easily changed.

Wirewrapping is the preferred method of breadboarding digital circuits having more than a few chips. It is not recommended for analog circuits. There is a variety of prototyping boards that have holes in a 0.1″ grid suitable for wirewrap sockets, complete with gold plated edge connectors. Alternatively, you may use a piece of perf board with holes on 0.1″ grid, and flat cable connectors to connect to the outside world.

Finally, the PC board gives the neatest and easiest assembly, but is difficult to make extensive changes once the PC is made, so it is not practical for breadboarding. In some projects PC layout is given so you can make your own boards. These PC boards are also available from MICROSIGNAL, write for prices and availability.

MOUNTING CONNECTORS ON PERF BOARD

Card edge connectors (used to plug into the user port and cassette port) can be mounted on perf board as follows:

1. Form the connector tabs so that their spacing is about the thickness of the board (1/16″). This is done by pressing the connector against a hard flat surface as shown in Fig. 1.6, so that the tabs bend uniformly.

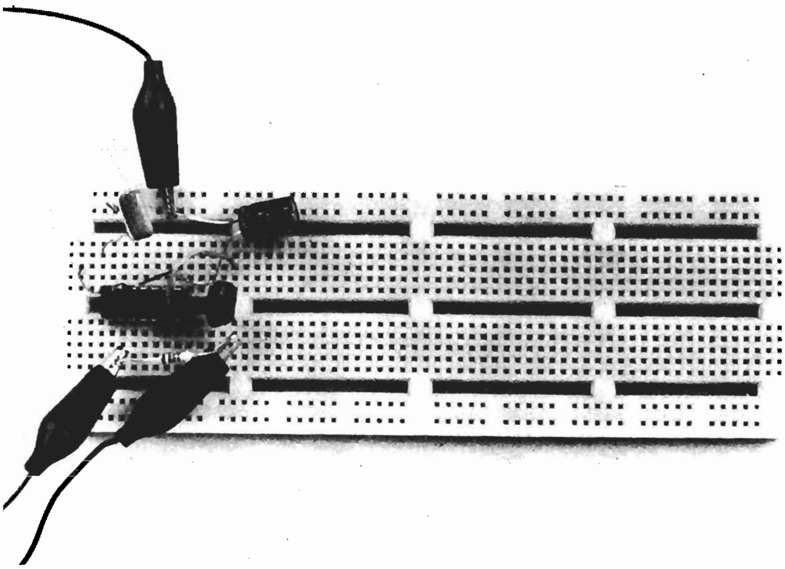


Figure 1.5. The Superstrip

2. Slip the connector on the board and use a bare #20 or heavier wire in a loop going through two holes in the board and then through the eyelets of the endpoints of the connector. You can do this only on the eyelets at each end because the end pins on both sides are ground and it does not matter if they are shorted together. All other pins have different signals.
3. Tighten and then crush the loop with a pair of pliers to make it support the board. See Fig. I.7.
4. Solder the wire and eyelets together to immobilize the assembly.

This method of attachment will have enough strength to allow using the board as handle to insert and extract the connector.

If wires must go from your board to another device, make sure there is adequate strain relief. To do this, drill a 0.125" hole close to the edge of the board and pass a nylon cable tie through it. Use the tie to clamp the wire securely on the board (see Fig. I.8) so that the stress goes on the insulation of the wire and not on the conductor which might break.

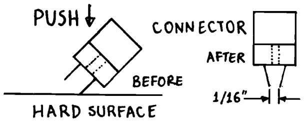


Figure 1.6. How to Bend the Leads of a Connector

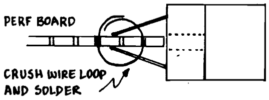


Figure 1.7. Attaching the Connector to the Perf Board

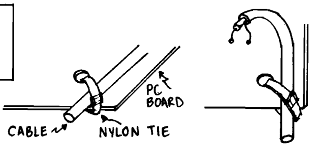


Figure 1.8. Providing Strain Relief

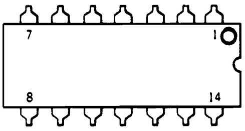


Figure 1.9. Orientation Marks of DIP Packages

COMPONENT ORIENTATION

Make sure that IC's are plugged in the right way. Check the component placement guide (if one is given) for proper orientation. Pin 1 of Dual In Line (DIP) IC packages is marked with a dot embossed on the package next to pin 1. Also, there is a notch embossed in the plastic on the side that pin 1 is (see Fig. I.9).

PC board layouts mark the location of pin 1 by using a round pad for this pin instead of the usual rectangular one.

The leads of the transistors do not follow a general assignment scheme. You must get their function from the spec sheet of the transistor or a data book.

Diodes must be inserted as shown in the diagram. The cathode is marked by a black band around the body of the diode closest to the lead that is the cathode.

Electrolytic capacitors have polarity which you must observe. One of the leads is marked + or − on the body of the capacitor and it must be wired to conform with the polarity shown in the schematic.

SOURCES OF MATERIALS

A good nearby source of materials is your neighborhood Radio Shack store. If you cannot find something there, check the Yellow Pages under Electronic Parts, Retail to see if there is a company in your area that can supply it. If you cannot find it locally or if you prefer mail order (usually you get better prices mail order and you don't pay state tax), you can get addresses of mail order firms from the back pages of magazines like Byte or Computers & Electronics. When buying mail order keep in mind that if a business has been around more than a few months, most likely they are not dishonest. However you cannot judge the service you will get nor the quality of the materials until you try them or hear from somebody that has experience with them. Based on our own experiences with them, the following mail order companies are above average in quality of service and merchandise:

JAMECO ELECTRONICS
1355 SHOREWAY ROAD
BELMONT, CA. 94002
(415- 592-8097

DIGITAL RESEARCH: PARTS
P.O. Box 401247
GARLAND, TX 75040
(214) 271-2461

CALIFORNIA DIGITAL
P.O. Box 3097B
TORRANCE, CA. 90503
(800) 421-5041

DIGI-KEY CORP.
HIGHWAY 32 SOUTH
THIEF RIVER FALLS, MN 56701
(800) 346-5144

WALLEN ELECTRONICS
108 SAW TELL AVE.
BROCKTON, MA 02402
(617) 588-6440

MOUSER ELECTRONICS
11433 WOODSIDE AVE.
SANTEE, CA 92071
(619) 449-2222

Most of these companies have catalogs. Write for your copy.

PC boards for the projects in this book are available from MICROSIGNAL. Write for information on prices and availability.

CHAPTER 2

IMPROVING THE 64

PROJECT **1**

USER PORT BREADBOARDING

Many projects in this book employ the USER PORT on the 64. Here is a breadboarding system that lets you experiment easily, neatly and safely with the USER PORT. All the important signals on the port are brought out to a solderless breadboard ('superstrip') so that you can build circuits and make connections using only insulated wire – no soldering is necessary.

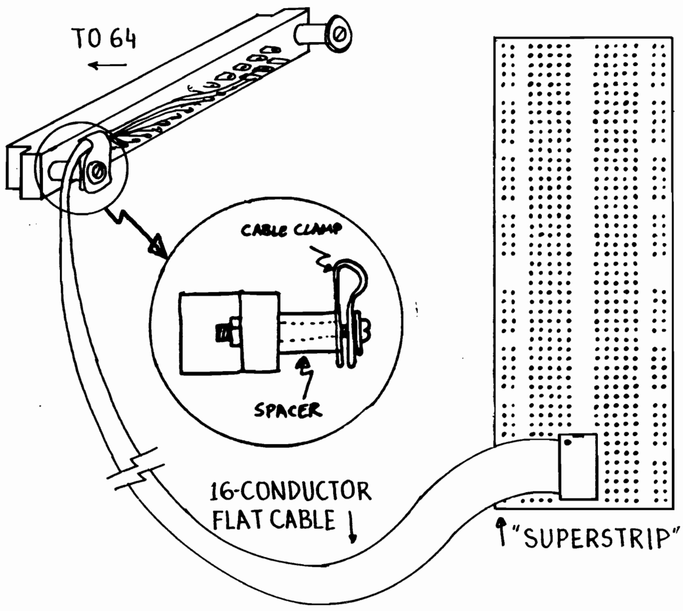
THEORY OF OPERATION

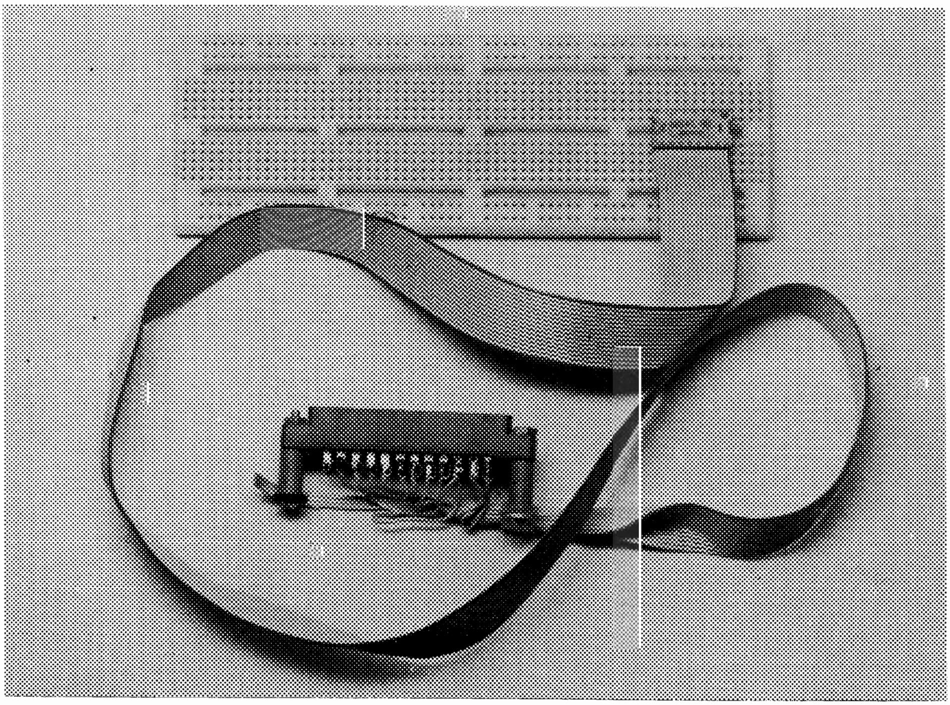
The idea behind the USER PORT breadboard is simple. Instead of having wires flying in all directions, here is a neat arrangement consisting of a connector to the USER PORT, a 16 conductor flat cable to bring the signals to a convenient location and a solderless breadboard to work on. Solderless breadboards like the 'superstrip' work very well and are excellent for experimenting. When you have a circuit that works to your satisfaction you can transfer it to a wirewrap implementation or a PC board, freeing your breadboard for the next job.

The signals brought to the connector are all the USER PORT signals, +5V, ground and RESET. RESET is brought out to be used as an output to reset peripheral circuits. See project 6 on how to properly reset the 64.

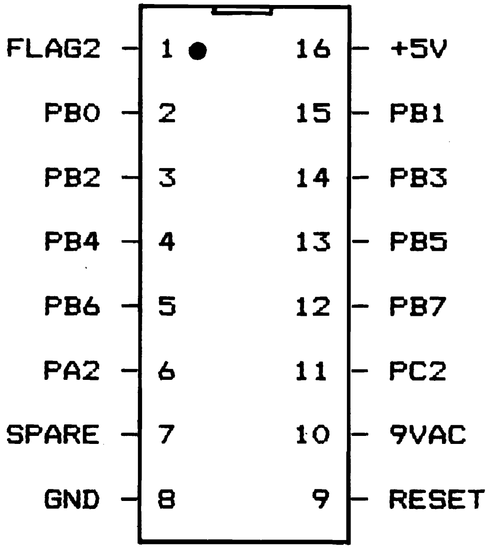
The pin arrangement on the connector was chosen to reduce crosstalk between the various signals. The basic philosophy is to group together signals that change at the same time.

There are two pins on the 16-pin connector that are not connected directly on the PC board. Instead they can be connected via jumpers if needed. Jumpering points A-B brings 9VAC to pin 10 of the DIP plug. Unless you have specific plans to use 9VAC in your circuit, do not connect this jumper. In breadboarding, accidental contacts happen and 9 VAC can be lethal to the 64 or to your LSTTL circuits.


TO 64
CABLE CLAMP
SPACER
16-CONDUCTOR FLAT CABLE
"SUPERSTRIP"



SIGNAL	USER PORT	DIP PLUG
FLAG2	PIN B	PIN 1
PB0	PIN C	PIN 2
PB2	PIN E	PIN 3
PB4	PIN H	PIN 4
PB6	PIN K	PIN 5
PA2	PIN M	PIN 6
SPARE	--	PIN 7
GND	PIN 1,A	PIN 8
RESET	PIN 3	PIN 9
9VAC	PIN 11	PIN 10
PC2	PIN 7	PIN 11
PB7	PIN L	PIN 12
PB5	PIN J	PIN 13
PB3	PIN F	PIN 14
PB1	PIN D	PIN 15
+5V	PIN 2	PIN 16



The distance between points A-B is 0.1″ so you can use a two pin header with a removable shorting plug. Installing the plug will supply the 9VAC and removing it will disconnect them without soldering.

The other free pin is a spare and can be connected via a piece of insulated wire to any of the signals that are left out.

The capacitor provides decoupling of the +5V supply at the exit point to reduce supply noise. Additional decoupling capacitors should be used on the superstrip.

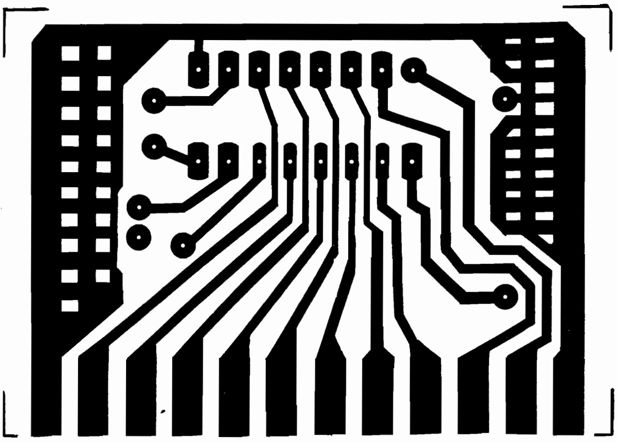
CONSTRUCTION HINTS

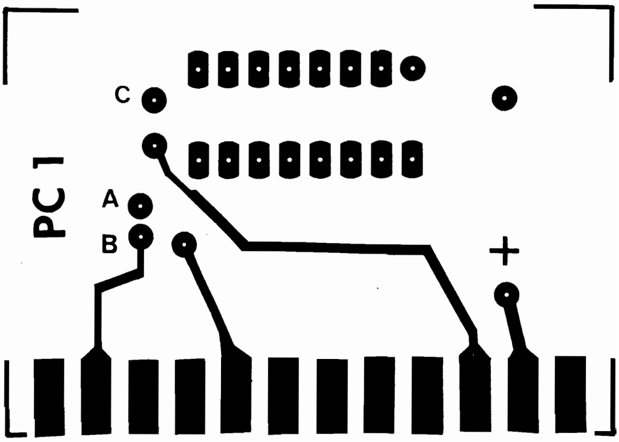
The breadboard system can be assembled without a PC board, as shown in the sketch and photo. However, the PC board is highly recommended here because it gives a very clean and sturdy assembly. When you are trying to debug a circuit the least you want is problems with the breadboarding set up itself.

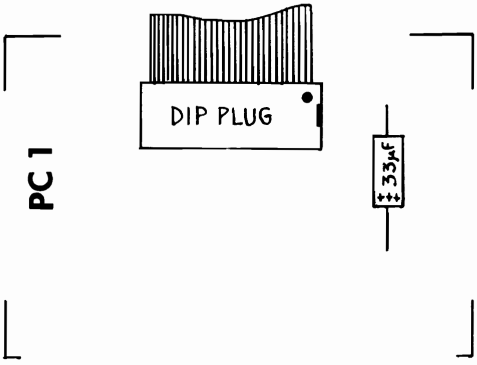
Be very careful not to bend the pins on the DIP plug assembly. Take these two precautions: 1. When not plugged in the socket of the PC board or in the superstrip, plug it in a piece of styrofoam and attach the foam with some tape so it cannot fall off. 2. Use an IC extractor tool to remove the DIP plug from the socket or superstrip. If you do not have such a tool, pry the plug out carefully using a small screwdriver.

PARTS LIST

1. PC board #PC1.
2. 16 pin solder tail socket. High quality socket recommended. Best sockets are the ones with gold plated screw machine contacts. Make sure the DIP plug pins fit in their contacts.
3. 33 uF/16V electrolytic capacitor, axial lead. Not critical, any value between 16 uF and 50 uF OK.
4. 12/24 pin, 0.156″ spacing PC connector, solder eyelet recommended (TRW CINCH 50-24A-30 or equiv.)
5. 16 conductor flat cable with DIP plugs attached to each end, up to 36″ long (Digi-Key R-116-24-ND for 24″ length or R116-36-ND for 36″ long).




PC 1
C
A
B
+


DIP PLUG
PC 1
33µF

PROJECT 2
EXTERNAL AUDIO AND VIDEO

The 64 will perform adequately when connected to a TV set, assuming the TV is in good condition. In order to get the best possible video and audio out of the 64 you will need direct connections to an external monitor and HI-FI amplifier. Here is how to do it safely and properly.

THEORY OF OPERATION

The 64 generates video and audio signals that are available through the DIN connector on its back. These signals are also fed into an RF modulator inside the 64 which converts them to a RF (Radio Frequency) signal similar to that broadcast by a TV station.

The RF signal is available at the RCA jack on the back of the 64. When connected to a TV set, it will be received, amplified, filtered and demodulated. Thus an estimate of the original video and audio signals will be obtained and it will be used to drive the CRT of the TV set to produce the picture.

This process of encoding and decoding is designed for broadcast video. When used on a digital video signal like the one produced by 64 it degrades it and it also introduces unnecessary distortion in the audio signal. Even when the TV set is brand new and properly adjusted the result leaves a lot to be desired. Of course most of the TV sets are at least a bit old and quite a bit removed from perfect alignment. Thus, direct video and audio connections are most desirable.

The main problems with direct connection are incompatibilities of impedance and voltage levels and the possibility of getting 110VAC (or worse) into you 64 (this kind of voltage can fry your 64 to crisp if it sneaks inside it).

To avoid the last problem you can use direct cabling and before inserting the cables, check for 110V grounding

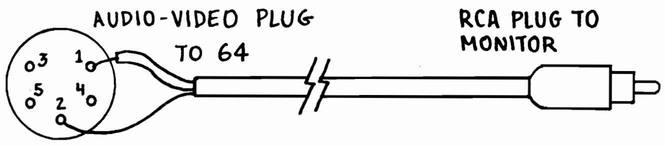


Figure 2.1 Black-and-White Monitor Connection

problems. Or, alternatively, you can build a foolproof system with proper isolation so that it will work under any conditions.

First the direct cabling. If you have a black-and-white monitor, you use only the LUMINANCE signal from the Audio-Video jack. If you have a color monitor like the Commodore 1701 which has both LUMA and CHROMA inputs, then you connect the LUMINANCE output of the 64 to the LUMA input and the VIDEO output to the CHROMA input of the monitor. The 1701 also has an AUDIO input to which the AUDIO OUT output if connected. See the schematic for the connections to the 1701 or similar monitor. All cabling must be shielded cable, preferably of the RF type with woven shield.

The commonly available RGB type of color monitor cannot be directly connected to the 64. A special decoder must be added to convert the video signals coming out of the 64 to the separate Red Green and Blue signals needed to drive such monitor.

The least expensive method of getting a video monitor for the 64 is to modify your color TV set so that you connect directly to its video and chrominance circuits. Unless you know exactly what you are doing this is risky business (the voltages within a color TV set can fry you in addition to the 64). No detailed how-to instructions can be given because the circuits within the color TV sets vary quite a bit. Also, if your TV set has a "hot chassis" it is best to forget any direct connections. It is just too much trouble to do it safely.

To connect to your stereo, you go from the AUDIO OUT pin on the Audio-Video connector to the AUX IN of the stereo. The sound output of the 64 is monophonic, so plug

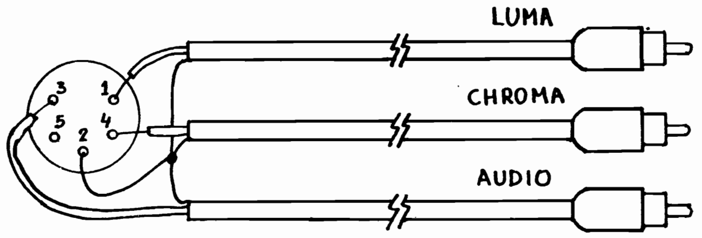


Figure 2.2 Commodore 1701 Color Monitor Connection

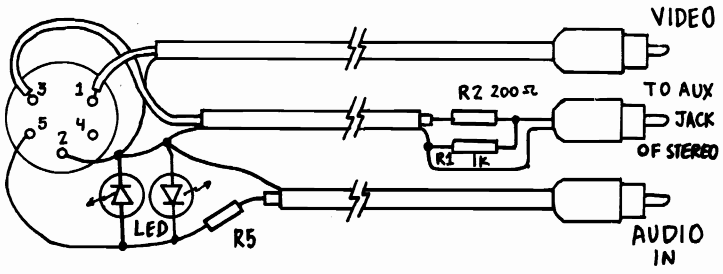


Figure 2.3 Black-and White Monitor Connection with Audio Out and Audio In.

in either channel and set the amplifier to MONO. The audio output of the 64 has a maximum of 2V p-p (peak to peak) level with an output impedance of about 100 Ohms. Thus there will be no problem from the impedance standpoint but you may have a bit too much volume for some stereos. To reduce the level at the auxiliary input you may use a voltage divider. R1 should be 1K and R2 should be selected so that 2×(R2/(R1+R2)) is equal to the maximum P-P voltage specified for the AUX input of your stereo.

The 64 can also receive audio input from external sources which it processes and adds to the output of the synthesizer. The audio input has an 100K impedance and 3V p-p maximum input level. Here we definitely need some protection to avoid excessive input levels. It is provided by the 1K resistor and the two LED's. The LED's draw no current when the voltage across them is less than 1.7 volts. When the voltage across them exceeds 1.7 volts they draw a lot of current so that the voltage drop across the 1K resistor prevents the voltage from rising at the 64 input. In effect, the LED's act as 1.7V zener diodes but with much better performance than a zener diode of the same voltage.

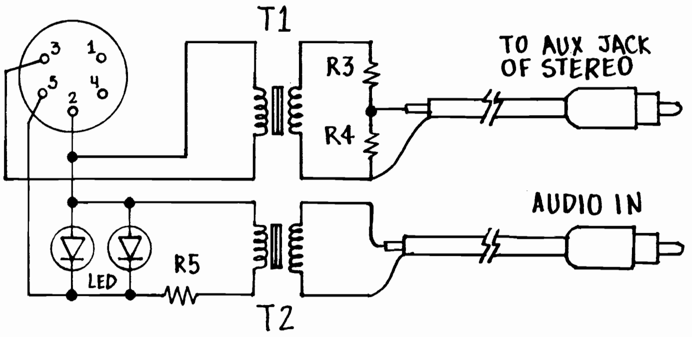


Figure 2.4 Transformer Coupled Audio Connections

When you construct the cable assembly, tape with insulating tape the exposed wires of the LED's and resistors. Or you may use heat shrinkable tubing instead of tape for a much neater assembly.

To test your set up for compatibility, plug your cable assembly into the DIN jack on the 64 and turn the power on to your 64, monitor and stereo. Using an AC voltmeter measure between the ground of each plug and the ground of the corresponding jack. If the indication on all connections is zero volts, you can plug in safely. If it is not, there is a grounding problem. If your stereo or monitor has a two prong power plug, reverse it and measure again. If you now get zero volts, you can plug in. But there is one caveat. If somebody reverses that power plug, you may be in for trouble.

The bulletproof way of connecting to your stereo is to use audio transformers, as shown in the schematic. Any miniature audio interstage transformer with impedances between 0.5K and 2K will work. R3+R4 should equal the impedance of the side of the transformer to which they are connected. Their ratio determines audio volume as explained above. The disadvantage of this method is that there may be some loss of audio quality with poor quality transformers.

PROJECT **3**

UNIVERSAL CASSETTE INTERFACE

If you already have a portable cassette recorder, you may be wondering whether it is possible to adapt it for use with your 64. The answer is a qualified "yes". Your tape recorder must have good high frequency response and it must be in good shape. You will have to construct an interface to connect it to your 64. And you will lose some of the conveniences of the Commodore Datasette. On the other hand, you will save the cost of a Datasette in addition to building a fun project.

THEORY OF OPERATION

Writing and reading digital information on a system designed for audio use is a tricky proposition. Some systems convert data into a string of audio tones, let's say 1200Hz for a logic "0" and 2400Hz for a logic "1". This audio signal (called FSK for Frequency Shift Keying) can be recorded on the cassette tape using an inexpensive portable cassette recorder. Later, when played back, the tones are converted to data for input to the computer. The problem with FSK is that it cannot tolerate speed variations in the recorder. If the playback speed is different from the recording speeds, the frequency of the tones changes and errors result.

Commodore has used a different approach. Instead of user supplied standard audio cassette recorders, they decided to employ modified audio cassette recorders and record the data digitally. They order the case and mechanism of an audio cassette recorder and they add their own electronics which are specified for digital use.

The recording technique they chose is PWM (Pulse Width Modulation) which works as follows: Logic "1" is represented by a pulse of length 0.75T followed by a low level of duration 0.25T. T is the period of the PWM signal and its actual value is not important in determining

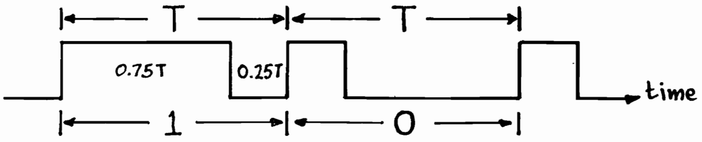


Figure 3.1. A PWM Signal

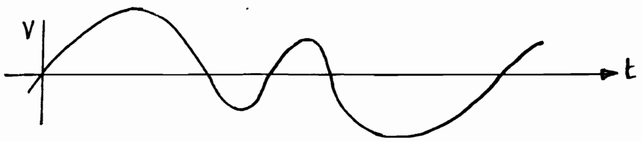


Figure 3.2 PWM Signal During Playback

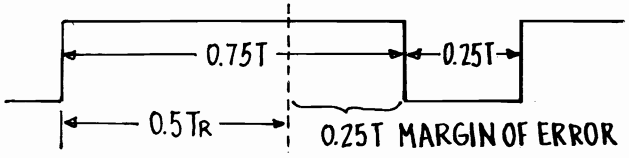


Figure 3.2 The Margin of Error for a PWM Detector that Looks at 0.5T.

whether we have a "0" or "1". Logic "0" is represented by a pulse of length 0.25T and a low level of length 0.75T as shown in Fig. 3.1.

The advantage of PWM in tape recording is its insensitivity to speed variations. If the playback speed changes, T will change but the relationships of 0.75T for an "1" and 0.25T for a "0" will not change. Even if the detector looks at 0.5 of the original T after the rising edge of the incoming signal to see if it has a "0" or an "1", the margin of error is 0.25T, quite large (see Fig. 3.2).

The PWM signal is generated by the 64 and recorded directly on the cassette tape. During playback the signal coming out of the recording head looks like the one shown in Fig. 3.3. The circuit inside the Datasette converts

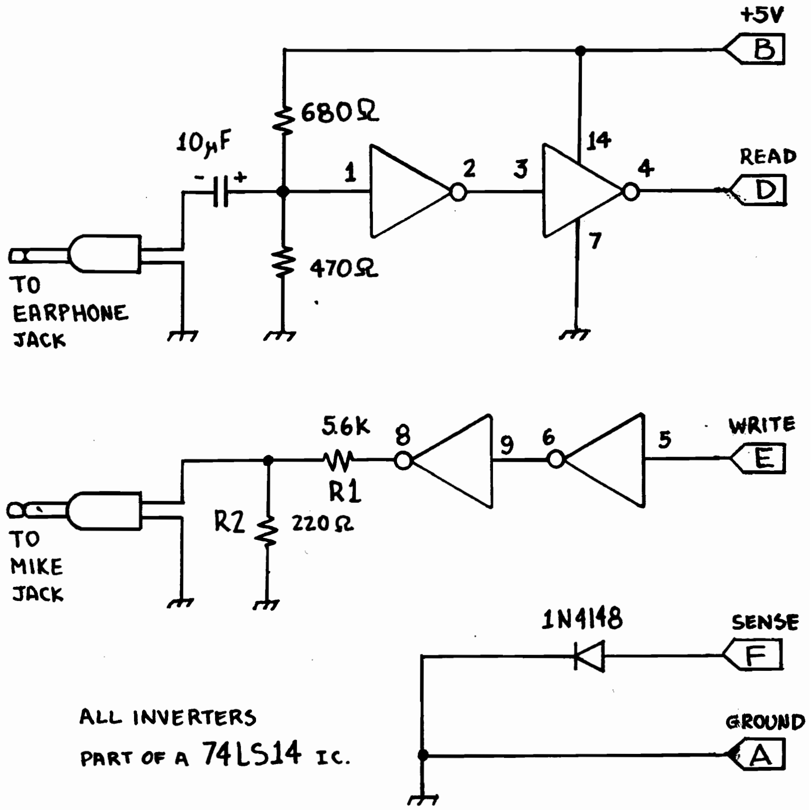


Figure 3.4 Cassette Adaptor Circuit 1

this signal to a digital PWM signal (hopefully identical to the original PWM signal) that is fed into the 64 and decoded into data.

In addition, the 64 controls the motor of the cassette recorder (turns it on and off as needed by supplying power to it) and senses if any of the buttons have been pressed.

It is not practical (nor desirable) to open up your cassette recorder and replace its electronics with a copy of Commodore's circuit. Instead, it is possible and relatively simple, to add a circuit external to the cassette recorder that allows it to read and write a PWM signal. The cassette recorder must be of good quality (most units costing over $40 are O.K.) and in good condition. If the recording head is worn from use, it will have poor high frequency response. A cassette deck, requiring an amplifier or receiver

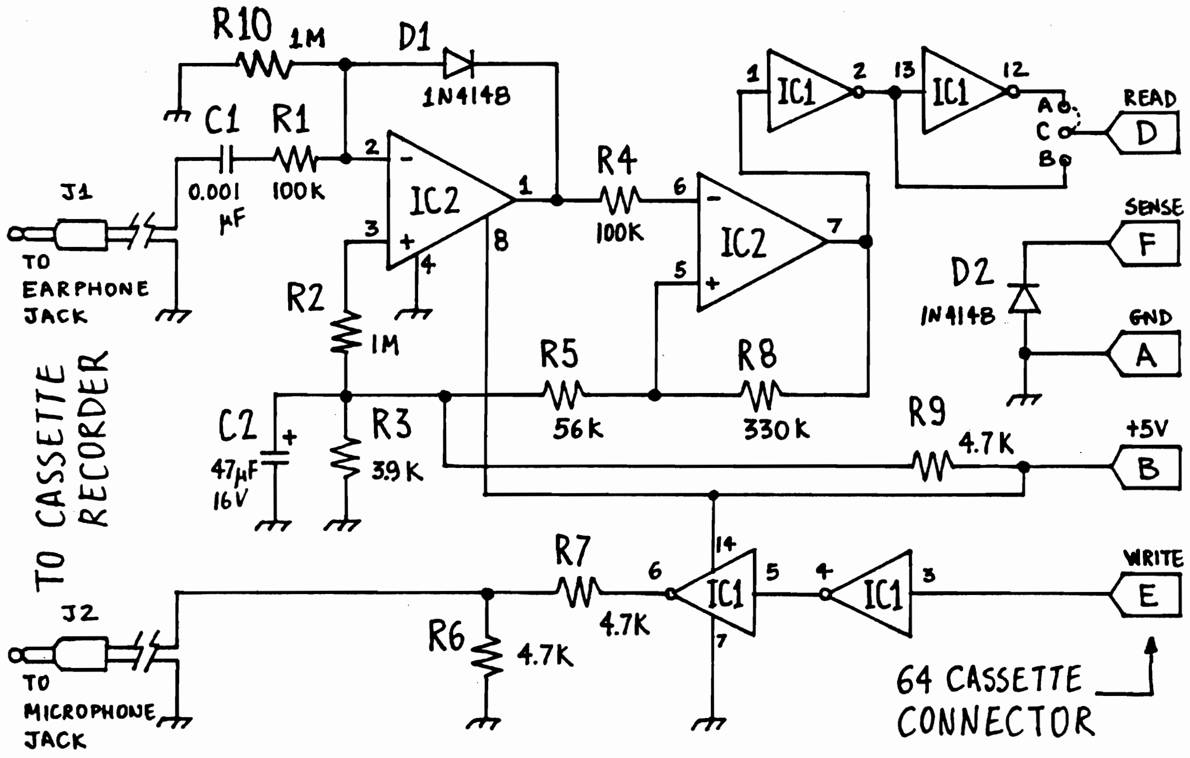


Figure 3.5 Cassette Adaptor Circuit 2

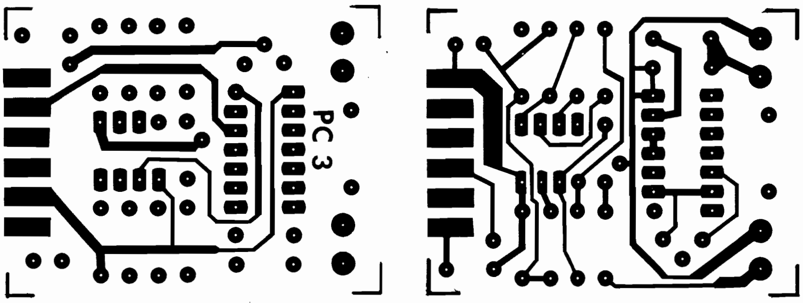
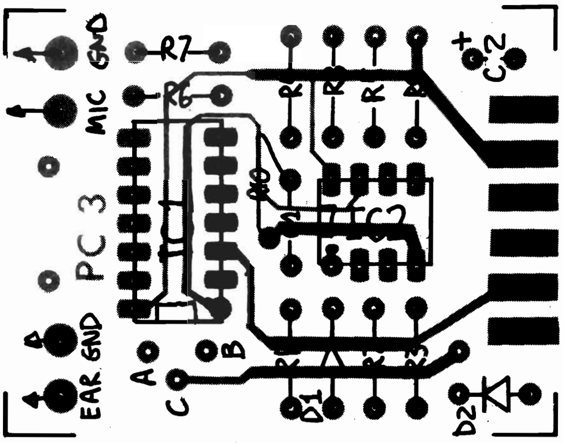


Figure 3.6 PC Board Layout for Circuit 2



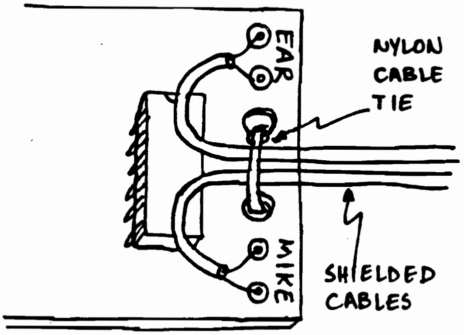


Figure 3.8. Construction Details

to play, will not work because its output is not sufficient to drive the circuits shown here.

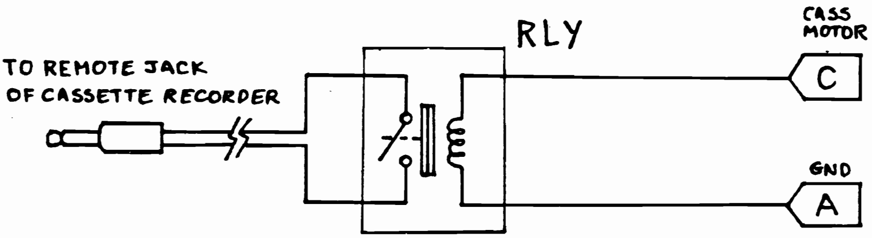
Some high quality stereo cassette recorders will work with the 64 cassette port directly, without interface. This is extremely poor practice and may result in degradation or in damage to the 64. Do not do it.

Two interfaces will be shown here. The first is simple and straightforward and it should work with most cassette recorders. The second is more complex and it should work even with difficult cases.

The first circuit is based on the 74LS14 Hex Schmitt Trigger IC. The audio from the earphone jack is coupled to two Schmitt triggers in series to square it up and to adjust its voltage and impedance levels for compatibility with the 64. Output can be taken either true or inverted to compensate for some recorders that invert the signal. In the record mode, the IC is used to buffer the signal feeding into the microphone jack. A voltage divider made up of R1 and R2 reduces the amplitude of the record signal to avoid overloading the cassette recorder input.

The second circuit is identical to the first for recording, with the exception that R1 and R2 are now equal at 4.7K. You may also try the same values as the first circuit. During playback, the signal is high-pass filtered by the 0.001 uF capacitor and the 100K resistor and amplified by A1. Diode D1 clamps the output to a minimum of about +1.3V during the negative swing. During the positive swing A1 is allowed to saturate. A2 is wired as a Schmitt trigger with high gain to square up the signal. Final squaring is done by the 74LS14 which also converts the signal to LSTTL specifications. As in the first circuit, both true and inverted outputs are available.

To adjust the playback volume, use a tape recorded in a Datasette (you can use the interface to record the tape if you cannot get such a tape). Experiment with various settings of the volume control to find the one that will give you reliable loading every time. With most recorders, this will be somewhere in the upper half of the volume control travel, perhaps at full volume. If there are any tone controls or switches set them for maximum high frequency



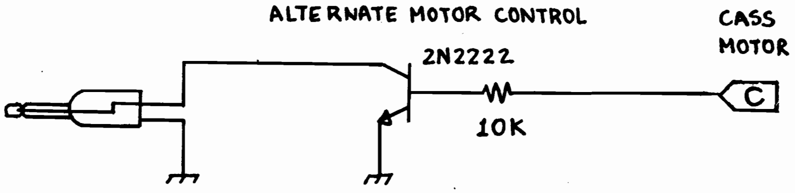


Figure 3.7. Cassette Motor Control Circuits

response. Once you have found the right setting of the volume control, mark it with a drop of paint or typewriter white-out liquid for future reference.

You may also need to experiment with record levels to get optimum recording. If you have trouble with recording (a program saved with the interface should load right away on a Datasette), try various values for R1 from 1K to 10K.

There is no way for the 64 to sense whether a button has been pushed on the recorder, so the SENSE input is wired low. If we connect it directly to the ground, the 64 will behave strangely during recording, so instead, it is grounded via a diode. This works well although it is not clear why.

Because SENSE is always activated, you will never get the messages "PRESS PLAY ON TAPE" or "PRESS RECORD & PLAY ON TAPE". Instead the 64 will start loading or saving as soon as you hit RETURN on the load or save statement. This presents no problem during loading, but before saving, you must make sure that the recorder is in the record mode, otherwise data will be lost.

Motor control of the cassette is accomplished by a 5V relay driven by the MOTOR output. The relay controls the recorder via the REMOTE jack available on most portable cassette recorders. An alternate simpler circuit can be used on cassette recorders that have a positive tip remote control. Most of the newer recorders fit this description. To make sure, plug a 2.5 mm plug into the remote control jack on the recorder and push the PLAY button. Check the polarity of the voltage of the center conductor (tip) with respect to the outer conductor (jacket). If the tip is positive you can use the alternate circuit and save the cost and bulk of the relay.

PARTS LIST, CIRCUIT 1

1. IC: 74LS14
2. Resistors: All 1/4W, 5%. 220, 270, 680 Ohm. 4.7K.
3. D1: 1N4148 or 1N914 diode.
4. C1: 10 uf/16V electrolytic.
5. Two 3.5 mm Jacks.
6. PC connector, 6/12 position, 0.156″ spacing.
7. Shielded audio cable, plastic cable tie.

PARTS LIST, CIRCUIT 2

1. IC1: SN74LS14N Hex Schmitt trigger. IC2: LM358
2. D1, D2: 1N4148 or 1N914 diodes
3. C1: 1000 pF mylar. C2: 47 uF/16V electrolytic
4. All resistors 1/4 W, 5%.
   R1, R4 = 100K, R2,R10 = 1M, R3 = 3.9K, R5 = 56K
   R6, R7 = 4.7K, R8 = 330K, R9 = 4.7K
5. J1, J2: 3.5 mm plugs to fit cassette recorder.
6. Connector: 6/12 position, 0.156″ spacing PC connector.
7. Shielded cable, plastic cable tie.

PROJECT **4**

PARALLEL PRINTER INTERFACE

Dot matrix printers like the EPSON RX-80 or the NEC 8020A with parallel interfaces are inexpensive and offer printing quality far better than the Commodore 1525. This project details a very simple interface to any parallel printer via the USER PORT. While limited in applications (it will not print the Commodore graphics symbols nor will it respond to PRINT # commands), it works well for program listings and general printing applications.

THEORY OF OPERATION

Most printers come standard with what is known as a 'Centronics Parallel Interface', named after a company EPSON almost put out of business. Centronics, a U.S. company, was the first to introduce and make popular the low cost dot matrix printer and their interface became the defacto industry standard (the NEC 8020A and the C. Itoh PROWRITER use the same interface and this project applies to them as well).

The Centronics interface is a handshaking 8-bit parallel interface that works as follows: The computer puts the data (ASCII code of character to be printed) out on the port and generates a low pulse on a line called 'STROBE' to indicate to the printer that valid data is available. The printer accepts the data and replies by pulsing low a line called ACKNLG (ACKNowLedGe). If the printer is busy no data is lost because it does not generate the acknowledge until it has accepted the data, and the computer waits for the ACKNLG pulse before proceeding. The timing diagram of the process is shown in Fig. 4.1.

This type of output with handshake suits the USER PORT perfectly. PB0-PB7 can carry the data, PC2 generates the strobe and FLAG2 senses the ACKNLG pulse. The hardware for the interface is a cable with connectors to plug to the USER PORT on one end and to the printer on the other.

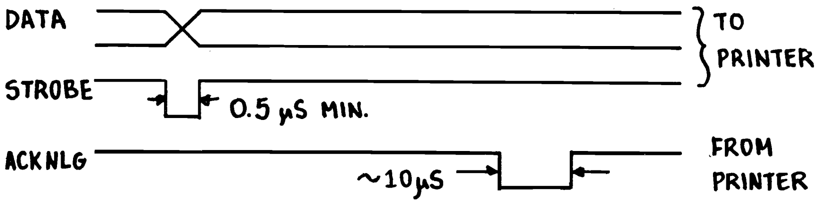


Figure 4.1 Timing Diagram of the Printer Handshake

CONSTRUCTION NOTES

The connector on the 64 side is the same type as the one in project 1, with the flat cable attached in identical manner. The cable can have a length of 3-4 feet. An 11-conductor flat cable will do nicely. However, this width is not normally available, so it will have to be made by separating a 14 or 16 conductor flat cable. Once started, the extra conductors will peel right off. The connector at the printer end is a CINCH 57-30360. The wiring of the cable follows the list shown. Double check the finished cable with an Ohmmeter to make sure the connections are wired properly.

PROGRAMMING

First the user port should be made an output port by a POKE 56579,255. Then FLAG2 should be enabled by a POKE 56589, PEEK (56589) OR 16. Finally a short machine language subroutine is put in memory starting at location 52992 ($CF00). The subroutine outputs the contents of register A and waits for an ACKNLG pulse. Then it goes to the screen printing routine.

By changing the output vector (in locations 806 and 807), which normally points to the screen print routine, so that it points to our machine language subroutine, all output routed to the screen will be printed on the printer first and then displayed on the screen as well. However, the printer will not print graphics which the screen can print. Instead you may get funny looking results, like font changes or overwrites. Also, the printer will not print lowercase letters because Commodore does not use standard ASCII codes. This could be remedied by changing the subroutine to detect lowercase codes and change them to standard ASCII.

COMPUTER		PRINTER	
SIGNAL	PIN	SIGNAL	PIN
PC2	8	STROBE	1
PB7	L	DATA 8	9
PB6	K	DATA 7	8
PB5	J	DATA 6	7
PB4	H	DATA 5	6
PB3	F	DATA 4	5
PB2	E	DATA 3	4
PB1	D	DATA 2	3
PBO	C	DATA 1	2
FLAG2	B	ACKLNG	10
GND	A	GROUND	19

Figure 4.2 Wiring List for the Printer Interface

```
1 REM ** PRINTER DRIVER **
2 REM
10 POKE 56579,255:POKE 56589,16
20 FORI=0TO14:READ A:POKE52992+I,A:NEXTI
30 POKE806,0:POKE807,207
40 DATA 72,141,1,221,173,13,221,41
50 DATA 16,240,249,104,76,202,241
```

The easiest way to set up the port and put the machine language subroutine in place is to enter and run the BASIC program given here. You could also append this program to your own programs and call it in the beginning to set up the printer. When it runs, it will enable the printer. To disable the printer, POKE 806,202: POKE 807,241. To enable the printer again, POKE 806,0: POKE 807,207.

Remember to set up your printer to execute a line feed every time it executs a return. Otherwise it will print all lines one on the top of the other because the 64 does not send line feeds to the screen. Instructions on how to set up the printer to do a line feed in every return are given in its manual. For most printers all it takes is changing a DIP switch.

PROJECT 5
EXPANSION PORT BUFFER

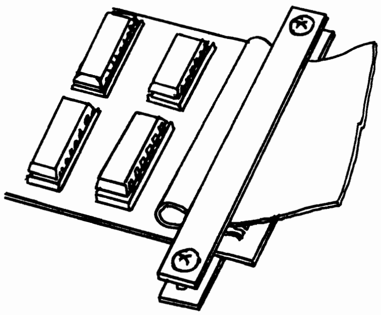
The expansion port is the 44-pin female connector recessed inside the back of the 64. It allows adding ROM cartridges containing software like games and up to 512 I/O ports. Apparently in an effort to reduce costs, Commodore did not buffer the lines coming to the port. No buffering, especially if you connect to the port more than a few cards (as it is possible with any of the commercially available expanders), can mean erratic operation and lost data. If you plan to use the expansion port and you want to do it right, here is how to do it.

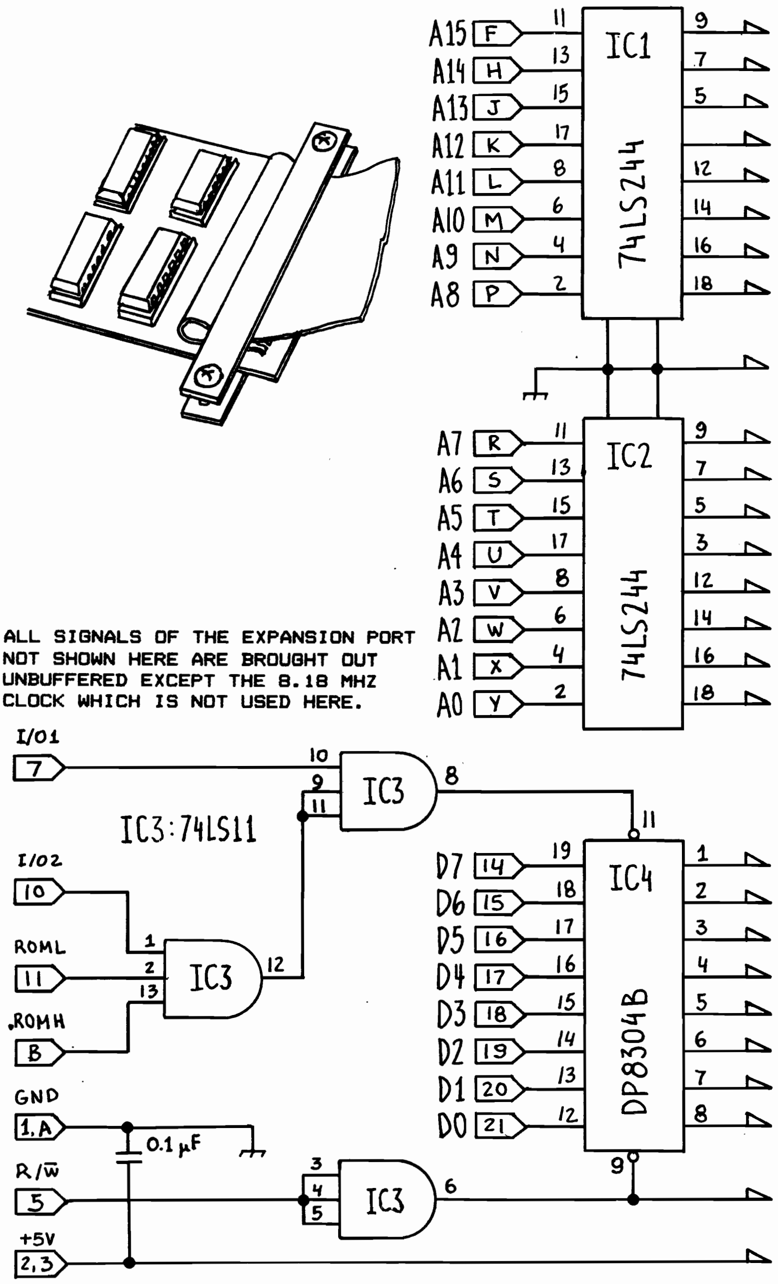
THEORY OF OPERATION

The data and address lines coming to the expansion port on the 64 are directly driven by the 6510 CPU. The drive capability of the 6510 is limited (one TTL load) and internal circuitry already uses some of this capacity. In fact, Commodore specifies one LSTTL load as the maximum allowable loading on the expansion port lines. Unless buffering is used to increase the drive capability of the port, the danger exists that operation of a system with more than a few cards connected to the expansion port will be marginal, giving rise to soft (noise generated and not repeatable) errors. Of course, with heavy loading, the system will stop.

An address and data buffer can be made with only four IC's. It will guarantee correct operation with up to 200 LSTTL loads, with the added benefit of providing a physical barrier to external voltages in case of improper connections (if something goes wrong, chances are the buffer IC's will be destroyed leaving the 64 intact).

Looking at the schematic, two LS244 octal buffers are used to buffer the address lines. The R/W signal is buffered by one of the AND gates. The data bus buffer is bidirectional and its direction is switched by the R/W sig-





ALL SIGNALS OF THE EXPANSION PORT NOT SHOWN HERE ARE BROUGHT OUT UNBUFFERED EXCEPT THE 8.18 MHZ CLOCK WHICH IS NOT USED HERE.

nal. It is enabled only when external access is requested (I/O1, I/O2, ROML or ROMH signals active) so there are no conflicts when accessing locations internal to the 64.

The +5V supply from the 64 is also brought out to be used by peripherals. Be careful not to draw too much current from this line. Commodore specifies 450mA maximum allowable current draw from this supply when nothing is connected to the USER PORT. In order to play it safe, if your peripherals connected to the expansion port draw more than 300 mA, use an external power supply to power them.

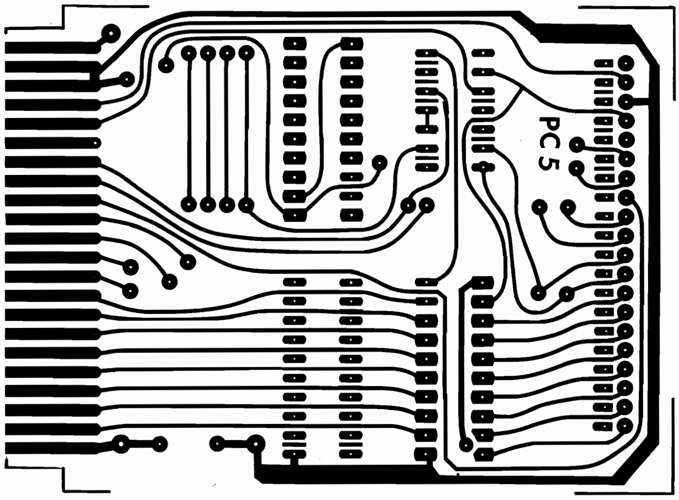
All other lines to the expansion port should also be brought out to be used during expansion. This is not shown in the schematic to avoid clutter.

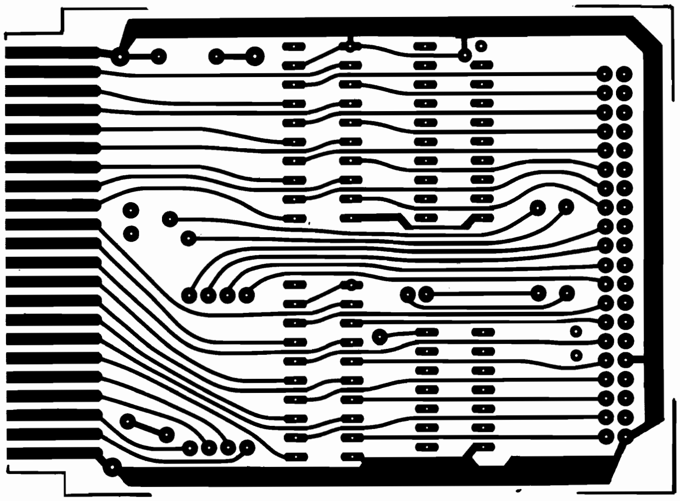
CONSTRUCTION NOTES

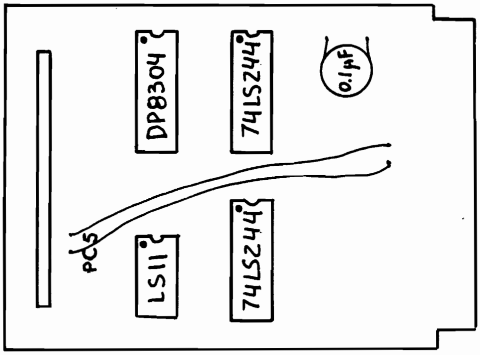
The best way to implement the buffer is by a PC board. The layout given is for a two sided PC board with plated through holes. It is designed to be connected with an expansion chassis using a piece of 40-conductor flat cable, up to 24″ long. Although the connector on the 64 has 44 pins, a 40-pin cable and connectors were used for the expansion because this is the closest size available. The next size up, 50 pins, was not needed because in the expansion port there are duplicate power and ground lines which were omitted. Also omitted was the 8.18 MHz dot clock which is not used in any of the projects here, and which, if included, would radiate unacceptable RF interference.

The other end of the flat cable can lead to a card cage with standard 44-pin, 0.156″ spacing connectors (which are easier to find, cheaper and easier to work with). In fact, such a system can use standard breadboard cards easily available from Vector Electronics and many other sources. For convenience in reading the schematics, it is a good idea to maintain the same pin arrangement as in the 64.

Using an external card cage in a good enclosure will result in an interfacing facility that looks professional and works as good as it looks. Any project in this book that is originally designed to be connected to the USER PORT can


PC 5





be easily changed to work with ports implemented on cards in the card cage.

There is a disadvantage in using a card cage with connectors different than the ones in the 64 expansion port. Commercially available peripherals and game cartridges designed to plug in the expansion port connector will not be compatible with the connectors in the card cage. There are two ways to remedy this problem, One is to add some 64-compatible connectors in your card cage. The other is to build or purchase on the commercially available expansion busses that have no buffering and plug into the expansion port, offering a few expansion slots. You will plug the expansion port buffer you will build in the last slot and use the others for your store bought equipment.

The flat cable can be either soldered directly on the PC board (make sure you provide strain relief as shown in the sketch) or you may solder a header on the board and use a ready made flat cable with connectors on both end to plug into the header. This second method increases the cost but results in faster (you will have to strip and tin 40 wires if you use the first method) and better assembly.

PARTS LIST

1. Two 74LS244N octal buffers.
2. One DP8304B or 74LS245 octal bus transceiver
3. 74S11 triple 3-input AND gate
4. 0.1 uF disk ceramic capacitor
5. PC board, sockets, 40 connector flat cable with connectors as needed (see Construction Notes).

PROJECT **6**

RESET SWITCH

There are times when the 64 stubbornly refuses to respond to the keyboard, including the RUN/STOP-RESTORE key conbination. When this happens, it is time to press RESET or cycle the power switch OFF-ON. Alas, the memory contents of the 64 cannot survive a power switch cycle and you lose your program and data. And you will look in vain for the RESET switch. The 64 doesn't have one. So if you would like the convenience of a RESET switch, you must install it yourself.

THEORY OF OPERATION

The 6510 (and most other microprocessors) must be started at a known internal state after power up. To accomplish this, the 6510 CPU has a pin called RESET which, when grounded, brings the CPU to the initial state. The RESTORE function available in the 64 is different from RESET in that it activates the NMI (Non Maskable Interrupt) line. But, if a program gets lost and scrambles important memory locations, the NMI sequence will not be able to restore operation. Thus the 64 "hangs up". It is also possible for the 6510 to enter a "hung" state during which it performs no operations and does not respond to NMI.

When the 64 becomes unresponsive to the keyboard, the only thing you could do is turn it off and then on. This will destroy the programs in memory and any data you might have entered. It is particularly frustrating to have to do this after you enter a lot of data or after you type in a long program.

A RESET switch will enable you to restart your 64 without altering anything in its memory. The expansion connector has a pin labeled RESET, so you might be tempted to ground this with a push button switch to reset the 64. In fact there is in the market at least one expansion board that does just that. While this might work most of the time, it is not good practice. The reset pin on the expansion connector connects directly to the reset pin on the 6510. If you ground it directly, you will reset the 6510 but due to switch bounce you might get one or more additional incomplete resets resulting in erratic operation.

To do it right, you must either replicate the circuit used inside the 64 as shown in the schematic or use that circuit directly. Adding a reset switch by soldering inside the 64 will void your warranty, so wait until it has expired before you do it. You will not have to wait long!

CONSTRUCTION NOTES

Open the case of the 64 and locate the 556 timer IC at the lower left hand corner of the PC board. There are at least two versions of the 64 reset circuit. One is shown in the schematic in the back of the Programmer's Reference Guide (first edition). The other is found in the 64 used to write this book. It was purchased in June 1983 and presumably it is in all 64's manufactured after that date and perhaps many moths earlier.

In order to determine which circuit is in your own 64, measure (using the lowest Ohm scale in your VOM) the resistance between pin 9 of the 556 and pin 3 of the USER PORT. If you get a short (direct connection) you have the newer circuit.

In either case, the switch is mounted on the upper left side of the keyboard. Twist together two 22 gage stranded insulated wires, about 6" long. Solder very carefully one to pin 8 of the 556 and the other to pin 14 of the same IC if you have the newer circuit. If you have the older circuit, connect the second wire to pin 7 of the 556.

Solder the other ends to the terminals of a good quality miniature pushbutton switch. Then drill a hole on the left side of the keyboard to mount the switch. Use a plastic internal cap from a spray can as shown in the sketch to protect the switch from accidental damage.

PROGRAMMING NOTES

The reset switch will not erase any BASIC program in memory but it will alter two pointers: 1. The link field in

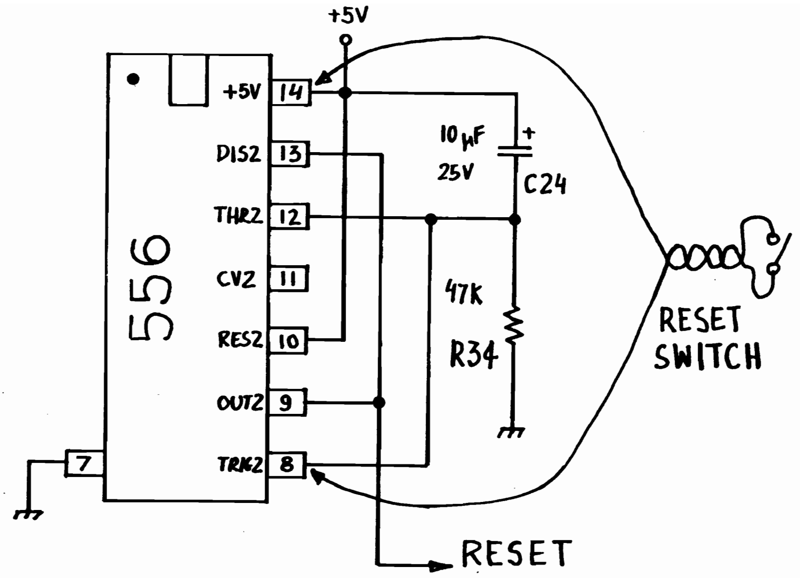


Figure 6.1 Reset Circuit of Later Model 64. The RESET Switch is Added Between Pins 14 and 8

the first line of BASIC 2. The pointer to the end of the program. After reset these pointers must be restored to their original values. Otherwise, the 64 will not recognize the program in memory.

If you know what these pointers are (let's say be PEEKing their value and writing it down before the crash) you could POKE their values after the reset and restoration would be complete.

The link pointer is in locations 2049 and 2050 ($801 and $802). Location 2048 ($800) must be set to zero for the 64 to recognize the BASIC program. The end-of-program pointer is at locations 45 and 46 ($2D and $2E).

The 64 is likely to hang up only when you run a program and not during editing or listing. Thus, it is best to be prepared by finding out what are the contents of the link pointer and of the end-of-program pointer before you run a program. Thus, if the program crashes and the 64 hangs up so that you have to reset it, you will be able to restore the pointers and get back your BASIC program.

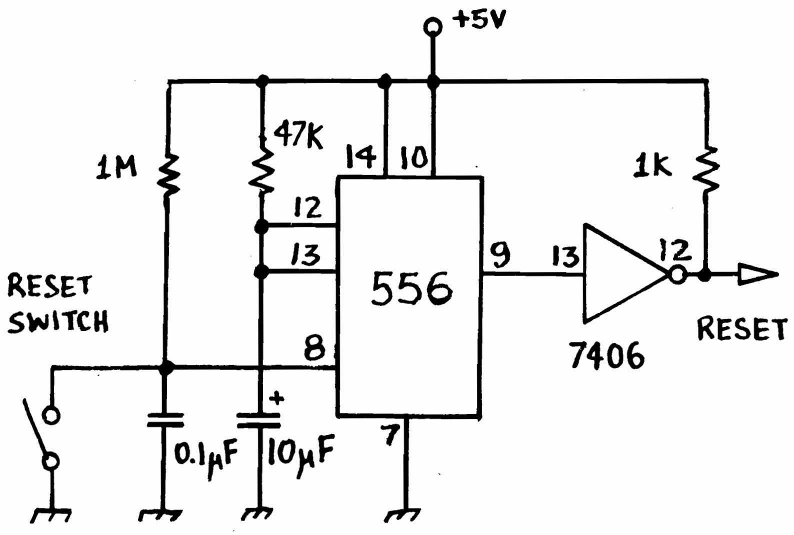


Figure 6.2 Reset Circuit of Earlier Model 64. The RESET Switch is Added Between pins 8 and 7

In order to simplify things, you can start all your programs in a standard way so that the link pointer will always be the same. This is done by using 1REM (without any spaces) as the first statement in your programs. Adding this statement to a program will not alter its function in any way and it will result in the following link pointer: Location 2048 will contain 0, location 2049 will contain 7 and location 2050 will contain 8.

Another trick is to use the contents of locations 45 and 46 (end-of-program-pointer) as part of the name under which the program is saved. For example, ACCOUNTS 25.68. The "." is used as separator, 25 goes to location 45 and 68 goes to location 46. This mnemonic trick also gives you an indication of how recent is the version of the program. During development, program length is likely to increase, so names with a larger suffix are more recent.

Let us work an example to see how we can ressurect a BASIC program after RESET.

1. Enter the following BASIC program as example. Type it in exactly as shown. Do not insert any extra spaces. If you do, the number in location 45 will be different from the one shown.

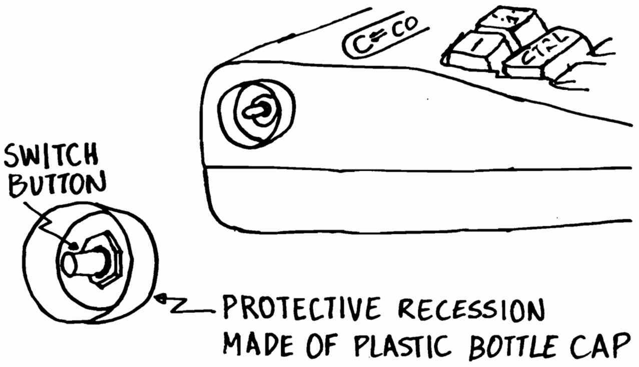


Figure 6.3. Construction Detail of the Reset Switch

```
  1REM
100PRINT"HELLO"
110A$="THIS IS A TEST"
120PRINTA$
130END
```

2. Find the contents of locations 45 and 46 by:

```
PRINT PEEK(45),PEEK(46)
```

You should get 60 8. If you were to save this program under the name TEST using our trick, its full name for saving would be TEST60.8.

3. Press the RESET button momentarily. The screen of the 64 will blank out and after a few seconds it will display the power-up message (**** COMMODORE 64 BASIC V2 **** etc.).

4. Try to LIST the program. You will get nothing. The program is not lost though. It hibernates and is ready to come back to life when you restore its pointers.

5. Do the following POKEs:

```
POKE 2048,0: POKE 2049,7: POKE 2050,8
POKE 45,60: POKE 46,8
```

6. Initialize the BASIC pointers with the CLR command. This is not usually necessary. Your BASIC program has now been restored and you can LIST it or SAVE it or RUN it as if nothing happened. Sure beats having to retype a long program!

PROJECT 7
LIGHT PEN

A light pen lets you input data by pointing to the screen of you computer. Let's say you have a menu of 5 items on the screen. With a light pen you could select one of the 5 options by pointing to your selection without even touching the keyboard. The 64 has built-in circuitry to detect the light pen signal, so all you need to add is the pen itself, which, in effect is a light sensor.

THEORY OF OPERATION

The image on the screen is made up of lines that scan from left to right. Each line takes approximately 65 us to complete and the whole screen is scanned in about 15 ms. Thus, an individual point on the screen will be emitting light only for a brief period (less than 0.5 us) every 15 ms. If we place a photodetector on the screen (small enough and fast enough to detect light from one scan line only), we will get a pulse every time the beam scans past that point and light is emitted.

We can find the horizontal position of the photodetector by measuring the elapsed time from the beginning of the horizontal scan that resulted in the output pulse to the occurrence of the pulse. Similarly, we can find the vertical position by counting the number of horizontal scans (starting from the top of the screen) that have occured before the output pulse.

The 64 contains all the required timing and counting circuitry necessary for light pen operation. All you need to add is the photodetector that generates the output pulse at the appropriate time.

This can be rather difficult or rather easy depending on the performance you require. A light pen of good contruction that has a small enough aperture to detect a single scan line and is fast enough to give accurate readings during horizontal measurement costs over $100.

If you forego both fast operation and small aperture, you can build a light pen that will give reliable performance with a vertical accuracy of about 20 positions for less than a couple of dollars! Surprisingly, this capability is quite sufficient for most applications. You will not be able to do high resolution x-y doodling with it but chances are you won't need to either.

The circuit is straight forward. The current generated by the phototransistor is amplified by Q1 and the output is a logic low level every time sufficient light enters the phototransistor (you may have to increase the intensity of your monitor or TV to get good performance). The output of the pen connects directly to the LIGHT PEN input of the 64. This input is pin 6 of CONTROL PORT 1, on the right side of the 64. Power (+5V) and ground are also taken from this port.

The phototransistor should have a built-in lens and high sensitivity. Switch S1 is used to signal to the 64 that the pen is in position and ready to take a measurement. It can be mounted on the pen or better yet, incorporated in the mechanical design of the pen so that when you press the pen against the screen at the desired spot, it is activated. You can totally omit the switch and use a key in the keyboard to signal to the 64 to take a reading. You can detect if the key is pressed by the GET statement.

CONSTRUCTION NOTES

Construction of the probe can be as simple or as elaborate as you desire. The prototype constructed to test this project is made of the barrel of a Papermate ball point pen (because it fits the Radio Shack phototransistor used) and it has no switch. This is adequate and perhaps a good starting point for experimentation. After you build and use this simple light pen you might experiment with more intricate mechanical designs. In any case, the parts cost is so low you can build several versions of the light pen and keep the one you like best, giving the others as gifts to friends.

The LIGHT PEN is available at both the user port (pin 7) and at the game I/O connector (pin 6). Either connector can be used. When you use the light pen you will not be

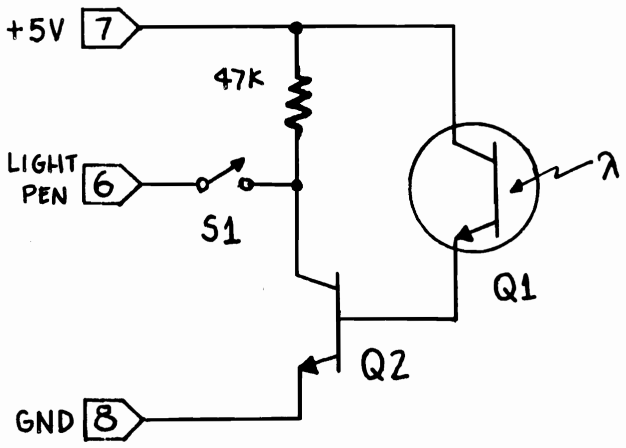


Figure 7.1. Light Pen Schematic

able to use the joystick simultaneously because the firing button of the joystick connects to the same input used by the light pen.

If you wish to incorporate a push-to-activate switch in your design, you may want to consider this approach: The transistors are mounted at the end of a rigid element, for example a wooden dowel .125" dia. a few inches long. The other end of the dowel is mounted on the push button of a push-to-activate switch using glue. The whole assembly is mounted in a tube, so that the lens of the phototransistor protrudes from one end of the switch and there is enough travel to activate the switch when pushed against the screen. Refer to the sketch for more details.

You can scrounge mechanical parts from various ball point pens and markers. The mechanical quality of the resulting pen will depend on your ingenuity in fitting and adapting parts, on the quality of the workmanship and, last but not least, the quality of the switch. A good switch that activates with a "click" sound will give you the best performance.

Finally, you will need some means to grasp the connector to unplug it from the 64. The plastic hood made for the connector is preferred for this but you can make a satisfactory handle using spacers and washers as shown in the sketch.

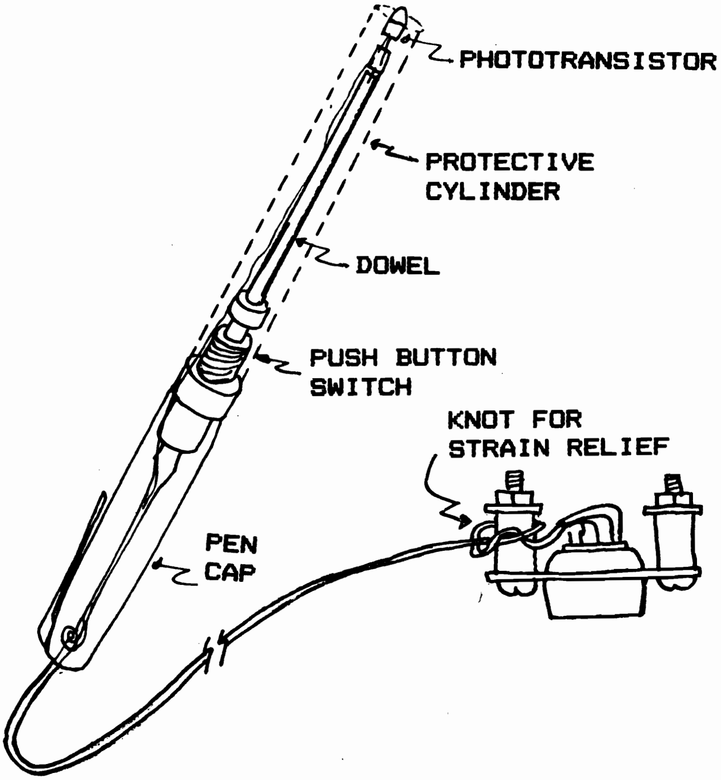


Figure 7.2. Construction Detail of a Push-to-Activate Light Pen

PROGRAMMING NOTES

The X coordinate of the light pen is stored at location 53267 and the Y coordinate at location 53268. To obtain the coordinates, just PEEK these locations. When switch S1 is off you will get either a default value (zero) or the previous value. In order to detect closure of S1 (indicating that it is time to take a reading) your program must look for a change in the coordinate value. Here is a program that prints out the Y coordinate of the pen every time S1 is closed:

```
10 A=PEEK(53268):A=PEEK(53268)
20 IF PEEK(53268)=A GOTO 10
30 PRINT A : GOTO 10
```

```
100 REM  ** LIGHTPEN TEST **
101 REM
110 PRINT CHR$(147)
120 FOR I=1 TO 10
130 PRINT CHR$(18);"      ";CHR$(146)
140 PRINT: NEXT I
150 FOR J=1 TO 3
160 FOR I=1 TO 40
170 PRINT CHR$(18);" ";CHR$(146);" ";
180 NEXT I
190 NEXT J
200 X=PEEK(53267): Y=PEEK(53268)
210 PRINT CHR$(19); TAB(22);
220 PRINT"                "
230 PRINT CHR$(19); TAB(20);
240 PRINT"X=";X;"  Y=";Y
250 GOTO 200
```

Line 10 is used to stabilize the reading, however it may not be much good. Another approach would be to take a few readings and use their average as the actual reading.

If switch S1 is omitted from the design, you can substitute for it any key of the keyboard which you press when you want a reading. Here is a program for this:

```
10 GET A$ : IF A$ = "" GOTO 10
20 A=PEEK(53268) : PRINT PEEK(53268)
30 GOTO 10
```

To check the resolution and stability of the light pen in both coordinates, use the program LIGHT PEN TEST. It creates a test pattern of horizontal and vertical bars and constantly gives you a reading of the X and Y coordinates. As you move the light pen across the pattern, you will be getting different readings corresponding to its position in the pattern.

PARTS LIST

1. Q1: Sensitive phototransistor with built-in lens, NPN type. See text for more details. Radio shack models will work O.K.
2. Q2: Silicon NPN transistor, 2N2222, MPS6566 or equivalent.
3. Resistor: 47K, 5%,1/4W or less.
4. Connector: D type, 9-pin socket. Radio Shack part# 276-1538 or equivalent. Hood (optional) part# 276-1539.
5. About 3 feet of 3-wire flat cable, mechanical parts according to your design.

PROJECT 8
RS-232C INTERFACE

The RS-232C standard is the most common interface standard between computers and peripherals. Your 64 contains software to implement this universal communications channel. Add a couple of IC's and it can start talking to peripherals such as modems, printers and plotters without any further interfacing.

THEORY OF OPERATION

The RS-232C standard defines the voltage and impedance levels for the transmission of digital data between a computer and a modem. It is a serial data bus where logic "0" is represented by a voltage between 5 and 15 volts and logic "1" is represented by a voltage in the range of −5 to −15 volts.

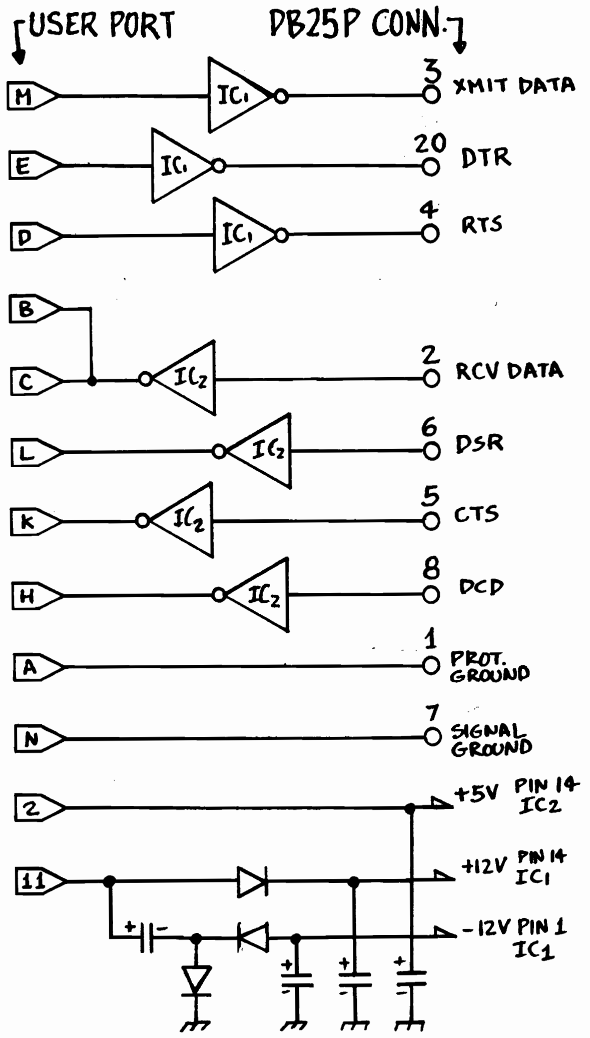
The 64 offers software support for the standard but does not provide the required voltages and connector. In addition, there are certain shortcomings in the 64 implementation, as follows:

1. The USER PORT is also used for the signals intended for the RS-232 interface. This means that if you use the inteface, you will not be able to use the USER PORT for anything else.

2. When the RS-232 interface is active, you will not be able to use the serial bus that connects to the disk drives and the Commodore printer and you will not be able to use the cassette.

3. Only a portion of the RS-232 interface signals is implemented in the 64 software. While these will be sufficient for most applications, there will be instances that they are not enough.

4. If you program in BASIC, the highest baud rate you can use is 2400. The highest baud rate allowable by the RS-232 standard is 19600 baud.



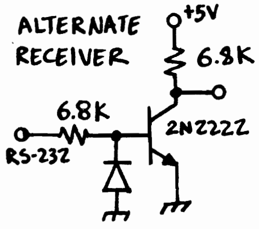
ALL CAPACITORS : 100 µF/25V

ALL DIODES : 1N4001

IC1 : MC1488 or SN75188N

IC2 : MC1489 or SN75189N

PINS 2,5,9,12 of IC2 MUST BE TIED VIA INDIVIDUAL 12K RESISTORS TO +5V.



On the plus side, the implementation offers software defined parameters (such as baud rate), 256 character buffers for both receive and transmit and it is integrated in the BASIC operating system so that it can accept commands like LIST and PRINT#.

The RS-232C interface uses a quad driver IC (Motorola MC 1488 or Texas Instruments SN75188N) to translate 64's voltage levels to RS-232C levels. This IC requires a ±12V supply which is generated from the 9VAC connection on the USER PORT.

The receiver can be another IC (MC1489 or SN75189N) or it can be built using a transistor and a few other components, as shown in the schematic.

Most modems and some printers will work with only the data signals and ground (the so called 3-line interface) without any additional control signals. However, in some instances other pins should be connected together to give default values to some control signals. These connections are usually referred to as the "null modem". It is a good idea to include it in your 3-line interfaces (which will become in reality four line) because it allows operation with a wider range of equipment at virtually no additional circuitry.

Building the interface is not at all difficult and any construction method will work well. The RS-232C specifies a maximum cable length of 50 feet. However, unless you have a specific need for a long cable, keep it short (5-10 feet) for ease of handling.

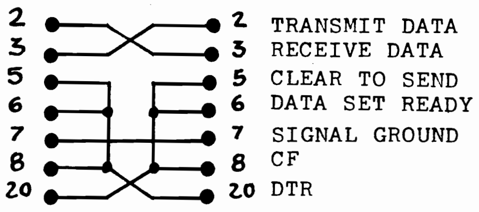
PROGRAMMING NOTES

The Programmers Reference Guide by Commodore is necessary reading if you plan to write your own programs for the interface. Here we will just outline some information on how to use the interface with BASIC.

First, you must OPEN device 2 which is the RS-232 interface. The OPEN statement must also specify signalling speed, word length parity and type of handshake. For example, let's say that we want a 300 baud channel, use 8-bit word length, no parity and no handshake (3-line interface).

```
100 REM TERMINAL PROGRAM
105 REM
110 OPEN 2,2,3,CHR$(6+32)+CHR$(32+128)
120 GETK$: IFK$="" THEN 170
130 IF K$=CHR(147) THEN 210
140 K=ASC(K$)AND127
150 IF K=20 THEN PRINT#2,CHR$(8): GOTO170
160 IFK>31 OR K=13 THEN PRINT#2,CHR$(K)
170 GET#2,K$: IF K$="" GOTO 120
180 K=ASC(K$) AND 127
190 IF K=8 THEN PRINT CHR$(20): GOTO 120
200 IF K=13 OR K>31 THEN PRINT CHR$(K)
210 CLOSE 2 : END
```

NULL MODEM CONNECTION FOR USE WITH 3-LINE IMPLEM.



The OPEN statement will be:

OPEN 2,2,3CHR$(6)+CHR$(0)

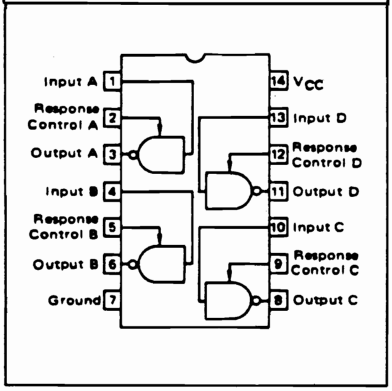
To send data over the channel use PRINT#2, for example:

PRINT#2,"TEST"

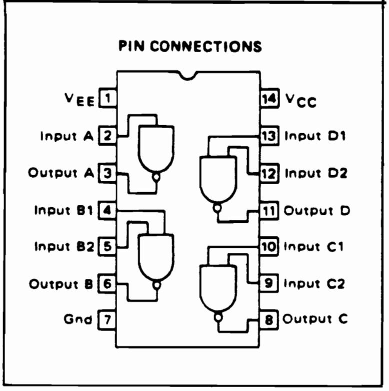
To receive data you can use INPUT#2. However, there is the chance of hanging up the 64 if RETURN is not received at the end of the line. The INPUT# statement must receive a RETURN to terminate and some RS-232 devices might send a line feed instead. As an alternative, you may use the GET# statement. If you do, use it often enough so that you unload the buffer before it gets a chance to overflow. If the receive buffer overflows, data will be lost.

When you open the RS-232 channel, the 64 immediately reserves two 256-byte buffers at the top of memory. It does not check to see if it is OK to do so, it just does it. Therefore, there is a danger it will overwrite strings or data

MC1489L
MC1489AL



MC1488



stored there. Thus, the OPEN statement must precede all DIM statements and if possible, be executed in the very beginning of the program. When you no longer need the channel, make sure that you close it with a CLOSE statement.

When you open the RS-232 channel, the BASIC variable ST loses all previous meanings and becomes solely an error flag for the channel. If its value is not zero, there is an error. Each time it is read, it is cleared to zero. For the meaning of the various error codes see the Programmers Reference Guide.

As an example of the use of the RS-232 interface, a very simple terminal program is presented written in BASIC. The program opens the RS-232 channel and looks for input at the 64 keyboard and at the channel alternatingly. Input from the keyboard is transmitted and input over the channel is displayed on the screen. To exit the terminal program, press the CLEAR SCREEN key. The OPEN statement assumes 300 baud, seven data bits and mark parity. The modified ASCII code in the 64 is not corrected so only uppercase letters are transmitted and the received data will print as lowercase.

PROJECT **9**

ADDING A USER PORT

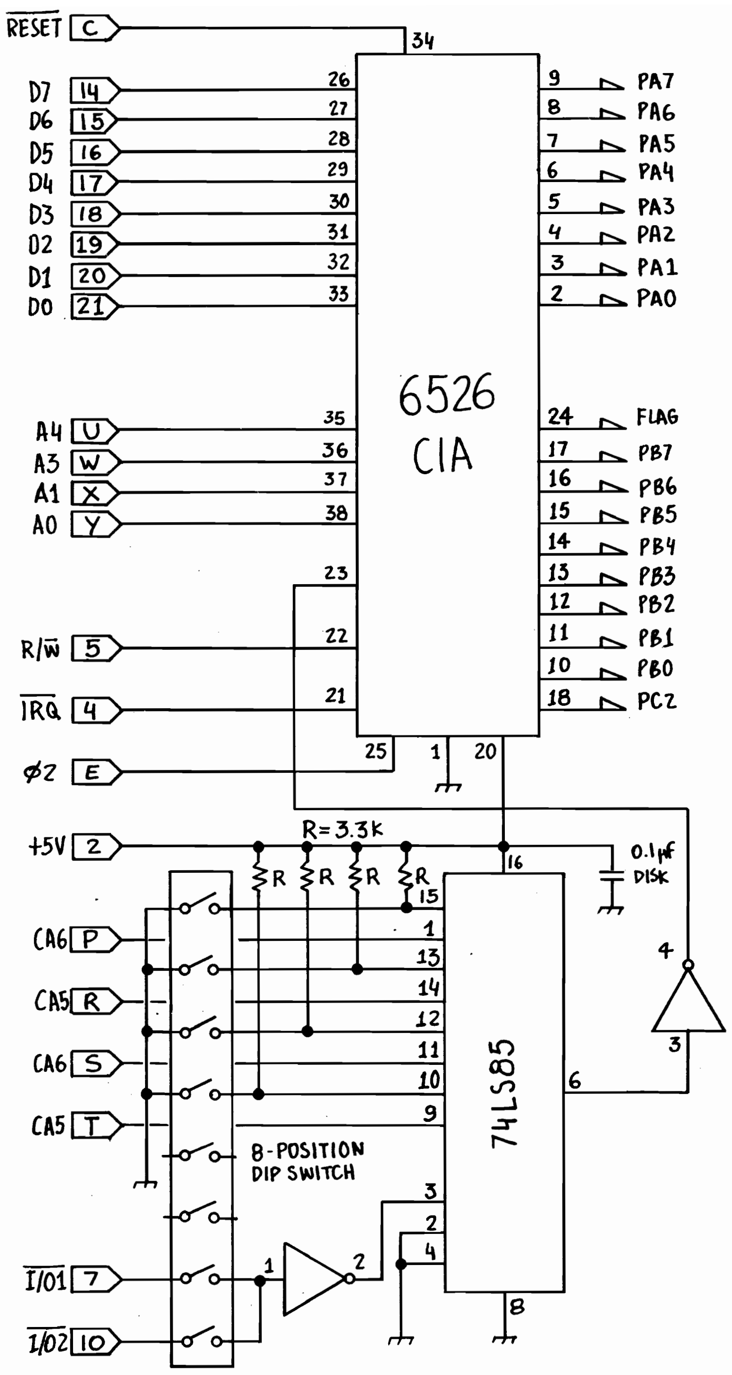
The USER PORT on the 64 is a very convenient feature that simplifies interfacing significantly. Since more of a good thing is a better thing, here is a way to add one (or more) User Ports to your 64. Or, if you are not interested in full compatibility, the same circuit will serve as two separate 8-bit I/O ports.

THEORY OF OPERATION

The USER PORT on the 64 is based on the 6526 CIA chip. The 6526 contains two 8-bit I/O ports, port A and port B, and other circuits like a time-of-day clock which are not needed for our purposes. The design of the USER PORT is such that PA2 of port A is used for handshake purposes for port B. Thus if we emulate the USER PORT, we will seriously impair the functionality of port A. However, the 7 remaining bits of port A can be used for I/O purposes as needed. If duplication of the USER PORT handshake is not required, the circuit presented here will function as two independent 8-bit ports, one with handshake capabilities (port B) and one without (port A).

Even if the port is configured to emulate the USER PORT, there are some things our port cannot do. One is to be used with an RS-232 interface. Second, since it plugs into the expansion connector, it cannot supply the AC voltage that may be needed by some peripherals (remember, the USER PORT has two lines labeled 9VAC). Third, it may be difficult to change commercial programs that drive the USER PORT because source listings of these programs are confidential. The programs in this book can be easily changed to drive the new port.

Interfacing the CIA is straightforward. The designers of the 64 have reserved two 256-byte blocks for I/O expansion. I/O block 1 are the locations between adresses 56832



RESET C 34
D7 14 26
D6 15 27
D5 16 28
D4 17 29
D3 18 30
D2 19 31
D1 20 32
D0 21 33
A4 U 35
A3 W 36
A1 X 37
A0 Y 38
23
R/W 5 22
IRQ 4 21
φ2 E 25
1
20
6526 CIA
9 PA7
8 PA6
7 PA5
6 PA4
5 PA3
4 PA2
3 PA1
2 PA0
24 FLAG
17 PB7
16 PB6
15 PB5
14 PB4
13 PB3
12 PB2
11 PB1
10 PB0
18 PC2
+5V 2
R=3.3k
R R R R
0.1μf DISK
16
15
1
13
14
12
11
10
9
3
2
4
8
6
74LS85
4
3
CA6 P
CA5 R
CA6 S
CA5 T
8-POSITION DIP SWITCH
I/O1 7
I/O2 10
1
2

and 57087 inclusive ($DEOO-$DEFF). I/O block 2 ranges between 57088 and 57343 inclusive ($DFOO-$DFFF). Two signals, I/O1 and I/O2, available at the expansion port, become active low when the corresponding I/O block is addressed.

The CIA itself looks to the CPU like a 16-byte memory. These 16 bytes are the ports plus all control registers. The CIA is place on one of the sixteen 16-byte segments within the I/O block by a 4-bit comparator. The comparator is enabled by either I/O1 or I/O2 depending on the block selected. Then address lines A4 to A7 are compared with the settings of the DIP switch to generate the chip select on the CIA.

PROGRAMMING NOTES

Port B is programmed just like the USER PORT. The only difference is that you should add a displacement to USER PORT addresses to get the address of the new port. The displacement depends on the switch settings on the new port. If you select I/O1, you add 256. If you select I/O2, you add 512. Then you add the setting of the address switches multiplied by 16.

Here is an example: Switch set to select I/O1. Address switches set to 0101. We want to access the DDR of the new port. Since the DDR of the new port is at 56579, we start with this number and add: First 256, because we use I/O1. Second, since the address switches are set to 5 (decimal) we add 5 × 16 = 80. So the address of the DDR of the new port as configured is 56579+256+80=56915.

Programming port A proceeds along the same lines. The base address for the 16 registers of the CIA is 56832 plus the two numbers computed as shown above.

PARTS LIST

1. Commodore 6526 CIA. See text for alternative.
2. 74LS85, 4-bit comparator.
3. 74LS04, Hex inverter.
4. 40-pin socket, 2×16-pin sockets, 14-pin socket.
5. 0.1 uF ceramic disc capacitor.
6. 4×3.3K, 5%, 1/4W resistors.
7. 8 position DIP switch.

PROJECT 10
MORE I/O PORTS

In many applications the flexibility of a bidirectional I/O port (like the one in the USER PORT) is not necessary. Thus separate input and output ports can be built using standard LS TTL parts, saving both money and board space. Programming is also simpler because there are no control registers to set up. There are also savings in address space, because each port occupies just one memory location. Finally, it is possible to implement handshaking but additional circuits must be used.

THEORY OF OPERATION

Additional I/O ports for the 64 are best located in the two 256-byte address blocks reserved for this purpose, as described in the previous project. The I/O1 and I/O2 select signals indicate which block is currently addressed.

Our circuit must implement the port itself, an address decoder to place the port within the I/O address block and means for detecting whether the port is being read or written. This last requirement is necessary only when there are both an input and output port on the same address. If the input ports have different addresses from the output ports, it is only necessary to make sure that the program does not try to read an output port and vice versa.

The input port is basically a tristate buffer feeding the data bus of the CPU. It remains in the OFF state until it receives an enable from the address decoder indicating the CPU is ready to read the input data from this particular port. The IC commonly used as input port is the 74LS244 octal tristate buffer. If six or less input bits are sufficient, the 74LS367 Hex tristate buffer can be used, saving board space and cost.

The output port latches the contents of the CPU data bus when the address decoder signals that the bus con-

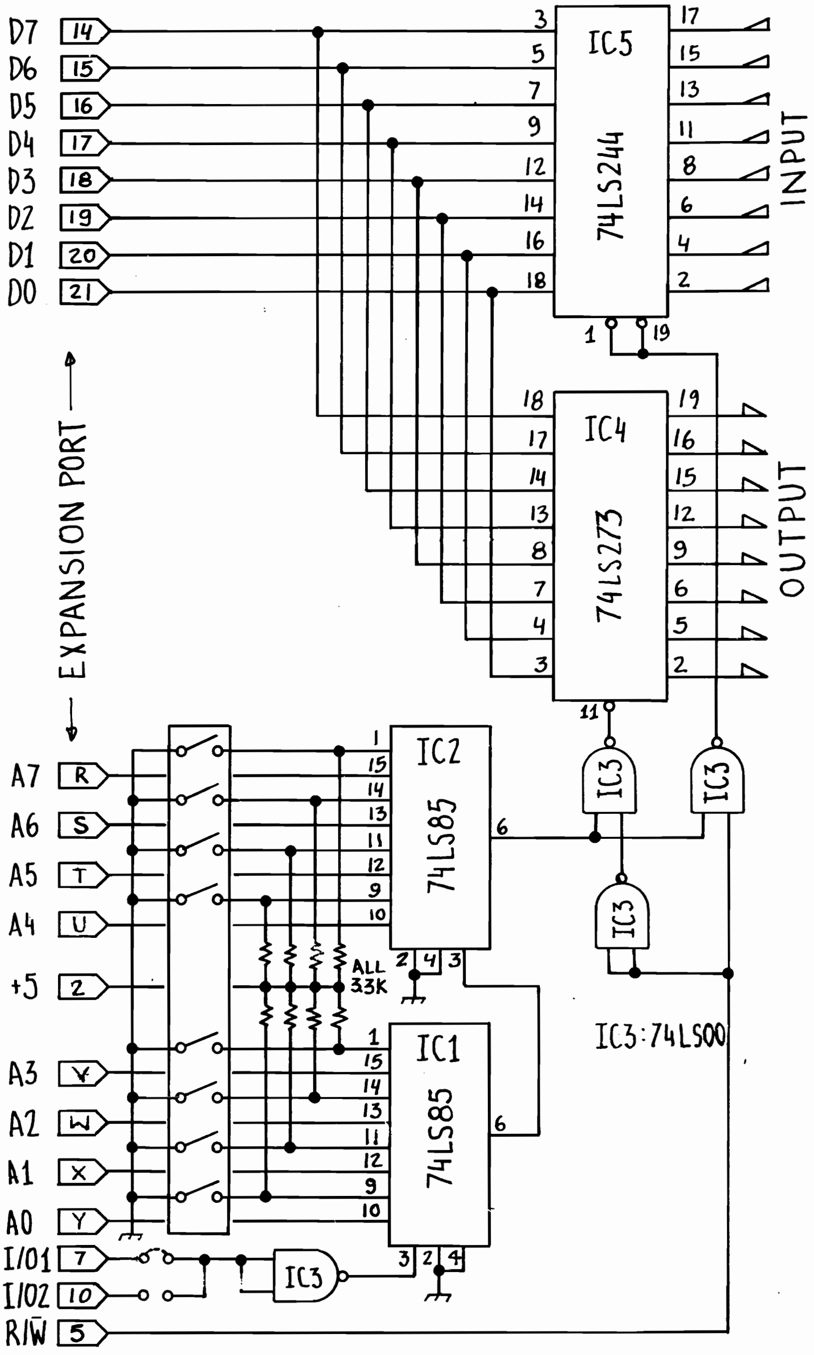


Figure 10.1 User Port, Schematic 1

tains data for the port. Almost any octal storage LS IC will work as output port. The common choices are the 74LS273 or the 74LS374 octal latches. The first has a clear input which can be tied to the RESET signal of the 64, so when the CPU is reset, the port comes up cleared. The second has tristate outputs which offer also higher drive capability. These features can prove useful in some applications.

The newer 74LS533 and 74LS534 are equivalent in function with their 373 and 374 predecessors but they have a pinout that simplifies PC board layout (all inputs on one side and all outputs on the other). If six or less outputs bit will do, the choice IC is the 74LS174 hex D-type flip-flop. It also has a master clear input to be tied to the RESET line.

There are many ways to implement the address decoder. For example, if the system needs only one or two more ports, we can dispense with the address decoder altogether. We just use the I/O1 and I/O2 signals to enable the ports. In this case, every time an address within the 256-byte I/O block is accessed, the port will be read or written depending on the type of access (read or write). With address decoding, we can specify exactly which address within the block is occupied by the port, so that a maximum of 256 input and 256 output ports can be implemented within each I/O block.

As an example of a fully decoded I/O port let us look at schematic 1. The lower 8 address lines of the CPU are compared (using two 74LS85 4-bit comparator IC's in cascade) with the settings of the DIP switch. When they are equal to the setting and the chosen I/O signal goes low, the output of the comparators is "1". At this point, the NAND gates generate the port enable signal based on the state of the R/W line. If it is high, signifying a read, the buffer is enabled. If it is low, signifying a read, the latch is activated.

If the second comparator is omitted and output is taken from the first comparator (pin 6, IC1) the circuit will still work fine but the maximum number of ports per I/O block is reduced to 16 input and 16 output ports, because each I/O port will respond to 16 different addresses within the block. If the extra ports are not needed (it is unlikely that

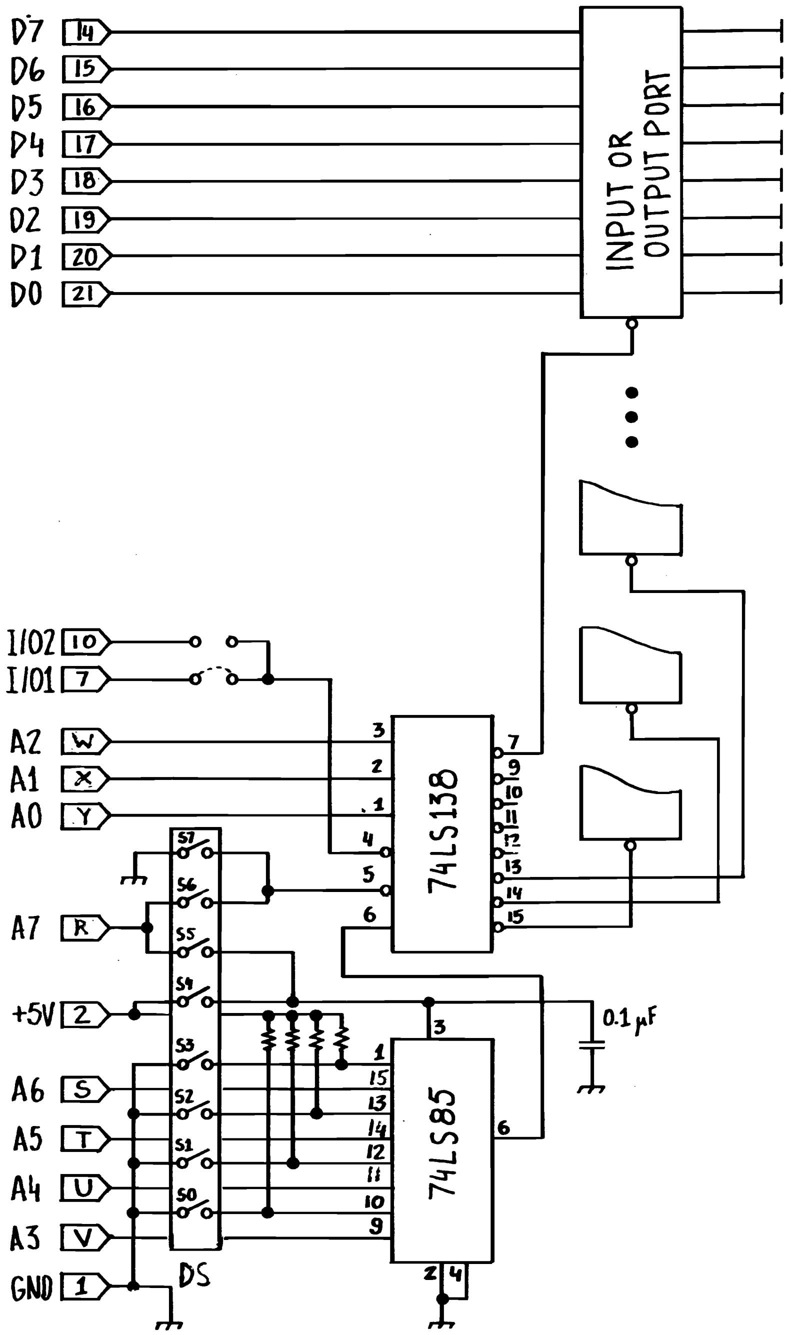


Figure 10.2 User Port, Schematic 2

you will need that many ports) this change saves one IC.

It should be noted here that the schematic as shown provides one input and one output port. To add more ports, you will have to duplicate the circuit. Also, be careful when you use other peripherals that have ports within the I/O blocks. There may be conflicting addresses. In addition, incomplete decoding to save chips may not be possible. To avoid conflicts, select your port addresses after consulting the memory map of all peripherals connected to the expansion port.

Schematic 2 shows the preferred I/O port decoding arrangement. It offers full decoding so no address space is wasted. Also up to 8 input or output ports can be driven by the same decoder. It has the possible disadvantage that each address can be used for an input or an output port but not for both ports simultaneously as is the case in schematic 1. This happens because the R/W signal is not used in this circuit. In practice, this disadvantage has no importance. There are plenty of addresses available to accommodate all reasonable I/O port arrays, and the circuit is recommended because it offers the lowest chip count for an 8-port address decoder.

The DIP switch is set as follows: The first four (S0-S3) switches are set to select A3-A6 of the port address, as explained in the programming notes. The upper four switches (S4-S7) select the 128 address space within the I/O block. To select the lower block, set: S4-ON, S5-OFF, S6-ON, S7-OFF. To select the upper block, reverse these settings.

The desired I/O block is selected by soldering in a jumper on the assumption this connection is not likely to change often. In fact, if you do not plan to change the address of the port, you can replace the DIP switch with soldered in jumpers.

PROGRAMMING NOTES

The ports are accessed by reading or writing (depending on the type of port) at their address. No set up is required. The starting address of I/O block 1 is 56832 and that of I/O block 2 is 57088. If incomplete decoding is used, as for example when only one comparator is used in schematic one, each port will respond to multiple adresses. It is good practice to address such ports using only their lowest valid address.

The address decoder in schematic 2 requires some summation to find the port address. We start with the starting address of the I/O block selected. Then, if switches S4-S7 are set to select the upper 128 address section, we add 128. Next we look at the Switches S0-S3. They form a binary number (when the switch is ON, the bit is "0"). We find its decimal equivalent multiply it by 8 and add it to the sum. Finally, we add the position of the output of the LS138 that drives the enable of our port. The first position has value 0, the last 7.

Here is an example to illustrate the computation. I/O1 is used, switches S4-S7 are set to select the lower 128 addresses, switches S0-S3 set to 1011, input port connected to first output line of the LS138, output port connected to the fifth output line of the LS138. Wanted is the address of the ports.

1. The starting address is 56832 because we chose I/O1
2. The lower 128 addresses are selected within the I/O block, so we do not anything to the sum here.
3. Convert 1011 to decimal. It is 11. So we add 8*11=88
4. The input port is enabled by the first LS138, so we add zero. For the output port we add four.

Thus the input port address is 56832+0+88+0=56920 and the output port address is 56832+0+88+4=56924.

PROJECT **11**

8K RAM WITH DATA RETENTION

When power fails, the contents of RAM memory disappear unless, of course, it has a data retention feature. It is not possible to add this feature to the RAM memory of the 64 because it is made of dynamic memory IC's whose contents need constant refreshing and are not designed for power down operation. We can, however, replace an 8K RAM block between 32768 and 40959 ($8000 to $9FFF) with RAM designed to operate in a low power, standby mode, when power goes down.

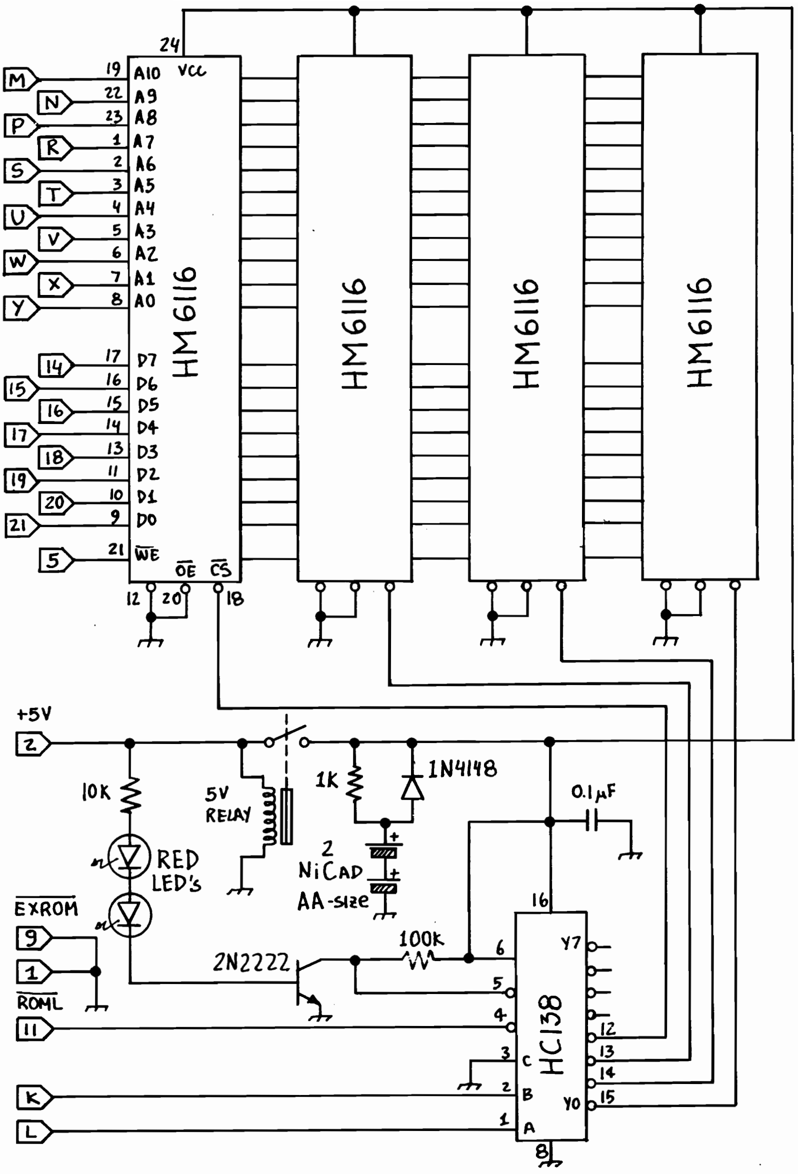
THEORY OF OPERATION

CMOS 2KX8 memories like the HITACHI HM6116 have a standby operation mode in which they draw very little power, typically under 100uA at 3 volts.

A memory board can be built with these memories using some extra circuitry to activate the standby feature and a couple of batteries to provide standby power. Such a board would retain data for a few hundred hours after the power to the 64 is turned off. The batteries will recharge when the power is turned on again, ready for the next power down cycle.

Referring to the schematic, when power from the 64 is present, the relay closes and the circuit operates like a normal 8K RAM. Because pin 9 (EXROM) of the expansion connector is grounded, this RAM replaces the RAM in the 64 between 32K and 40K. The 1000 Ohm resistor in parallel with the diode trickle charges the NiCad batteries.

The two LED's act as zener diodes with a voltage drop of 1.7 volts each (at low voltages, LED's offer better characteristics than zeners). If the power goes down and the +5V supply decreases to below 4 volts, the transistor will turn off. This will happen because the voltage drop on the two LED's and the base-emitter junction of the transistor is 1.7+1.7+0.7=4.1 volts.



M
N
P
R
S
T
U
V
W
X
Y
14
15
16
17
18
19
20
21
5
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
WE
OE
CS
VCC
24
HM6116
HM6116
HM6116
HM6116
+5V
2
10K
RED LED's
5V RELAY
1K
1N4148
2 NiCAD AA-size
0.1μF
EXROM
9
1
ROML
11
K
L
2N2222
100K
HC138
Y7
Y0
C
B
A
16
8

When the transistor is off, the HC138 decoder is deselected and so are the memory chips. During power down the memory IC's must not be selected because if they are, memory contents may be lost. Also, do not substitute other decoder chips for the 74HC138. It offers very low power operation required during the standby phase and at the same time is very fast so that is does not affect the operation of the 64.

When the supply voltage drops below 2.7 volts, the diode will conduct and the battery will start supplying power to the memory array. When it drops even further down, below about 2 volts, the relay opens, isolating the 64 from the battery. If no relay is used, the 64 will drain the battery quickly.

NOTE: NEVER solder directly on a battery (unless it has wires welded on it by its manufacturer). Use a battery holder instead and solder on its terminals while the batteries are out.

The memory is used just like the memory it replaces. However, advantage of its data retention capability can be taken only when the programmer uses it to store important data that must not be lost. A simple way to store and recall data from this memory is by using PEEK and POKE instructions. BASIC variables and arrays cannot be automatically placed in this memory because the BASIC interpreter determines their position. However, it may be possible to trick the 64 so it stores these quantities in the protected memory. You may try manipulating the pointers in locations 45-52.

PROJECT **12**

ADDING ANOTHER 64K OF RAM

The microprocessor CPU within the 64 can address up to 64K of memory. Only 39K of this space is user accesible when running BASIC. The rest is used by the operating system and BASIC interpeter. This project lets you add another 64K of RAM to your 64 using a trick called paging.

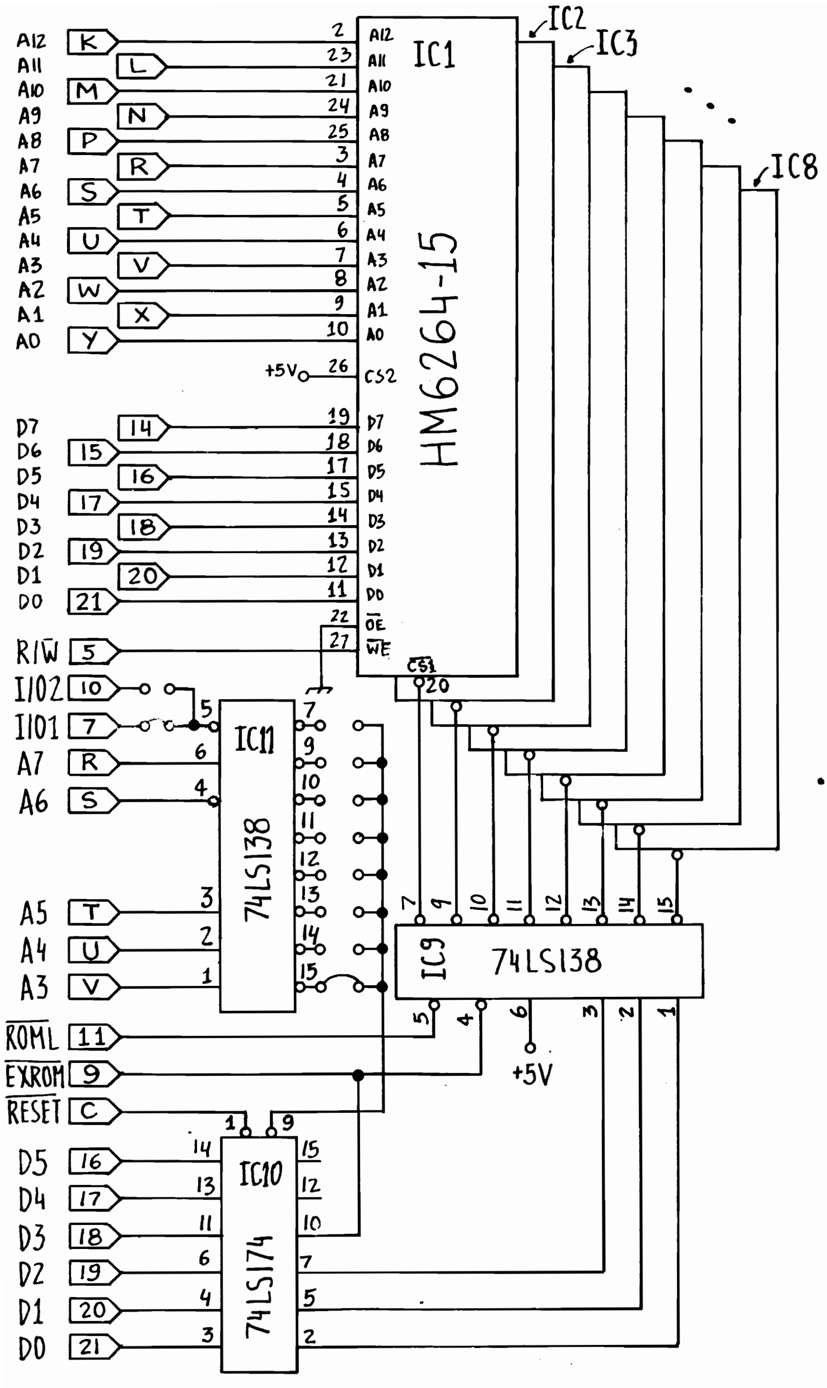
THEORY OF OPERATION

The 6510 CPU used in the 64 has 16 address lines. Thus the maximum memory it can directly address is 64K, or 2 to the power 16. To increase the size of the directly addressable RAM, the CPU chip must be modified. This obviously cannot be done because the 6510 is cast in silicon and it is not possible to change it. However, if something other than directly accessible RAM is acceptable, there are techniques that will allow us to implement very large memories.

A very simple technique to increase addressable memory is paging. A long time ago, scrolls were replaced by books and the concept of a paging became established in publishing. A page is actually a small window to the text. You can only view a portion of the text at a time (as opposed to the whole text on a fully unrolled scroll). In a book, accessing the text by flipping pages does not change the space of your desk taken up by the book.

Book paging directly translates to paging done in a computer. A page can be let's say 8K. You position a stack of pages (book) at any available and convenient location on the address space (desktop). At any given time you can only access the contents of the current page. To access the contents of another page it must be made current by swapping (flipping) pages.

Seems like a wonderfully simple idea but how complicated is it to build? Actually it is rather easy to implement



IC1
HM6264-15
IC2
IC3
IC8
IC9
74LS138
IC11
74LS138
IC10
74LS174
A12 K
A11 L
A10 M
A9 N
A8 P
A7 R
A6 S
A5 T
A4 U
A3 V
A2 W
A1 X
A0 Y
D7 14
D6 15
D5 16
D4 17
D3 18
D2 19
D1 20
D0 21
R/W 5
I/O2 10
I/O1 7
A7 R
A6 S
A5 T
A4 U
A3 V
ROML 11
EXROM 9
RESET C
D5 16
D4 17
D3 18
D2 19
D1 20
D0 21
+5V
CS2
CS1
OE
WE

paging, at least for the hardware portion of it. A good page size for the 64 is 8K. We can stack our pages in the address space starting at 32768 ($8000). This particular area is designed for operation with external ROM, for adding game cartridges or the extended BASIC. Thus, there are provisions in the 64 to switch its own RAM out of this space when external memory is used. The external memory signals its presence by grounding a pin in the expansion port labeled EXROM.

The 8 kilobyte page can be bought as a single static memory IC, the 8K×8 Hitachi HM6264P-15 or equivalent. Eight of these are needed to implement the full 64K extension. Right now this may be costly because these IC's have been in the market for less than a year. New memory IC prices do not stay high for long.

Page turning is controlled by the software via an output port. IC11 in the schematic decodes the port address as 10ABCXXX where X are don't cares and the value of ABC depends on which output of the decoder is used. As shown in the schematic, A=B=C=0. If a more elaborate decoding is preferred, see project 10. You have your choice of I/O1 or I/O2 which are jumper selectable. In the schematic, I/O1 is jumpered and assuming the don't cares are zero, we get a port address of 56832+128=56960.

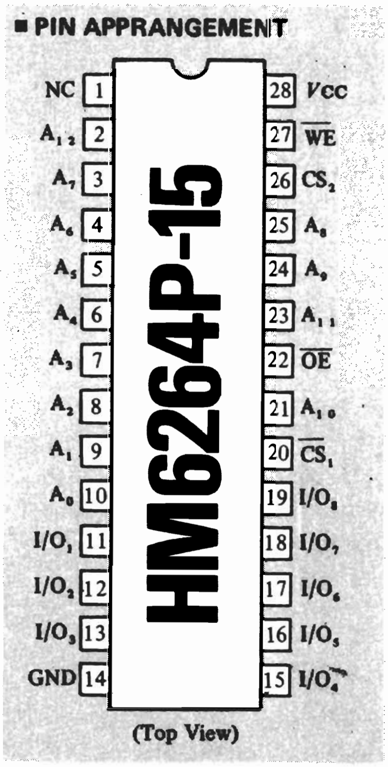
The port itself is IC10. Its lowest 3 bits select the page. The fourth bit, when set to zero, exchanges the RAM of the 64 with the current page. This is done by selecting IC9 and driving the EXROM signal low. On power up, the port is cleared, so page 0 is selected. To get back the 64 RAM, POKE into the port address the number 8.

PROGRAMMING NOTES

So far so good, but what's the catch? Well the catch is the software. For one, the BASIC in the 64 has not been written to accomodate paging. Not that it is impossible to do (on the contrary) but memory has become cheap only recently while MICROSOFT'S BASIC was designed 7 years ago. However it may still be possible to work things around these limitations by manipulation of the pointers to BASIC text stored in page zero.

Without any modification to BASIC we can use the extra RAM for data storage, as long as we access the data using PEEK and POKE. If we leave BASIC altogether and go to machine language, the limitations are few. You must be careful not to swap pages from a program that is on the page being swapped out (doing that is equivalent to standing on a tree branch you are chopping off). And of course you must keep track of where the data and programs are and swap pages at the appropriate times.

BASIC programs can be stored in ROM or RAM pages and loaded into the target location using a simple program in machine language. When the program is loaded, an unused RAM page is swapped in so the BASIC program sees all memory present and is not affected by paging. Much faster than a disk and much more reliable!



8192-word x 8-bit High Speed Static CMOS RAM

CHAPTER 3

SENSOR INTERFACING

PROJECT **13**

COMPUTERIZED THERMOMETER

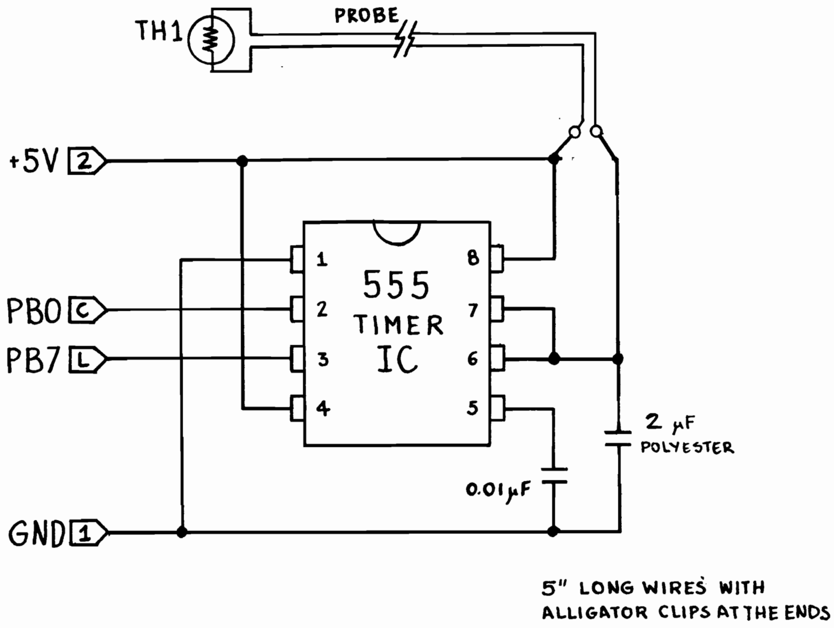
Is your 64 getting too hot? What is the room temperature right now? You can answer these questions with any thermometer. But what if you would like to record temperatures over time, let's say the room temperature in a 24-hour period measured every 10 minutes? In this case a computerized thermometer will be almost a necessity. And if you would like to control temperature with your 64 you will need to sense it first, with a thermometer your computer can read. This project tells you how to add temperature measuring capabilities to your 64 using only four components and a short program.

THEORY OF OPERATION

There are basically three ways to sense electrically temperature variations. Thermistors are special resistors that vary with temprature. Thermocouples are junctions of disimilar metals. Due to a phenomenon called the thermoelectric effect, these junctions develop a small voltage (a few mV) which is almost a linear function of temperature. Finally, semiconductor junctions exhibit a voltage drop which is a function of temperature.

Thermistors are the easiest temperature sensors to use of all three types. They have four disadvantages: First, their variation of resistance is not a linear function of temperature, as it can be seen in Fig. 13.1. Second, thermistors are never identical even if they have the same part number. Their characteristics are only specificied within a tolerance, usually 10%. Third, since all wires have some resistance which itself is temperature sensitive, we cannot use very long wires with thermistors. And fourth, thermistors can operate only over limited temperature ranges, usually −50 to 150 degrees C.

Fortunately, a computer can easily make thermistors quite accurate temperature sensors within their operating



temperature range. For precision scientific measurements over large temperature ranges one must use thermocouples that are much harder to interface.

The resistance of the thermistor is converted to a pulse width by a 555 timer IC connected as monostable multivibrator. In this mode, the 555 has excellent stability and linearity.

In operation, the computer generates a trigger pulse for the 555 by outputting a "0" and then an "1" to the trigger input of the timer. When triggered, the output of the timer goes high and remains high for a time which is a linear function of the resistance of the thermistor. The machine language program in the computer measures the width of the output pulse with a resolution of 25 uS. For more details on the machine language program, see project 33, "Capacitance Meter".

The thermistor nominal resistance and capacitor value were chosen for good rejection of 60Hz noise. A smaller capacitor with a thermistor with higher resistance would also work, but there might be 60Hz hum modulation of the pulse width which would show as constantly changing readings when the temperature is stable.

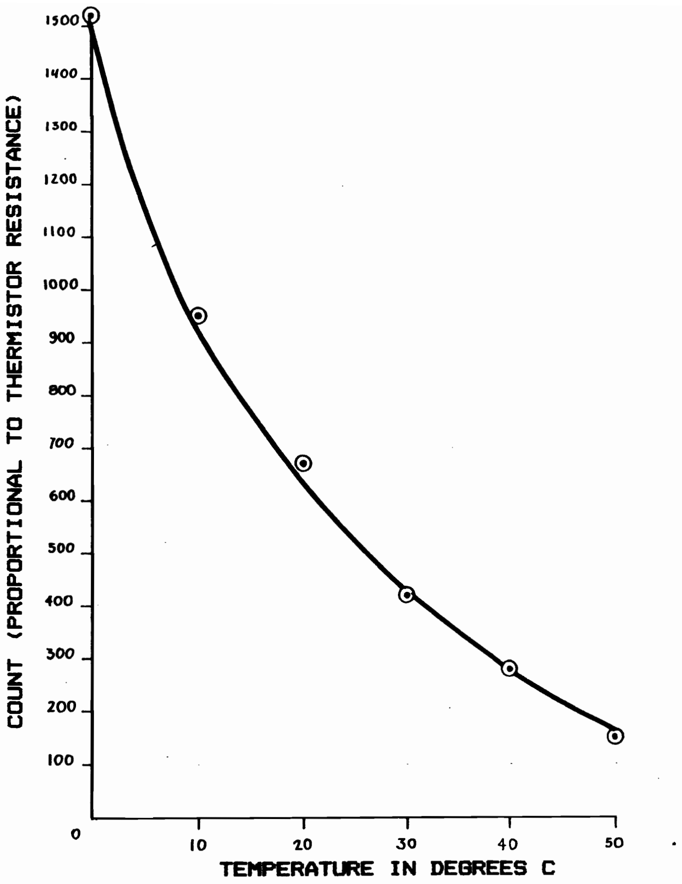
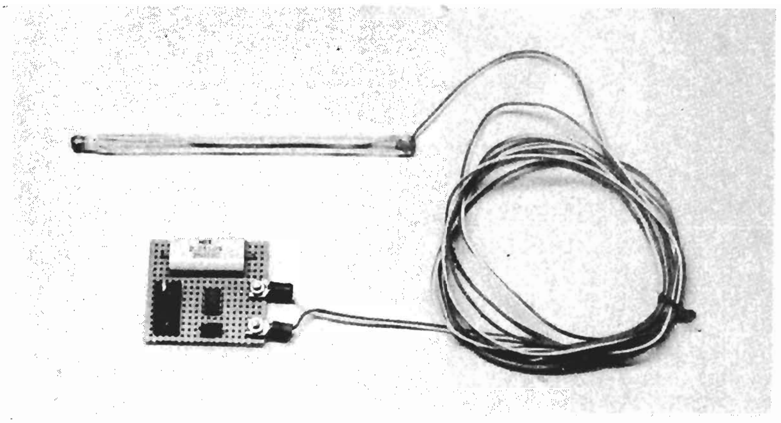


Figure 13.1 Variation of Resistance Versus Temperature for a Typical Thermistor

As it was mentioned before, the resistance of the thermistor (and hence the pulse width) is not a linear function of temperature. Therefore, it is necessary to correct this nonlinearity by a program. There are two ways this can be done: One is to combine linearization and calibration in one step by using a table that gives the temperature for each pulse width. The other is to use a polynomial function to approximate the resistance versus temperature curve of the thermistor. We will use the table lookup method with linear interpolation between entries.



LINEARIZATION AND CALIBRATION

Let us assume that we will use the thermometer between 0 and 50 degrees C. We can build a calibration set-up as follows: Take a reference thermometer and our thermometer and immerse them both in a pot containing ice cubes and water.

Using the program in the capacitance meter project modified to print the variable COUNT, get a count for the above condition which represents 0 degrees C. Then heat the pot until the ice melts and the reference shows 10 degrees C. Take down the value of count and proceed likewise to get the values of count corresponding to readings of multiples of 10 degrees C.

When 50 degrees are reached, there will be 6 entries in the table and in the future we will be able to convert COUNT values to temperature readings by referring to the table. However, accuracy will be within 5 degrees C which is not very good. To increase accuracy, we can take more readings and have more entries in the table, let's say every 5 degrees. Alternately, we can assume that resistance varies linearly with temperature within the 10 degree segments. This is not true so this assumption will result in some error depending on how much the actual curve differs from a straight line. In our case it works very well, giving an error of about one degree, a factor of five improvement over straight table lookup.

```
100 REM ** THERMOMETER **
110 REM
120 POKE 56579,1
130 REM SETUP DRIVER
140 FOR I=1 TO 35
150 READ A: POKE 49152+I,A: NEXT I
300 REM SET UP CALTABLE
310 DIM CALTBL(20)
320 FOR I=1 TO 6
330 READ A: CALTBL(I)=A: NEXT I
400 REM TAKE COUNT
410 I=0
420 I=I+1
430 SYS 49153
440 COUNT=PEEK(139)+256*PEEK(140)
450 CT=CT+COUNT
460 IFI<50 GOTO420
470 COUNT=CT/50:CT=0
480 PRINT CHR$(147)
500 REM LINEARIZATION
510 P=0
520 P=P+1
530 IF COUNT<CALTBL(P) GOTO520
550 CPD=(CALTBL(P-1)-CALTBL(P))/10
560 TEMP=((CALTBL(P-1)-COUNT)/CPD)+(P-2)*10
570 TEMP=(INT(TEMP*10))/10
600 REM CONV TO FARENHEIT
610 FTEMP=32+TEMP*(9/5)
620 FTEMP=INT(FTEMP*10)/10
700 PRINT TAB(10);"** 64 THERMOMETER **"
710 PRINT
720 PRINT TAB(10);"COUNT IS =";COUNT
730 PRINT:PRINT
740 PRINT TAB(10);"TEMP IN DEG C =";TEMP
750 PRINT:PRINT
760 PRINT TAB(10);"TEMP IN DEG F =";FTEMP
770 GOTO 400
5000 DATA 120,169,0,141,1,221,133
5010 DATA 139,133,140,169,1,141,1
5020 DATA 221,165,139,24,105,1,133
5030 DATA 139,165,140,105,0,133,140
5040 DATA 173,1,221,48,238,88,96
5050 DATA 1540,950,670,420,280,180,150
```

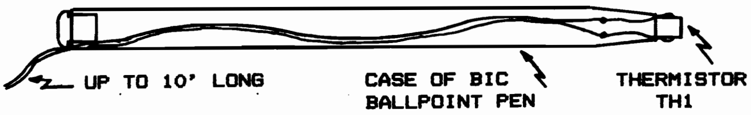


Figure 13.2 Construction Detail of the Thermistor Probe

PROGRAMMING NOTES

The BASIC listing is a complete thermometer program. It will set up the machine language driver, get the data and perform linearization and calibration and display the reading in both degrees C and F.

The program takes 50 measurements and finds the average count which is then used for further calculations. This reduces jitter and increases accuracy but it slows down the rate by which readings are updated to once every 2.4 seconds. This is not a problem because the thermistor specified has a much longer time constant.

Statement 5050 is the calibration table, listing count versus increasing temperature. The data as shown are for 0 to 50 degrees C in 10 degree increments. You can substitute the data for your particular thermistor or data for any other temperature range. If you have a different number of data points change line 320 to reflect this. If calibration data is in increments other than 10 degrees C, you will need to modify lines 550 and 560. These statements do the linearization and calibration.

CONSTRUCTION NOTES

Use a two conductor flexible cable no longer than 10 feet to connect the thermistor to the circuit. Make a probe for the thermistor out of the housing of a ball point pen as shown. Apply "3M Super Strenth Household Cement" or equivalent around the thermistor to seal it to the body of the pen. Use the plug that capped the back of the pen to jam the cable against the pen housing and keep it from moving.

PROJECT 14
LIGHT SENSOR

If your 64 can't tell day from night, don't be alarmed. It has no eyes but its birth defect can be corrected rather easily. This project tells you how to connect a photocell to your 64 so that it can sense light.

THEORY OF OPERATION

The resistance of a photo conductor like CdS (Cadmium Sulfide) changes with incident light. Photocells are packaged pieces of photoconductor material with leads attached. A typical photocell would have a resistance of 500 kiloOhms in the dark and less than a thousand Ohms under sunlight.

By using a 555 timer circuit and by replacing the timing resistor with a photocell, we can convert variations of light to variations of pulse width which the 64 can measure. The circuit is essentially that of the previous project with the thermistor replaced by a CdS photocell.

The variation of the resistance of the photocell is not a linear function of incident light. It is possible of course to correct this using the 64 and to calibrate the system against a known light source. In addition, using a thermometer circuit, the ambient temperature can be measured and used to correct the readings (the resistance of the photocell also varies with temperature).

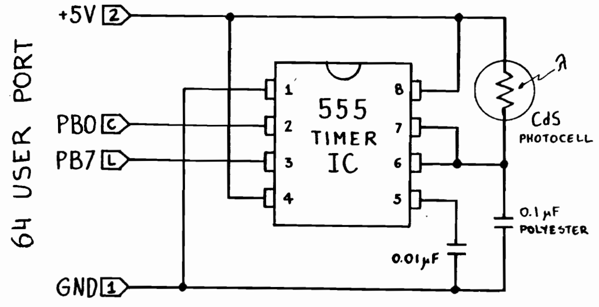
PROGRAMMING

The program is essentially that used in the thermometer. The count obtained by calling the machine language program can be directly printed on screen or used for further calculations. Since most artificial light sources are modulated by 60Hz, you may need to average 5 to 10 measurements to get a steady reading.

APPLICATIONS

There can be many innovative applications for the light sensing system. Here are a few mundane ones:

1. Turn on outside lights when darkness falls.
2. Detect interruptions of light beams. Using a photocell illuminated by a light source placed some distance from it you can detect people entering and1or leaving an area for example. Or a model railroad approaching a crossing. Or an opaque liquid rising in a tank.
3. Make a light seeking robot. Place the photocell on the shaft of a motor controlled by the 64. Write a program so that motor turns until maximum light falls on the photocell. Mount the head of a plastic doll on the shaft for an impressive effect of face looking into light source (a flashlight) and following it.
4. Use it to point solar collectors to the sun. A lens can be used to collect light from one direction so that the photocell becomes more directive. An old camera can provide both the lens and the lightproof enclosure. Mount the photocell so that its photoconductor is exactly on the film plane behind the lens.
5. Measure the light dosage of your plants. Measure intensity every let's say 5 minutes and accumulate it. The total for each 24 hour period is the daily light dosage. Keep in mind that you must linearize the readings before adding them up – otherwise result will be meaningless.
6. Make a darkroom exposure meter that computes correct exposure time and times the exposure. If you bring the 64 in the darkroom, cover the screen of your monitor with a piece of rubilith (ask a printer for one) to render its light safe.



PROJECT 15
LIQUID LEVEL SENSOR

It's a rainy night and you are away from home. Is your basement flooded? The liquid level sensor will tell your 64 and you will be able to call it on the phone to report to you the status of you cellar. Impressive but only one of the possible applications of a liquid level sensor.

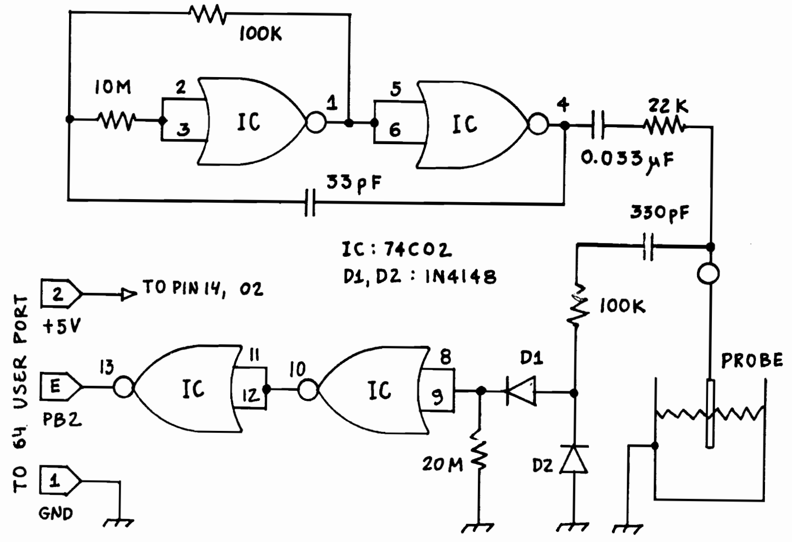
THEORY OF OPERATION

Current can flow through water (but perfectly pure water is an insulator). By placing two electrodes in a container you can tell if it contains a water based solution based on whether current flows. If the applied voltage is DC, electrolysis will take place which will soon corrode the electrodes. AC will have no ill effects because the products of the electrolysis will be opposite in consecutive cycles and will recombine.

The circuit shown here uses a CMOS NOR gate to act as an oscillator to generate an AC voltage. If the probe is not immersed in liquid this voltage is converted to DC by diodes (configured as a voltage doubler), amplified and used to present a logical "1" to the user port. If liquid touches the probe, current flows through the probe and the liquid to the ground and the voltage at point A drops so that the output is now a "0".

An alternate circuit could utilize the LM1830 fluid detector IC from National Semiconductor. The principle of operation is the same as in the discrete circuit. The disadvantage of the IC fluid detector is that it needs a high supply voltage, 15V, which is not available at the 64 and must be supplied externally.

The probe can be just a piece of steel wire if the liquid is water in a tank made out of metal. In this case the tank must be electrically connected to the circuit ground. For corrosive liquids the probe should be made out of stainless steel or be gold plated.



If the container is the liquid if not conductive (for example the basement mentioned above), the probe must have two separate conductors (one of which connects to the circuit ground). In some cases, a phono plug will work fine as the probe. In others, two parallel strips can be etched on a piece of circuit board and used as the probe.

PROGRAMMING

Connect the output of the sensor circuit to any bit of the USER PORT. Make this bit an input by placing a 0 at the corresponding bit of the data direction register.

To check the output of the sensor, assuming you have connected the sensor to PB2 use the following:

```
 90 POKE 56578,0
100 T = PEEK(56577) AND 4
110 IF T = 1 THEN PRINT "HIGH"
120 IF T = 0 THEN PRINT "LOW"
130 GO TO 100.
```

The '4' in statement 100, is a mask to isolate the third bit of the USER PORT.

PROJECT 16
FORCE SENSOR

Measuring force can be quite tricky if high accuracy and repeatability are required. However, if you can live with relaxed specs there is a novel and very simple way to sense force that will allow the 64 to be applied in some unusual ways.

THEORY OF OPERATION

Force is usually converted to an electrical signal by attaching a strain gage or a piezoelectric chip on an elastic beam subject to the force. The beam deforms under the stress of the force and the resistance of the strain gage changes or the piezoelectric material generates a voltage. These are electrical quantities that can be amplified, processed and digitized. Generally, an elaborate setup is required to measure force this way.

There exists, however, a little known method that allows you to sense force with minimal trouble. It is based on the fact that the black conductive foam rubber used in the packaging of IC's (in order to avoid static charges) has a resistance that changes with applied force. You can prove this easily to yourself. Take a piece of the foam, put a coin on each side of it, lay the probe tips of an ohmmeter on the coins and grasp the whole assembly between your thumb and index finger. As you will apply force, you will notice that the resistance of the assembly decreases.

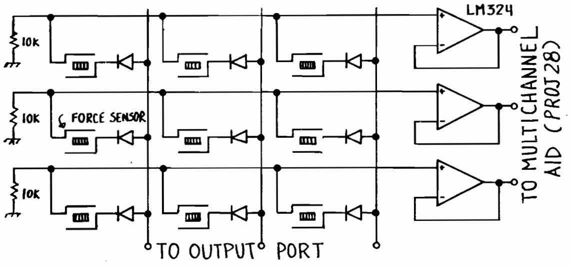
The 555 timer circuit will work just fine here to convert the variation in resistance to a digital signal. You can use the same circuit as the thermometer. Instead of the thermistor, you will use the conductive foam sensor. You may have to reduce the value of the 2 uF capacitor to 1 uF depdning on the characteristics of your particular piece of foam.

It is not likely that you will be able to purchase the conductive foam rubber in small quantities. If you do not have some, check with local electronics parts distributors or electronics companies or even the electronics shop of a nearby university or college. A small piece is all you need and it doesn't cost anything, people throw this material away when they use the chips they were packaged in it. You should be able to get some by asking politely.

The tranducer can be made in the shape and size needed to fit the application. As long as it consists of the conductive foam sandwiched between two electrodes, it will work. For example, if you want to computerize your door mat for security purposes, you can make a 5″ × 5″ sandwich of conductive foam between two pieces of unetched PC board material. After attaching electrodes, the sandwich is heat or tape sealed in a heavy duty poly bag and placed under the regular door mat. Your 64 will then be able to determine if somebody is standing at the door and perhaps even tell if it is a child or an adult.

Another application involves making a music keyboard for the 64 that is responsive to how hard the key is pressed. Music keyboards in use with microcomputers now lack this realism.

And of course, there is the obvious application of adding touch to robot fingers. A transducer only 1/4 inch in diameter will work just fine. You can make one by using a hole punch to punch out mini-pill sized round pieces of conductive foam. In fact, you can simulate skin by making a waffle like structure of many small transducers, sequentially addressed and sensed by an X-Y multiplexing scheme. An isolating diode (1N4148) must be used in series with each element of the matrix, as shown in the example of the 3×3 matrix below.



CHAPTER 4

TELEPHONE INTERFACES

PROJECT 17
TELEPHONE RING DETECTOR

Did the telephone ring when you were away? When? How about keeping a log of telephone rings with your 64. A ring detector can also be the basis for a telephone answering machine. This project shows how to build a reliable optically isolated ring detector and how to integrate it into a computerized telephone answering machine.

THEORY OF OPERATION

The telephone ringing voltage is a 20Hz, 110 volt AC signal, superimposed with a 400Hz signal. The 20Hz signal activates the bell in your telephone set while the 400Hz signal is heard by the calling party to indicate that the phone rings.

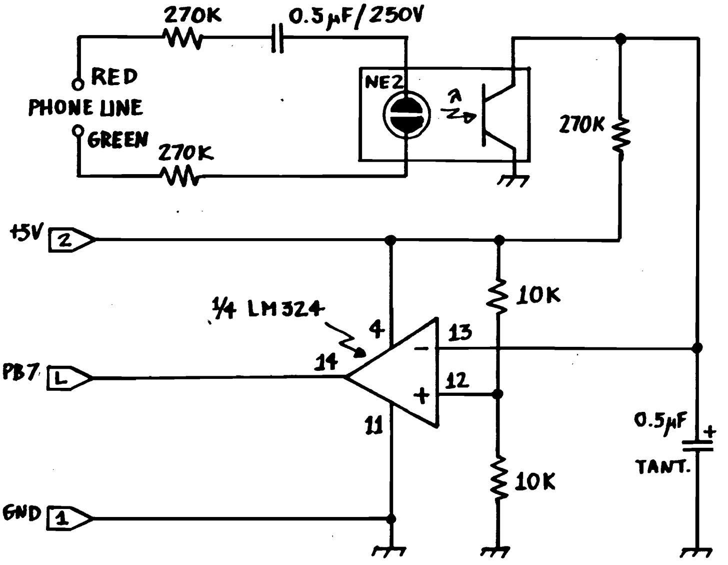
In building anything that connects to the phone lines it is important to take extraordinary safety precautions. While you may do anything you wish at your home, you have no right to damage or make unsafe (by sending high voltages) the telephone network which is used by all.

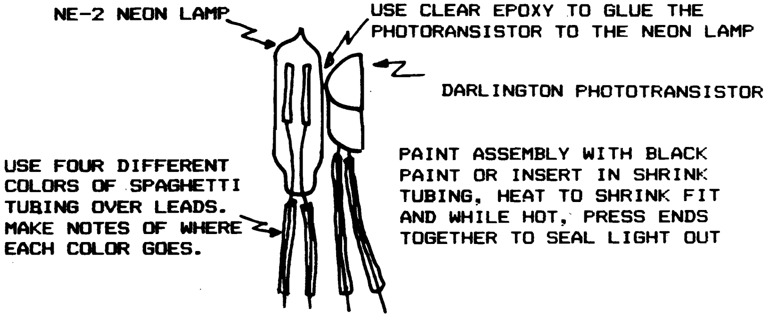
Telephone company regulations are very strict in this matter. The device you connect to the phone lines must be either FCC approved or you must rent a coupler from the phone company. The coupler makes sure that nothing damaging gets to the phone line.

Although required by law to make arrangements for you (for a fee) to connect anything reasonable to their lines, your local telephone company may not like the idea and try to refuse helping you.

In case you want to exercise your alternate option, the ring detector is designed to be undetectable by electrical inspections of the line by the phone company.

This is achieved by presenting a very high impedance, more than 100K, to the telephone line. A small neon lamp is used to convert the ringing voltage to light which is

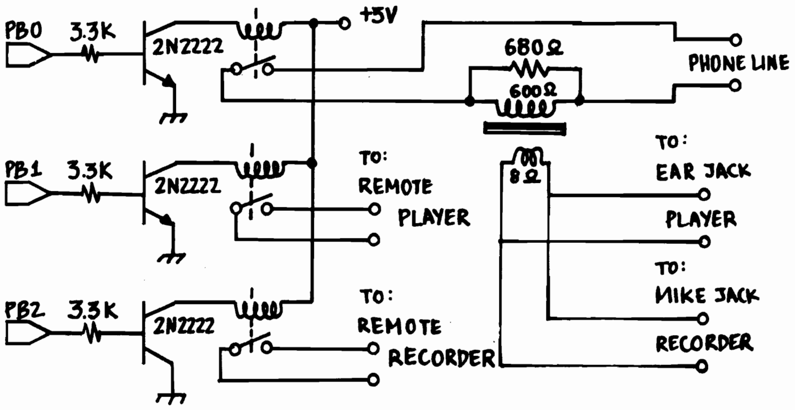



NE-2 NEON LAMP
USE CLEAR EPOXY TO GLUE THE PHOTORANSISTOR TO THE NEON LAMP
DARLINGTON PHOTOTRANSISTOR
USE FOUR DIFFERENT COLORS OF SPAGHETTI TUBING OVER LEADS. MAKE NOTES OF WHERE EACH COLOR GOES.
PAINT ASSEMBLY WITH BLACK PAINT OR INSERT IN SHRINK TUBING, HEAT TO SHRINK FIT AND WHILE HOT, PRESS ENDS TOGETHER TO SEAL LIGHT OUT

picked up by a Darlington phototransistor. The resulting signal is amplified by an op-amp and presented to the user port as a logical '0' or '1'. The use of the neon lamp gives excellent noise immunity because the lamp does not fire at voltages less than 70 volts. Thus line noises, clicks, etc. do not give false readings. In addition, it offers perfect isolation of the phone lines since no electrical connection is made between your circuit and the line.

Programming is identical to that of the liquid level sensor, since both devices have a 1-bit output. Use the TI$ variable to create a clock to log the time of telephone ringings. You could also keep a count of how many times the phone rang each time.

ANSWERING MACHINE



AN ANSWERING MACHINE

To answer the telephone when you detect a ring, you only need connect a 680 Ohm resistor across the lines. This can be done using a small 5V relay, driven by one of the bits of the user port.

The connections for a complete answering machine are shown in the schematic. The cassette player should use an endless loop tape to deliver the message. It is turned on after the desired number of rings is detected, and the answer relay is activated. After a delay equivalent to the length of the message (usually 20 seconds) the cassette recorder is turned on for a specified duration, let's say a minute. A program that does this can be written in BASIC.

PROJECT 18
PULSE TELEPHONE DIALER

This project gives your 64 the capability to dial a telephone. Pulse dialing is accepted by all exchanges in the USA (as opposed to tone dialing) so this system has universal appeal.

THEORY OF OPERATION

Pulse dialing operates by interrupting the current flow in the telephone line. To send a digit, let's say '5', the line is interrupted five times. The interruptions last 40 ms with a pause of 60 ms between interruptions. There is a minimum of 800 ms pause between digits. Pulsing with these specifications is called 10pps (pulse per second) dialing. Some telephone exchanges may require 20pps dialing where all timing values are half of these given above.

The circuit consists of a transistor driving a reed relay that interrupts the line. A BASIC program drives the relay through the user port, creating the required pulses.

A reed relay is specified because it is fast. It has the disadvantage that reed relays when not activated are usually in the open state. Here switch S1 is used to manually restore continuity of the line when the 64 is not operating.

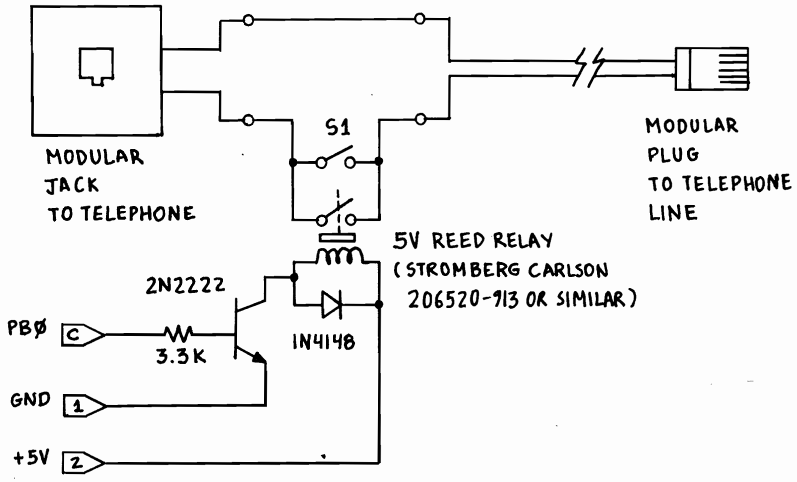
NOTE: The telephone headset must be picked up to establish connection before dialing can proceed.

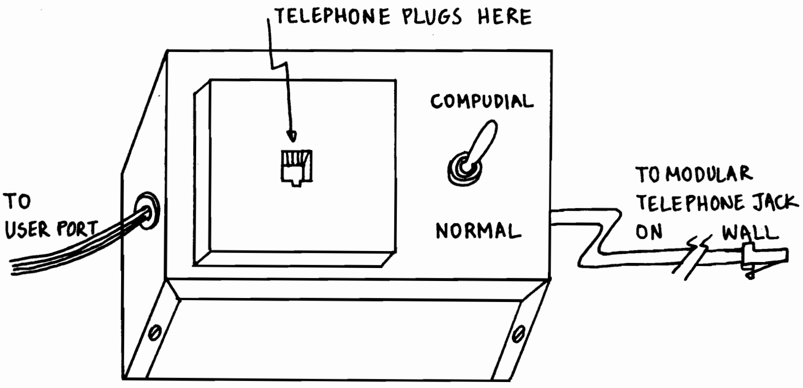
PROGRAMMING NOTES

Lines 120 to 150 input the string of digits to be dialed. There is no limit to the length of this string. Lines 160 to 210 extract the digits from the string one at a time and convert the ASCII codes to the actual numbers (0 is replaced by 10 because 10 pulses represent a zero). Lines 300 to 410 form the subroutine that does the actual pusling. Lines 250 to 290 add a redial feature where the same number is dialed again, if, for example, the line is busy.

```
100 REM  ** PULSE DIALER **
101 REM
105 POKE 56579,255: P=56577
110 POKE P,255
115 PRINTCHR$(147): PRINT
120 PRINT"ENTER TELEPHONE NUMBER"
150 PRINT: INPUT N$
160 L=LEN(N$)
170 FOR M=1 TO L
180 D$=MID$(N$,M,1)
190 IF D$="" GOTO 250
200 IF D$="0" GOTO 240
210 D=ASC(D$)-48
220 GOSUB 300
230 GOTO250
240 D=10: GOSUB 300
250 NEXT M
260 PRINT: PRINT"TO REDIAL PRESS Y ELSE N"
270 GETA$: IF A$="" GOTO 270
280 IF A$="Y" GOTO 160
285 GETA$: IF A$="" GOTO 285
290 GOTO 120
295 GOTO 200
300 POKE P,255
310 FOR I=1 TO D
320 FOR K=1 TO 21
330 NEXT K
340 POKE P,0
350 FOR K=1 TO 34
360 NEXT K
370 POKE P,255
380 NEXT I
390 FOR K=1 TO 450
400 NEXT K
410 RETURN
```

The program can be modified so that is dials a group of numbers sequentially (perhaps to deliver an advertising message) or dials numbers derived from a mailing list database (to play a recorded announcement to friends or customers for example).





PARTS LIST

1. Modular telephone jack
2. Modular telephone plug with cord attached
3. Single pole single throw miniature toggle switch
4. 5V reed relay
5. IN4148 or equivalent diode
6. 2N2222 or equivalent transistor
7. 3.3K, 0.25W, 5% resistor
8. Case and hardware as needed.

PROJECT 19
DTMF TELEPHONE DIALER

If your telephone line can accept touch-tone phones (this type of service is optional and costs more in most exchanges) you can use the dialer described in this project to let your computer dial out numbers much faster than it is possible with a pulse dialer.

THEORY OF OPERATION

Dual Tone Multifrequency (DTMF) dialling also known by the trademark Touch-Tone provides quick dialling by sending out each number as a pair of tones as opposed to pulse dialling that sends out a series of pulses. The tones must last a minimum of 40 ms and between digits there should be a pause of 40 ms. For the standard seven digit number, total dialling time is 7×40+6×40=520 ms. Compare this with the average dialling time of a pulse dialer which is 4 seconds. Pulse dialling offers no advantage over DTMF dialling and it is only a poor sense of cost savings on the part of the phone companies that prevents it from being the standard service in the US.

Figure 19.1 shows the frequencies that correspond to the various keys on the telephone keyboard. The fourth column, with keys having letter designations is not normally used for dialling but is available for supervisory functions. As an example of how to read the figure, the number "5" is represented by a combination of 770 Hz and 1336Hz tones.

Generation of the DTMF frequencies is very easy if we use a chip made for this purpose. The MOSTEK MK5087 and MK5089, the TI TCM5089, the GI AY-9-9559 or the AMI S2559 DTMF dialer IC's are equivalent and any one will work fine in this project. They synthesize the DTMF pairs digitally from a built-in oscillator that uses an external 3.58 MHz TV colorburst crystal.

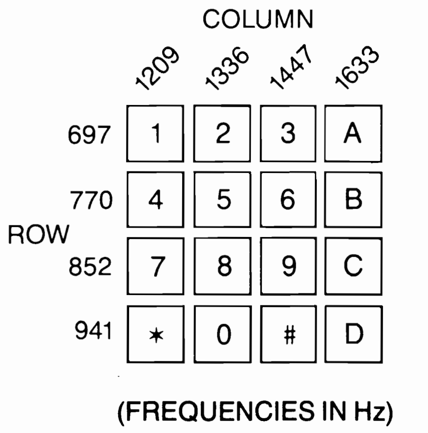


FIGURE 6 DTMF FREQUENCY ASSIGNMENTS

Figure 19.1 DTMF Frequency Assignments

Figure 19.2 shows the correspondence between the 8 inputs of the dialer IC and the tones produced. To produce a tone pair, both inputs corresponding to the desired tones must be set high.

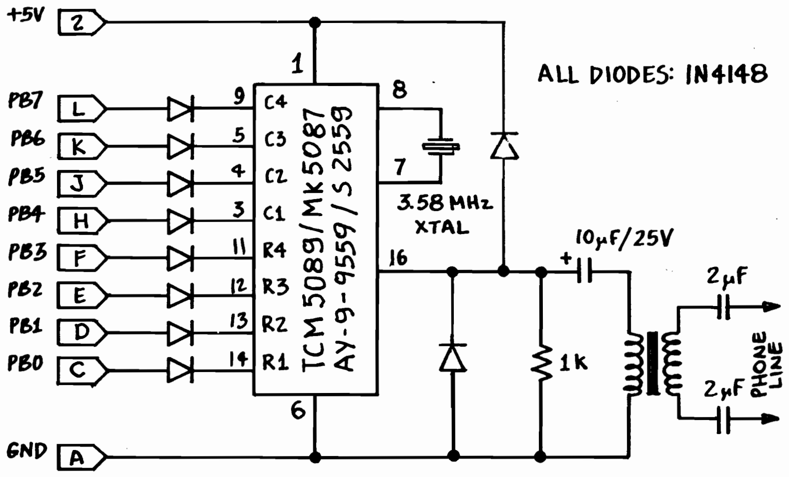
Coming to our schematic, the USER PORT drives the dialer IC directly. The diodes (1N4148 or 1N914) are needed to avoid drawing current when no tones are produced. To produce a tone pair, a pair of USER PORT bits must be set to "1" for more than 40 ms. For example, to send the number 5, we must output the binary number 00100010 which in Hex is 22 and in decimal 34.

If the dialer IC is connected to a port other than the user port, you must use either a CMOS IC for the port or a CMOS buffer followed by the diodes and then the dialer. This is necessary because the tone select inputs of the dialer IC must go to +5V to select a tone and LSTTL outputs go to only 3.5V when high.

The output of the dialer is connected to the phone line using a 600 Ohm interstage transformer to provide isola-

#	TONE1	TONE2	C4	C3	C2	C1	R4	R3	R2	R1	CODE
1	697	1209	0	0	0	1	0	0	0	1	17
2	697	1336	0	0	1	0	0	0	0	1	33
3	697	1477	0	1	0	0	0	0	0	1	65
4	770	1209	0	0	0	1	0	0	1	0	18
5	770	1336	0	0	1	0	0	0	1	0	34
6	770	1477	0	1	0	0	0	0	1	0	66
7	852	1209	0	0	0	1	0	1	0	0	20
8	852	1336	0	0	1	0	0	1	0	0	36
9	852	1477	0	1	0	0	0	1	0	0	68
0	941	1336	0	0	1	0	1	0	0	0	40

Figure 19.2 Pin-out and Frequencies Generated by the DTMF Dialer IC



tion. A good quality transformer is needed here for reliable operation. The important specification is flat frequency response in the DTMF frequencies (600 to 1700 Hz).

PROGRAMMING NOTES

The DTMF dialer can be programmed easily in BASIC because the 40 ms timings for tone and pause are well within its capabilities. In addition these timings are lower bounds only and longer tones or pauses will work fine.

```
100 REM  ** DTMF DIALER **
101 REM
110 POKE 56579,255: P=56577
120 POKE P,0
130 DIM NE(10)
140 NE(0)=40: NE(1)=17: NE(2)=33
150 NE(3)=65: NE(4)=18: NE(5)=34
160 NE(6)=66: NE(7)=20: NE(8)=36
170 NE(9)=68
180 PRINTCHR$(147): PRINT
200 PRINT"ENTER TELEPHONE NUMBER"
210 PRINT: INPUT N$
220 L=LEN(N$)
230 FOR M=1 TO L
240 D$=MID$(N$,M,1)
250 IF D$="" GOTO 270
260 D=ASC(D$)-48
265 GOSUB 300
270 NEXT M
280 PRINT: PRINT"TO REDIAL PRESS Y ELSE N"
285 GETA$: IF A$="" GOTO 285
290 IF A$="Y" GOTO 230
295 GOTO 200
300 POKE P,NE(D)
310 FOR I=1 TO 30: NEXT I
320 POKE P,0
330 FOR I=1 TO 30: NEXT I
340 RETURN
```

The program shown here is similar to the one shown in the previous project. Lines 100 to 170 are set-up, lines 200 to 295 get the number from the user and lines from 300 on do the dialling. As shown, the program offers dialling and automatic redial. It can be modified to add features such as sequential dialling or the capability to look up a number from the name of the person you are calling and dial it.

PROJECT **20**

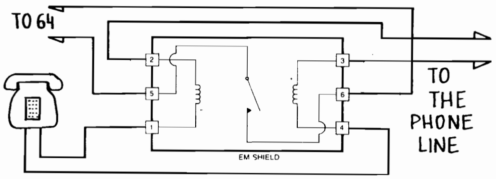
PHONE LINE STATUS MONITOR

When you interface your 64 to the telephone line, unless it has its own number, it is a good idea to enable it to detect the status of the line. This will prevent it, among other things, from coming on line and trying to dial out when you are using the phone.

THEORY OF OPERATION

The telephone set draws no current from the line when the receiver is on hook. If you pick up the receiver, the set will draw current (20 to 50 mA) needed for its operation. This current can be sensed to reliably detect the off-hook condition. Because of the need to do things right when dealing with a public resource like the telephone system, it is best to sense the status of the line with a device built specifically for this purpose.

The TELTONE M-949 line sense relay fits the bill perfectly. It is designed to connect in series with the phone line as shown in the schematic. It has two identical coils so that it does not unbalance the line. The resistance of the coils is very low (20 Ohms) compared to the resistance of the telephone set. It responds to line changes very fast, within 1 ms. And finally, it offers excellent (up to 1500V) isolation between the phone line and the equipment connected to its contacts.



The line sense relay can also be used to count the rotary dial pulses and to recover the number dialed out, either for recordkeeping purposes or just to find out who is calling whom and when.

PROJECT 21
DTMF RECEIVER

A receiver for DTMF (Touch-Tone) signals has many uses. The most important is remote control of the computer via the telephone. You can call your 64 when you are away from home and, using the pushbuttons on the telephone set, command it to give you information via a voice synthesizer. Or you may give it instructions to remotely perform functions like turning on the lights.

THEORY OF OPERATION

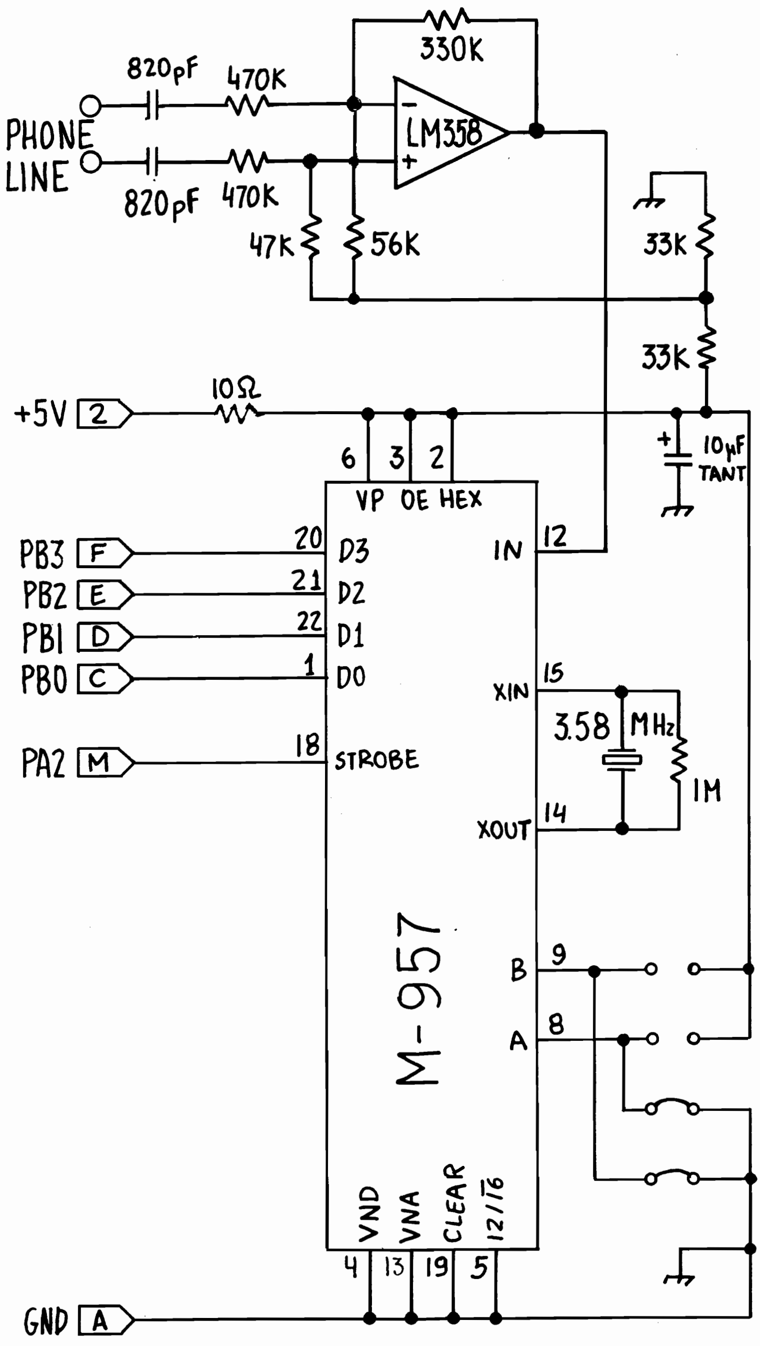
The DTMF signals allow fast dialling and they are transmitted by the intercity telephone network which does not transmit rotary dialling pulses. This last feature allows you to call your computer from anywhere in the world to enter commands and data. For a discussion of the DTMF signals see project 19.

Reliable reception of the DTMF signals requires very specialized circuitry because the phone lines can introduce all kinds of noise and distortion to the information they transmit. Long distance connections are notorious for poor transmission qualities.

It is only fortunate that we can now purchase (at very reasonable prices) a DTMF receiver of good quality integrated on a single silicon chip, the TELTONE M-957 (TELTONE Corp., P.O. Box 657, Kirkland, WA 98033-0657, USA).

The M-957 connects to the phone line via an op-amp configured as differential amplifier to cancel any 60Hz common mode noise present on the line. AC coupling of both inputs with small (820 pF) capacitors isolates the line from the receiver. In addition, this arrangement offers a very small and balanced load to the phone line.

The output of the receiver is four bit binary number corresponding to the number represented by the received DTMF signal. A valid output exists only when the STROBE


PHONE LINE
820pF
820pF
470K
470K
330K
LM358
47K
56K
33K
33K
10Ω
+5V
2
10μF TANT
6
3
2
VP OE HEX
PB3
F
PB2
E
PB1
D
PB0
C
PA2
M
20
21
22
1
18
D3
D2
D1
D0
STROBE
IN
12
XIN
15
3.58 MHz
1M
XOUT
14
M-957
B
9
A
8
VND
VNA
CLEAR
12/16
4
13
19
5
GND
A

output is "1". The data output of the receiver is connected to bits PBO-PB3 of the USER PORT, while the STROBE output is connected to PA2.

The M-957 has two inputs, labeled A and B that act as digital gain control to adjust its input sensitivity.

When A=B=0, maximum input signal for correct operation is −2 dbm (dbm means the reference is 1mW, i.e. 0dbm = 1 mW), while minimum detectable signal level is −32dbm. Other values of A and B increase the sensitivity with maximum sensitivity when A=B=1 (maximum signal −8dbm, minimum signal −38dbm).

These inputs can be controlled from the 64 but it is questionable if the improvement in performance would be worth the trouble. Instead, it is suggested to use jumpers and set them initially at 0. If you need more sensitivity, set them to increasing binary values.

APPLICATIONS

A practical DTMF input arrangement would consist of a phone line status monitor, a ring detector, means for answering the phone and the DTMF receiver as shown in the block diagram.

The 64 can check if the telephone set is on the line via the line status monitor. If the phone is off the hook, any incoming DTMF signal will be from the phone dialling out. This information can be used to record outgoing telephone call data, like time, number called and who is calling. This last piece of information can be obtained if the caller, after entering the number he wants to call, enters the symbol "#" and then his assigned code number. Entering excess digits will not affect the number being dialed. If a DTMF dialer is also controlled by the 64, a redial feature can be added, using the "*" key. Or, with a tape recorder connection, recording of the call can be initiated. In fact, this arrangement converts a simple Touch-Tone telephone set to a programmable telephone system with as many functions as you care to program.

When the telephone set is on-hook, the computer can answer the phone after a programmable number of rings.

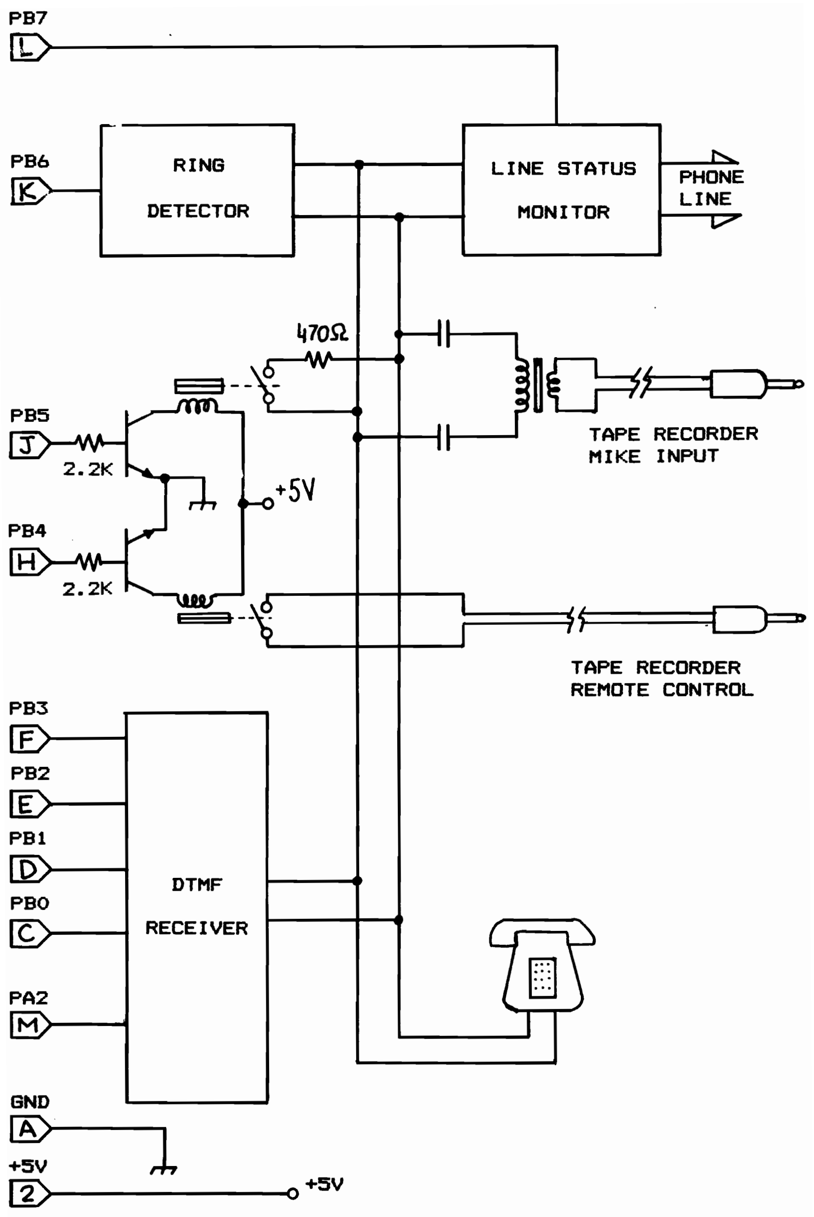


Figure 21.1 Block Diagram of DTMF Receiving Setup

Then, any DTMF signals received will be considered as commands to the computer. With a voice synthesizer like the one described in project 24, the computer could report back on the condition of whatever it monitors.

Here is an example of how things might go. You call up the computer, which after two rings answers and greets you with the synthesized message. "Please enter your access code". You enter 3769# which is your access code. The computer replies: "access code verified, ready for command input". Entering 1# (where 1 stands for command one) results in the computer reading back to you the times the computer received phone calls and any messages left. A message could simply be a number to call back, entered with a similar procedure. Entering 2# can tell you the room temperature right now. #3 turns on the lights and so on.

Another application is voice mail, as described in the next project.

It is a good idea to warn your callers that your computer will answer the phone and perhaps have a demonstration of synthesizer talk. Not long ago, a dentist had the bright idea of letting his office computer call up all patients with appointments one day before the appointment and remind them via a synthesizer of their appointment time. This idea was not received well from his patients who thought they were victims of obscene phone calls. When the dentist printed a brochure explaining his system and had his receptionist demonstrate it in operation, he had a lot more success with his system.

PARTS LIST

1. IC's: LM358 dual op-amp, TELTONE M-957 DTMF receiver.
2. Resistors: 0.25W, 5%. 1M, 2×470K, 330K, 56K, 47K, 2×33K, 10 Ohm.
3. Capacitors: 2×820pF/500V, 10uF/10V tantalum.
4. Crystal: 3.58 Mhz, TV colorburst crystal.
5. Connector for USER PORT, PC or perf board.

PROJECT 22 VOICE MAIL

In voice mail (also known as voice store and forward) the "letters" are recorded messages exchanged via the telephone. Each user has a "mailbox" where messages are left for him. He can call periodically to get his messages or the computer can call him to play back his messages. Commercial voice mail systems start at over $10,000. In this project we examine a limited but very inexpensive voice mail system based on the 64.

THEORY OF OPERATION

A voice mail system consists of two parts: The controller and the medium where the messages are stored. The controller accepts user codes and commands entered via DTMF tones. In the future voice recognition is likely to replace the DTMF tones but right now it is not reliable enough for this function. The controller process the DTMF data and activates playback of recording of the messages in the voice memory. In commercial systems the voice memory is a hard disk and speech is stored using some form of data compression quantization, usually Delta encoding.

Here we will use a cassette tape deck to store messages. This will give slow access (up to 30 seconds) and it will accomodate a limited number of mailboxes, about five or so. In contrast, a commercial system might have more than a thousand mailboxes and response times less than 200 ms. The reason for using a tape deck here is of course, its low cost.

This project is the most difficult to build in this book and it is only for experienced builders. The instructions and information tend to be general in nature because, quite a bit of how the system will finally look depends of the availability of the right cassette deck and how far you want to go in squeezing the best performance out of it.

Three characteristics of the cassette deck are important in this project. The basic requirement is that it must be solenoid operated. Fortunately, most decks over $150 nowdays have this feature. In a solenoid operated deck the buttons for play, stop, rewind etc. do not operate the mechanism of the deck directly. They are electrical switches that send current to solenoid actuators connected to the deck mechanism.

A very helpful feature is high-speed rewind and fast-forward (FF) functions. The deck will be operated as a random access device, so it will have to be able to move fast to the position your computer commands it to go. The rewind and FF times are usually shown in the specifications of the deck.

Finally, a deck with two separate motors, one for play and the other for FF and rewind is very desirable if you plan to record and play at reduced speed to increase storage capacity.

The usually important specifications for a Hi-Fi deck like frequency response, distortion, dynamic range etc. are not significant here. Any deck that is solenoid operated is close enough to the top of the line to have excellent performance in these areas, at least as far as our application is concerned.

The cassette deck need to be modified to add the contacts of small relays across its controls and to add some form of tape position sensing. Bring all wires to a connector mounted on the back of the deck for easy disconnect. Some decks have such a connector for remote operation.

In order to double the available storage without increasing access time, it is recommended to modify the deck so that each channel can be written separately. The recording and erase signals must be switched by reed relays mounted inside the deck to avoid noise pick-up.

Referring to the block diagram of the voice mail system we have a full phone line interface consisting of a line status monitor, ring detector, DTMF receiver, voice synthesizer and ability for the computer to answer a call (by presenting a 350 Ohm DC load to the line). The tape deck is controlled by the computer via relays connected in parallel with its push buttons. The deck is also connected to the phone line to record and play back messages.

The 64 must maintain an internal tape counter so that it knows current tape position every time it initiates an access. For reliable operation, a sensor must be used on one of the pulleys directly connected to the reels of the cassette. An optical sensor can either be a notched disk that interrupts a light beam or a reflecting target (aluminum foil strips) that reflects the light of an LED to a phototransistor.

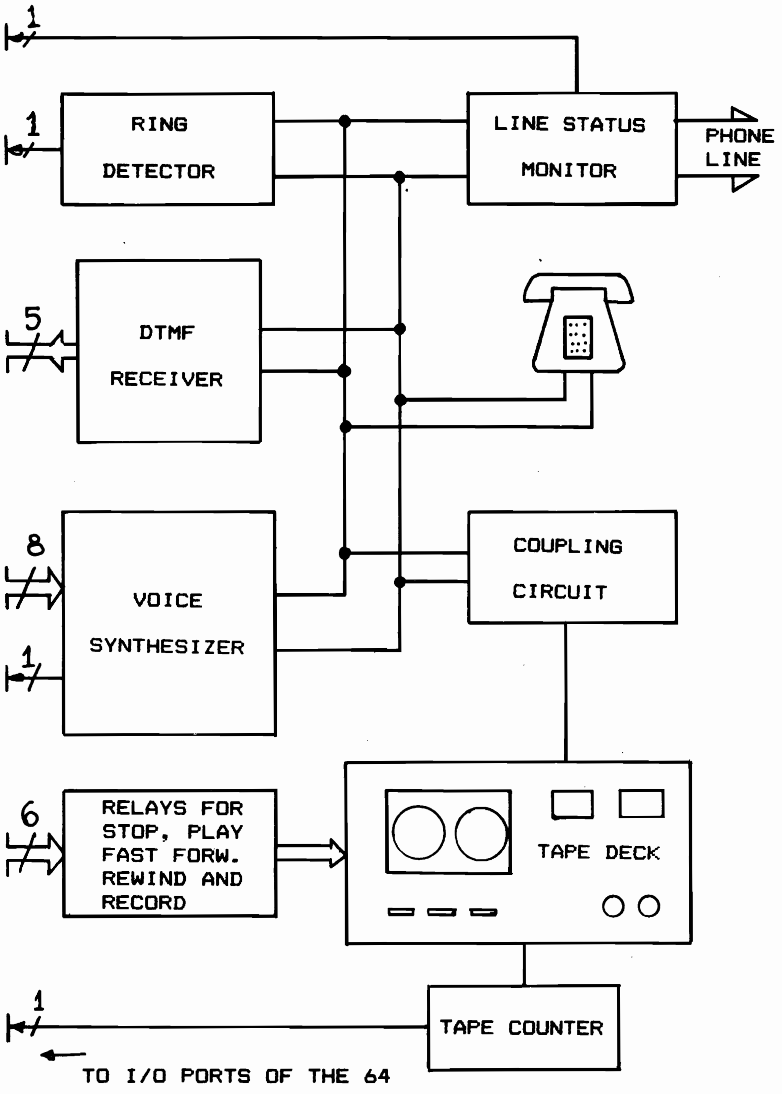
Some decks have a brush that makes contact with conductive sectors on a pulley. They use this to detect when the tape stops moving so as to turn off the motor. You can use this feature directly by counting the pulses generated by the brush.

Another approach is to glue a small magnet on the tape counter driving shaft and place next to the magnet a reed switch which closes when the magnet approaches it as it rotates. This solution is the easiest to implement if done carefully. In some decks it might not be possible because of space limitations. Be careful if you decide to place the magnet on a fast moving part directly connected to the reels, for example the driving pulley. It may imbalance the rotational system causing vibrations or if it is too big it may slow it down due to the added mass.

Only the highest quality cassettes must be used to avoid binding and slipping that will result in loss of position, or worse stoppage of the tape movement altogether.

The pulses generated by any of the above systems are counted by the 64 to determine position on tape. In practice, you need to know only a small number of positions. If you use a 30 minute cassette (about the upper limit for reasonable access times) each side will be 15 minutes playing time, enough for seven, 2-minute messages. This number is doubled when you use both tracks.

The program in the 64 must be able to position the tape to any of the 7 starting locations for the messages and select one of the two tracks. We need seven numbers corresponding to the count of pulses between each consecutive pair of message starting locations. These numbers will



BLOCK DIAGRAM OF VOICE MAIL SYSTEM

be different because tape is moving faster as the driven reel fills up. They must be determined experimentally.

Due to tape and mechanism slippage, it is best to start the message a few seconds after the computed position. Periodic recalibration might also be necessary. Recalibration requires only to bring the tape to the starting point manually and then cold start the program.

A tape deck interfaced as described can also be used for data and program storage. However, it is not a good idea to go to all this trouble only for digital storage because with about the same amount of money you can buy a disk drive which is much better than the deck for this purpose.

If you need more message storage, either message time must be reduced or another deck added. If tape length is increased instead, searching time will increase to perhaps unacceptable levels. Since you are paying for long distance calls, it might even make financial sense to use a second deck rather than pay the phone company while waiting for the tape to be positioned over your message.

Another way to increase message storage capacity is to reduce tape speed during recording and playback. Frequency response for a given tape deck is directly proportional to tape speed. A deck with a bandwidth of 12KHz can be operated at 30% of its rated speed (1 7/8 ips) and still get a bandwidth of 3.6 KHz which is better than the bandwidth the phone company is supposed to provide in a long distance phone call.

A tape deck with two motors is required for reduced speed operation. The easiest way to change tape speed is to change the size of the pulley that drives the capstan. If the pulley cannot be changed easily (perhaps your local machinist wants too much money for making another pulley), you will have to add some form of electronic servo controller to the motor. Your deck may already have such a controller. If it does, before changing pulleys, try to figure out how to adjust it for lower speed.

AN EXAMPLE OF OPERATION

The best way to describe the voice mail system is to give an example of how it would operate. Bill, a user who has a mailbox in the system calls it up when out of town. After two rings, the system answers and in synthesized voice it says: "Please enter your access code. If you have no access code, wait on the line and somebody will talk to you". If no one is there to answer the call, the second sentence will be "If you have no access code leave your message after the tone. Your message can be up to two minutes long".

Bill enters his access code, which is 1234, using the dial pad of his Touch-Tone phone or, if the phone has a rotary dial, a battery operated DTMF transmitter (Radio Shack sells one for less than $30). There are only five users of the system and a single digit access code would be sufficient to determine who is requesting access. The long access code acts as key and prevents unauthorized use of the system.

The system recognizes a valid access code and looking up the user's name responds "Hi Bill, ready for your commands". A command starts with the # symbol, consists of two digits and ends with the * symbol. This gives 100 possible commands which are more than enough. In this system, the first digit is odd and the second digit even, reducing the number of available commands to 25 but providing at the same time a means to check for valid codes (obviously, a code whose first digit is even indicates an error).

The first thing Bill wants to do is listen to any messages that came in since his last call. So he enters #10*, the command code for playback. The synthesizer comes on with "Message one. Received two zero hours five six minutes. Caller is John. Wait for message access". After 10 seconds John's voice tells him to remember to pay the rent tomorrow because he won't be in. The synthesizer then informs him that this was the only message saying "End of private messages".

Bill wonders what time is it now so he enters #50* and he hears the synthesizer say "The time is one one hours two seven minutes". Next he enters #12* to see if any messages are left from non-users. The synthesizer says "No public messages". He then enters #30#4* which indicates that he wants to leave a message to user 4 which is

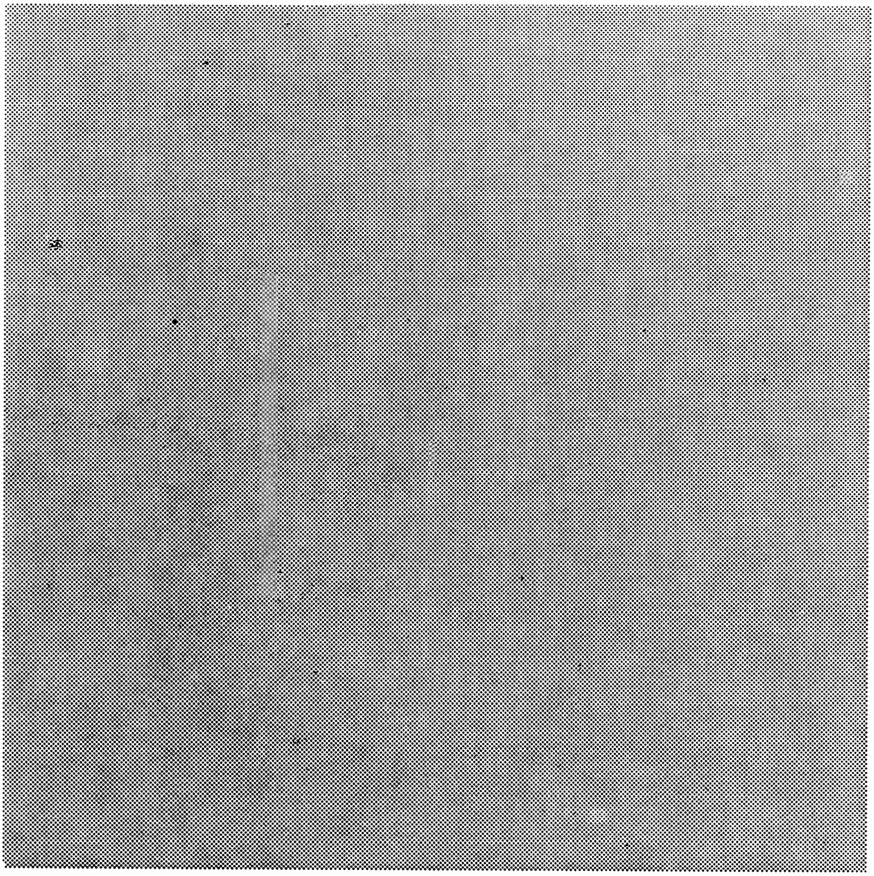
Kathy. The synthesizer says "Accepting message for Kathy" and after 20 seconds it signals him with a beep to start saying his message. Another beep at 15 seconds before the end of the 2-minute time warns him of the upcoming end of message and a double beep indicates the message time has run out.

Bill is ready to finish; so he enters #90#00* to indicate that he wants to end the session and erase all personal messages. To this the synthesizer replies "Good bye" and hangs up. If the first digit of the second number was different from zero, it would indicate the number of message Bill would like to erase. If the second digit of the second number was 1, it would indicate that the message selected is to be saved on the archival tape for future reference before it is erased from the mailbox.

This example does not cover all possible functions of the voice mail system. Giving access to some of your messages to another user, messages broadcast to all users, direct data entry to the computer, monitoring and reporting by the computer or message forwarding (the computer calls you to another number to deliver your messages or gives your new number to callers) are some of the functions possible with the voice mail system not described here.

CHAPTER 5

VOICE INTERFACES



PROJECT 23
VOICE OUTPUT

If your 64 could talk, what could it say? Find out by giving your computer the gift of speech. This project lets you digitize and play back speech with low data rate by taking advantage of an unusual property of the speech signal.

THEORY OF OPERATION

If we connect an oscilloscope to a microphone and speak into it, the screen will show a signal similar to trace A in Fig. 23.1. If we digitize this signal using an A/D converter and store it in the memory of the 64, we could play it back using a D/A converter and an amplifier. Thus the 64 would act just like a tape recorder with one important difference: random access. If we record the words "hello", "Bill", "John" and "says" on a tape recorder, every time we play the tape, we will get these words played back in the same order. If we record them in the 64, we could play them back in any order to make various phrases, such as "hello Bill" or "John says hello" and so on. In other words, we have random access to the words (vocabulary) we recorded.

While in theory this scheme would work well, it is not very practical because it requires too much memory for the storage of speech data. For example, using an 8-bit A/D converter and sampling at 8 kilosamples per second (these parameters will give telephone quality speech) we would be gobbling up memory at the rate of 8 kilobytes for every second of speech we record!

Fortunately, there is a way out of this high data rate. Human speech is a very unusual signal in many ways, so very unusual digitization schemes will work with it. For example, the speech signal has the fascinating property that its zero crossings contain just about all the intelligibility of the original. In other words, if we take the speech signal

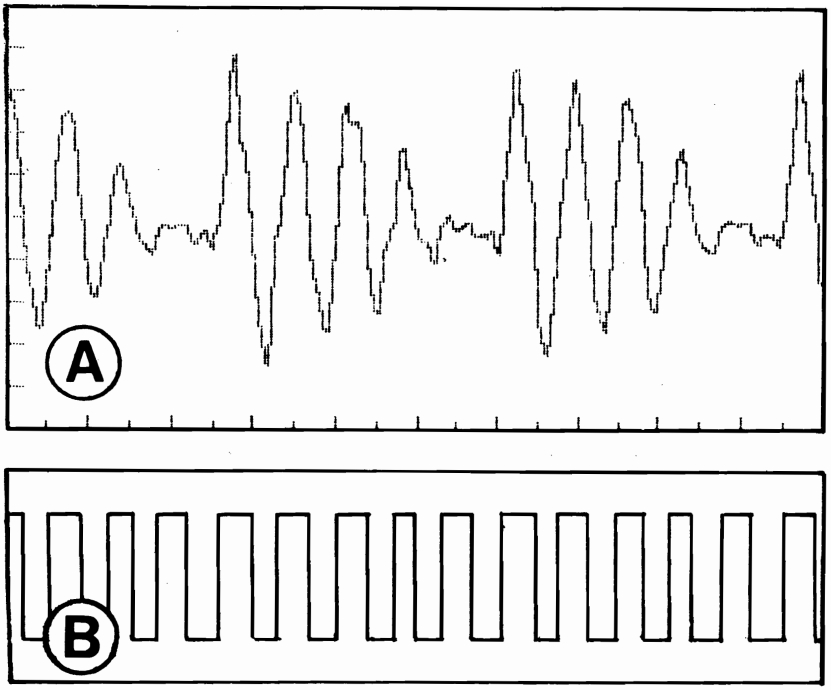
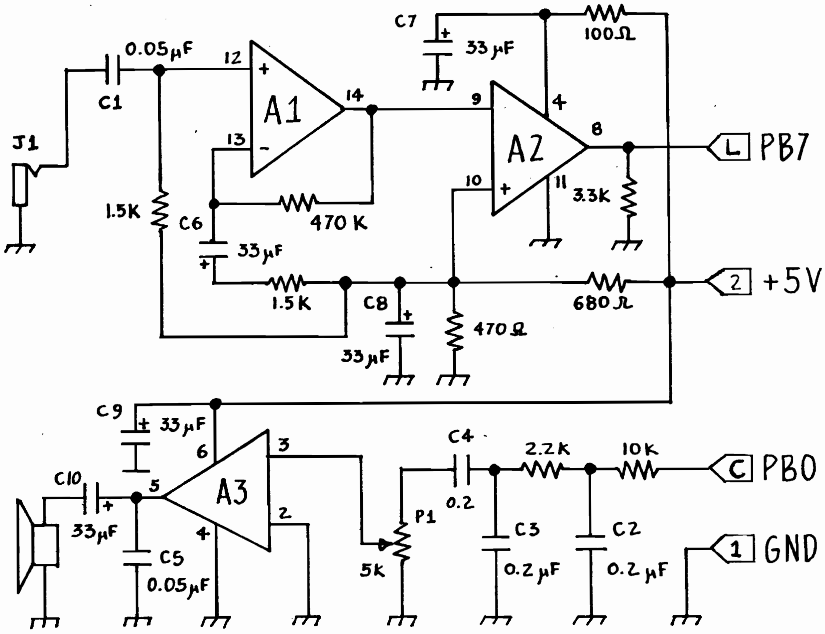


Figure 23.1 Speech Signal and its Zero Crossings

coming out of the microphone and throw away all information except the times the signal crosses zero volts, whatever remains may sound distorted and noisy but it is easily understood!

We can get the zero crossing signal by hard clipping the original as shown in trace B in Fig. 23.1. The zero crossing signal is either high or low (actually the term "zero crossing" is a sonorous misnomer. A more accurate term is "sign signal"). This is a signal easily accomodated by the 64, without further interfacing. The schematic shows a circuit that takes the speech signal from the microphone and generates the zero crossings by amplifying and clipping it. The resultant digital signal is fed into bit 7 (PB7) of the USER PORT and a machine language driver program is used to sample and store the signal.

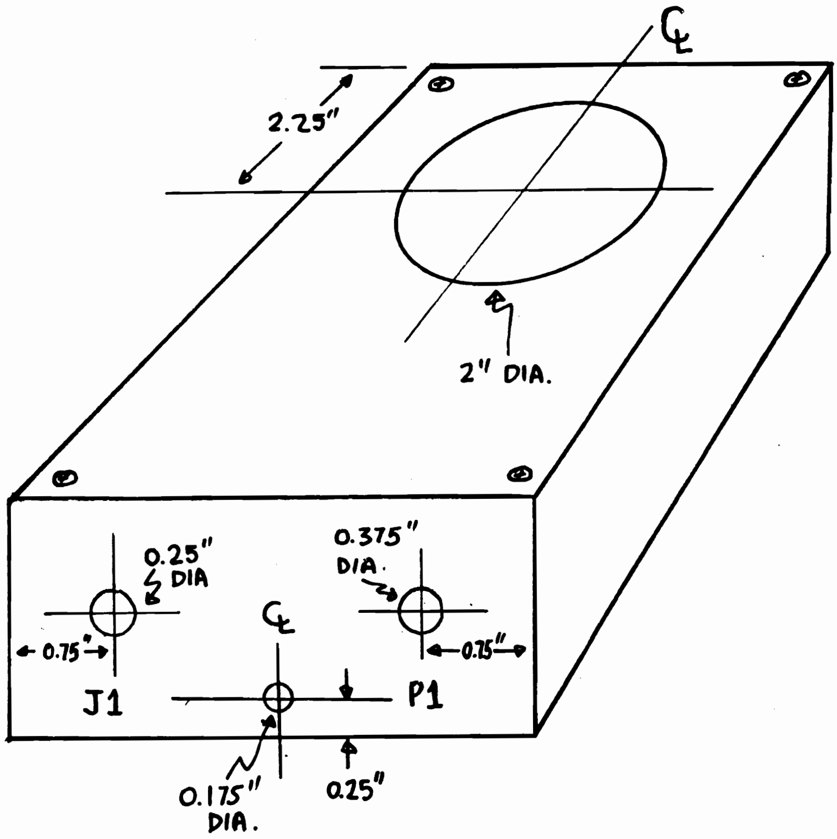


The program works as follows: The zero crossing signal is sampled every 170us. The information (either a "0" or a "1") is shifted into a temporary location. After 8 shifts, a byte is formed and stored. This goes on until the allocated memory is filled. The process is reversed to play back the digitized speech. The zero crossing signal is output via bit 0 (PB0) of the USER PORT and is low pass filtered by a resistor-capacitor network. The filtered signal is then amplified by an audio power amplifier IC, the LM386, driving a 3"speaker. This arrangement will provide adequate audio volume and quality for most applications.

CONSTRUCTION NOTES

This project can be built and packaged satisfactorily in many ways. For example, the circuit can be built on a perf board and mounted inside a small speaker cabinet, with the volume control and microphone jack attached to the back of the cabinet.

Another approach that results in a compact package that works quite well uses a plastic box to house the circuit and speaker. The circuit can be assembled either on perf board



or on a Printed Circuit (PC) board made from the artwork given. If you use a perf board, it is suggested that you follow the component placement on the PC.

The assembly is mounted in a 6×3.75×2 inch plastic box with an aluminum cover. A 2″ hole is cut out in the cover, a piece of window screen material is glued on the back and then the speaker is glued on. Follow the drawing for dimensions and placement of various holes.

A four conductor flat cable carries the signals and power from the 64. A connector with handles and strain relief as shown in Project 1 is used to plug into the USER PORT.

PROGRAMMING NOTES

The digitizing and playback is done by two different driver programs which are quite similar. Both programs use for temporary storage the random number seed area,

```
PROGRAM TO DIGITIZE VOICE
----------------------------

                  PTRL=139
                  PTRH=140
                  CTR=141
                  TEMP=142
                  DEL=143
                  PRT=56577
                  DDR=56579


78                SEI
A901      INITL   LDA #1
8D03DD            STA DDR
A900              LDA #0
858B              STA PTRL
A940              LDA #64
858C              STA PTRH
A9FB              LDA #248
858D              STA CTR
A000              LDY #0
A9F0      DELAY   LDA #240
858F              STA DEL
E68F      DELOP   INC DEL
D0FC              BNE DELOP
AD01DD    GDATA   LDA PRT
2A                ROL A
268E              ROL TEMP
E68D              INC CTR
F009              BEQ INCRM
E68F      DUMMY   INC DEL
E68F              INC DEL
E68F              INC DEL
38                SEC
B0E5              BCS DELAY
A58E      INCRM   LDA TEMP
918B              STA (PTRL),Y
A9F8              LDA #248
858D              STA CTR
E68B              INC PTRL
D0D9              BNE DELAY
E68C      ENDCK   INC PTRH
A996              LDA #150
C58C              CMP PTRH
D0D1              BNE DELAY
58                CLI
60                RTS
```

```
PROGRAM TO PLAY BACK VOICE
----------------------------

                  PTRL=139
                  PTRH=140
                  CTR=141
                  TEMP=142
                  DEL=143
                  PRT=56577
                  DDR=56579


78                SEI
A901      INITL   LDA #1
8D03DD            STA DDR
A900              LDA #0
858B              STA PTRL
A940              LDA #64
858C              STA PTRH
A9FB              LDA #248
858D              STA CTR
A000              LDY #0
A9F0      DELAY   LDA #240
858F              STA DEL
E68F      DELOP   INC DEL
D0FC              BNE DELOP
268E      GDATA   ROL TEMP
2A                ROL A
8D01DD            STA PRT
E68D              INC CTR
F009              BEQ INCRM
E68F      DUMMY   INC DEL
E68F              INC DEL
E68F              INC DEL
38                SEC
B0E5              BCS DELAY
B18B      INCRM   LDA (PTRL),Y
858E              STA TEMP
A9F8              LDA #248
858D              STA CTR
E68B              INC PTRL
D0D9              BNE DELAY
E68C      ENDCK   INC PTRH
A996              LDA #150
C58C              CMP PTRH
D0D1              BNE DELAY
58                CLI
60                RTS
```

so it is advisable not to use the RND function in BASIC if your program has voice output.

The sampling time is determined by the time it takes to execute a loop through the program. Thus, every path through the program must have the same execution time, equal to the sampling period. To equalize path lengths, a program segment labelled DUMMY is included.

The digitized data goes in memory starting at a location defined by the initial values of PTRH and PTRL (starting location=PTRH*256+PTRL). In the listing as shown, starting location for data storage is 64*256+0=16384. The last location where data is stored is determined by the constant in the routine ENDCK. This constant is the high order byte of this address. The low order byte is always 0. Thus the end of speech data is allocated in 256 byte increments. The value of 150 ($96) shown in the listings indicates an end-of-data address of 38400 ($9600). By changing this constant the end of data can be moved anywhere in memory space.

During playback, a portion of the recording can be played back by POKEing different values in the locations the begin and end pointers are stored. This way you can have random access to the vocabulary. Since voice data is stored as bytes in memory, it can be edited using memory move, insert and delete commands from a monitor program like 64MON.

If you would rather not get into the programming details, you can type in the BASIC program that will set the drivers in place and allow you to record and playback speech easily by making selections from a menu.

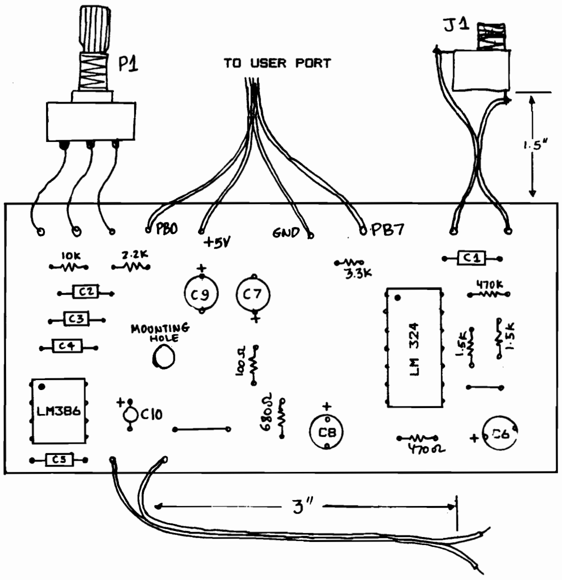
The BASIC program set up the following memory map:

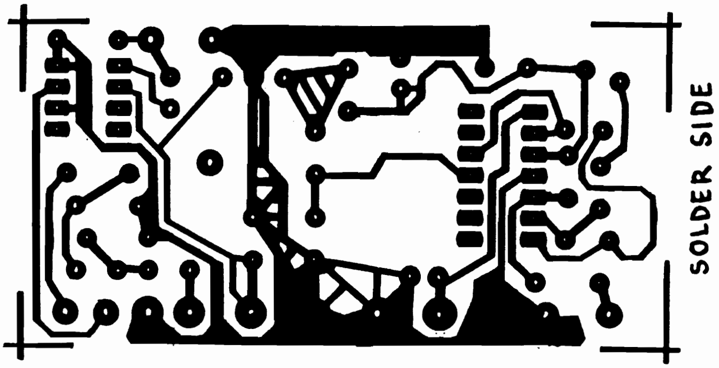
```
 2048: Start of BASIC Workspace
16128: End of BASIC Workspace
16129: Start of Record Driver
16198: Start of Playback Driver
16205: PTRL PLAYBACK
16209: PTRH PLAYBACK
16384: Start of Voice Data
38400: End of Voice Data
```

If you would like to get sophisticated, you could store the speech data in the RAM from address $A000 to $FFFF and thus leave the BASIC workspace to BASIC alone. This RAM is hidden under the operating system and BASIC ROM's and remains unused. It can be switched on by setting some bits in the port on-board the CPU (mapped as memory location 1). After the memory switch, BASIC will dissapear, so all programs must be written in machine language. Only experienced programmers in machine language have a chance of pulling this off successfully.

```
100 REM  ** VOICE OUT **
101 REM
110 REM SET TOP OF BASIC
120 POKE 55,0: POKE 56,63
125 POKE 51,0: POKE 52,63: CLR
130 REM PUT DRIVERS IN PLACE
140 FOR I=1 TO 138
150 READ A: POKE 16128+I,A:NEXT I
200 REM CREATE MENU
210 PRINT CHR$(147):PRINT: PRINT
220 PRINT TAB(12);"1. DIGITIZE VOICE"
225 PRINT
230 PRINT TAB(12);"2. PLAYBACK VOICE"
250 PRINT:PRINT
255 PRINT TAB(10);"PRESS THE NUMBER NEXT"
257 PRINT
260 PRINT TAB(13);"TO THE FUNCTION"
263 PRINT
265 PRINT TAB(15);"TO SELECT IT"
270 GET A$: IFA$=""GOTO270
280 IF ASC(A$)=50 GOTO400
295 IF ASC(A$)<>49 GOTO 200
300 REM DIGITIZE VOICE
310 PRINT CHR$(147)
320 PRINT"  HOLD THE MICROPHONE"
330 PRINT: PRINT"CLOSE TO YOUR LIPS AND"
340 PRINT: PRINT " SPEAK LOUD AND CLEAR"
350 PRINT: PRINT "    FOR 30 SECONDS"
360 PRINT: PRINT "PRESS ANY KEY TO START
370 GET A$:IF A$="" GOTO370
380 SYS 16129
390 GOTO 200
400 REM PLAYBACK
410 PRINT CHR$(147):PRINT
420 PRINT"CHOOSE PLAYBACK MODE": PRINT
430 PRINT" 1. WHOLE VOICE BUFFER"
440 PRINT " 2. PORTION OF VOICE BUFFER"
445 PRINT:PRINT
450 GET A$:IFA$=""GOTO450
460 IF ASC(A$)=49 GOTO 490
470 IF ASC(A$)=50 GOTO 500
480 GOTO 400
490 SYS 16198: GOTO200
```

```
500 PRINT "GIVE STARTING ADDRESS IN"
504 PRINT "DECIMAL. ADDRESS MUST BE"
507 PRINT "BETWEEN 16384 AND 38400."
510 PRINT: INPUT SA
520 UA=INT(SA/256)
530 LA=SA-256*UA
540 POKE 16205,LA
550 POKE 16209,UA
560 PRINT:PRINT "GIVE ENDING ADDRESS"
562 PRINT "ABOVE CONSTRAINTS APPLY."
564 PRINT "ALSO, ENDING ADDRESS MUST"
566 PRINT "BE AT LEAST 260 MORE THAN"
568 PRINT "THE STARTING ADDRESS."
570 PRINT:INPUT EA
580 EA=INT(EA/256)
590 POKE 16260,EA
600 SYS 16198
605 PRINT
610 PRINT "IF YOU WANT TO HEAR IT"
615 PRINT "AGAIN PRESS Y ELSE N"
620 GETA$:IF A$=""GOTO620
630 IF A$<>"Y" GOTO 650
640 SYS 16198: GOTO610
650 POKE 16202,0: POKE16206,64
660 POKE 16257,150: GOTO 200
5000 DATA 120,169,1,141,3,221,169,0,133
5010 DATA 139,169,64,133,140,169,248,133
5020 DATA 141,160,0,169,240,133,143,230
5030 DATA 143,208,252,173,1,221,42,38
5040 DATA 142,230,141,240,9,230,143,230
5060 DATA 143,230,143,56,176,229,165,142
5070 DATA 145,139,169,248,133,141,230
5080 DATA 139,208,217,230,140,169
5090 DATA 150,197,140,208,209,88,96
5100 DATA 120,169,1,141,3,221,169,0,133
5110 DATA 139,169,64,133,140,169,248,133
5120 DATA 141,160,0,169,240,133,143,230
5130 DATA 143,208,252,38,142,42,141,1
5140 DATA 221,230,141,240,9,230,143,230
5160 DATA 143,230,143,56,176,229,177,139
5170 DATA 133,142,169,248,133,141,230
5180 DATA 139,208,217,230,140,169,150
5190 DATA 197,140,208,209,88,96
```





PARTS LIST

1. PC 23
2. Microphone: Dynamic, 200-600 Ohm impedance
3. Speaker: 3 inch diameter or larger
4. J1: 3.5 mm miniature jack
5. A1,A2: 1/4 LM324N
6. A3: LM386N
7. C1,C5: 0.05 uF mylar capacitor
8. C2,C3,C4: 0.2 uF ceramic disk capacitors
9. All other capacitors: 33 uF, 16V electrolytics
10. All resistors: 1/4 W, 5%
11. P1: 5K miniature potentiometer.

PROJECT **24**

ALLOPHONE SPEECH SYNTHESIZER

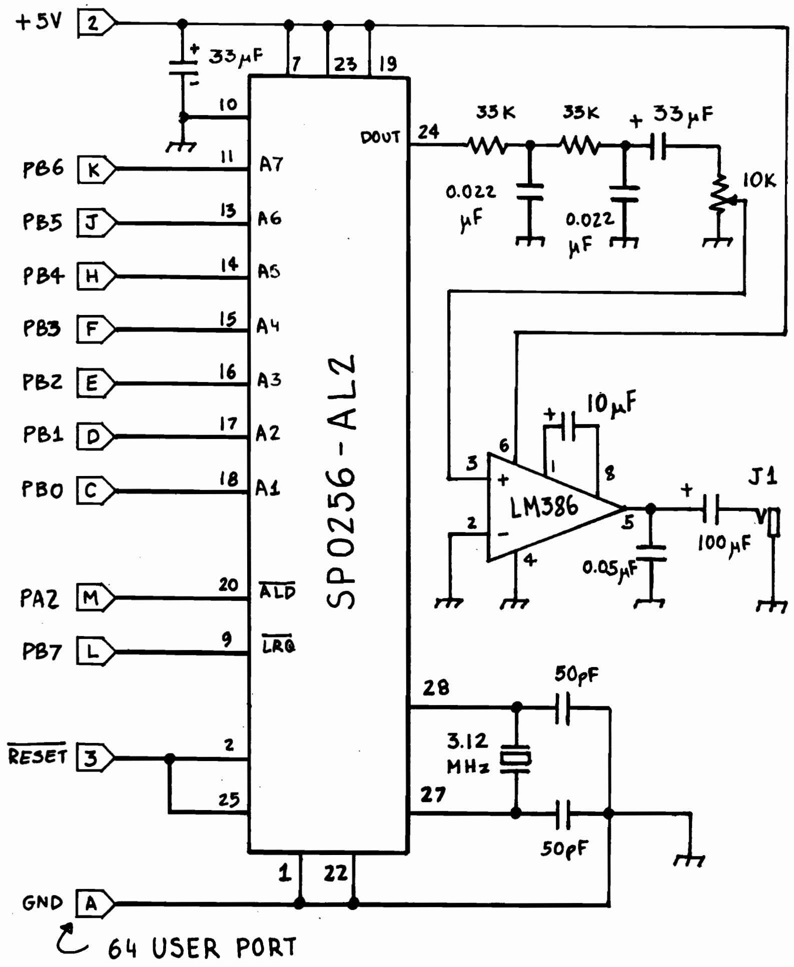
A Speech Synthesizer produces speech by pronouncing together strings of basic speech sounds like phonemes or allophones. Its advantages are extremely low data rate (only a few bytes are required to say a word) and unlimited vocabulary. It has a mechanical sounding voice that is just right for games. Using only two IC's, this project will give you many hours of fun, building words and phrases by stringing together allophones.

THEORY OF OPERATION

This speech synthesizer is based on the General Instrument SP-0256-AL2 speech synthesizer IC chip. Internally, the chip contains the equivalent of an electronic vocal tract, 2K ROM and a microprocessor. Although the IC and the theory behind its operation are very complex, the average user needs no special knowledge to use it effectively.

For our purposes, the important signals on the synthesizer are: 8 address lines (A1-A8), 1 audio output line (DOUT), 1 address latch strobe input (ALD) and a line indicating that the synthesizer is talking (LRQ).

In operation, when LRQ is low, the synthesizer is ready to accept data. The data is a number from 0 to 63, signifying one of the 64 allophones the synthesizer can pronounce. The number corresponding to the desired allophone is placed on the address lines and the ALD signal is brought momentarily low to latch the data (the data is called "address" because in effect it addresses one of the 64 allophones stored in the synthesizer). After the address is latched, the synthesizer brings LRQ high to signify it is talking. The speech signal corresponding to the selected allophone comes out of the DOUT pin as a pulse width modulated square wave. It is filtered by the RC network and amplified by the second IC, the LM386 audio amplifier



and spoken through an external speaker. When the talking is finished, LRQ goes low signifying the synthesizer is ready for another allophone.

The synthesizer has a built-in oscillator that requires a 3.12 MHz crystal. A 3.579545 MHz crystal used in color TV's is much easier to find and it will work satisfactorily.

For further details on the operation of the synthesizer IC the reader is referred to its specification sheet and the application note titled "Allophone Speech Synthesis Technique" by Janet May, both available from: General Instrument Microelectronics Division, 600 West John St., Hicksville, NY 11802.

The synthesizer is interfaced to the USER PORT of the 64. The ALD strobe is generated by pulsing PA2 every time the port is written. LRQ is wired to the MSB of the port and it is sensed by reading the port. If it is set, the reading will be larger than 127.

The circuit is powered from the 5V supply of the 64. The synthesizer chip draws about 90 mA from this supply. Unlike the zero crossing voice output, an external speaker is used, because good audio quality is required. Radio Shack sells a number of small speakers in enclosures priced under $20. Any one of them will do fine. Or you may use the speaker from an old stereo.

PROGRAMMING NOTES

No machine language driver is required for this project because the synthesizer chip does all the fast footwork within itself. Programming from BASIC, you will first need to initialize the USER PORT so that all bits except the most significant (MSB) are outputs (POKE 56579,127). To say an allophone, you POKE its number to the USER PORT after making sure that the synthesizer is ready to accept it (it is ready when it sets the MSB of the USER PORT to zero). The allophone number is accepted by the synthesizer when PA2 is momentarily pulsed high.

At this point, you have interfaced the synthesizer, it has been initialized and you are ready to have some fun. Check the table for the allophones and their codes (numbers). The allophones do not look much like the alphabet. Rather, they are elementary sounds from which all the spoken words of English can be approximated. The allophone sequence required to say a word usually has little to do with spelling. Say the word, listening for the sounds you make. Then pick what you think will be an equivalent sequence of allophones.

For example, the word "computer" in allophones will be:

ALLOPHONE TABLE

		Silence	
PA1	0	PAUSE	10ms
PA2	1	PAUSE	30ms
PA3	2	PAUSE	50ms
PA3	3	PAUSE	100ms
PA3	4	PAUSE	200ms
		Short Vowels	
*/IH/	12	S**I**T	70ms
*/EH/	7	**E**ND	70ms
*/AE/	26	H**A**T	120ms
*/UH/	30	B**OO**K	100ms
*/AO/	23	**AU**GHT	100ms
*/AX/	15	S**U**CCEED	70ms
*/AA/	24	H**O**T	100ms
		Long Vowels	
/IY/	19	S**EE**	250ms
/EY/	20	B**EI**GE	280ms
/AY/	6	SK**Y**	260ms
/OY/	5	B**OY**	420ms
/UW1/	22	T**O**	100ms
/UW2/	31	T**O**	260ms
/OW/	53	B**EAU**	240ms
/AW/	32	**OU**T	370ms
/EL/	62	SADD**LE**	190ms
		R — Colored Vowels	
/ER1/	51	F**IR**	160ms
/ER2/	52	F**IR**	300ms
/OR/	58	ST**ORE**	330ms
/AR/	59	AL**AR**M	290ms
/YR/	60	CL**EAR**	350ms
/XR/	47	RE**PAIR**	360ms
		Resonants	
/WW/	46	**W**OOL	180ms
/RR1/	14	**R**URAL	170ms
/RR2/	39	B**R**AIN	120ms
/LL/	45	**L**AKE	110ms
/YY1/	49	**Y**ES	130ms
/YY2/	25	**Y**ES	180ms

Voiced Fricat.

/VV/	35	**V**EST	190ms
/DH1/	18	**TH**EY	290ms
/DH2/	54	**TH**EY	120ms
/ZZ/	43	**Z**OO	210ms
/ZH/	38	A**Z**URE	190ms

Voiceless Fricat.

*/FF/	40	**F**OOD	150ms
*/TH/	29	**TH**IN	180ms
*/SS/	55	VE**S**T	90ms
/SH/	37	**SH**IP	160ms
/HH1/	27	**H**E	130ms
/HH2/	57	**H**OE	180ms
/WH/	48	**WH**IG	200ms

Voiced Stop Cons.

/BB1/	28	**B**USINESS	(SOFT)	50ms
/BB2/	63	**B**USINESS		50ms
/DD1/	21	COUL**D**		70ms
/DD2/	33	**D**O		160ms
/GG1/	36	**G**UEST		80ms
/GG2/	61	**G**OT		50ms
/GG3/	34	WI**G**		160ms

Voiceless Stop Cons.

/PP/	9	**P**OW	210ms
/TT1/	17	PAR**T**	100ms
/TT2/	13	**T**O	140ms
/KK1/	42	**C**AN'T	160ms
/KK2/	41	S**K**Y	190ms
/KK3/	8	**C**OMB	120ms

Affricate

/CH/	50	**CH**URCH	190ms
/JH/	10	DO**DG**E	140ms

Nasal

/MM/	16	**M**ILK	180ms
/NN1/	11	THI**N**	140ms
/NN2/	56	**N**O	190ms
/NG/	44	A**N**CHOR	220ms

** These allophones can be doubled.*

```
100 REM ALLOPHONE SYNTHESIZER
101 REM
110 REM DATA STATEMENTS ARE CODES
120 REM FOR WORDS 0-9
130 REM
200 REM INITIALIZATION
201 REM
210 ADR=56576:DAD=56578
220 BDR=56577:DBD=56579
230 POKE DBD,127
240 A=PEEK(DAD)OR 4
250 POKE DAD,A
260 POKE ADR,(PEEK(ADR) OR 4)
270 DIM SP(100)
280 N=0:I=0
299 REM
300 REM SET UP ALLOPHONE ARRAY
301 REM
310 I=I+1
320 READ A:IF A=255 GOTO 400
330 SP(I)=A:N=N+1
340 GOTO 310
399 REM
400 REM SPEAK
401 REM
410 FOR I=1 TO N
420 IF PEEK(BDR)>127 GOTO 420
430 POKE BDR,SP(I)
440 POKE ADR,(PEEK(ADR) OR 4)
445 POKE ADR,(PEEK(ADR) AND 251)
450 NEXT I
460 GET A$:IF A$="" GOTO 460
470 GOTO 400
500 DATA 43,19,14,53,4,4,46,15,15,11,4,4
510 DATA 13,31,4,4,29,51,19,4,4
520 DATA 40,58,4,4,40,6,19,35,4,4
530 DATA 55,12,41,55,4,4,55,7,35,7,11,4,4
540 DATA 20,13,4,4,56,6,11,0,255
```

KK1-AX-MM-PP-YY1-UW1-TT2-ER1

Quite different from spelling! Looking up the codes for the allophones we get:

42-15-16-9-49-22-13-51

If we POKE these numbers consecutively into the USER PORT we will hear a synthetic "computer" followed by a steady hum. No, it did not break. What happens is the synthesizer will keep saying the last allophone entered, in this case ER1. To turn the hum off, POKE one of the pauses into the USER PORT.

Do not hesitate to experiment with various allophones to get things sounding right. Keep in mind that stressed vowels are always longer than unstressed ones. If the stress is in the wrong place try lengthening the allophone (if that is permissible – check with the table) or try a similar but longer allophone.

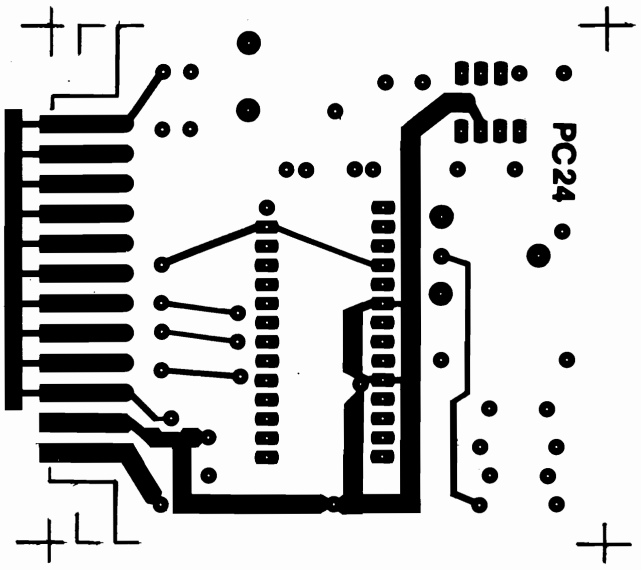
It is also helpful to remember that there are many cases where no matter how hard you try, the results will be marginal. 64 allophones are simply not enough to cover all the subtleties of human speech.

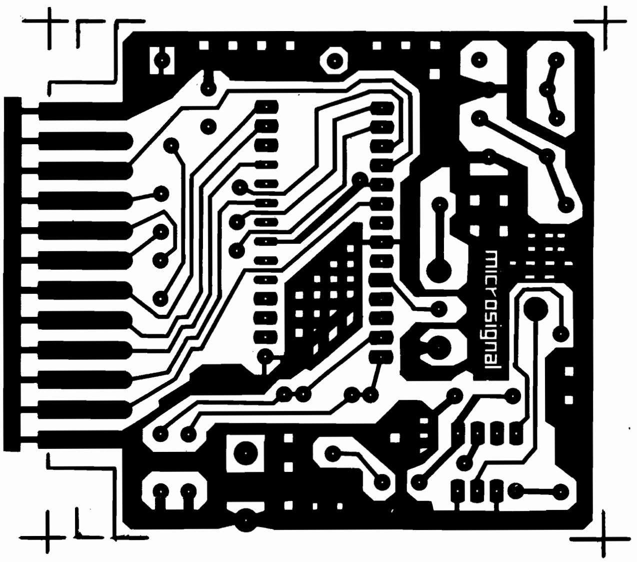
The BASIC program makes it easy to try different sounds. The allophone codes are in the DATA statements. 255 signifies end of data. Always make sure the last two data is 0 followed by 255. Zero will turn off the synthesizer output and 255 will terminate the data input. Once the data is spoken, you can replay it by pressing the space bar on the keyboard. To exit, press the RUN/STOP key. The data shown in the BASIC program listing will result in the synthesizer speaking the numbers 0 through 9.

You may notice the following interesting phenomenon: A friend hears the synthesizer talk but does not understand most of it. When you tell him what the synthesizer said and he listens to it again, he can understand everything clearly. There is a similar phenomenon with vision.

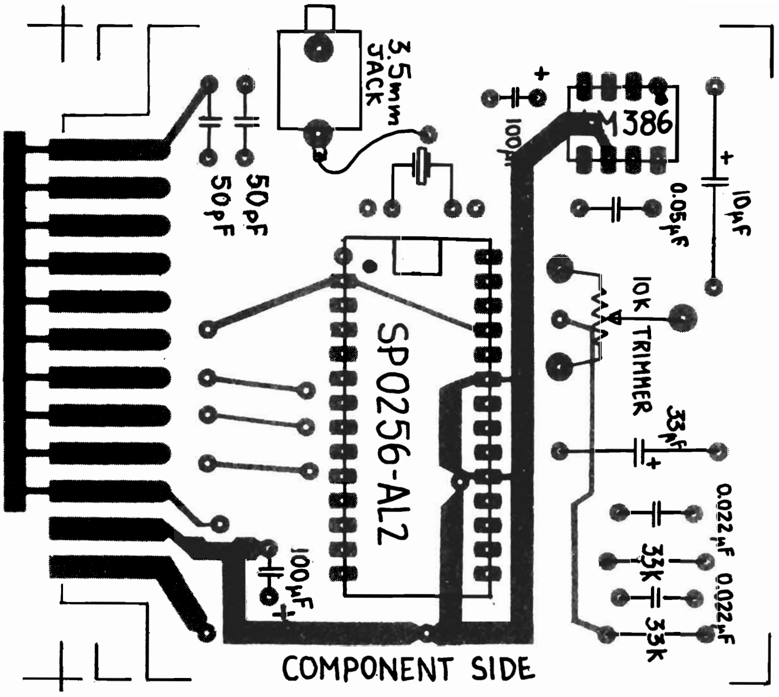
PARTS LIST

1. PC # 24
2. IC's: SP-0256-AL2. LM 386. Note: Radio Shack now sells SP-0256-AL2. PART # 276-1784

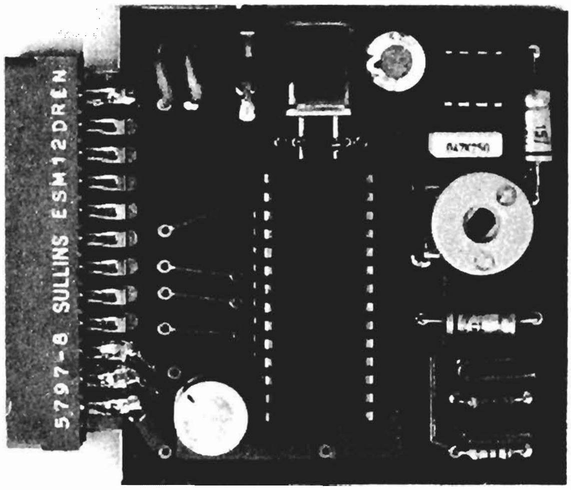




3. XTAL: 3.12 MHz Crystal. A TV colorburst crystal will also work, you get higher pitched voice.
4. 10K trimmer pot. See board for lead configuration.
5. Resistors: 2×33k, 1/4W, 5%
6. Capacitors: 2×50 pf, 2×0.022 uF mylar, 0.05 mylar or



COMPONENT SIDE

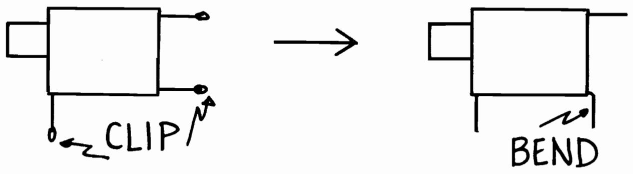


ceramic disk, 2×100 uF/16V radial lead electrolytics, 10 uF/16 volt axial lead electrolytic.

7. PC connector, 12/24 contacts, 0.156 spacing, TRW CINCH 50-24A-30
8. 3.5 mm, plastic enclosed jack.

ASSEMBLY INSTRUCTIONS

1. Use sockets for IC's. All components go to the component side. Soldering is done on the other side. Board plugs in with component side facing up.
2. Form connector tabs so that their spacing is about the thickness of the board and then slip it on the board and solder.
3. If you cannot get a trimmer that fits the PC hole pattern, improvise with whatever you can get. Bend or file pins to fit. Or use heavy wire to make the connections and at the same time provide support.
4. Use a plastic encased 3.5 mm jack. There are a few different kinds of such jacks but the PC board is made for the most common type. Clip and bend the leads as follows:



   If you have trouble finding a jack that fits, glue a small bracket on the board with epoxy so that you can mount any jack, or forget about the jack and directly solder a cable on board which on the other end has a plug that fits your speaker. Drill a hole on the board and use it to tie the cable for strain relief.

5. There are four holes for the XTAL because their lead spacing varies from style to style. Use the holes that work best for your crystal.
6. Do not plug the synthesizer in while the computer is on. Besides taking a risk of ruining them both, it won't work. It needs the power-on reset signal from the computer to initialize.

PROJECT 25
VOICE INPUT

Saying "Print 3 plus 4" is much easier than typing the same thing. But can your computer understand your voice? Here is how to build a voice input peripheral, which, although not a speech recognizer, will let you control your computer by voice to amaze your friends and neighbors.

THEORY OF OPERATION

Speech is the natural means of communication between humans, so it would be ideal if we could communicate the same way with computers. Unfortunately, the information conveyed by the speech signal is encoded in the variations of the signal in a way that remains largely unknown. Thus, building voice recognition machines is a black art.

Commercially available speech recognizers today have a vocabulary of 16 to 64 words. They must be trained to the vocabulary by the user and, once trained, they can only recognize words drawn from their vocabulary and spoken by the person that trained them. Even then, they recognize about 95-98% of the words spoken. Most of them cost thousands of dollars and thus they are not within the budget of most experimenters. Despite their limitations, these recognizers have many uses and are fascinating to play with.

There is a way, however, to experiment (and play) with voice input that avoids all difficulties and cost of speech recognition. Instead of recognizing what each word in a sentence means, we just count the number of words. What good is that? A lot if used properly. Take for example George. "George" was the name of a toy van that was controlled by voice and appeared in the market several years ago.

George would go straight, left, right, back and stop

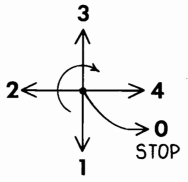
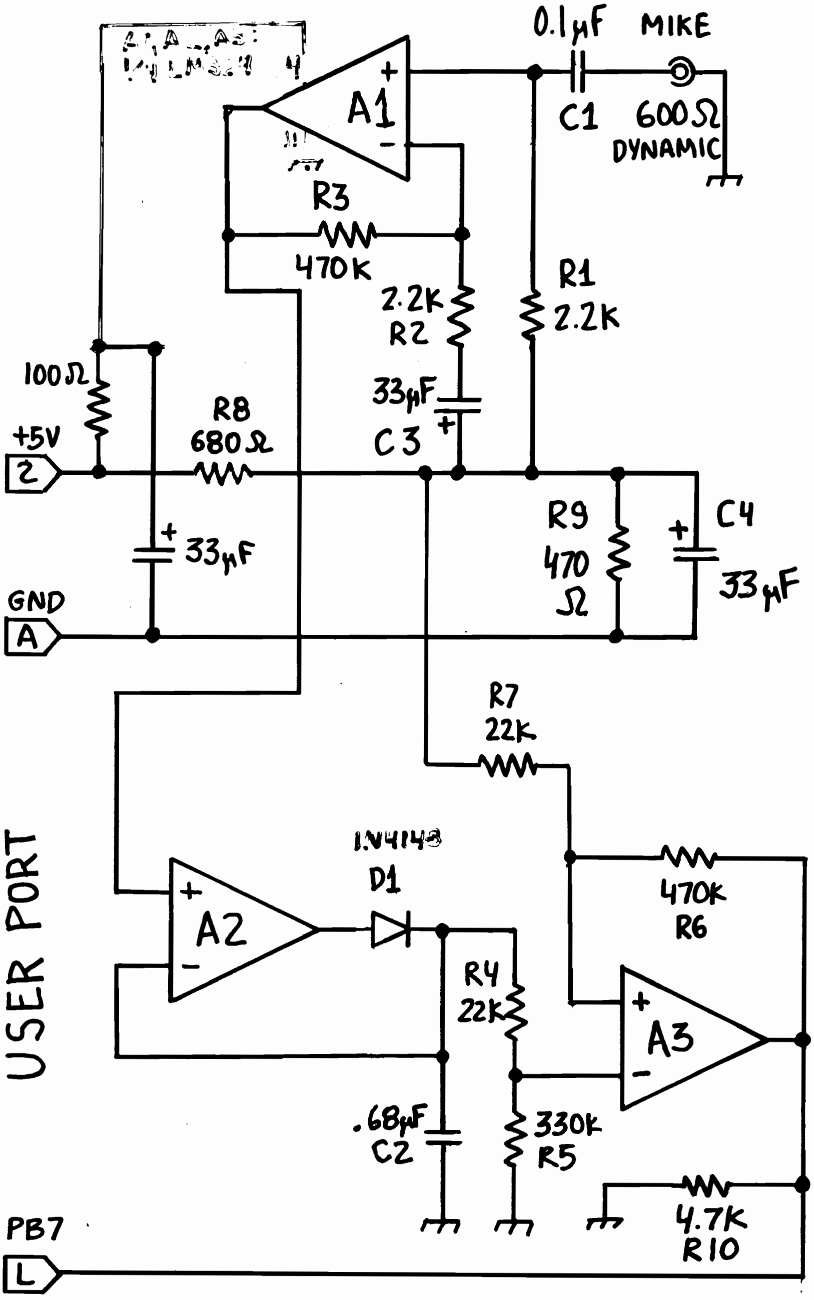


Figure 25.1 "George" Changes Direction of Motion Depending on the Value of the Modulo Counter

under voice control. To determine the action George was to take, the number of words heard by George was counted. A counter inside George could count from 0 to 4 and then start again from 0. The counter was incremented every time a word was heard. When the counter indicated 0, George stopped. When it indicated 1, it went straight. Two send it to the left, 3 to the right, and 4 made it back up. See Fig. 25.1.

This of course if fun but does not amount to much by itself. Now take an operator who can think and talk at the same time. Suppose that George is stopped at the moment and the operator wants it to move straight. Straight is 1, so he says "Straight" and George starts moving forward. To stop him again, he must say four words to bring the counter to zero. If he says "Stop" four times George would stop but it would be rather boring after a few times. So he says "George please stop now". And lo and behold, George obeys as if he understood what the operator said. This makes for a very impressive demonstration as the number of words for a function varies with the current position of the counter. In addition, different words, making humorous suggestions, can be used each time.

The trick to making George understand human speech is to remember the current position of the counter, calculate quickly how many words are needed to bring the counter to the desired position and then make a sentence that has the correct number of words, choosing the words so that they follow a theme (for example, George might be supposed to be reluctant to go under the table so you coax, beg and threaten him to go in the desired direction around any obstacles). Adding voice to George with the synthesizer can take the analogy (or parody) as far as your imagination will let you.


A1, A2, A3:
1/4 LM324
A1
+
−
0.1μF
C1
MIKE
600Ω
DYNAMIC
R3
470K
2.2K
R2
R1
2.2K
100Ω
+5V
2
R8
680Ω
33μF
C3
+
+
33μF
GND
A
R9
470
Ω
C4
+
33μF
R7
22K
1N4148
D1
470K
R6
USER PORT
A2
+
−
R4
22K
A3
+
−
.68μF
C2
330K
R5
4.7K
R10
PB7
L

This concept can be extended to tasks other than motion control. With a little creativity, some programming and a lot of practice, you can really amaze people at the understanding your computer shows. There are also some possibly practical applications. The circuit is basically a voice sensor so that it can be used to detect sound for security purposes, or to turn off lights when nobody is in the room or to sound an alarm (or start rocking the crib) when the baby wakes up and cries. Other applications include selection from menus on the screen (count out loud until the desired number is reached) and educational applications for children. Applications for aiding the handicapped are possible but in this case the expense of a true speech recognizer is certainly justified.

Coming to the circuit, the output of the microphone is amplified by A1, bringing it to an amplitude of about 3 volts p-p. A2 is connected as peak detector and as result, the capacitor C2 is forced to follow the positive peaks of the signal. A3 is connected as Schmitt trigger. When the voltage across C2 rises above a predetermined level, the output of A3 becomes zero. This is sensed by the computer via the USER PORT.

The sensitivity of the circuit is set rather low to avoid picking up extraneous noises. You must hold the microphone close to your mouth, almost touching the lips and you must talk a bit louder than normal. A second level of noise rejection is performed using criteria based on the minimum duration of a word.

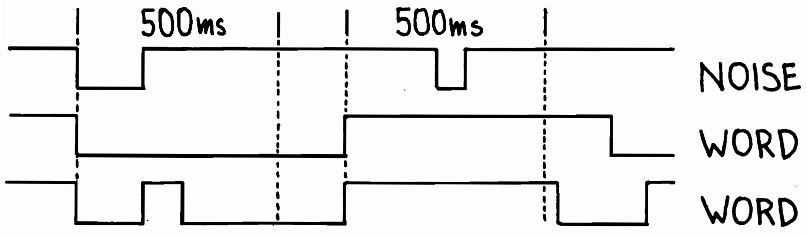
In conversational speech, words are often pronounced together in groups of two or more words without any silence separating them. When talking to voice input devices, each word should be pronounced separately. You can easily do this if you are a little careful while talking.

PROGRAMMING NOTES

All programming for this project can be done in BASIC because there are no timing requirements that BASIC cannot handle.

The output of the voice input circuit is zero when a sound is present. Due to the action of the peak holding

```
100 REM VOICE GEORGE
101 REM
105 PRINT CHR$(147)
108 PRINT "WAIT TEN SECONDS"
110 FOR I=1 TO 500: NEXT I
115 PRINT CHR$(147)
120 FOR I=0 TO 1023
125 POKE55296+I,1
130 NEXT I
135 POKE 56579,0: PRT=56577
140 DIM X(5): DIM Y(5)
145 X(1)=0: X(2)=-1: X(3)=0: X(4)=1
150 Y(1)=1: Y(2)=0: Y(3)=-1: Y(4)=0
155 X=20:Y=12: CTR=1
160 POKE 1063,49
165 DX=0: DY=1
170 TL=0: TH=0
180 IF CTR=0 GOTO 200
190 GOTO 400
200 IF PEEK(PRT)>127 GOTO 180
210 REM SOUND IS DETECTED
220 TL=TL+1
230 IF PEEK(PRT)<128 GOTO 220
240 REM SOUND STOPPED
250 TH=TH+1
260 IF PEEK(PRT)<128 GOTO 600
270 REM END OF SOUND?
280 IF TH<20 GOTO 250
290 REM NOISE PULSE?
300 IF TL<20 GOTO 170
310 REM UPDATE DIRECTION
320 CTR=CTR+1
330 IF CTR=5 THEN CTR=0
335 POKE 1063,CTR+48
340 IF CTR=0 GOTO 170
350 TH=TH+1
360 IF PEEK(PRT)<128 GOTO 390
370 IF TH<60 GOTO350
380 DX=X(CTR): DY=Y(CTR)
390 TL=0: TH=0
395 GOTO 200
400 REM NEXT STEP
410 X=X+DX: Y=Y+DY
420 IF X>39 GOTO 470
430 IF X<0 GOTO 490
440 GOTO 500
470 X=0: Y=Y+1
480 GOTO 500
490 X=39: Y=Y-1
500 IF Y>24 THEN Y=0
510 IF Y<0 THEN Y=24
520 L=1024+40*Y+X
530 P=102
540 IF PEEK(L)=102 THEN P=81
550 POKE L,P
560 FOR I=1 TO 50
570 IF PEEK(PRT)<128 GOTO 590
580 NEXT I
590 TL=0: TH=0: GOTO 200
600 REM GAP IN SOUND
610 TL=TL+TH: GOTO 220
```



capacitor, C2, the output does not present any jitter for the duration of the sound. A sound, however, is not a word. We define a word as a sound that lasts more than 500 ms. This way, almost all noises are rejected because they normally last less than 500 ms. On the other hand, all words are longer than that, especially if pronounced fully and clearly.

Certain words contain gaps of silence because they include plosive consonants like p,t,k. For example the word "aptitude" contains two gaps after the t's. We must not count parts of a word as separate words because this will throw the counter off. Fortunately, gaps are never longer than 500 ms so we can easily discriminate them from gaps between words that last always more than that (unless one is making a great effort to talk as fast as possible which of course is not advisable).

Figure 25.2 shows the timing diagrams for each of these cases. A sentence is considered ended when there is no sound for more than 1.5 seconds.

Our program then works as follows: It looks at the most significant bit (MSB) of the USER PORT which normally is "1". When it goes to "0", it starts measuring time (by incrementing the counter TL (Timer Low) and stops when it goes high again. It then measures the time the MSB remains high in counter TH (Timer High). If this time is longer than 500 ms (TL>20) then we decide the sound has ended. If the MSB of the USER PORT goes low before 500 ms, then TH is added to TL on the theory that we encountered a gap within a possible word. TL is then incremented until the MSB becomes high again. TH is cleared at this time and we try again to see if the time the MSB remains high exceeds 500 ms. If it does, we check TL. If TL is shorter than 500 ms, the sound is considered noise, both TL and TH are set to zero and the scanning of the MSB continues. If TL is longer than 500 ms, a word has been detected and CTR is incremented. CTR is a modulo N counter. After we increment it, we must check to see if it has exceeded N, in which case we set it to zero. The counter in "George" is modulo 4 so when it reaches four and a word occurs it must be set to zero.

The program "VOICE GEORGE" emulates the operation of George. As it moves about the screen, it lays a track. When it crosses its own track, the symbol he uses for the track changes. The value of CTR is always shown in the upper right tand corner. It is very useful while getting the hang of talking to George and making it go where you want to go. When it is zero, George is stopped. When it is one, he moves down, two he moves left, three he moves up and four he moves right.

The delay at the beginning of the program is due to setting the color memory so that the characters have the right color for the track to appear on screen. If this step is omitted, the track will not be visible because the track characters are not printed but directly poked in the screen memory. The color of the track is white with the listing as shown. To change its color, change the number POKED in line 125 to any value between 1 and 15.

The number 20 in line 280 is the minimum length a sound must have in order to be considered a word. Reducing this increases the sensitivity of George to quiet voices but at the same time it increases sensitivity to extraneous noises.

A feature that might be worthwhile to add is a routine that drives the SID in the 64 to produce a sound (perhaps a tick) every time George makes a step. A good place to insert the sound generating routine is right after line 550.

George is a fun toy for kids and most are captivated by it. With a little programming effort in BASIC, you can write various educational games around it, from sentence formation to modulo arithmetic.

CHAPTER 6

A/D AND D/A INTERFACES

PROJECT 26
SIMPLE D/A CONVERTER

A very simple bridge between the digital and analog worlds can be made using only resistors connected to the USER PORT. Although not accurate enough for precision work, this D/A can be used to generate audio signals and to demonstrate the operation of the D/A converter.

THEORY OF OPERATION

This project uses the R-2R ladder which is the heart of all modern D/A converters. One of its advantages is the use of only two values of resistance, R and 2R. Only the ratio of the resistances is important, not their actual value making it relatively insensitive to variations in temperature.

In operation, the output at the point marked 'LADDER OUT' is 1/2 the voltage at PB4, plus 1/4 the voltage at PB3, plus 1/8 the voltage at PB2, plus 1/16 the voltage at PB1, plus 1/32 the voltage at PB0. Thus the resulting output voltage is a properly weighted sum of the individual inputs and represents the value of the binary number at the inputs PB0 to PB4.

The output impedance of the ladder is R Ohms (regardless of how many steps it has. Electrical Engineering undergraduates usually have to prove this paradox as an exercise).

To avoid loading the ladder, an operational amplifier is used as buffer. It should be powered from a +9V supply to avoid non-linear operation around +5 volts. Resistor R1 raises the minimum output above the ground about 2 volts to avoid poor operation of the op-amp around that area too. An LM358 op-amp is shown in the schematic but a 741 can also be used as well. With an LM358, R1 could be omitted without serious effects in the linearity of the output.

The R-2R resistor ratio should be as close to 2 as possible. 1% tolerance resistors are preferred, but 5% will work

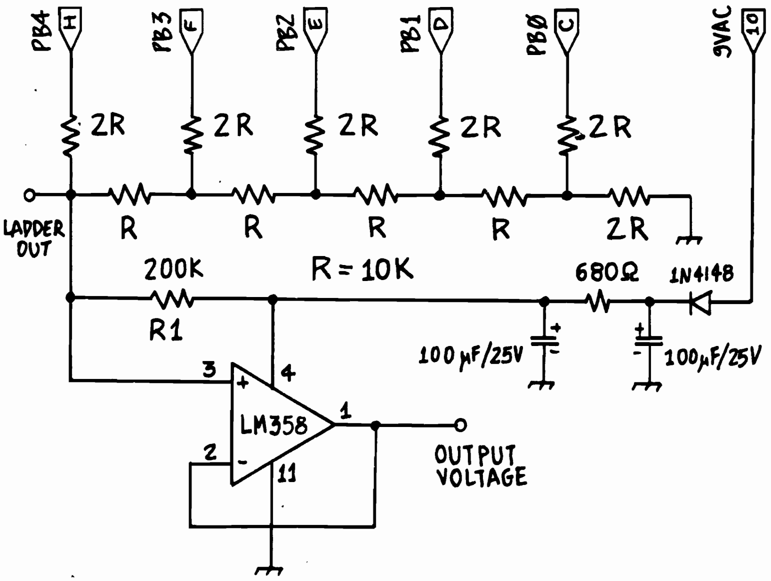
O.K. If you have 5% 10 K and 20K resistors to sort through, you may use the circuit shown elsewhere for the thermometer and substitute the resistors for the thermistor, printing out the count for each resistor. Resistors with counts as close as possible should be selected.

PROGRAMMING

Machine language programming is generally necessary to achieve output at audio frequencies. You only need to store to the 5 least significant bits of the output port the value you want to convert to analog. (Do not forget to set the data direction register to output first).

The following program in BASIC will generate a triangular wave made up of 32 steps in each side. It can be observed by using an oscilloscope at the output.

```
100 REM   ** D/A TEST **
101 REM
110 POKE 56579,255
120 P=56577
130 FOR I=0 TO 31: POKE P,I: NEXT I
140 FOR I=31 TO 0: POKE P,I: NEXT I
150 GOTO 120
```



PROJECT **27**

PRECISION 8-BIT D/A CONVERTER

This project describes a full 8-bit precision D/A converter with built in reference and voltage output. An 11-bit D/A converter chip is used giving excellent linearity, so this D/A approaches textbook performance.

THEORY OF OPERATION

An ideal D/A converter is impossible to make. However, if we want a very good 8-bit D/A converter, we can take a converter specified for higher resolution and configure it for the lower resolution we need. Its cost is somewhat higher but we get outstanding performance. In this project we use an 11-bit D/A converter IC configured for 8-bits plus sign.

The data lines of the USER PORT drive the data lines of the converter and PA2 drives the sign, controlling the polarity of the output.

The converter has a built-in voltage reference and voltage output so no extra components are necessary. However it needs a ± 15V power supply which must be supplied externally. Modular power supplies are available for these voltages or one can be built as shown in the schematic.

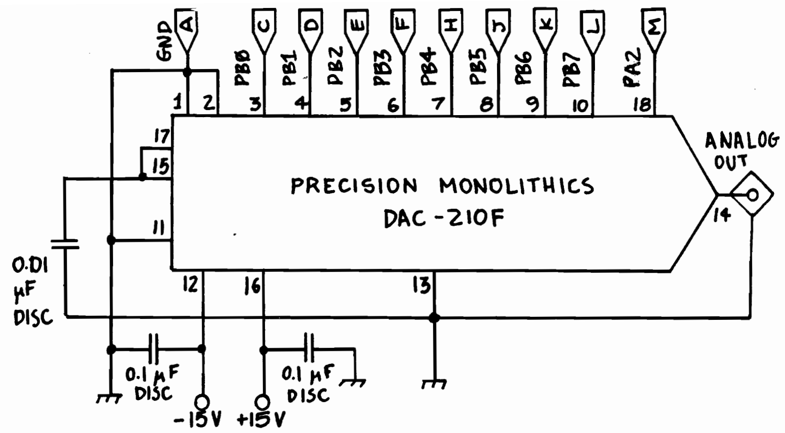
Full scale output can be adjusted to 10.24 volts exactly (for 40 mV steps) by using an external potentionmeter as shown.

Maximum output drive of the converter is 10 mA, which should be sufficient for most applications. If more output is desired a buffer must be used.

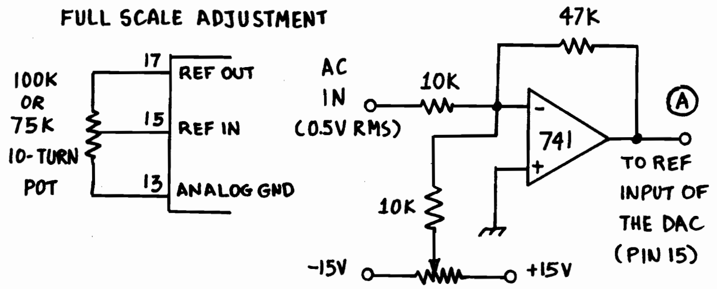
PROGRAMMING NOTES

First set the USER port to output (POKE 56579,255). Then PA2 is set to be an output as follows:

```
POKE 59578, PEEK (59578) OR 4
```



CIRCUIT FOR FULL SCALE ADJUSTMENT

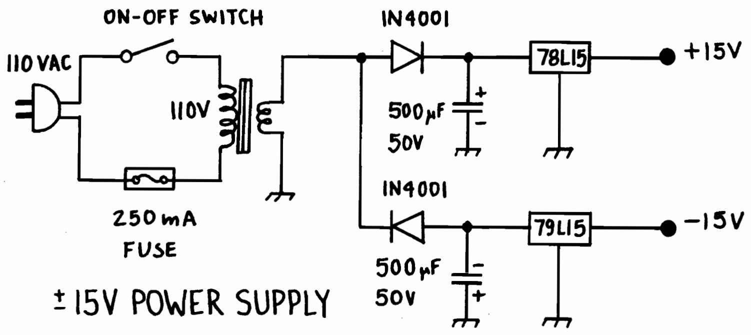


10—TURN, 10K POT. ADJUST FOR 6.5V OUTPUT AT POINT Ⓐ WITH NO INPUT.

For positive output, set PA2 high:
POKE 59576, PEEK (59576) OR 4
For negative output, set PA2 low:
POKE 59576, PEEK (59576) AND 251

Then POKE into the USER PORT the value you want to convert. For example, if you want to set the output to 5 volts, POKE 56577, INT(5000/40). Here 5000 is the desired output voltage in mV and 40 is the value of each step in mV. The output can be eitheir positive or negative depending on the value of PA2, so in effect the D/A has a resolution of 9 bits.

The DAC-210 settles within 1.5 us. Thus it can take data as fast as you can supply them, even if your program is in machine language.



± 15V POWER SUPPLY

APPLICATION NOTES

The typical application of the D/A converter is conversion of digital to analog signals. New values are fed to the converter at regular time intervals. The time interval (the sampling period) determines the frequencies in the output signal. If we output the same data with half the sampling interval, all frequencies in the output will be doubled. The output is usually low pass filtered to remove all frequencies above half the sampling frequency. In the time domain signal, the effects of such filtering are seen as removal of the quantization steps so that the signal is a continuous curve.

The converter described in this project can also be used as a digitally controlled voltage reference due to its high linearity and stability. Such a reference is useful to automatic testing applications.

It two converters are connected to the X and Y channels of an oscilloscope, high resolution (512×512) graphic displays will be possible. A machine language driver will be necessary to drive the D/A converter fast enough so that there is no visible flicker.

The D/A converter can be used as a multiplier if the reference input is driven by an external signal. The requirements are that the external signal varies between 3 and 10 volts. If the signal fed into the reference input is audio, the D/A converter can act as a digital attenuator to control volume or to create sound effects by modulating under digital control the input signal.

The schematic shows a level shifter required to feed audio in the reference input.

PROJECT **28**

8-CHANNEL 8-BIT A/D CONVERTER

The 555 timer forms an effective converter from an analog quantity such as resistance or capacitance to a number. An A/D converter converts a voltage to a corresponding number. In this project we interface an IC A/D converter to the user port, allowing the 64 to measure voltage directly, and indirectly any quantity that can be converted to a voltage.

THEORY OF OPERATION

The National ADC0809 is an inexpensive, 8-bit A/D converter with an 8-channel input multiplexer. The multiplexer has three digital inputs that select which one of the eight analog input lines is fed to the A/D converter.

The converter is controlled by two signals, ALE which latches internally the multiplexer address and START which initiates the conversion process. Conversion takes about 120 us and its result is available at the eight digital output lines as a straight binary number. The EOC (End of Conversion) output remains low during the conversion process and goes high when conversion is completed. The timing diagram of the process is shown in Fig. 28.1.

The converter is interfaced to the expansion port as shown in Fig. 28.2. Full decoding is used and it appears to the CPU as one input and one output port. The output port selects the analog channel and initiates the conversion. The data appears on the input port. No End-of-Conversion detection is provided because it would require an added input port. Instead, the program should wait 150 us after it orders a conversion and then read the input port.

A full handshake with EOC would make possible operation of the converter at maximum speed (around 100 us/conversion). However, due to lack of sample-and-hold, full 8-bit accuracy is obtained only for signals that vary less

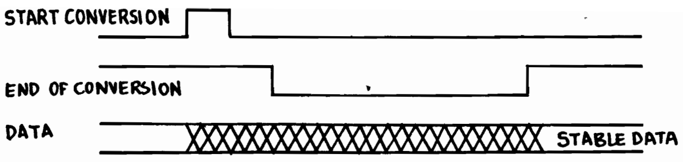


Figure 28.1 Timing Diagram of the Conversion Process

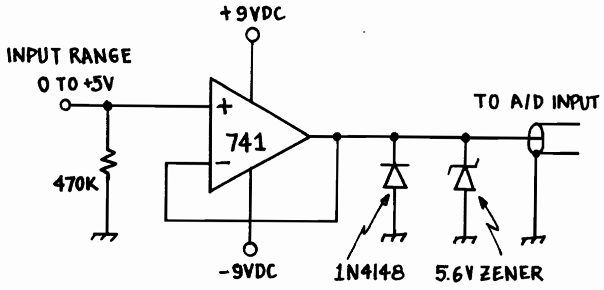


Figure 28.3 DC Buffer Amplifier

than 1 bit (20 mV) within the conversion time. Thus, the converter can be operated at rated accuracy when the input contains low frequency signals only, requiring sampling periods much larger than 100 us (typically 10 ms or more).

The input signal to the A/D converter can range from 0 to 5 volts. The 100 Ohm resistor and 0.01 uF capacitor on each input remove noise that might be picked up by external connections. They should be located physically close to the A/D IC itself. The driving impedance of the source must be less than 1000 Ohms.

It is good practice to locate a buffer amplifier close to the source of the signal and use a shielded cable to connect the output of the buffer to the A/D input. Two buffer designs are shown. The one in Fig. 28.3 is for DC signals and the one in Fig. 28.4 is for AC signals. AC signals must be level shifted and scaled so that they range between 0 and 5 volts before they can be inputted to the A/D converter.

The converter is ideally suited for ratiometric operation, that is, sensing the output of a resistive voltage divider where one of the resistors is variable. This type of input is shown in Fig. 28.5. The variable resistance can be a ther-

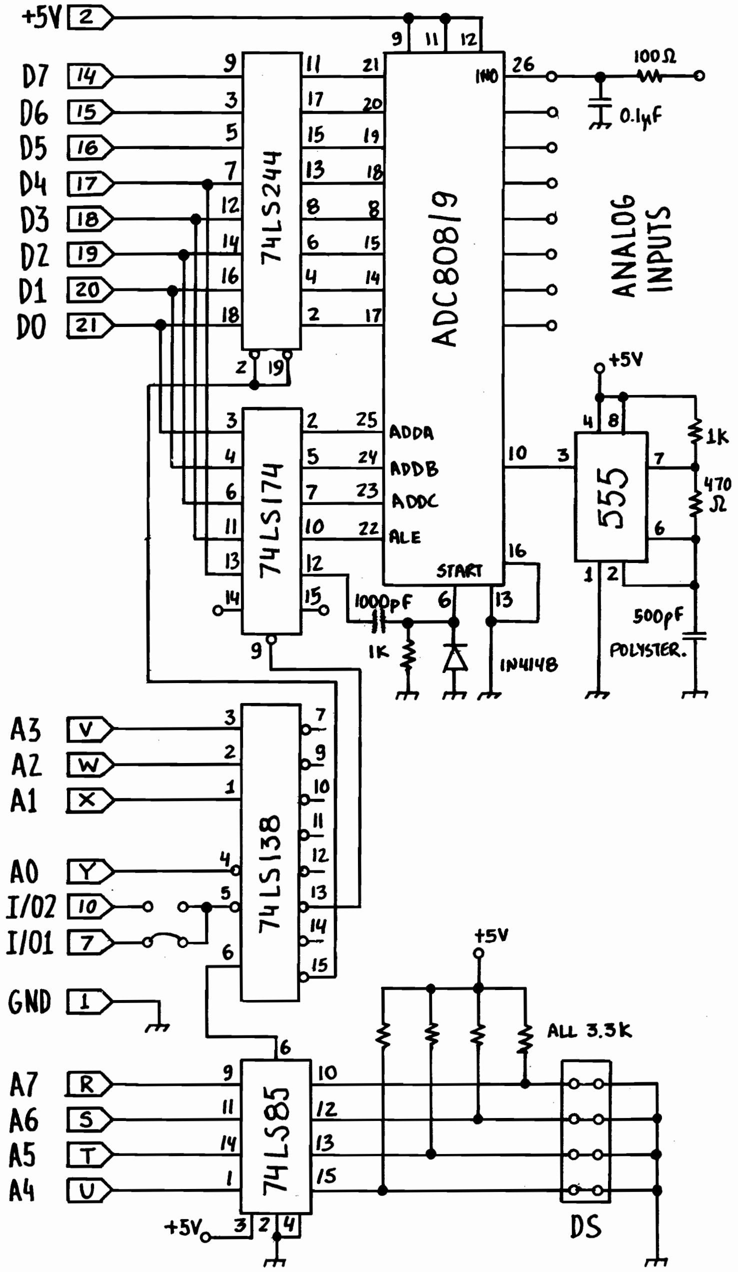


Figure 28.2 Schematic of the 8-Channel, 8-Bit A/D Converter

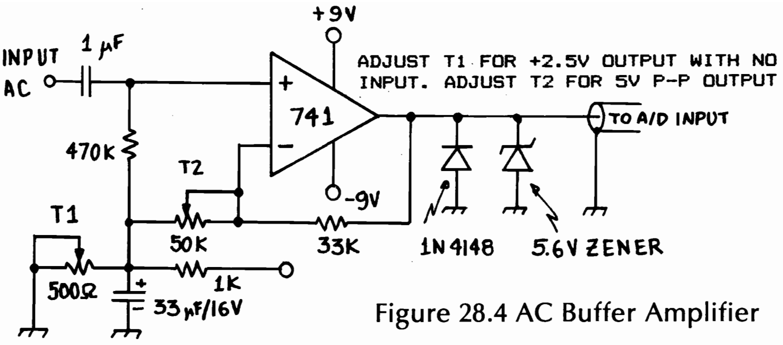


Figure 28.4 AC Buffer Amplifier

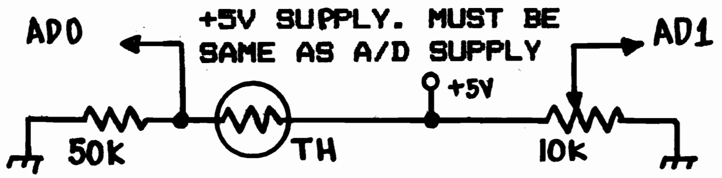


Figure 28.5 Resistance Measuring Circuit

mistor, photocell, pressure transducer etc. Or a potentiometer may be used to sense mechanical position. In fact, this project is the preferred resistive sensor interfacing method and it is recommended over the circuit shown in Chapter 3 because of its multichannel capability and ease of programming.

PROGRAMMING NOTES

The ports can be positioned on any two consecutive even addresses in the I/O blocks I/O1 or I/O2. With the switches on the DIP switch DS and the jumpers on the output of the LS138 as shown, the input port is at location 0000 0000 within the I/O1 space. It can be addressed then at location 52768+0=52768. The output port is at location 0000 0010 within the I/O1 space, so it can be addressed at location 52768+2=52770.

The three least significant bits of the input port select the input channel while the fourth bit latches the channel address. The fifth bit initiates conversion on its rising edge. It is fed into the A/D via a differentiator to satisfy the requirement for a maximum on-time of 1 us for the convert pulse.

PROJECT 29

12-BIT DATA ACQUISITION SYSTEM

The 64 has been marketed as a home computer. Yet it has the graphics and memory required for many scientific applications. Here is an istrument-quality data acquisition peripheral that will give you the basis for precision, low cost instrumentation.

THEORY OF OPERATION

For serious applications, you will need a data acquisition peripheral that has high resolution (12 or more bits) and stability and linearity at least equal to its resolution. And for many applications, fast conversion is essential.

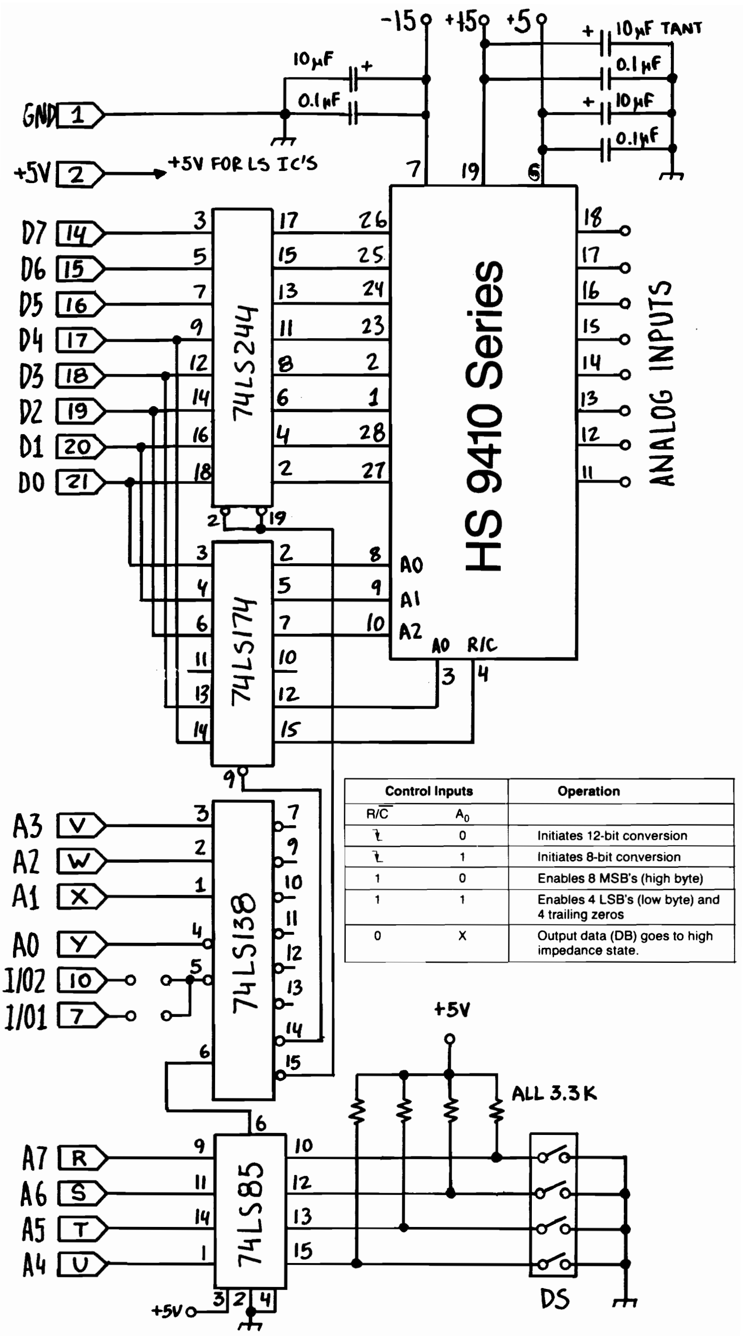
It is extremely difficult to build such a peripheral from scratch, so it is fortunate to find that somebody has already done all the hard work for you. Hybrid Systems HS 9412 is an 8-channel, 12-bit Data Acquisition System (DAS) in a 28-pin DIP offering 30 us conversion time and selling for less than $150.

Interfacing the HS 9412 to the 64 is quite easy and the circuit is essentially the one used in the previous project. The functions of the two control lines is the only difference as shown in Fig. 29.1. The 12-bit data must be taken from the converter in two fetches, one for the most significant byte and one for the least significant byte. The control inputs determine which one of the two bytes is read.

No handshake is implemented because the 64 will take more than 30 us to generate the control signals and receive and store the A/D data, even in machine language.

The 9412 requires three power supply voltages, +5V and ±15V. These should be generated by a power supply similar to the one shown in project 27.

The input voltage range of the HS 9412 is ±10V. There are also two other models identical in performance with the exception of input voltage range. Model HS 9411 has ±5V input range and model HS 9410 from 0 to 10V.



Control Inputs		Operation
$R/\overline{C}$	A_0	
⌊	0	Initiates 12-bit conversion
⌊	1	Initiates 8-bit conversion
1	0	Enables 8 MSB's (high byte)
1	1	Enables 4 LSB's (low byte) and 4 trailing zeros
0	X	Output data (DB) goes to high impedance state.

CONSTRUCTION NOTES

The important consideration in the construction of the converter is noise which must be minimized. Each quantization step is 1.22 mV and noise must be less than that, preferably much less. Noise can come from the power supply, from grounding problems and from RF pick-up, including capacitive and inductive coupling.

Power supplies external to the 64 with good decoupling with two capacitors (10 uF tantalum and 0.1 uF ceramic disc) right next to the DAS power pins help avoid power supply noise.

A ground plane under the DAS and covering as much of the PC board as possible helps reduce RF noise pick-up acting as a shield. It also provides a solid ground to reduce ground noise. To further reduce noise, digital circuits and lines should be kept as far as possible from the analog inputs.

For best results, in addition to the above precautions, the circutry, power supplies and analog input amplifiers, buffers and antialiasing filters that might be required, should be housed in a metal box that connects to the 64 via flat cable. On the expansion port of the 64, there should be a board that contains a data and address buffer, similar to the ones shown in project 5. Or better, the port decoder can be on the board that plugs into the 64, so that the address buffer is eliminated and the cable width is smaller. BNC connectors or RCA jacks must be used for analog inputs. This type of construction might seem excessive but it is required to insure 12-bit accuracy.

CHAPTER 7

CONTROLLER INTERFACES

PROJECT **30**

PWM MOTOR CONTROLLER

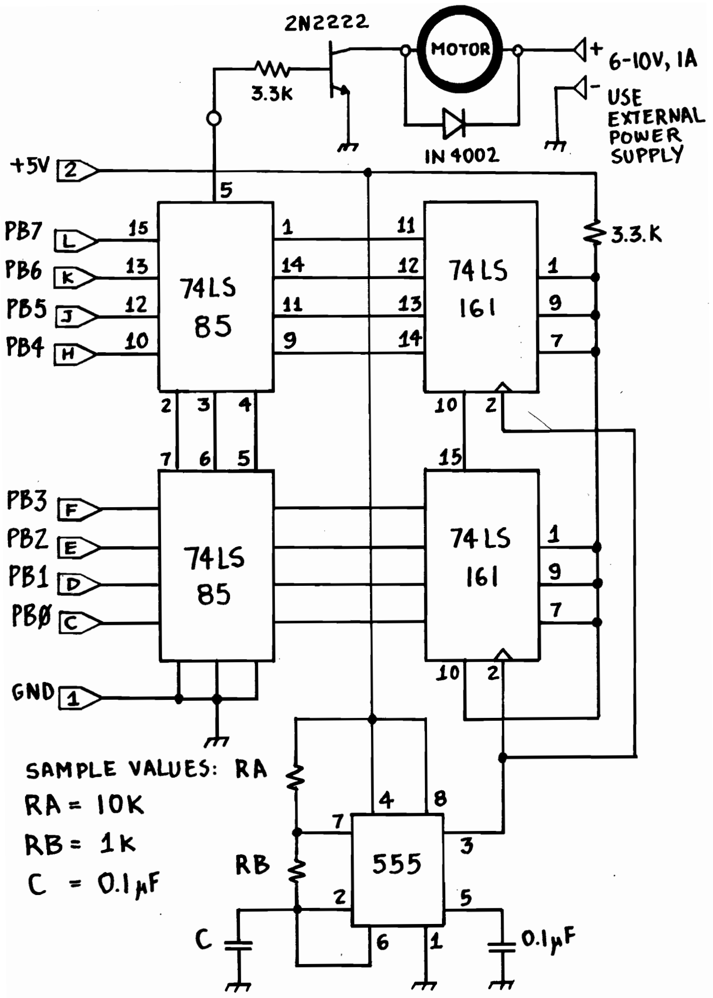
Operating a motor under program control is not only fun but has many applications. From automating a model rail-road to opening and closing the curtains of your house to increase energy savings, motor control enables your 64 to actually do something. In this project, a PWM technique is used to give precise control even on toy motors.

THEORY OF OPERATION

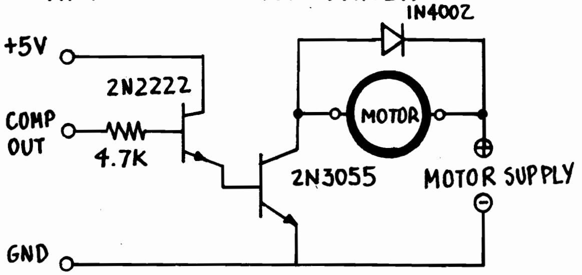
The small shunt DC motor is the most widely used type of electrical motor. It will run on both DC and AC and it is used in toys, cassette recorders, disc drives, drills, hair driers, mixers and so on.

Its speed can be controlled by adjusting the voltage applied to it. Unfortunately, torque depends on current and current decreases with decreasing applied voltage, so a point is reached when the torque due to the applied voltage is less than the friction of the bearings and the motor stops. For small toy motors this happens at around 500 RPM.

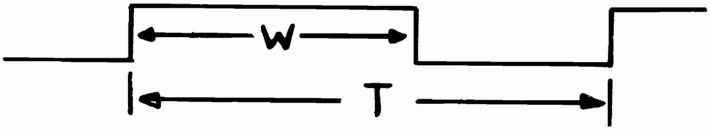
A solution to this problem is to use Pulse Width Modulation (PWM). Full voltage is applied for some amount of time and then power is turned off. By adjusting the ratio of power ON to power OFF the total energy going into the motor (and its speed) is controlled. And by making the ON-OFF switching fast enough, the inertia of the motor smooths out the pulses and we see a continuous motion. Using PWM, a toy motor can be controlled down to 1 RPM or less without stalling. Torque remains constant regardless of speed because when ON the motor operates at full voltage (and current). An additional significant benefit is that the power transistor that drives the motor is either ON or OFF, thus dissipating only a few percent of the power it controls.



HIGH POWER MOTOR DRIVER



The PWM circuit consists of an oscillator that drives a counter, a comparator and a control input (the 64 USER PORT). As long as the count is below the control input, the output of the comparator is an "1". When it is more than the control input, it is "0". The comparator output changes states once for every full cycling of the counter, thus the period of the output wave is constant (T). When it changes depends on the control input, thus the width W varies with the control input, giving us a PWM signal.



This is connected to a transistor driver that in turn controls the motor. The diode at the collector of the transistor dissipates the inductive pulse that occurs when the transistor turns off.

To control motor speed, first set the USER PORT to output (POKE 56579,255). Now you can POKE the desired pulse width to the port (range from 0 to 255) and the motor will run at the corresponding speed. For example, to operate the motor at half speed, POKE 56577,128. To stop the motor, POKE 56577,0.

CONSTRUCTION NOTE

Try various values of RA, RB and C in the 555 until operation is achieved at desired speeds. You will find that not all motors operate smoothly at all speeds. The better the motor quality the wider the range. Good quality miniature DC motors can be obtained from old cassette recorders. Turntable motors are usually AC only and will not operate on DC. A source of inexpensive medium power DC motors are motors used to power electric windows and seats in old luxury cars. Also check the electronic surplus stores in your area.

If you are using the high power circuit, build it solidly and provide some heat sinking for the 2N3055. Make a single ground connection from the digital circuit ground to the power circuit ground to avoid loops.

PROJECT **31**

AC POWER CONTROLLER

By giving the 64 the capability to control 110VAC power, you can open a whole new area of applications for your computer. Electrical appliances, from light bulbs to stoves to stereos operate on AC power. Controlling AC power is as close as your 64 can get to controlling the world.

THEORY OF OPERATION

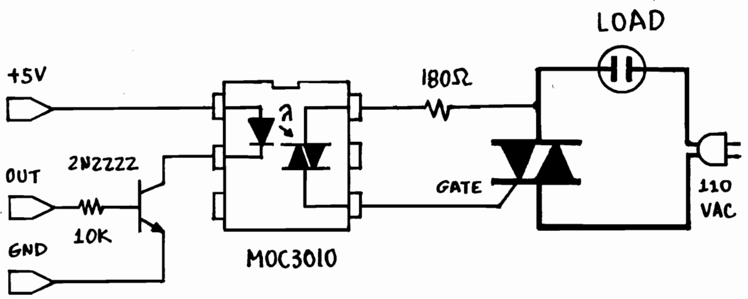
AC power control is surprisingly easy nowadays, due to a power semiconductor device called triac. A triac is a high power semiconductor switch that has three terminals. Two of the terminals perform the switching function carrying the power. The third terminal, called gate, turns the triac on and off. Only a small gate current is needed to control large currents through the triac. For example, in the Radio Shack #276-1000 triac, a 25mA gate current can switch up to 6A. The triac can only be used with AC power.

Interfacing a device that controls more than 600W of power at high voltages requires care. If you do it improperly, you may utterly destroy your 64 or even electrocute yourself or others.

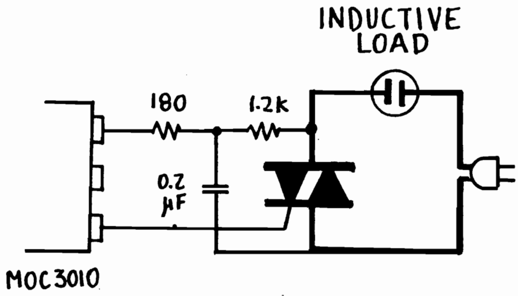
Fortunately, there is a second semiconductor marvel that helps significantly to avoid problems. The optoisolated triac. It is a small triac with a photosensitive gate, packaged together with a Light Emitting Diode (LED) in a small light-proof package. When current flows through the diode, light is emitted, striking the phototriac gate and turning it on. There is no electrical connection between the controlling circuit (LED) and the controlled circuit (phototriac). Perfect electrical isolation exists.

From this point on the interfacing is simple. The 64 via an output port line, turns the transistor on, causing current to flow through the LED. The LED emits light which activates the optoisolated triac, turning it on to supply gate current to the power control triac. Two different circuits

CIRCUIT FOR RESISTIVE LOADS



CIRCUIT FOR INDUCTIVE LOADS



are shown, one for resistive loads (lights, ovens etc.) and one for inductive loads (anything that contains a motor: fans, refrigerators, etc.).

CONSTRUCTION NOTES

In the unlikely event that you do not have anything connected to the USER PORT, you can use it to drive the power controller. Since it has eight output lines, it could control up to 8 AC circuits. If the USER PORT is not free, you can build an output-only port for the expansion bus as shown in the schematic, dedicated to triac control.

It is very important to build this peripheral in such à way that it is not possible to accidentally get 110VAC in your body or in the 64. It is suggested that a flat cable be used to connect the output port of the 64 to an aluminum box where the optoisolators and power triacs are. The box should be of sizeable dimensions (something like 8×6×2 inches or larger) and fully closed, with all circuitry inside and no exposed leads. All construction inside should be very neat, with wires adequately separated and components tied down securely so they cannot move and touch. There is no room for sloppiness in the power box. Only the highest quality workmanship will do.

The power triacs will need heat sinking. You can bolt them on the aluminum box which will act as a heat sink. Make sure to follow the instructions of the manufacturer for mounting the triacs. Some triacs may not have an isolated mounting tab, in which case you will have to use an insulating wafer and heat sinking compound to mount the triac properly on the box.

It is imperative that the grounds for AC and the 64 are not conected in any way. Isolation is the name of the game here and it should be strictly observed.

Do not try to switch loads of higher wattage than the triac is designed for and remember to use the appropriate circuit for inductive loads.

Finally, while miniaturization is a virtue in things electronic, it just isn't possible with power circuits. Make sure that the box is substantial and sturdy. It will control 2.5KW, a considerable power by an measure.

CHAPTER 8

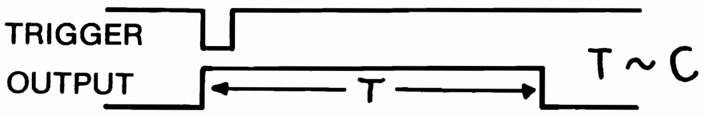
INSTRUMENTATION CIRCUITS

PROJECT 32
CAPACITANCE METER

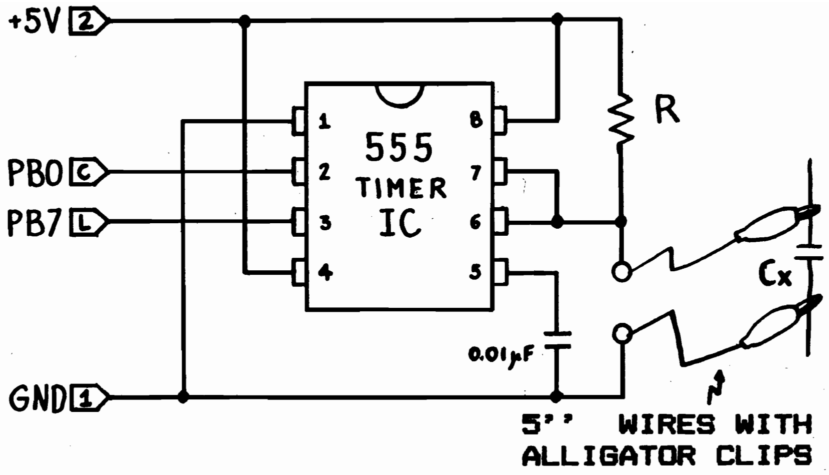
This meter will allow you to measure capacitance between 100pF and 50uF, yet it only uses 3 components (and your 64 which does most of the work). You will be surprised at the accuracy and stability of this little circuit, which can, in most cases, replace a standalone instrument costing as much as the 64.

THEORY OF OPERATION

The capacitance meter operates by measuring how long it takes to charge a capacitor to a predetermined voltage, while a known current is applied. A 555 timer IC in the monostable mode is used to perform the function of applying the current and determining if the threshhold has been reached. The 64 is used to initiate a measurement and count time. The accuracy of the meter depends on the 555 which was chosen for its exceptional stability charateristics. The 64 initiates a measurement cycle by bringing low then high the least significant bit of the user port (PBO). This triggers the 555 and its output goes high for as long as it takes to charge the unknown capacitor to 2/3 the supply voltge. The 64 measures the duration of this output pulse using a machine language program to obtain high resolution (25us per count). Here is a timing diagram of the process:



The amount of current flowing in the unknown capacitor is determined by R. 470K will give good results for unknown capacitors from 100pF to 1uF. For values over 1uF and electrolytics (that have high leakage currents) R should be changed to 2.2K.



```
                   CTR=$8B
                   PORT=$DD01

78                 SEI          ;DISABLE INTERR.
A900      TRIGG    LDA #0       ;OUTPUT A
8D01DD             STA PORT     ;ZERO AND
858B               STA CTR      ;CLEAR CTR
858C               STA CTR+1    ;AND CTR+1
A901               LDA #1       ;OUTPUT AN
8D01DD             STA PORT     ;ONE TO TRIGGER
A58B      COUNT    LDA CTR      ;INCREMENT
18                 CLC          ;CTR AND CTR+1
6901               ADC #1       ;AS A 16-BIT
858B               STA CTR      ;WORD TO COUNT
A58C               LDA CTR+1    ;DURATION OF 555
6900               ADC #0       ;OUTPUT PULSE
858C               STA CTR+1    ;AFTER TRIGGER
AD01DD    CHECK    LDA PORT     ;IF PULSE
30EE               BMI COUNT    ;HAS ENDED
58                 CLI          ;ENABLE INTERR.
60                 RTS          ;RETURN
```

```
100 REM ** CAPACITANCE METER **
101 REM
110 POKE 56579,1
120 FOR I=1 TO 35
130 READ A: POKE49152+I,A: NEXT I
140 POKE49152+I,A
150 NEXT I
200 SYS49153
210 COUNT=PEEK(139)+256*PEEK(140)
220 IF COUNT<2 GOTO 200
230 PRINT (COUNT/2364)*100;"NF"
240 GOTO 200
300 DATA 120,169,0,141,1,221,133
310 DATA 139,133,140,169,1,141,1
320 DATA 221,165,139,24,105,1,133
330 DATA 139,165,140,105,0,133,140
340 DATA 173,1,221,48,238,88,96
350 END
```

PROGRAMMING NOTES

The program listed will drive the capacitance meter and print out on the screen the value of Cx, the unknown capacitance. The printout stops when Cx is disconnected and resumes when a Cx of 100pF or more is connected.

The program is all in BASIC form to help entering and running. The DATA statements contain the machine language driver.

The machine language driver works as follows: First the interrupts to the 6502 are disabled. Unless this is done, the count will be erratic because if the 6502 gets interrupted, the counting will stop until the interrupt is serviced. The section labeled TRIGG clears the locations CTR and CRT+1 (storage for the count) and outputs the trigger pulse (by outputting a "0" and then a "1"). The section labeled COUNT increments locations CTR and CTR+1 taken as a 16 bit number. That is, the carry (if any) from CTR is added to CTR+1. Finally, CHECK checks the output of the 555. If it is high, the program goes to COUNT to increment the count. If it is low, the pulse has ended and counting is terminated. To exit, interrupts are enabled and a return from subroutine (RTS) instruction is executed.

The timing loop consists of the instructions from LDA CTR to BMI COUNT. Every time it is executed, the count is incremented. The time required to execute this instruction sequence (25 cycles or 25 microseconds) is the time resolution of the count.

CALIBRATION

The program as given will print out capacitance in nF (1nF = 1000 pF = 0.001 uF) for R = 470K. You should calibrate your circuit after building it. Calibration is based on a known capacitor with a value over 5 nF. Let's say you have an 10nF (0.01 uF) capacitor with 2% tolerance. Connect it to the meter and adjust the constant in line 230 of the BASIC program (in the listing it has the value of 2364) until the printout is as close to 10nF as possible. Your capacitance meter will now be calibrated with an accuracy equal to that of your calibrating capacitor (2% in this case).

PROJECT 33
FFT SPECTRUM ANALYZER

A spectrum analyzer takes as input a signal and computes and displays its spectrum, a frequency domain representation of the signal. In this project we build a Fast Fourier Transform (FFT) based spectrum analyzer that computes the energy of the input signal at 256 equally spaced frequencies.

THEORY OF OPERATION

An oscilloscope is used to display time domain (the x-axis is time) signals. A spectrum analyzer converts the time domain signals to a frequency domain (the x-axis is frequency) representation and displays them. A frequency domain representation is useful when the information we want is not obvious in the time domain signal. Take for example a square wave. We know it has harmonics but we cannot see them. It just looks square. If we look at it using a spectrum analyzer, the square wave will not be seen but instead we will see both the frequency and amplitude of the fundamental and each one of the harmonics that make up the square wave.

The spectral analysis is done in software, using the Fast Fourier Transform (FFT) algorithm. A 512 point transform is taken, giving 256 spectral samples. These can be easily displayed in the high resolution graphics mode of the 64.

The input signal is taken from locations 22000 to 22512. Each sample of the signal is expected to be a byte and they should be stored in consecutive locations. The signal can come from a Data Acquisition System (for example the one described in project 29) or from one of the test programs appended to the listing of the spectrum analyzer.

There are two such programs to facilitate testing of the analyzer. The one starting at line 1000 generates a pulse whose width is determined by line 1020. Its length is now

10 samples. You may change this number and see the different spectra, all of which are based on the sinx/x function. The other test program starts at line 1100 and generates a sinewave. You can change the frequency of the sinewave by changing line 1110 so that SIN(I) becomes SIN(I*K). If K is less than 1 the frequency will decrease, if greater than 1 it will increase.

The frequency in a digital signal processing system is always a function of the sampling frequency. The highest possible frequency is one half the sampling frequency. When an analog signal is coverted to a digital one using a DAS it is important to band limit the input signal using a low pass filter so that it does not contain any frequencies higher than half the sampling frequency. If no filtering is used and the signal contains a harmonic at lets say 0.75 of the sampling frequency, aliasing will occur and the harmonic will appear at a different frequency (0.25Fs).

The program computes the complex FFT in lines 400 to 600. To change the FFT size change line 290. NP is the number of points which must be a power of two and E is the power to which two is raised to get NP. In other words, 2+E=NP. SA is the starting address of the sampled time domain signal. If you use an FFT size other than 512, the display will still display 256 points. Thus if you set NP=256: E=8 you will get a symmetrical spectrum with 128 unique points.

The X array contains the real part of the FFT and the Y array the imaginary part. The input is always real, so the output of the FFT is symmetric around the middle.

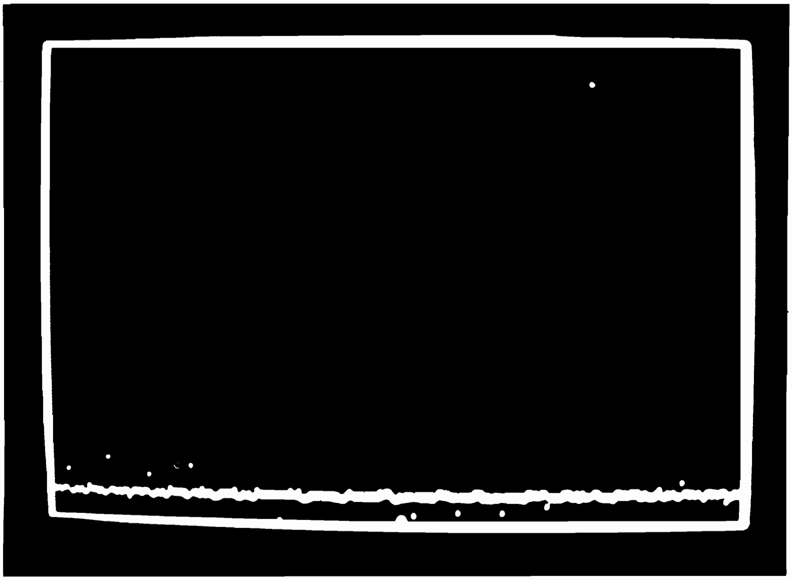
Lines 600-695 do the bit reversal, a necessary step to unscramble the order of the FFT outputs.

Lines 700 to 720 compute the spectrum which is defined as the square root of the sum of the squares of the real and imaginary parts of the FFT. If line 728 is omitted, the spectrum will be displayed. Line 728 computes the log of the spectrum with base 10 so that the result can be expressed in decibels. The log spectrum is useful in applications like frequency response analysis and speech processing. The spectrum or the log spectrum as the case might be, are stored in the X array while the Y array is cleared

```
100 REM ** SPECTRUM ANALYZER **
110 REM
120 DIM X(512), Y(512)
130 PRINT CHR$(147)
140 PRINT: PRINT: PRINT TAB(10);
150 PRINT "FFT SPECTRUM ANALYZER"
160 PRINT
170 PRINT "ERASE PREVIOUS PLOT?";
180 PRINT "(Y OR N)":
190 GET A$: IFA$="" GOTO 190
200 IF A$<>"Y" GOTO 290
210 FOR I=0 TO 27: READ A
220 POKE 23300+I,A: NEXT I
230 DATA 169,127,133,254,169,0
240 DATA 133,253,168,133,253
250 DATA 145,253,160,63,162
255 DATA 32,145,253,136,208,251
260 DATA 198,254,202,208,246,96
270 SYS 23300
280 PRINT "COMPUTATION TIME IS 5 MIN."
290 NP=512: E=9: SA=22000
300 REM GET A/D DATA
310 FOR I=1 TO NP-1
320 X(I)=PEEK(SA+I-1)
330 Y(I)=0
340 NEXT I
400 REM CALCULATE FFT
430 FOR LO=1 TO E
440 LNX=2^(E-LO): LX=LNX+LNX
445 SCL=6.283185/LX
450 FOR LM=1 TO LNX
460 ARG=(LM-1)*SCL
470 C=COS(ARG): S=SIN(ARG)
480 FOR LI=LX TO NP STEP LX
490 J1=LI-LX+LM
500 J2=J1+LNX
510 T1=X(J1)-X(J2)
520 T2=Y(J1)-Y(J2)
530 X(J1)=X(J1)+X(J2)
540 Y(J1)=Y(J1)+Y(J2)
550 X(J2)=C*T1+S*T2
560 Y(J2)=C*T2-S*T1
```

```
570 NEXTLI
580 NEXTLM
595 NEXT LO
600 REM BIT REVERSAL
610 J=1: NV2=NP/2: NM1=NP-1
620 FOR I=1 TO NM1
630 IF I>=J GOTO 670
640 T1=X(J): T2=Y(J)
650 X(J)=X(I): Y(J)=Y(I)
660 X(I)=T1: Y(I)=T2
670 K%=NV2
675 IF K%>=J GOTO 690
680 J=J-K%: K%=K%/2
685 GOTO 675
690 J=J+K%
695 NEXT I
700 REM LOG SPECTRA IN DB
705 FOR I=1 TO 256
710 A=X(I): B=Y(I): C=20/LOG(10)
715 A=A*A+B*B
720 A=SQR(A)
725 IF A=0 THEN A=0.00001
728 A=LOG(A)*C
730 X(I)=A: Y(I)=0
735 NEXT I
740 REM FIND MIN AND MAX
745 MIN=1E38: MAX=1E-38
750 FOR I=1 TO 256
755 IF X(I)<MIN THEN MIN=X(I)
760 IF X(I)>MAX THEN MAX=X(I)
765 NEXT I
770 REM NORMALIZE FOR DISPLAY
775 SP=(MAX-MIN)/144
780 FOR I=1 TO 256
785 Y(I)=167-INT((X(I)-MIN)/SP)
790 NEXT I
800 REM GENERATE GRID
805 POKE 56576,PEEK(56576) AND 252 OR 2
810 POKE 53272,PEEK(53272) AND 7 OR 120
815 POKE 53265,PEEK(53265) OR 32
820 FOR I=23552 TO 24551
825 POKE I,16: NEXT
```

```
830 FOR I=4 TO 35
835 POKE 24576+647+I*8,255
840 POKE 24576+6720+I*8,255
845 NEXT I
850 POKE 25503,254: POKE31576,254
855 POKE 31143,24: POKE31142,24
860 FOR J=0 TO 7
865 FOR I=1 TO 18
870 POKE 25248+I*320+J,192
875 POKE 25496+I*320+J,3
880 NEXT I
885 NEXT J
900 REM SUBROUTINE PLOT
940 FOR X=32TO 287
945 Y=Y(X-31)
950 CH=INT(X/8)
955 RO=INT(Y/8)
960 LN=Y AND 7
965 BY=24576+RO*320+8*CH+LN
970 BI=7-(X AND 7)
975 POKE BY,PEEK(BY) OR (2^BI)
980 NEXT X
985 GET A$:IF A$="" GOTO985
990 POKE53265,PEEK(53265) AND 223
995 POKE56576,PEEK(56576) OR 3
996 POKE53272,PEEK(53272) AND 7 OR 16
997 CLR: END
1000 FOR I=0 TO 512
1010 POKE22000+I,0: NEXT I
1020 FOR I=0 TO 10
1030 POKE 22000+I,255
1040 NEXT I
1050 END
1100 FOR I=0 TO 512
1110 S=INT((SIN((/256)*I*200)+1)*127)
1120 POKE 22000+I,S
1130 NEXTI
1140 END
```



SPECTRUM OF TEST SINEWAVE

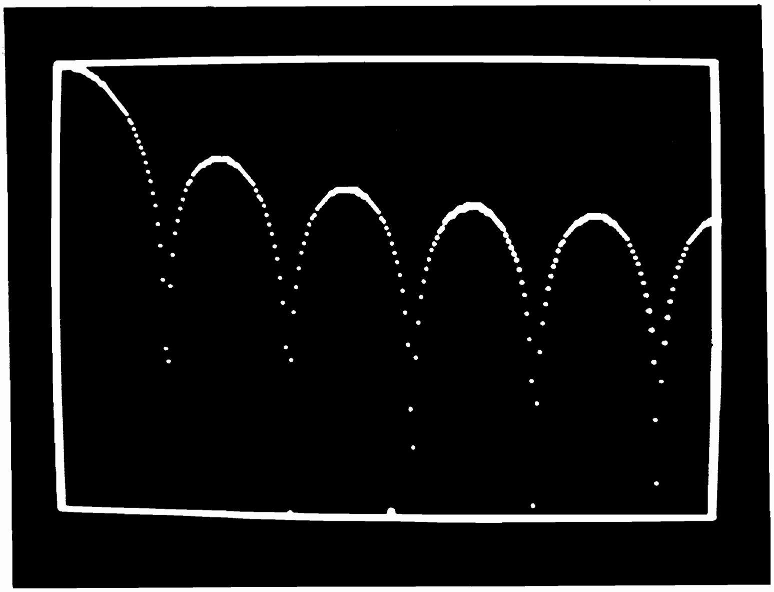


Figure 33.1 Log Spectrum of the Pulse Generated by the Test Program

because the result is always real. This is done in line 725.

Lines 740 to 765 scan through the X array to find the minimum and maximum values of the array elements. These values are used to normalize the array for display purposes. The normalizing routines is lines 770 to 790. The normalized values are stored in the Y array.

Lines 800 to 825 set up the high resolution display mode and lines 830 to 885 generate the outline of the display area. Finally, the output is plotted in lines 900 to 980. Line 985 scans the keyboard and when a key is pressed it gracefully exits the high resolution display mode and returns the screen to normal character operation.

The program can be improved by adding a calibration grid and a cursor (a vertical line that moves left or right in response to pressing the L and R keys on the keyboard). The values of X and Y at the point where the cursor is are then displayed below the plot.

The spectrum analyzer described here has equal or better performance than the best spectrum analyzer in the market as far as accuracy is concerned. It weak point is computational speed. It takes approximately 200 seconds to compute the 512 point complex FFT. This can be compared with less than 0.5 seconds for commercial FFT processors. There are some special FFT hardware that would do the same FFT in about 2 milliseconds, or 100,000 times faster! But if you can do something else while the 64 is computing the FFT or if you just don't mind waiting, you will be hard pressed to find a less expensive FFT based spectrum analyzer.

You may have at this point many questions about spectrum analysis in general and FFT in particular. Rather than offering a course here on these matters, we will refer you to the book "Theory and Application of Digital Signal Processing" by L. R. Rabiner and B. Gold, Prentice-Hall, 1975.

PROJECT 34
LOGIC ANALYZER

A logic analyzer catches fleeting digital signals and stores them for observation and study. Similar to an oscilloscope, its main advantage is that it allows you to see signals that occur only once while with the oscilloscope you can only see periodic signals.

THEORY OF OPERATION

The main component of the logic analyzer is a fast memory that can automatically sample and store the input signal for later perusal. This type of memory is very inexpensive today. And if we use the 64 for control and display purposes, we can build an inexpensive but quite capable logic analyzer.

The sampling frequency of the analyzer must be as high as possible in order to catch any narrow spikes in the signal and to clearly indicate timing differences between a number of channels. We use a 10 MHz clock giving a resolution of 100 ns which is adequate for most applications. Higher clock frequencies will require much more careful design and construction of the analyzer circuits.

Commercial logic analyzers have many channels, usually 16 or more and are elaborate and expensive instruments. Our analyzer has only eight channels, each with a capacity of 1024 samples. Its triggering capabilities are also very limited. It can be set to trigger on a combination of three input signals.

Depending on programming, the data can be scrolled across the screen of the 64 using block graphics or they can be displayed in the high resolution graphics mode. In this case, if one pixel represents a sample, all acquired data can be displayed in two screens.

As it can be seen in the block diagram of Fig. 34.1, the two 2148H 1Kx4 RAM IC's (IC1 and IC2) are used to store

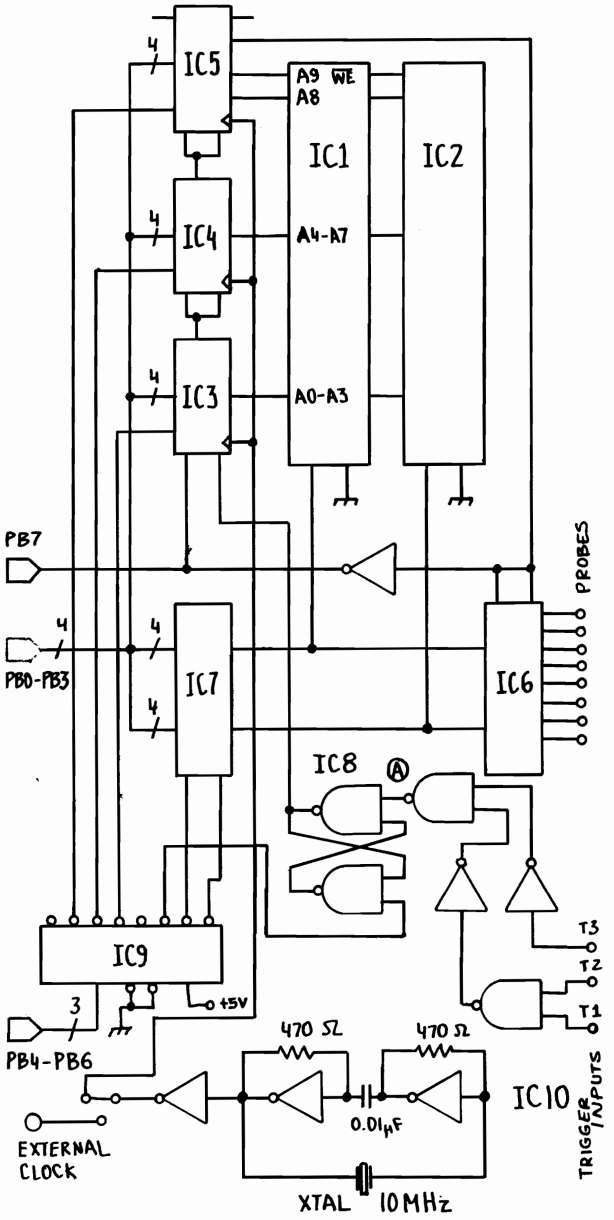


Figure 34.1 Block Diagram of the Logic Analyzer

the acquired data. The 10-bit counter made of the 74F161 counter IC's (IC3, IC4, IC5) is driven by the clock generator during data acquisition. Thus the memory is sequentially addressed. During the acquisition phase, the memory is always in the write mode so the data that appear at its inputs are written in consecutive memory locations every clock cycle (100 ns).

IC6 (F244) acts as buffer for the input signals. It has a small (about 0.3V) hysteresis around the transition region of its inputs which helps eliminate any noise picked up by the probes.

When all 1024 locations have been written, bit 3 of IC5 is toggled and becomes 1. Thus the memory goes in the read mode, IC6 is deselected and the counter stops because its controlling input, CEP, goes low. The data can now be read by the 64.

Interfacing to the 64 is easily done via the USER PORT. The four lower bits are used for data I/O and the next four for control. IC7 multiplexes the RAM data into the four lower bits of the port, under the control of IC9. IC9 also allows the 64 to load any of the counters with data from the lower four bits of the USER PORT.

The RAM contents are transferred to the 64 by loading sequential addresses in IC3, IC4 and IC5 and then reading each nybble of the data by controlling IC7 which acts both as buffer and as multiplexer.

PB7 is used by the analyzer to signal to the 64 that it has finished acquiring data. When it goes low, control can be passed to the 64.

The acquisition phase is started by making bit 3 of IC5 an 1. Triggering is armed by setting the flip-flop (made of two gates in IC8) via IC9 and by setting bit 3 of IC5 to zero. The acquisition will start only when the trigger inputs T2 and T3 are high and T1 is low. When this happens, the flip-flop will reset and it will enable the counter to start the acquisition phase. In order to satisfy the triggering requirement, any trigger input that is not used must be set to the trigger condition (i.e. to ground if it is T1 or to +5V if it is T2 or T3).

If this simple triggering scheme is not sufficient, you can

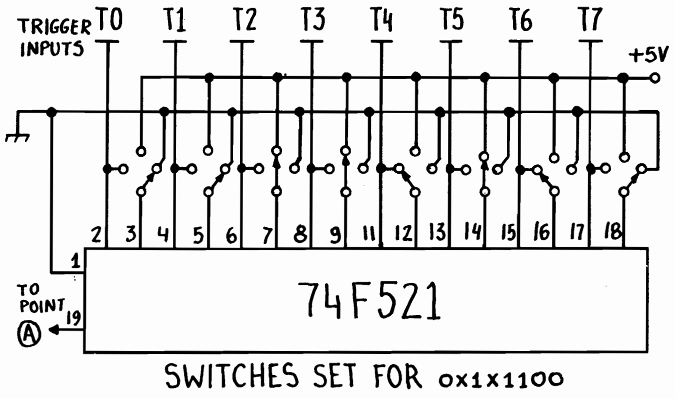


Figure 34.2 Eight Channel Triggering Circuit

go to a more elaborate triggering arrangement as shown in Fig. 34.2. There the trigger flip-flop is driven by an 8-bit comparator. Switches select whether each of the 8 trigger lines are 0, 1 or a don't care for a trigger to occur.

There is also provision for an external clock. It is useful when looking at microprocessor bus signals for example. All the signals in a microprocessor system are synchronized to the CPU, so it is best to use its clock to drive the analyzer. The external clock connection also offers the possibility of using a slower clock to increase observation time. By connecting a function generator as external clock and setting it to 5V pulse output, the analyzer clock becomes fully variable over a very wide range.

By substituting a fast, 8-bit A/D converter for the digital inputs, and using a function generator as variable external clock, the analyzer can be used as a digital storage oscilloscope with a single trace.

The analyzer can be easily extended to 2048 words by using a 2048×8 memory IC and addressing the extra address pin by the unused output in IC5. There may be some problem in finding a fast (70 ns) RAM chip of this size but memory speeds are constantly decreasing so this speed will be soon commonplace. In the meantime, you can increase the clock period to lets say 150 ns and use an 100 ns 2016 which is easily available. Or if you can select among several 100 ns IC's you can find one that accesses at around 80 ns at room temperature and try it out with the 100 ns clock. Chances are it will work O.K.

CONSTRUCTION NOTES

Wirewrap or PC construction is necessary for this project. Use the PC board of project 1 to connect to the 64 and a few feet of flat cable to connect to the analyzer, which can be housed in a small plastic box.

Good power supply decoupling is essential. Use a 0.1 uF disc ceramic capacitor from +5V to ground next to each IC. The +5V supply must be also decoupled with a 2 uF tantalum capacitor. Power is supplied by a "wall transformer" type of supply that supplies 9VDC at 600 mA or more. The current draw of the analyzer is too much to be supplied by the 64.

Use sockets for all chips. The input connectors should be mini clips attached to short (about 12 inches long) wires. It may be worthwhile to buy replacement clips from logic analyzers made by Hewlett-Packard or Tektronix instead of cheap mini-clips which may be hard to anchor to IC pins for measurements.

Do not subsitute the F series IC's with LS types unless you plan to operate at clock speeds of 5MHz or less.

PROGRAMMING NOTES

The signal analyzer can be programmed in BASIC but operation of the display will be very slow. It is recommended that the time consuming functions of data display and clearing the display screen be written in machine language. Then the BASIC program can call these subroutines as needed, while housekeeping and control remain in BASIC.

The clear screen routine can be the same as in project 33. The display routine should be able to get from the analyzer RAM one of the four 256-byte blocks it contains and display it graphically in the upper or lower portion of the screen. Thus to display all the contents of the RAM it is called four times with different parameters.

To display an 1, the second bit of a display byte is set. To display a zero, the 6th bit is set. This will give a picture of the signal similar to that of an oscilloscope, with each trace occupying one line of display byte. Three 256-byte blocks can be displayed this way on one screen but the display will be cluttered. It is recommended that you display two blocks and use the extra space for timing markers and comments and parameters.

If you are looking at the data bus of a microprocessor, you arelikely to prefer a display of the actual 0's and 1's which is easier to read when you are trying to relate the signals to instructions. You can write such a routine in BASIC with adequate performance.

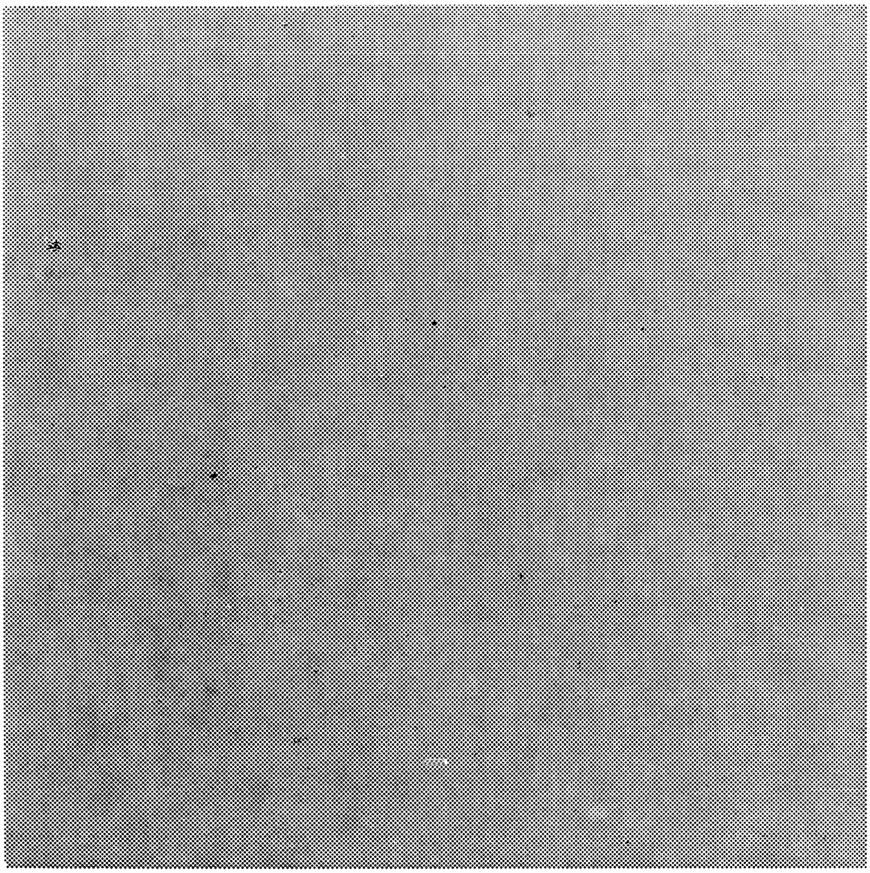
An interesting idea is to transfer the contents of the analyzer RAM into the 64 and use a disassembler to convert the acquired bus data into assembly instructions. This way, you could monitor execution of programs within your 64. This scheme, used skillfully, can simplify breaking protection systems of software manufacturers or just figuring out how undocumented programs work for reverse engineering purposes.

PARTS LIST

1. IC1, IC2: 2148H 1K×4 RAM memory.
2. IC3, IC4, IC5: 74F161 four bit counters.
3. IC6: 74F244 octal buffer.
4. IC7: 74LS244 octal buffer.
5. IC8: 74F00 quad NAND gate.
6. IC9: 74LS138 1 of 8 decoder.
7. IC10: 74S04, Texas Instruments part preferred.
8. 7805 five volt, 1A regulator.
9. 10 MHz crystal.
10. 12 pieces, 0.1 uF ceramic disc capacitors.
11. 2 uF/15V tantalum capacitor.
12. PC 1 and connector.
13. Two 210 ohm, 0.25 w resistors.
14. Sockets fov IC's, case, probes.

CHAPTER 9

HOME SECURITY SYSTEMS



PROJECT 35
PICKPROOF DIGITAL LOCK

A keypad, and a few resistors and capacitors interfaced to your 64 can send master safecrackers to the unemployment line. In addition, changing the combination to this pickproof lock is as easy as changing a BASIC line.

THEORY OF OPERATION

There are three ways to open a combination lock: 1. Use the key combination (obtained by legal or illegal means). 2. Go through all combinations one at a time. 3. Inspect the lock to figure out the combination. The last two methods are called "picking the lock" (from the use of picks on key locks). Method #3 is the one used by movie safecrackers on bank vaults where equipped with stethoscopes (or more recently, unspecified digital gizmos) they twirl the dial listening for the telltale clicks.

In our pickproof lock, there are too many combinations (one million) so method #2 will not work. In addition, every time a wrong combination is entered, the lock will not respond for 30 seconds (half a million minutes is a long time). Method #3 is also useless because the keypad and the lines to the keypad do not contain information about the combination, nor can the 64 be interrogated in any way through these lines.

In operation, the USER PORT data lines scan the keyboard matrix to determine which key (if any) has been pressed. When the correct combination is entered, PA2 is used to activate a solenoid to unlock the door. The scanning proceeds as follows: Each of the four least significant bits of the USER PORT is made a "0" and then the four most significant bits are checked to determine if a "0" is present, indicating that a key has been pressed using a table (KTABLE). If no key is pressed, the scanning continues.

The resistors in the circuit are there to insure that the inputs are "one" without signal and the capacitors help filter out noise picked up by the lines.

PROGRAMMING NOTES

The lock program is written in BASIC because execution speed is sufficient for real time operation and BASIC is much easier to use. To unlock, you enter on the keypad the combination (variable CKEY, line 240). The solenoid will be activated for 20 seconds to unlock the lock and then released to lock again. The keypad arrangement is an shown in Fig. 35.1.

Only 10 keys are used, arranged like the keypad on a pushbutton phone. Any other arrangement of 16 keys is possible by changing the KTABLE. The layout of the KTABLE corresponds to that of the keyboard. Unused keys are given codes to detect illegal key pushes.

The key combination in the listing is "1111111" to facilitate testing. You need only connect one key to check operation of the lock.

Line 515 prevents a keypress from registering more than once if a key is held pressed down. To enter the combination, keys must be pressed and released six times.

If you would like to increase the key combination to more digits, change line 240 to reflect the new key and also change line 350 to account for the increased digits. For example, for an 8 digit key change N<6 to N<8.

Line 380 determines how long the solenoid remains activated. You may change the constant (now 400) to suit your application.

For master key operation, change line 240 to read:

```
240 CKEY=111111: MKEY=222222
```

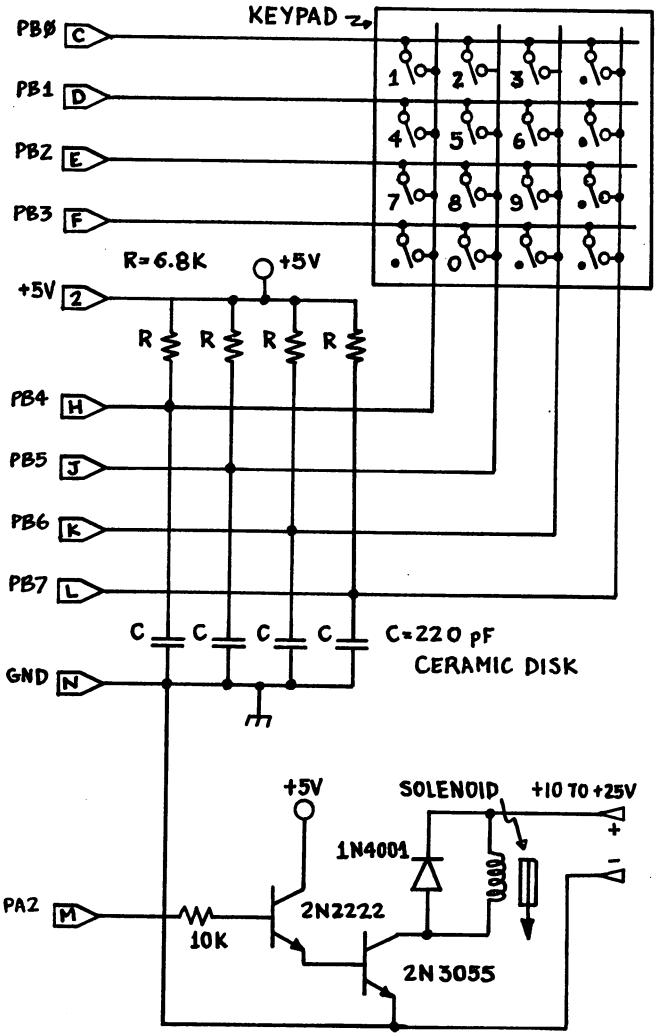
and line 360 to read:

```
360 IF INT (CODE) < > CKEY AND INT (CODE) < > MKEY
GOTO 700
```

Now CKEY is the regular key combination and MKEY is the master. The values assigned here are shown for easy testing, but keys can be any six digit number.

```
1 2 3 ·
4 5 6 ·
7 8 9 ·
· 0 · ·
```

Figure 35.1 Keypad Arrangement for the Digital Lock



```
100 REM  ** PICKPROOF LOCK **
105 REM
110 REM  SET UP KEY TABLE
115 REM
120 DIM KT(4,4)
130 KT(0,0)=1: KT(0,1)=2: KT(0,2)=3: KT(0,3)=13
140 KT(1,0)=4: KT(1,1)=5: KT(1,2)=6: KT(1,3)=12
150 KT(2,0)=7: KT(2,1)=8: KT(2,2)=9: KT(2,3)=11
160 KT(3,0)=14: KT(3,1)=0: KT(3,2)=14: KT(3,3)=10
195 REM
200 REM  SET UP PORT
205 REM
210 PRT=56577: POKE56579,15
220 POKE 59578, PEEK(59578) OR 4
230 POKE 59576,PEEK(59576) AND 251
240 CKEY=111111
295 REM
300 REM  LOCK OPERATION
305 REM
310 CODE=0: N=0
320 GOSUB 500
330 IF KPRESS>9 GOTO 700
340 CODE=CODE+KPRESS*(10^N)
350 N=N+1: IF N<6 GOTO 320
360 IF INT(CODE)<>CKEY GOTO 700
370 POKE 59576,PEEK(59576) OR 4
380 FOR I=1 TO 400: A=2^2: NEXT I
390 GOTO 230
395 REM
500 REM  KEYSCAN SUBROUTINE
505 REM
510 SB=1: I=0
515 IF (PEEK(PRT) AND 240)<>240 GOTO 515
520 POKE PRT,15-SB
530 Q=PEEK(PRT) AND 240
540 IF Q<>240 GOTO 570
550 SB=SB*2: I=I+1
560 IF SB<16 GOTO 520
565 GOTO 510
570 IF Q=224 THEN J=0: GOTO 620
580 IF Q=208 THEN J=1: GOTO 620
590 IF Q=176 THEN J=2: GOTO 620
600 IF Q=112 THEN J=3: GOTO 620
610 GOTO 700
620 KPRESS=KT(I,J)
650 RETURN
655 END
695 REM
700 REM  ERROR DELAY
701 REM
705 PRINT "ERROR"
710 FOR I=1 TO 900: A=2^2: NEXT I
720 GOTO 300
```

ADDED FEATURES

The program presented here performs only the basic lock function. It is easy to add various nice features to the basic lock, for example you can add a master key as described above. Another feature will be time-controlled access. Using the TI variable you can make the lock respond to certain keys only certain times of the day. For examble if the lock controls the power to a TV set, you could use it to determine when and for how long your children can watch TV. Still another use is logging of usage of lock. If every employee has his own code to enter and exit the premises you will know when he is there and when he is not. Finally, if you would like to get really advanced you can write the lock program in machine language and have a real time interrupt which will check the keypad every let's say 100 ms while the 64 is used in its normal mode.

GETTING SOPHISTICATED

The digital lock could still be picked if one could access radio emmissions from the 64 and decode its operation. A remote chance of course but paranoia is a necessary ingredient of a high security operation. If you need this extra protection, place the 64 in a cage made of fine mesh metal screen, soldered at the seams. The case of the 64 should be at least 5″ from the wire cage for effective shielding. Also wires going to the lock should be passed through ferrite beads just outside the case.

Continuous operation is a requirement for a lock, so provisions must be made for accidental power downs. An uninterruptible power supply based on batteries will maintain operation during power down. The lock is designed to remain locked in a power failure.

Finally, provisions should be made to break into the guarded area when the lock fails (for example, the 64 malfunctions) with minimum damage and cost, while unauthorized forced entry is very difficult.

PROJECT **36**

HOME SECURITY SYSTEM

The 64, equipped with the appropriate sensors, can become a 24-hour watchdog looking after the security of your home. Here we will see how to put together a home security system using some of the peripherals described in projects in this book.

THEORY OF OPERATION

A home security system provides warning against intrusion and problems like flooding or fire. It could also take action to scare away intruders (anything more drastic could create legal problems).

An effective warning system requires appropriate sensors strategically placed. In a home, the first step is to detect opening of doors and windows through which an intruder can enter. This is usually sensed magnetically. A reed switch is placed on the frame of the door and on the door, next to the switch, a magnet is placed that keeps the switch closed when the door is closed, as shown in Fig. 36.1. When the door opens, the switch opens signalling the computer.

You can buy the magnet-switch assemblies packaged for this purpose, with holes for the attaching screws and terminals for the wires. Each switch is connected to one bit of an input port and pulled high by a 3.3K resistor as shown in the schematic of Fig. 36.2. The 0.1 uF capacitor and 100 Ohm resistor help reduce noise that might be picked up by the wires going to the switch.

The zener diode and the fuse protect the inputs in case high voltages accidentally get in the wiring. This can happen for example if a nail is driven into the wall and pierces both a power line and a switch wire.

You may use a connon ground wire for a number of switches that are located in the same area, reducing the number of required wires. Do not run these wires in any

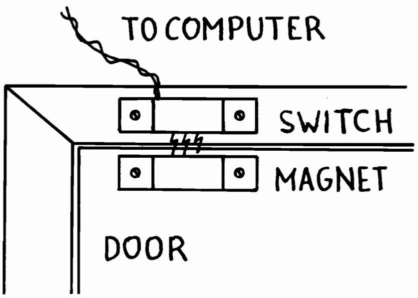


Figure 36.1 Placement of Magnetic Switch on Door

ducts or pipes used for power wiring. If you are building a house it is good idea to use a pipe network inside the walls through which various signal cables can be pulled. We are entering the era of the wired house so if you can afford it, it is best to be prepared. Use metal conduit pipe but if you cannot afford it you could use PVC pipes.

The 64 can scan the switches and print on its screen a drawing of a house using block graphics, showing the open and closed doors and windows. Arming and disarming the alarm system can be done by a hidden switch or by the lock described in project 35. When you leave, you enter the arming code. Then if a door or window opens, the alarm goes off. Using the circuit in project 18 or 19, your program can dial the police and deliver a message, saying that it is an unattended alarm device and that a break-in has occured at your address. Do not use a speech synthesizer to deliver the message, the police might be confused or think it is a prank. Use a cassette recorder instead. Also make sure to check with your local police department before giving their number to your 64. There have been problems in the past with automatic dialers and in some areas there may be restrictions on their use.

You can create a spectacular and sophisticated racket using your 64 that should be sufficient to scare away even professional thieves. The general guidelines are that sounds are more effective when they come from different directions with varying durations and timings. And that sounds are capable of creating strong avoidance emotions. Take for example the screeching of a chalk on the keyboard.

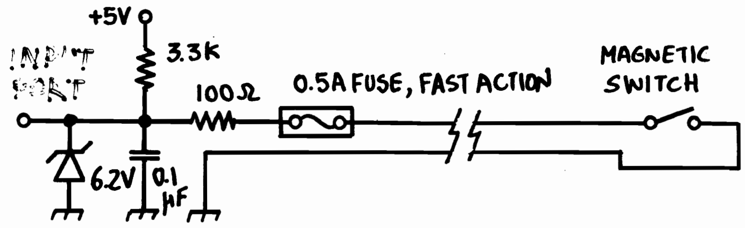


Figure 36.2 Schematic for the Connections of the Magnetic Switches to the 64

Use the sound synthesizer in the 64 and the speech synthesizer in project 24 to create just about anything your imagination can conjure. To switch between speakers in various rooms within the house, use reed relays driven by an output port.

The lights can be turned on and off using controllers like the one described in project 31. And finally, a circuit like the one shown in project 25 can be placed in every room to detect the noise the thief is making so that the sound effects can be orchestrated in that direction.

With these techniques and creativity, you can turn your house in a nightmare of a haunted house that can also be activated selectively for parties around Halloween.

The next step is to add to your security system the capability to detect fire and flooding. Use the circuit in project 15 to detect the presence of water. To detect fire, you could think of using a thermistor and look for high temperatures. This is not a good idea because if the thermistor is not close to the flames it may not give an indication until it is too late. The preferred method is to use a ceiling smoke detector and add in series with its speaker an LED optoisolator, as shown in Fig. 36.3. When the smoke alarm goes off, the 64 will be able to detect it and take action.

Using the DTMF receiver described in project 21, the home security system can be controlled by telephone to water the plants for example (using a solenoid activated valve and drip irrigation tubing) or interrogated to report the condition of the home while you are away.

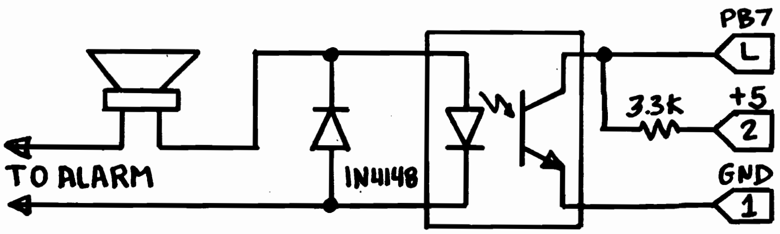


Figure 36.3 Connecting to a Smoke Alarm

These applications only scratch the surface of home controlling computers. Energy savings is another area. You could control solar heating systems very effectively, from solar water heaters to drawing the curtains to control sunlight entering a room.

PROGRAMMING NOTES

Time is not a critical element in the home security system, so all programs can be written in BASIC. In fact, once you get the required interfaces working, programming their integration into a home security system is straightforward.

However, if you wish, you could complicate this simple task in order to be able to use your computer while it controls the house. The scanning of the home security interfaces does not take very much time at all, so it can be done as a background task. That is, if all the security programs are in machine language, every time there is a jiffy interrupt (every 15 ms) you could go into the security scanning mode, make sure everything is OK and then come back to the program you are running to continue doing what you have been doing.

This way, you could use your 64 as usual, while the home security system is running as well. The only difference you might notice is a slightly longer time for some of your programs.

You can run in this background/foreground mode any BASIC program and all assembly programs that have no time critical loops.