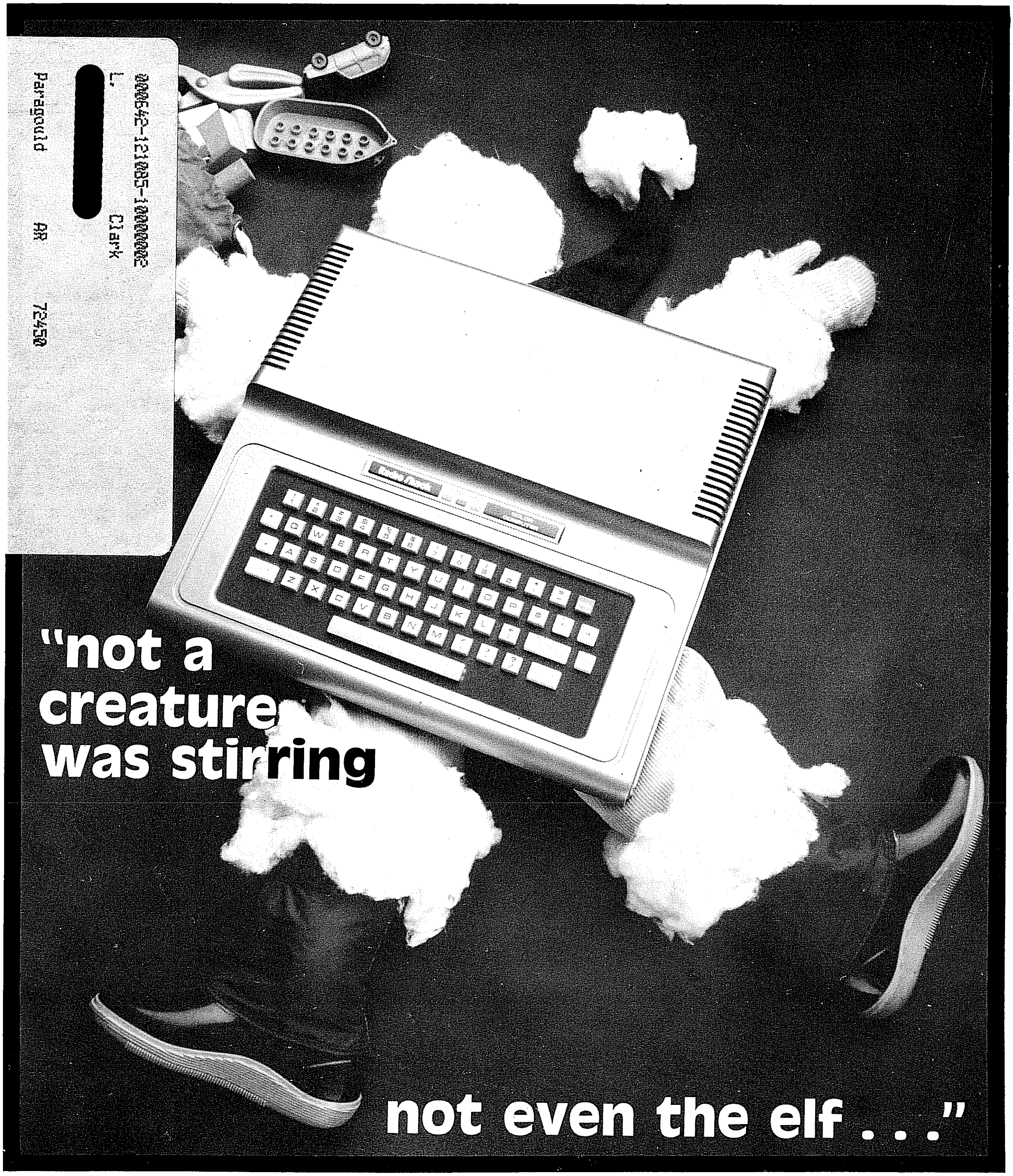
UNDERCOLOR

Volume 1 No. 2

December 25, 1984



Who's Who

Editor
Debra A. Marshall

Technical
Dennis Bathory Kitsz

Contributing Photographer
Charley Freiberg

Production Manager
Clare McCarthy

Paste-Up
Janet Patterson

Logistics Coordinator
Joanne Trottier

Technical Consultant and Publisher
Dennis Kitsz

Advertising:
Judy Knapp
(802)485-6139

Advertising:
Spencer Knowlton
(207)785-4151

Office
(802)485-6440

Editorial Office
(207)785-5148

Published By: ColorPlus Inc., Box 6809, Roxbury, VT 05669. (802)485-6440.

Circulation Problems? Contact David Gunn, Box 6809, Roxbury, VT 05669, 802-485-6440.
Subscriptions: Until Dec. 1, 1984: $20. After Dec. 1: $33. Add $14 for first-class postage. Foreign: until Dec. 1, 1984: Canada/Mexico $30 U.S.; other countries $40 U.S. *Under Color* is published 26 times a year. Call 802-485-6440, or write to *Under Color*, Box 6809, Roxbury, VT 05669, Attn: Perley Swann.
Advertisers: rates available.
Editorial Office: c/o Debra Marshall, RFD 1, Box 786, Union, ME 04862 207-785-5148. Submissions are welcome; include S.A.S.E. for returns. To receive author's guide, write: Author's Guide, Box 6809, Roxbury, VT 05669. Letters to Editor, manuscripts, article queries, etc. should be sent to editorial office.

Table of Contents

Cover Photo by Charles Freiberg

DefUSR by Terry Kepner

I have one of the early 4K Color Computers. It has been modestly used, but was a store demonstrator before I bought it, so it may have had more use than I realize. The Enter key doesn't always make solid contact when pressed. Sometimes it doesn't work, other times it seems to bounce and act as if I hit it twice. Can I remove the board and clean the contacts? Or could something be wrong with the chips or computer power supply?

Edwin McLean
Danville, CA

Yes, you can take apart and clean the keyboard, but you have to be *very* careful or you'll end up with tiny parts all over the place and will face the monumental task of trying to put them all back where they belong.

I would suggest you invest the money in a new keyboard, of which several are now available (Radio Shack $39.95; Mark Data Products SuperPro Keyboard $69.96; and SuperPro with adapter $70.95; HJL's Keyboard $79.95; and Keytronic's keyboard $89.95).

If you decide to clean the computer, get "contact cleaner" from an electronics store to do the job right. Rubbing alcohol and cotton swabs will also do the trick, but not as well, or with as lasting effect.

I have a 64K Color Computer with Tandon 100-1 (drive zero) and Shugart 400L (drive one) disk drives. My problem is consistent failures when making two-drive back-ups (single drive back-ups work just fine). I also have unexplained crashes at wide intervals.

I have software for checking drive speed, and both drives are within a few rpm's of 300.

Now for my questions: must both drives have the same type head loading? Can you provide me with the proper program shunt settings for the Shugart and Tandon drives? Can you give me an address where I can get the service manuals for these drives? Would a disk drive analyzer enable me to adjust my drives?

C.C. Nichols
New Braunfels, TX

If by head loading you mean whether or not both drives have to load the heads in the same manner, the answer is yes. If they don't, the operating system will think a drive is ready when it really isn't. The head should load as soon as the drive is selected (the LED comes on).

I can't help you with the Shunt settings since I don't have the manuals myself, but you can order the Shugart Manual from Hamilton/Avnet, 800-527-3387 (Shugart has a $100 minimum order quantity, Hamilton/Avnet doesn't). Ask for Shugart Part #39028. The repair manual should cost around $30. And the Tandon Manual is available from Tandon, P.O. Box 2107, Chatsworth, CA, 91311. Order part #179022-001, $29.50.

Definitely a disk drive analyzer will help you adjust your drives; it's the next best thing to using an oscilloscope on your drives (the necessary documentation is supplied with the program).

I recently purchased a 64K Color Computer II, and have a question: is it possible to modify the Color Computer 2 so it will supply 12 volts to the cartridge slot so you don't have to buy the Multipak Interface for many hardware products?

Ian Mount
Phoenixville, PA

You could do that, but I wouldn't recommend it: you'll probably overload the power supply and burn it out. The conversion is difficult, because the Color Computer 2 doesn't have the 12 volt supply on board. You would have to tap into the power transformer and build your own 12 volt circuitry.

All in all, while it's more expensive to buy the Multipak Interface, in the long run it's a better choice.

Does anyone know how to copy ROMpaks to cassette, and do it successfully? Does anyone know of a program that will let me create my own characters?

Wayne Tracer
Chiloquin, OR

Copying a ROMpak to cassette is simple: cover the ROMpak initialization pin (pin one), plug it into your computer, set up your tape recorder, turn on your computer, and type: CSAVEM"filename",&HC000, ?HDFFF,?HC000. This will put a copy of the ROMpak on tape, but it won't do you any good unless you have a 64K RAM computer. The ROMpaks use memory locations C000–DFFF, located above the ROMs.

If you have a 16K or 32K computer, the CLOADM routine tries to put the ROMpak program back where it was originally, starting at C000, where there's nothing now. Hence, nothing happens because there's no RAM in which to load the program.

There are only two solutions: relocate the program lower in memory where there is RAM (and take a chance that the program doesn't just happen to need that lower RAM for graphics and other data), or reload the program into a 64K computer, switching to 64K mode via special programs.

I don't know of any programs that let you design your own character set; does anyone else?

In July's The Color Computer Magazine *DEFUSR, you stated the Break disable wasn't possible even though the routines are in ROM and can be intercepted. There have been several assembly language routines in all the Color Computer magazines, but most are elaborate and some just don't work. The following program uses no memory and works on all Color Computers:*

```
0 FOR X = 248 TO 254
1 READ Y:POKE X,Y:NEXT X
2 POKE 410,126:POKE 411,0:POKE 412,248
2 DATA 50,98,28,178,126,173,165
4 FOR N= 1 TO 2 NEXT N
```

This must be placed at the beginning of a program. Line four is needed to start the program. If your program uses a For...Next command early on, line four can be omitted. The Break key can be turned back on via: POKE 411,130:POKE 412,185. The routine also disables the Pause (Shift @) and trace (TRON) functions.

The only problem occurs when you PEEK at the keyboard addresses. Usually the values are not passed on. It appears as if no keys are being pressed.

Dave Satterfield
Carson City, NV

I didn't say it was impossible, I said there wasn't a single POKE you could use to disable the Break key, as there is on the Model I and III computers. I said you need an assembly language routine to intercept the keyboard routine and filter out the Break key; which is what your short program does.

Your program does occupy memory; it is, however, below normal user RAM, down in page zero. You must be careful about using that area since it is reserved for future use by Radio Shack, and some commercial programmers are using that area for special tricks with their programs. It's possible that a loaded-in program could erase your disable Break routine.

I recently purchased a Radio Shack "Deluxe RS-232 Program Pak" (Cat. # 26-2226), and I have a few questions. First, can I use a Y adapter to plug in a disk drive with the RS-232 Pak? Do you know of any programs for this Pak? There are many faults in the program itself, along with the manual. Finally, why does it give me an SN ERROR when I download a Basic program, and why can't I transmit many assembly language programs through the Pak? The assembly language programs will transmit, but won't execute on either my machine or another after I have entered and exited the Pak.

I'm using a 64K Extended Basic computer with Modem I. Also note that there are problems using the Pak with the Radio Shack Multipak Interface; programs are lost when switching from Disk Basic to Basic.

Barton Fraize
Mount Pearl, Newfoundland, Canada

Are you planning to buy the Y connector to replace the Multipak Interface because of the lost programs when switching from Disk Basic to Extended Basic? If so, switching to the Y cable won't really help: the RS-232 Program Pak and Disk Drive ROMpak use the same addresses, so conflicts will arise when you try to use either unit.

The problem in switching from Disk Basic to Extended Basic is due to interfacing Disk Basic with Extended Basic. Flipping the switch is a bit like performing a lobotomy on your computer, with anesthesia: the computer usually dies and everything in memory is lost. With most program paks you aren't trying to save something in memory, so all you get is a quick reset.

If you're trying to transfer a program from Disk Basic to Basic, the best way is to save the program to tape while in Disk Basic, turn off your computer, disconnect the Disk ROMpak, turn your computer back on, and reload the tape.

Have you examined your Basic program after downloading to make sure everything is OK before trying to run it? And are you sure the programs you're downloading are Color Computer compatible?

After downloading a program, first check it out in the buffer for errors in transmission. The first line should start with a number. If it looks good, save the program to tape; turn off your compter; disconnect the RS-232 ROMpak; and reload the program. If you get a DS error, it means that Basic found a string of characters after a carriage return not starting with number. One way to fix this is to write a short Basic program that loads the program as a data file and stores it in an array, then find the error by scanning the array, fix it, write it back to tape, and try again.

The assembly language programs are more complex. You have to know their starting, ending, and execution addresses to use them. For details recheck your ROMpak manual.

I own a revision E board 32K Color Basic 1.1 version computer. One day I sat down at my computer desk, turned on the computer and display, and the keyboard failed to work; the computer ignored what I typed.

After resetting the computer several times the keyboard worked again, but it froze once more. Now the keyboard has been frozen for several days.

After examining the computer, I believe that one of my 6821's (U8) is fried or worn. What's wrong and what needs to be fixed?

Gary Mitchel
Mesquite, TX

You have three possible problems: the keyboard may be at fault, the keyboard/computer cable may be loose, or the PIA chip (U8) may be bad.

First, swap your two PIA chips. If U8 is bad, changing it will make the keyboard work properly. If that happens, you need to buy a new PIA chip to replace the bad one. Spectrum Projects (P.O. Box 21272, Woodhaven, NY, 11421, 212-441-2807) sells them for $9.95 each; the industrial grade PIA costs $14.95.

If changing the chips doesn't cure the problem, then either the cable or the keyboard is at fault. Unplug the keyboard (carefully) from your computer and put it back. If you still have a problem, try removing the cable from both the keyboard and the computer and putting it back on, reversed (the end that went to the computer now connects to the keyboard, and vice versa). If that fails to cure the problem, you'll need to replace the keyboard. A new one ranges from $39.95 to $89.95, depending on what you want and from whom you purchase it (Radio Shack $39.95; Mark Data Products SuperPro Keyboard $69.95; and SuperPro with adapter $70.95; HJL's Keyboard, $79.95; and Keytronic's Keyboard $89.95).

Good luck.

WILLIAM BARDEN, JR.

CROSS TALK

Part I

Or a way to make your computer talk to machines it has nothing in common with

I'm up to my ears in computers! I mean it — I have more computers than I want. I have Radio Shack Models I, II, III, IV, three Color Computers, two MC-10s, an IBM PC and PCjr, a Sanyo MBC-550, a Commodore VIC-20 and 64, and Timex/Sinclair TS2068s, TS1500s, and TS1000s, not to mention several other vintage systems. Do you know what most of these systems have in common? No, not MicroSoft Basic. Guess again. No, not the microprocessor. I'm sorry, your time is up... let's see what's behind curtain number three...

Most computer systems I own have RS-232C ports! RS-232C ports are a standard way to let a computer talk to the outside world — other computer systems, line printers, plotters, modems, and other kinds of peripheral equipment and devices.

The Color Computer uses its RS-232C port to connect to printers, modems, and other devices. In this article we'll look at how you can use this port so your Color Computer can communicate with other devices. And we'll bypass the ROM software so we're removed from its constraints.

RS-232C? An RS-232C interface is a standard way to connect electronic devices. The standard defines both the electronic and physical aspects of the connection. One of the nicest things about an RS-232C interface is that devices such as computers and computer equipment can be connected over fairly long distances, say, thousands of feet, and still transfer data without errors.

RS-232C is often called *serial* communication, and devices that use RS-232C are called serial devices. This is so because data is transferred as a string of bits (rather than all bits in parallel as a byte) in RS-232C communication.

The RS-232C standard defines the format of the serial data, which looks like Figure 1. A byte of data is converted to a string of eight bits with a leading start bit and a trailing stop bit or bits.

In the example in Figure 1, the eight-bit byte has been changed into a ten-bit stream of bits, with the eight data bits in the middle of the stream. The ten bit times are all the same length, so the total time will always be the same.

How are the bits sent? The easiest way is to transmit over two wires. Imagine a switch and battery at location A and a buzzer at location B, as shown in Figure 2. You're at location A, and you want to signal a Radio Shack store manager at location B. By prearrangement, the switch is closed and the buzzer sounds continuously. When you want to start transmission, you'll open the switch for one second. As soon as the store manager hears the buzzer stop, he will start counting seconds, using a Realistic 101 Timer. By prearrangement he knows that each bit time will be one second so that it'll take ten seconds to receive the entire ten bits. As soon as he detects the silent period, he'll wait 1½ seconds, putting him in the middle of the second bit time. He'll then note whether the buzzer is on or off, recording a one if the buzzer is on, and a zero if it isn't.

The manager will then wait another second; he will be in the middle of the third bit time allotment. He'll again record one or zero, depending on whether the buzzer is sounding. He'll do this for eight data bits. At the end of the eight bits (nine elapsed seconds), you'll send a continuous buzz for one second, and then leave the switch set so that the buzzer continues to sound.

The Radio Shack store manager will now take the eight data bits and arrange them as an eight-bit byte, placing the first bit received as the least significant. Looking up the byte in a table of ASCII characters, he'll convert the eight-bit value into a text character.

The process can be repeated for as many characters as you'd like to send. Each character will take ten seconds — one start bit of zero (no buzzer), eight data bits (buzz or no buzz), and one stop bit (buzz). The time in-between characters could be zero seconds, if you're sending a long message, or it might be minutes or hours, if you don't have any more text to send. The silent start bit (no buzz) alerts the manager that data is coming.

The process of passing data over these two wires is exactly analogous to RS-232C *asynchronous* communication. There may be varying times between characters, but once the start bit comes in, the receiving end counts time to get to the middle of each bit time, being as precise as possible about the timing.

RS-232C communication works in the same fashion, except that timing is much more rapid. Instead of one second for each bit time, there's only thousandths of a second for the bit times: from about 100 to 9600 bits or more can be sent each second.

TWO WAY TRANSMISSION. If you want to be able to signal the Radio Shack store manager, and also to receive data from him, you could just add another wire to the two wires you're using to make a three-wire system instead of two (Figure 3). You'd now have two buzzers, one at each end, and two sets of switches and batteries. The "common" wire would be the so-called *ground* wire. You could actually signal each other at the same time, except that it might be confusing.

The first system (the ability to send data in one direction only) is called *simplex*. The second system (the ability to send data in both directions) is called *duplex*. If the Radio Shack store manager and you are easily confused and can't receive while the other is transmitting, the duplex operation is called *half duplex* — transmission can only be performed in one direction at a time. If you both are very coordinated, the operation is called *full duplex* — you can both send and receive data simultaneously.

MORE CHARACTERISTICS. A number of different formats are used in RS-232C. We've been talking about ten bits. Often, though, one start bit, seven data bits, and two stop bits are used. Here again, there are ten total bits, but only seven data bits. This format is often used for ASCII, or character data, as ASCII codes use only seven bits to represent the alphabet, digits, or special characters.

Other formats might use as few as five data bits. The most frequently used format in computers, though, is one start bit of zero, seven or eight data bits, and one or two stop bits of one.

Another bit is sometimes thrown in as well. A *parity* bit is occasionally used to check on the data. The parity bit is the last data bit sent, and is set to a one or zero to make the total number of one bits in the data even or odd. We won't be using a parity bit in the cases we're talking about. *No parity* is often used in communications systems because there are other data checks. One frequently-used check is to send out a character and then have the receiving sytem *echo back* the same character, so the sending system can compare the character received with the one sent. This system is often used in full duplex systems, where you press a key to send a character without displaying the character on the screen. The receiving system sends back the character received which *is* displayed on the screen. The process is so rapid that it appears you typed the character and it was simultaneously displayed on the screen; it actually came from the other terminal!

The RS-232C standard uses a number of different data rates expressed as *bauds*. A baud is a unit of information transmission speed which is not necessarily equal to bits per second. Bauds commonly used on the Color Computer and other systems are 300, 600, 1200, and 2400 baud. In 300 baud transmission, there are ten bit times per character or byte, so that 30 characters per second can be sent. In 600 baud, 60 characters per second can be sent. In 1200 and 2400 baud, 120 and 240 characters or bytes per second can be sent.

To find the length of a bit time, divide one by the baud. A baud of 600, for example, has a bit time of 1/600, or 1.666 milliseconds (thousandths of a second). The total length for ten bits is 16.66 milliseconds.

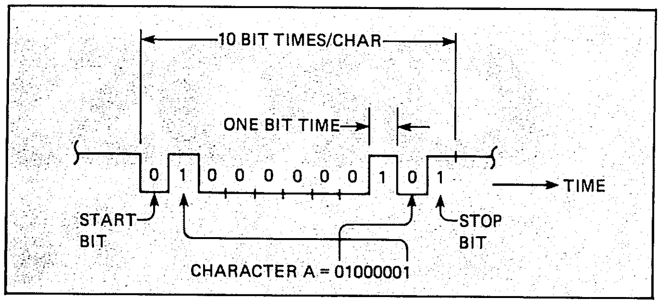


Figure 1. Typical Format: RS-232C Serial Data

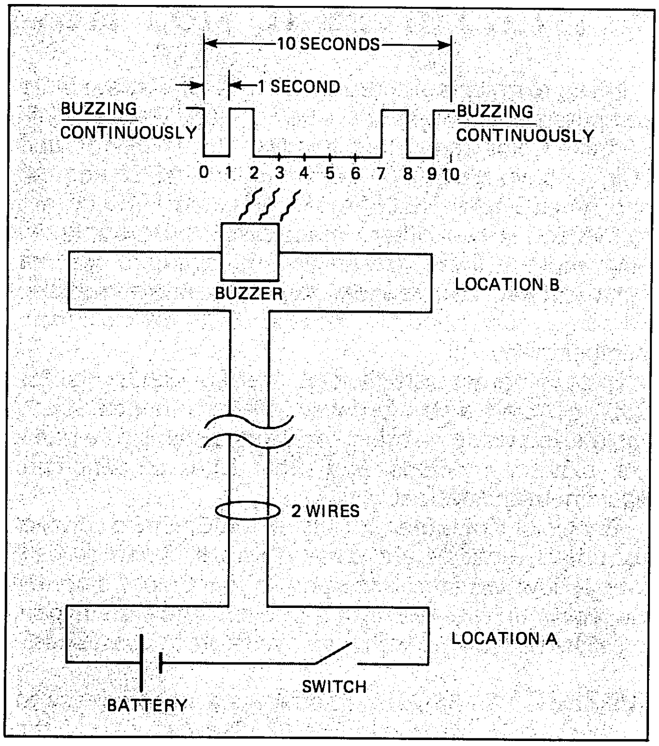


Figure 2. Remote Data Communication

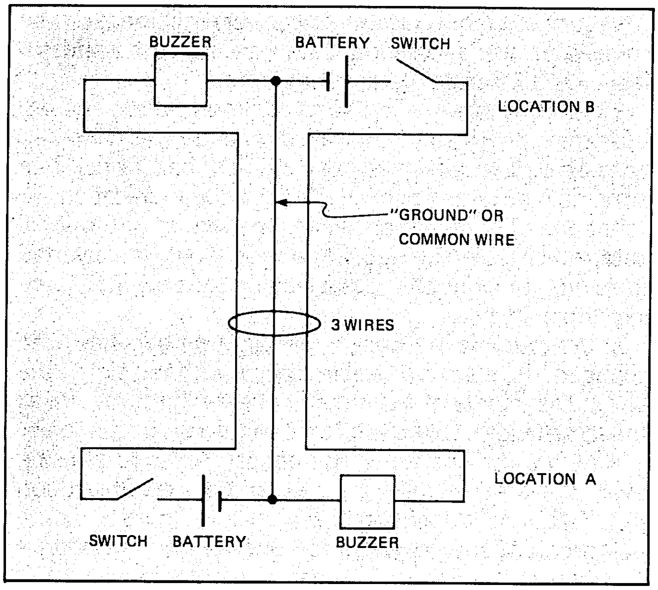


Figure 3. Two-Way Remote Data Communication

MORE SIGNALS. We discussed a three-wire system earlier. One wire is ground, or common. Two other wires transmit data from one end of the system to the other. On one end, one wire is called *TD* for transmit data and the other wire is called *RD* for receive data. On the other end, the names and connections are reversed.

There are many other signals present in RS-232C systems, however. Many are necessary for modem use. *Modems* are devices that take serial data and change it into audio tones so that data can be transmitted over telephone lines. One signal commonly used is *CD*, or Carrier Detect. This signal indicates to the computer that the modem is receiving the carrier tone of the sending device. In fact, this carrier is actually the stop condition of the TD line, the continuous buzzing in our earlier example. Another signal is *RTS*, or Request To Send. This signals the other end of the RS-232C connection that data is ready for transmission. *CTS*, or Clear To Send, informs the computer that it is all right to start transmission.

There are 22 signals in the RS-232C specification, representing various conditions and states. The 22 signals are connected via a 25-pin connector called a DB-25 connector, shown in Figure 4. The DB-25 connector can be seen on modems and other serial devices, and is used on the Radio Shack Model I, II, III, IV, Model 100, and other computer systems. The Color Computer, however, doesn't use a 25-pin connector. It uses a four-pin DIN connector, shown in Figure 5.

Though the complete set of RS-232C signals are useful, the most important of the signals are still ground, TD, and RD. The Color Computer RS-232C throws in CD for modem applications.

Using these four signals it's possible to connect to a modem, transfer data to a serial printer, or connect to another computer system.

HOW OURS DIFFERS. The connector used on the Color Computer isn't the only difference between the Color Computer and other computer system RS-232C interfaces. The biggest difference is that the Color Computer RS-232C signalling is accomplished primarily by software, rather than hardware, as on other systems.

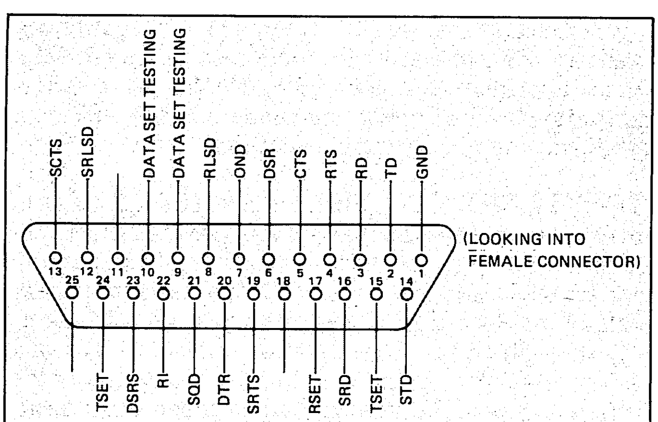


Figure 4. DB-25 Connector

Systems such as the Radio Shack Model IV have built-in hardware that handles the task of reading in the serial bit stream and converting it to a parallel byte, or sending out data after conversion from a parallel byte. This operation goes on independently from other operations in the system. About all a program must do is write a byte to be transmitted to the RS-232C hardware, or read in a received byte.

In the Color Computer a program must convert parallel bytes to serial bit streams, add start and stop bits, and then send the bit streams out over the TD line, or, alternately, read the RD line, strip off the start and stop bits, and assemble the received byte.

HOW RS-232C WORKS. It's important to know how the Color Computer hardware implements RS-232C communications so you can use the hardware in your own data communications applications. If you're not a hardware type, please bear with me: I'll make it as painless as possible.

The Color Computer has a number of devices called PIAs, *Peripheral Interface Adapters*. You can envisage the PIAs as memory locations; they are addressed in the same manner as other memory locations in the Color Computer. The addresses of the PIAs, however, are in the $FFXX area of the memory map of the Color Computer. There are PIAs to control sound, cassette I/O, read the keyboard, and other operations.

PIAs are unlike memory locations in that each bit of the PIAs is a signal line routed to various Color Computer functions. Once that bit in the PIA is set it remains set until new data is stored in the PIA. Alternately, a PIA bit might be an input bit that comes from a signal line. The on/off condition of the signal line can be determined by reading the PIA location. In digital engineering terms, the PIAs are *latches* that read and store binary data routed to or from signal lines.

Two PIA addresses handle RS-232C data. The PIA addressed by address value $FF22 holds the current state of the RD line in bit seven (the most significant bit). Whenever you PEEK at location $FF22, bit seven represents the zero or one state of the RD line. By properly timing the PIA reading, you can position the read in the middle of the bit time and decode incoming RS-232C data.

The PIA addressed by address value $FF20 controls the TD line. Putting a one bit into bit one of that PIA address using a POKE or other means will output a one bit on the

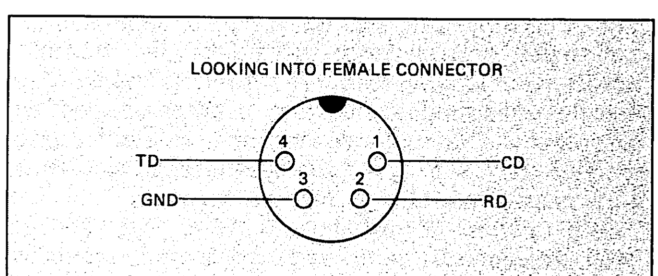


Figure 5. Color Computer 4-Pin DIN Connector

TD line. The one bit will stay there until a new output to the PIA arrives. By properly timing the output to PIA $FF20, you can create an RS-232C stream of TD bits.

A third bit is used to read the state of the CD line from the Color Computer RS-232C connector. This PIA works somewhat differently than the other PIA lines, as it controls an interrupt input. We won't talk about this bit, as we can do most everything we want with the RD, TD, and ground lines.

This is probably a good time to mention that actual signal levels found on the DIN connector for RS-232C signals are not what you would expect. Outputting a one bit on the TD line, for example, actually results in a −12 Vdc level. Outputting a zero bit results in a +12 Vdc level. This is compatible with the RS-232C specification. Be aware of this fact if you will be measuring signal levels with a voltmeter or oscilliscope. (Editor's Note: The Color Computer 2 does not have + or −12 volts available from its power supply, so it uses + and −5 volts on its RS-232C data lines. This works quite well over the usual — room sized — distances between computer and printer or modem.)

RS-232C AND BASIC. Unfortunately, RS-232C and Basic are not very compatible! The reason is speed — Basic is simply too slow to keep up with the speed of RS-232C data. Consider the 600 baud used for LLISTing Basic programs, for example. We said earlier that 600 baud was about 1.666 milliseconds per bit time. That represents 600 possible changes per second. A Basic loop such as:

```
100 FOR I=1 TO 6000
110 NEXT I
```

takes a little under nine seconds, or about 666 counts per second. If more processing is added, it's obvious that Basic can't keep up with 600 baud.

To do anything with the Color Computer RS-232C, then, we're forced to use assembly language. You won't have to actually *do* anything in assembly language, however. I've done all the hard work, and you can reap the benefits! For the sake of you assembly language buffs, however, I'll explain what's happening in the program I've prepared. Also, you'll need to know the general way the program works to be able to use it from Basic.

THE PROGRAM. The RS-232C program (Listing 1) consists of two parts, an output character part and an input character part. The output character portion will send out a single byte over the TD line to another computer system or serial device, such as a printer. The input character portion will read in from one to many bytes from the RD line and store the received bytes in a specified memory area. The input character portion will end when a specified number of bytes have been received or when a key is pressed on the Color Computer keyboard.

Both routines will work at bauds of 300, 600, 1200, or 2400. I regularly use 2400 baud in my computer room to transfer data between the Color Computer and my Model III, and I experience virtually no errors. You should be able to use 2400 bauds over moderately long distances (a few hundred feet) without problems.

Any number of data bits can be used with the program, in addition to any number of stop bits. There is no parity bit provision, so you'll have to verify the data with a checksum or other means.

Both programs use a common parameter block to define the speed and parameters of the RS-232C transmission, as shown in Figure 6. Location $3F00 (16128) holds the number of data bits. Location $3F01 (16129) holds the number of stop bits. Location $3F02 holds the baud in encoded form. Location $3F03 (16130) holds the character to be transmitted or the last character received. Locations $3F04,5 (16132, 16133) hold the starting memory address for storage of received data and locations $3F06,7 (16134, 16135) hold the ending address for received data.

$3F00 (16128)		NO. OF DATA BITS (1–8)
$3F01 (16129)		NO. OF STOP BITS (1–2)
$3F02 (16130)		BAUD RATE CODE
$3F03 (16131)		CHARACTER
$3F04 (16132)		START MEMORY ADDRESS
$3F05 (16133)		FOR RECEIVE
$3F06 (16134)		END MEMORY ADDRESS
$3F07 (16135)		FOR RECEIVE

Figure 6. Parameter Block

To use the output character routine, POKE the number of data bits, the number of stop bits and the baud (0=300, 1=600, 2=1200, and 3=2400) into the proper locations. You need only do this once. Then POKE the character to be transmitted into location $3F03 and call location $3F18 (16132) with a USR call. After the character has been transmitted the USR function will return to the Basic program.

To use the input character routine, set up the same parameters (or use the existing parameters) except for the baud. The baud codes are 4 – 7 for bauds of 300, 600, 1200, and 2400. POKE the starting and ending addresses of the data block to be used as a buffer. You may use the text screen addresses of $400 (4,0) and $5FF (5,255) if you want the data to appear on the screen as it's received. Then call location $3F43 (16195) with a USR call. You'll return after the last memory location has been filled or when you press any key on the keyboard.

SAMPLE BASIC DRIVER. Program Listing 2 is a sample Basic driver program that lets you use your Color Computer as a dumb terminal. The first portion of this program is the machine code of the assembly language program as data values. Regardless of your Basic code, use statements 110 through 260 to move the code to the $3F00 area. If you have a Color Computer with Extended Basic, use the program as is. If you do not have Extended Color Basic, change the four lines as shown in the listing.

Next month — applications! ■

PROGRAM LISTING 1
RS-232C ASSEMBLY LANGUAGE PROGRAM

16K Extended Color Basic

```
3F00                  ORG     $3F00
                * RS-232-C OUTPUT CHARACTER
3F00      07    NOBITS  FCB     7         ; #DATA BITS
3F01      02    NOSTOP  FCB     2         ; #STOP BITS
                * XMIT BAUD RATE: 0=300 1=600 2=1200 3=2400
                * RCV BAUD RATE : 4=300 5=600 6=1200 7=2400
3F02      00    BAUDR   FCB     0         ; BAUD RATE
3F03      00    CHAR    FCB     0         ; IN OR OUT CHAR
3F04      3F03  MEMS    FDB     CHAR      ; MEMORY START
3F06      3F03  MEME    FDB     CHAR      ; MEMORY END
3F08      016E  BAUDTB  FDB     366       ; 300 BAUD TRANSMIT
3F0A      00B0          FDB     176       ; 600
3F0C      0050          FDB     80        ; 1200
3F0E      0020          FDB     32        ; 2400
3F10      015A          FDB     346       ; 300 BAUD RECEIVE
3F12      00B0          FDB     176       ; 600
3F14      0050          FDB     80        ; 1200
3F16      001E          FDB     30        ; 2400
3F18 1A   50    RSOUT   ORCC    #$50      ; RESET FIRQ, IRQ
3F1A B6   3F03          LDA     CHAR      ; GET CHARACTER
3F1D F6   3F00          LDB     NOBITS    ; GET #DATA BITS
3F20 49                 ROLA              ; ALIGN FOR OUTPUT
3F21 34   03            PSHS    A,CC      ; SAVE CHARACTER
3F23 4F                 CLRA              ; 0 BIT IN BIT POSITION 1
3F24 8D   17            BSR     OUTPUT    ; OUTPUT BIT POSITION 1
3F26 35   03            PULS    A,CC      ; RESTORE CHARACTER
3F28 34   03    LOOP1   PSHS    A,CC      ; SAVE CHARACTER
3F2A 8D   11            BSR     OUTPUT    ; OUTPUT AND DELAY
3F2C 35   03            PULS    A,CC      ; RESTORE CHARACTER
3F2E 46                 RORA              ; ALIGN NEXT BIT
3F2F 5A                 DECB              ; DECREMENT # OF BITS
3F30 26   F6            BNE     LOOP1     ; LOOP IF NOT DONE
3F32 F6   3F01          LDB     NOSTOP    ; GET 3 OF STOP BITS
3F35 86   02    LOOP2   LDA     #2        ; 1 BIT=STOP
3F37 8D   04            BSR     OUTPUT    ; OUTPUT STOP BIT
3F39 5A                 DECB              ; DECREMENT # OF STOP BITS
3F3A 26   F9            BNE     LOOP2     ; GO IF NOT DONE
3F3C 39                 RTS               ; RETURN
                * OUTPUT SUBROUTINE
3F3D B7   FF20  OUTPUT  STA     $FF20     ; OUTPUT BIT POSITION 1
3F40 8D   58            BSR     DELAY     ; DELAY ONE BIT TIME
3F42 39                 RTS
                * RS-232-C INPUT CHARACTER
3F43 1A   50    RSIN    ORCC    #$50      ; RESET FIRQ, IRQ
3F45 10BE 3F04  LOOP9   LDY     MEMS      ; INITIALIZE BUFFER START
3F49 F6   3F00  LOOP10  LDB     NOBITS    ; GET # DATA BITS
3F4C 8D   3F    LOOP11  BSR     INPUT     ; GET INPUT BIT
3F4E 81   80            CMPA    #$80      ; TEST MS BIT
3F50 25   0C            BLO     LOOP15    ; GO IF 0
3F52 4F                 CLRA              ; FOR KEYBOARD
3F53 B7   FF02          STA     $FF02     ; OUTPUT TO KB
3F56 B6   FF00          LDA     $FF00     ; READ ALL ROWS
3F59 43                 COMA              ; KEYPRESS=1
3F5A 26   30            BNE     LOOP50    ; GO IF KEYPRESS
3F5C 20   EE            BRA     LOOP11    ; CONTINUE INPUT IF NOT
3F5E 8D   41    LOOP15  BSR     DELAYH    ; HALF BIT DELAY
3F60 8D   38            BSR     DELAY     ; TOTAL= 1 1/2 BIT TIMES
3F62 4F                 CLRA              ; INITIALIZE CHAR
3F63 8D   28    LOOP21  BSR     INPUT     ; GET INPUT BIT
3F65 8D   33            BSR     DELAY     ; DELAY BIT TIME
3F67 5A                 DECB              ; DECREMENT COUNT
3F68 26   F9            BNE     LOOP21    ; GO IF NOT ALL BITS
3F6A C6   08            LDB     #8        ; FINAGLE
3F6C F0   3F00          SUBB    NOBITS    ; FIND BITS TO SHIFT
3F6F 27   04            BEQ     LOOP25    ; GO IF NONE TO SHIFT
3F71 44         LOOP24  LSRA              ; ALIGN TO RIGHT
3F72 5A                 DECB              ; DECREMENT COUNT
3F73 26   FC            BNE     LOOP24    ; CONTINUE
3F75 F6   3F01  LOOP25  LDB     NOSTOP    ; GET # STOP BITS
3F78 8D   20    LOOP31  BSR     DELAY     ; DELAY ONE BIT TIME
3F7A 5A                 DECB              ; DECREMENT # OF STOP BITS
3F7B 26   FB            BNE     LOOP31    ; GO IF NOT DONE
3F7D B7   3F03          STA     CHAR      ; STORE CHARACTER
3F80 A7   A4            STA     ,Y        ; STORE IN MEMORY
```

Color Count

What's a picture worth? Well . . . a Color hi-res screen picture is 256 pixels wide by 192 pixels high, which comes to a total of 49,152 pixels. Each pixel is coded by a single bit in that mode. Divide by 8 (8 bits to a byte) and you discover a Color hi-res picture contains 6114 (6K) bytes of information. Now, typically a single English character is coded by a single byte. Thus, that Color hi-res picture contains 6K characters worth of information. The average word is about five letters long, and is typically accompanied by a space. Thus, the average word takes six characters, total. Six into 6K is about 1000. And so, we discover that a picture . . . a hi-res Color picture at any rate . . . is worth almost precisely 1000 words.

—Marty Goodman

Short Stuff

```
10 PMODE4,1: SCREEN1,1:T=RND
   (-TIMER)
20 PCLS: FORX=1 TO 40: LINE-
   (RND(255),RND(191)),PSET: NEXT:
   GOTO20
```

PROGRAM LISTING 1 (CONT.)

```
3F82 10BC 3F06      00840          CMPY     MEME     ; COMPARE TO END
3F86 27   04        00850          BEQ      LOOP50   ; GO IF AT END
3F88 31   21        00860          LEAY     1,Y      ; BUMP POINTER
3F8A 20   BD        00870          BRA      LOOP10   ; CONTINUE
3F8C 39             00880 LOOP50   RTS               ; RETURN TO CALLING
3F8D 44             00890 INPUT    LSRA              ; ALIGN A
3F8E 34   02        00900          PSHS     A        ; SAVE IN STACK
3F90 B6   FF22      00910          LDA      $FF22    ; GET RS-232C DATA INPUT
3F93 46             00920          RORA              ; ALIGN TO BIT 7
3F94 46             00930          RORA
3F95 84   80        00940          ANDA     #$80     ; MASK OUT REST
3F97 AA   E0        00950          ORA      ,S+      ; MERGE IN REMAINDER OF CHAR
3F99 39             00960          RTS
3F9A 8D   0C        00970 DELAY    BSR      GETCNT   ; GET DELAY COUNT
3F9C 30   1F        00980 LOOP41   LEAX     -1,X     ; DECREMENT DELAY COUNT
3F9E 26   FC        00990          BNE      LOOP41   ; GO IF NOT DONE
3FA0 39             01000          RTS               ; RETURN
3FA1 8D   05        01010 DELAYH   BSR      GETCNT   ; GET DELAY COUNT
3FA3 30   1E        01020 LOOP51   LEAX     -2,X     ; DECREMENT
3FA5 26   FC        01030          BNE      LOOP51   ; GO IF NOT DONE
3FA7 39             01040          RTS               ; RETURN
3FA8 34   06        01050 GETCNT   PSHS     D        ; SAVE A,B
3FAA B6   3F02      01060          LDA      BAUDR    ; GET BAUD RATE
3FAD C6   02        01070          LDB      #2
3FAF 3D             01080          MUL               ; D NOW 2 TIMES INDEX
3FB0 C3   3F08      01090          ADDD     #BAUDTB  ; POINT TO DELAY COUNT
3FB3 1F   01        01100          TFR      D,X      ; DELAY COUNT NOW IN X
3FB5 AE   84        01110          LDX      ,X       ; GET ACTUAL COUNT
3FB7 35   06        01120          PULS     D        ; RESTORE A,B
3FB9 39             01130          RTS
          0000      01140          END
00000 TOTAL ERRORS

BAUDR    3F02        LOOP11   3F4C        LOOP51   3FA3
BAUDTB   3F08        LOOP15   3F5E        LOOP9    3F45
CHAR     3F03        LOOP2    3F35        MEME     3F06
DELAY    3F9A        LOOP21   3F63        MEMS     3F04
DELAYH   3FA1        LOOP24   3F71        NOBITS   3F00
GETCNT   3FA8        LOOP25   3F75        NOSTOP   3F01
INPUT    3F8D        LOOP31   3F78        OUTPUT   3F3D
LOOP1    3F28        LOOP41   3F9C        RSIN     3F43
LOOP10   3F49        LOOP50   3F8C        RSOUT    3F18
```

PROGRAM LISTING 2

BASIC DRIVER

16K Extended Color Basic

```
100 REM  TERMINAL PROGRAM
110 CLEAR 100,16127
120 DATA 7,2,0,0,63,3,63,3,1,110
,0,176,0,80,0,32
130 DATA 1,90,0,176,0,80,0,30,26
,80,182,63,3,246,63,0
140 DATA 73,52,3,79,141,23,53,3,
52,3,141,17,53,3,70,90
150 DATA 38,246,246,63,1,134,2,1
41,4,90,38,249,57,183,255,32
160 DATA 141,88,57,26,80,16,190,
63,4,246,63,0,141,63,129,128
170 DATA 37,12,79,183,255,2,182,
255,0,67,38,48,32,238,141,65
180 DATA 141,56,79,141,40,141,51
,90,38,249,198,8,240,63,0,39
190 DATA 4,68,90,38,252,246,63,1
,141,32,90,38,251,183,63,3
200 DATA 167,164,16,188,63,6,39,
4,49,33,32,189,57,68,52,2
210 DATA182,255,34,70,70,132,128
,170,224,57,141,12,48,31,38,252
220 DATA 57,141,5,48,30,38,252,5
7,52,6,182,63,2,198,2,61
230 DATA 195,63,8,31,1,174,132,5
3,6,57,15308
240 CK=0:FOR I = 16128 TO 16313
250 READ A: POKE I, A: CK=CK+A
260 NEXT I: READ A: IF CK<>A THE
N PRINT " DATA INCORRECT, PLEASE
 CHECK": STOP
270 POKE 16132,4: POKE 16133,0
280 POKE 16134,5: POKE 16135,255

290 POKE 16130,4
300 DEFUSR0 = 16195 'POKE 275,63
: POKE 276,67 in Standard BASIC
310 A = USR0(0)  'A = USR(0) in
Standard BASIC
320 A$ = INKEY$: IF A$ = "" THEN
 320
330 PRINT A$;
340 POKE 16131, ASC(A$)
350 POKE 16130, 0
360 DEFUSR0 = 16152 'POKE 275,63
: POKE 276,24 in Standard BASIC
370 A = USR0(0)  'A = USR(0) in
Standard BASIC
380 GOTO 290
```

DENNIS KITSZ

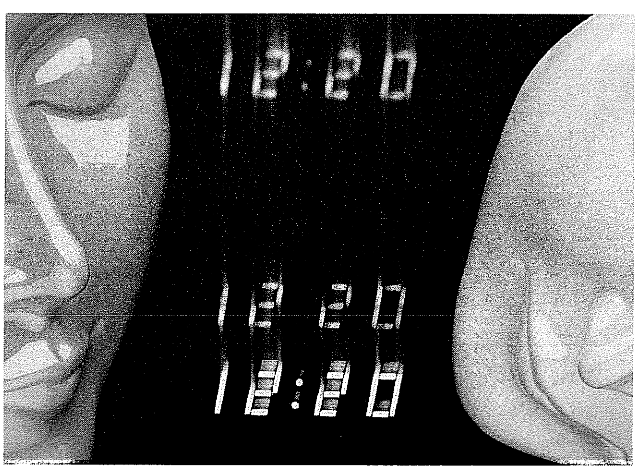
A Real-Time Talking Clock

Part II

A real-time clock with voice synthesis!

Ed's Note: Articles referred to, except Part I of this series, are found in back issues of The Color Computer Magazine, *unless otherwise noted.*

The idea of computer-generated speech always reminds my of my elementary school days, when Bell Laboratories amazed the world with its first talking computer. It sang *Daisy* in a hollow, inflectionless voice. I heard that same voice about ten years later in *2001: A Space Odyssey*. Director Stanley Kubrick paid homage to that first digital talker as the errant supercomputer HAL is dismantled, board by electronic board. It first begins to lose its intelligence and other human traits; its speech becomes simpler, then mindless. Finally HAL begins singing *Daisy*—in that hollow, inflectionless voice.



Bell Labs' *Daisy* is nearly a quarter-century past. Last month I presented the concepts of digitally recording your own speech, and this month I've got a simple circuit that will provide you with high-quality speech with a built-in clock vocabulary. With just a little more work, a speech device with full, software-controlled inflection can be built.

At the heart of this speechmaking is the General Instrument SP0256 vocal tract synthesizer circuit. Vocal tract synthesizers work by emulating the vocal tract—lungs, vocal cords, throat, nose, mouth, tongue and lips. Every word makes use of a significant portion of the vocal tract, which can be thought of as a *tone generator*, a *noise generator*, and a *filter.*

Your Personal Synthesizer

To see how your built-in generators and filters work, try this. Hum a note in the middle of your voice range—the *pitch generator* at work. Now sing "eeee" on that note. Change that sung "eeee" slowly to an "oooo" sound on the same note. Now stop the note and produce the unvoiced "th" sound, like you find at the end of "myth."

Hook them together slowly: "eeee" changing the "oooo" ending with "th." That's the first step. You've sounded a note, and filtered it with your mouth as "eeee" changed to "oooo." The "th" sound is your noise generator. The next step is to change the pitch. Again, sing "eeee" changing to "oooo," but make the pitch drop as you speak, that is, slide from a higher to a lower singing note while singing "eeee-oooo." End it in "th" again: EEEE-OOooo-th.

The final step is to shorten the "eeee" sound, and put the whole thing together quickly—pitch, filtering and noise. EE-OOoooo-th. Eooth. Youth. There's the word: youth. It consists of a sliding tone, a changing filtered sound, and a bit of noise.

Even the most complex sounds can be broken down into basic chunks of sound. There are the filtered pitches such as ee, oo, oh, ah, ih, eh; noises like th, f, s, sh; and filtered voice-pitch combinations that produce b, p, m, n, l, r, ch and so forth. Pitch changes provide inflection to the words. All these pieces are called *allophones,* and can be emulated with electronic pitch generators, noise generators, and filters. Such electronic allophones are the basis of many speech synthesis circuits, such as the General Instrument SP0256 device that interfaces so easily with the Color Computer. Inside the SP0256 are the sound generators and filters, plus a small microprocessor which, given sets of instructions, can produce complex, intelligible sound.

Interfacing The Synthesizer

The SP0256-017 and its companion SPR016-117 serial speech ROM form a talking clock set; these are the most accessible speech synthesis parts, available for $10 from Radio Shack (catalog number 276-1783). The interface schematic is shown in Figure 1. It uses the CoCoPort input/output board (Custom Color, April–May, 1983), making it compatible with disk drives and cartridges. The entire Color Computer talking clock can be constructed for under $25.

Construction is simple, using wire-wrapping or point-to-point soldering. Use sockets for all integrated circuits, and be cautious when handling the synthesizer chip and speech ROM. These are static-sensitive parts, and should be left in their packages until you are ready to put them in their sockets. Sufficient power can be obtained from the computer itself (5 volts is found on pin 9 of the Color Computer's edge connector).

The final step is getting the clock talking. Listing 1 contains the complete interrupt-driven clock display software presented last time, but integrated into this listing is a driver for the speech board. Have a look at the listing, beginning at the speech driver (Line 940), and the pin diagram (Figure 2).

The overall point of this series is timing, so consider that there's one important requirement for speaking, since in computer terms speaking is a very slow process: your program has to know when the last spoken word is complete. The SP0256 accepts a signal on ALD (Address Load, pin 20) to tell it to start talking, and provides the LRQ (Load Request, pin 9) to inform the host computer when its input buffer is full and cannot accept any more information.

The program in Listing 1 accommodates these special signals, and thus addresses the demands of two real-time problems: maintaining the correct real time, and keeping track of when the speech synthesizer is ready to speak. As you can imagine, the interrupt process—with its regular, predictable, short bursts of programming—is the ideal way of making both timekeeping and speechmaking invisible to any other programs which may be running.

Turn first to the flowchart in Figure 3, which outlines the sequence needed to announce the time. The actual interrupt-driven timekeeping was described in the first part of this project.

Talking Is Tricky

Speaking the time is second nature to

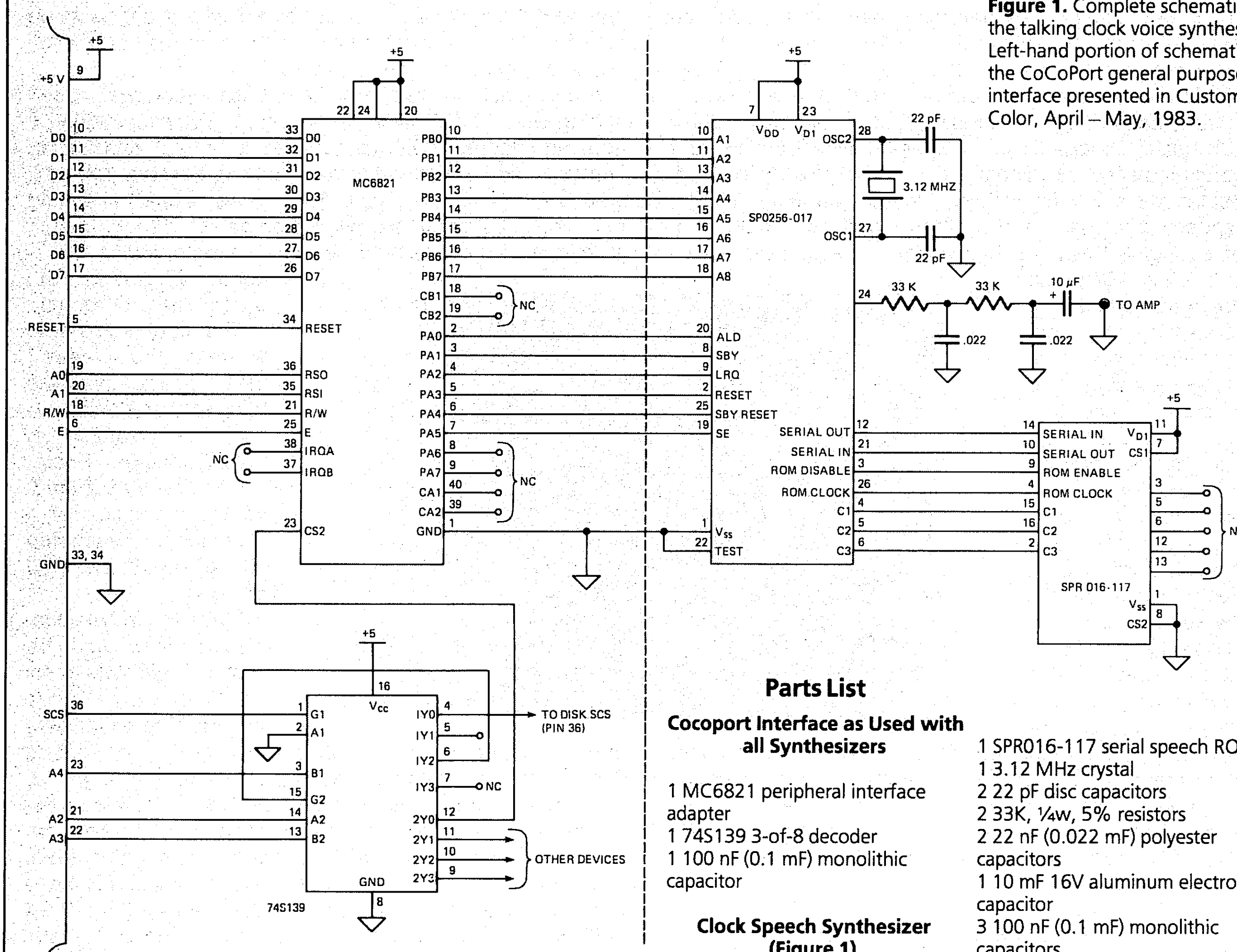


Figure 1. Complete schematic of the talking clock voice synthesizer. Left-hand portion of schematic is the CoCoPort general purpose interface presented in Custom Color, April – May, 1983.

Parts List

Cocoport Interface as Used with all Synthesizers

1 MC6821 peripheral interface adapter
1 74S139 3-of-8 decoder
1 100 nF (0.1 mF) monolithic capacitor

Clock Speech Synthesizer (Figure 1)

1 CoCoPort interface
1 SP0256-017 vocal tract synthesizer
1 SPR016-117 serial speech ROM
1 3.12 MHz crystal
2 22 pF disc capacitors
2 33K, ¼w, 5% resistors
2 22 nF (0.022 mF) polyester capacitors
1 10 mF 16V aluminum electrolyic capacitor
3 100 nF (0.1 mF) monolithic capacitors

(SP0256-017 and SPR016-117 are available from Radio Shack, catalog no. 276-1783.)

us, but it turns out to be a tricky programming task. Consider that from the number 0–20 there are unique written names for each number (zero, one, two . . . eighteen, nineteen, twenty), but after that only every tenth digit (thirty, forty, fifty, etc.) has a unique name. Hours only need the numbers through twelve, but minutes need the numbers through 59.

Furthermore, leading zeroes (such as 01:15) have to be quashed (you'd say one-fifteen, not oh-one-fifteen), although internal zeroes (11:03) have to be said aloud (eleven-oh-three, not eleven-three). For a 12-hour clock a.m. and p.m. need to be indicated, and exact hours (04:00, for example) have to spoken correctly (four a.m. rather than four-oh-oh a.m.). Counting anomalies have to be handled correctly; for example, the hour after 11:15 p.m., though a larger numerical value, is actually 12:15 a.m.

The original time display from last month was military time; I've kept that display. But I've decided that the spoken time is to be in ordinary a.m.-p.m. format, so 12 hours must be subtracted from any displayed hours over 12 to obtain the spoken hours: 13:17 becomes 1:17 p.m.

In other words, all options must be checked and any of several different paths might be followed for the correct expression of different times through the day. It's not difficult to organize—children do it easily—but it is tedious, boring programming.

Program Description

For a run-through, look back at Listing 1. The talking clock subroutine is dormant until triggered, which happens when a full minute rolls over during the software clock interrupt service routine; this roll-over trigger is found at Line 580. (If you like, you change which time roll-over triggers the talking clock. Move the position of LBSR CLOXER from Line 580 to Line 535 for 10 second announcements; to Line 605 for 10 minute speaking; or to Line 635 for one hour timing.) At the roll-over point, the subroutine Cloxer is entered (Line 1080), which saves all registers, sets the clock-talking flag CLKON, and checks to be sure the synthesizer is ready to speak. At this time, the phrase "It is . . ." is triggered and the announcement begins.

The Cloxer subroutine serves a double purpose, so after the "It is . . ." announcement, verbal descriptions get a little hazy. If the clock is talking (that is, sound is being output and the synthesizer's buffer is full), this subroutine returns to the main interrupt service subroutine (the time keeper), finishes its work, and resumes normal computing in Basic. If the clock is not talking or there's still room in the buffer, the program loads the value stored at WHICH and makes an indirect jump to the program steps that properly complete the subroutine. The first time through, at the one minute roll-over, the value at Which is zero, and the phrase "It is . . ." is spoken.

I talked about indirect jumps in my article on the Game of Life (April, 1984), so if you're not familiar with them, review that discussion. In general, the Which value provides an offset to any one of four program segments: routine It is, which speaks the words "It is . . ."; routine Hours, which announces the current hour; routine Minutes, which announces the current minute; and routine Ampm, which speaks the phrase "a.m." or "p.m." Separate routines are needed because the speech synthesizer's buffer isn't large enough to handle an entire time phrase such as "It is eleven forty-five p.m." at once. Naturally, the program can't wait for the speech to be completed, or one of the values of a fast, interrupt-driven subroutine would be sacrificed. A companion need for the individual subroutine segments is that, after having triggered the speech synthesizer to speak each phrase and return to the main program, the routine has to be able to continue where it left off. Otherwise, it might find itself saying "It is . . . it is . . . it is . . . it is . . . it is . . ."

Once the indirect jumb JMP ◄A,Y► is made to one of the four possible program branches from Line 1210, the program is off and running according to the flowchart in Figure 3.

Recall that I mentioned the main subroutine is first triggered by the one-minute roll-over when it enters the service routine Cloxer, and begins the phrase "It is . . . ." Back at Line 950 are three indicators: the Ampm flag for a.m. and p.m., the value Which to identify that the speaking routine is in progress, and the flag CLKON to define whether or not the clock is presently announcing.

Address	Word	Address	Word
0	Oh	18	Eighteen
1	One	19	Nineteen
2	Two	20	Twenty
3	Three	21	Thirty
4	Four	22	Forty
5	Five	23	Fifty
6	Six	24	It Is
7	Seven	25	A.M.
8	Eight	26	P.M.
9	Nine	27	Hour
10	Ten	28	Minute
11	Eleven	29	Hundred Hour
12	Twelve	30	Good Morning
13	Thirteen	31	Attention Please
14	Fourteen	32	Please Hurry
15	Fifteen	33	Melody A
16	Sixteen	34	Melody B
17	Seventeen	35	Melody C

Table 1. Complete list of words spoken by the SP0256-017 speech synthesizer when paired with the SPR016-117 serial speech ROM.

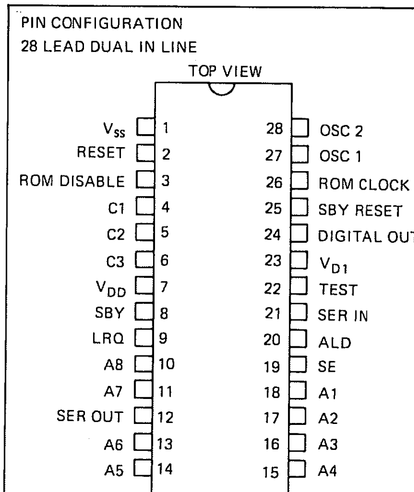


Figure 2. Pin-out of the SP0256 vocal tract synthesizer. The SE line (pin 19) is used with SPR serial speech ROMs (the serial speech ROM pin-out is found in Figure 4).

DENNIS KITSZ

Once the announcing starts, however, the interrupt service routine is subsequently handles by Line 990—before Cloxer—where the routine CLKTST is found. CLKTST evaluates announcement-in-progress flag CLKON every tenth of a second (branching from Line 470). If the synthesizer is not in the midst of announcing, CLKTST merely returns to the main interrupt service routine. If the clock is supposed to be talking—which could only have been triggered by the execution of Cloxer at the one-minute rollover—the CLKTST routine executes Cloxer.

But this time the circumstances are different. Cloxer would already have done some work (the Itis routine) and would be ready to go on to the other speaking roles (hours, minutes, am/pm). By the way, the complete set of words and phrases available to the SP0256-017/SPR16-117 speech synthesis pair is shown in Table 1.

The multiple use of Cloxer is tricky and doesn't lent itself to quick descriptive summaries. Fortunately, it's not necessary to understand my logic in detail unless you want to modify or expand my program. In summary, the interrupt service routine executes CLKTST to check (every 1/10 second) for clock synthesizer announcements in progress. Usually nothing happens, but at the roll-over of one minute, the interrupt service routine executes Cloxer to say "It is . . ." and to spark the clock announcement process into action. The CLKON flag is set. After the announcement is begun and when the synthesizer buffer becomes available, CLKTST continues to execute Cloxer and its four indirectly accessed subroutines until all the synthetic speaking is complete. Finally, the CLKON flag is turned off until the next one minute roll-over triggers the process anew.

Using The Program

Enter the source code in Listing 1 using an editor/assembler. Save the source to tape (W TALKCLOK), assemble the program to tape (A CLOXER), and turn the computer off. Remove the Edtasm cartridge, insert the talking clock board, turn the machine on, protect memory (CLEAR200,&H3E00), load the assembled program from tape (CLOADM"CLOXER), and execute it (EXEC&H3E00). The program will patch into the interrupt routine, the clock will be displayed on the screen, and the clock will begin announcing each minute.

You'll want to set the clock, which will initialize to 00:00:0.0. Load and execute the short Basic time-setting program in

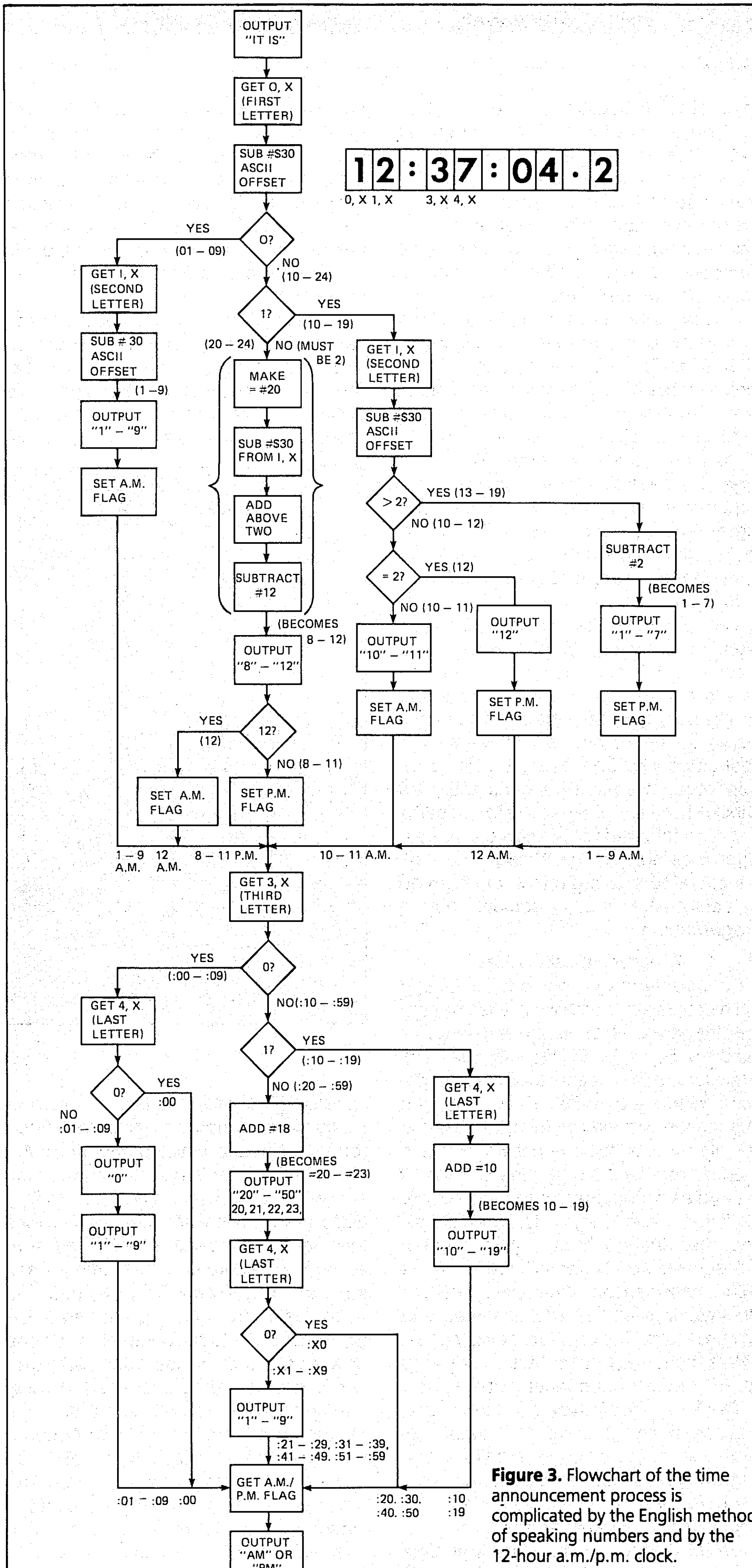


Figure 3. Flowchart of the time announcement process is complicated by the English method of speaking numbers and by the 12-hour a.m./p.m. clock.

Program Listing 1. Complete assembly language listing to maintain a software real-time clock with synthesized speech time.

```
3E00                    ORG     $3E00
                *
          FF52  VOICE   EQU     $FF52
                *
3E00 1A   50    INTOFF  ORCC    #$50     * TURN INTERRUPTS OFF
3E02 8E   3E33          LDX     #START   * POINT X TO SERVICE ROUTINE
3E05 BF   010D          STX     $010D    * STORE ROUTINE TO IRQ VECTOR
3E08 86   37            LDA     #$37     * VALUE 00110111 FOR MASKING
3E0A B7   FF03          STA     $FF03    * TURN ON VERTICAL SYNC
3E0D 8E   FF50          LDX     #VOICE-2        * POINT TO PORT OUPUTS
3E10 4F                 CLRA             * PREPARE TO OPEN PORT
3E11 A7   01            STA     1,X      * OPEN PORT A
3E13 86   39            LDA     #$39     * PREPARE DIRECTION
3E15 A7   84            STA     ,X       * SET SPEECH I/O BITS
3E17 86   04            LDA     #$04     * PREPARE TO CLOSE
3E19 A7   01            STA     1,X      * CLOSE PORT A
3E1B 4F                 CLRA             * PREPARE TO OPEN B
3E1C A7   03            STA     3,X      * OPEN PORT B
3E1E 4A                 DECA             * PREPARE ALL OUTPUTS
3E1F A7   02            STA     2,X      * SET ALL BITS OUTPUT
3E21 86   04            LDA     #$04     * PREPARE TO CLOSE B
3E23 A7   03            STA     3,X      * CLOSE PORT B
3E25 86   20            LDA     #$20     * PREPARE TO RESET
3E27 A7   84            STA     ,X       * RESET SPEECH DEVICE
3E29 86   38            LDA     #$38     * CLEAR SPEECH BUFFER
3E2B A7   84            STA     ,X       * SEND CLEARING PULSE
3E2D 4C                 INCA             * READY FOR RECEIVING
3E2E A7   84            STA     ,X       * SET RECEIVING PULSE
3E30 1C   EF            ANDCC   #$EF     * INTERRUPTS BACK ON
3E32 39                 RTS              * AND BACK TO BASIC "OK"
                *
3E33 8E   3EA9  START   LDX     #IMAGE+10       * POINT X TO 1/10 SEC.
3E36 C6   30            LDB     #$30     * B BECOMES ASCII OFFSET
3E38 6C   84            INC     ,X       * INCREMENT 1/10 SECONDS
3E3A A6   84            LDA     ,X       * GET 1/10 SECONDS VALUE
3E3C 81   36            CMPA    #$36     * IS 6/10 SECONDS COUNTED?
3E3E 2D   3E            BLT     OUT      * IF NOT 6/10 SECONDS, OUT
3E40 8D   6B            BSR     CLKTST   * SEE IF CLOCK TALKING
3E42 8D   4D            BSR     DEC1     * ELSE BAC UP 1 MEM. LOCATION
3E44 81   3A            CMPA    #$3A     * IS IT 1 SECOND YET?
3E46 2D   36            BLT     OUT      * IF NOT 1 SECOND, OUT
3E48 8D   4E            BSR     DEC2     * ELSE BACK UP 2 MEM. LOCNS.
3E4A 81   3A            CMPA    #$3A     * IS IT 10 SECONDS YET?
3E4C 2D   30            BLT     OUT      * IF NOT 10 SECONDS, OUT
3E4E 8D   41            BSR     DEC1     * BACK UP 1 MEM. LOCATION
3E50 81   36            CMPA    #$36     * IS IT 60 SECONDS YET?
3E52 2D   2A            BLT     OUT      * IF NOT 60 SECONDS, OUT
3E54 8D   42            BSR     DEC2     * ELSE BACK UP 2 MEM. LOCNS.
3E56 17   0060          LBSR    CLOXER
3E59 81   3A            CMPA    #$3A     * IS IT 10 MINUTES YET?
3E5B 2D   21            BLT     OUT      * IF NOT 10 MINUTES, OUT
3E5D 8D   32            BSR     DEC1     * ELSE BACK UP 1 MEM. LOCATION
3E5F 81   36            CMPA    #$36     * IS IT 60 MINUTES YET?
3E61 2D   1B            BLT     OUT      * IF NOT 60 MINUTES, OUT
3E63 8D   33            BSR     DEC2     * ELSE BACK UP 2 MEM. LOCNS.
3E65 81   35            CMPA    #$35     * IS IT 5 HOURS?
3E67 26   0D            BNE     NOT24    * IF NOT, TEST FOR 10
3E69 A6   1F            LDA     -1,X     * GET NEXT LOCATION
3E6B 81   32            CMPA    #$32     * IS IT 25 HOURS?
3E6D 26   0F            BNE     OUT      * IF NOT, THEN OUT
3E6F E7   1F            STB     -1,X     * MAKE 05 HOURS
3E71 5C                 INCB             * GET  VALUE "1"
3E72 E7   84            STB     ,X       * MAKE 01 HOURS
3E74 20   08            BRA     OUT      AND BE DONE WITH IT
3E76 81   3A    NOT24   CMPA    #$3A     * IS IT 10 HOURS YET?
3E78 2D   04            BLT     OUT      * IF NOT 10 HOURS, OUT
3E7A E7   84            STB     ,X       * PLACE #$30 (ASCII ZERO)
3E7C 6C   82            INC     ,-X      * BACK UP ONE MEM. LOCATION
                *
3E7E 108E 0416  OUT     LDY     #$0416   * POINT TO RIGHT SCREEN
3E82 8E   3E9F          LDX     #IMAGE   * POINT X TO CLOCK IMAGE
```

Listing 2. You can now use the talking clock with your Basic programs and it will provide a display and audible output for the correct time of day until you turn the machine off, or use program cartridges or assembly language programs which occupy the same memory area as the clock software. 32K and 64K computer users can relocate the clock in other areas of memory as needed.

Note that if you press the Reset button the peripheral interface adaptor on the clock board will be reset as well. Use EXEC&H3E00 to restore the operation of the speech synthesizer.

The SND Input

As shown in the schematic, the output of the speech synthesizer feeds the input of an external amplifier and speaker. It can, however, by made to sound through your television speaker. Connect the output of the speech synthesizer to pin 35 of the Color Computer's edge connector. Then enter the following line:

```
X = &HFF00 : POKE X + 1,PEEK(X + 1) AND 247 : POKE X+3,PEEK(X+3) OR 8 : POKE X+35) OR 8
```

The sound will now feed through to the television speaker. This sound input is useful whenever you want to mix sound from an external device with the sound from the computer. For example, other kinds of speech and sound boards, analog input devices, and so forth, can be fed through the SND input. The Basic Sound and Input/Output commands and the Reset button reset the original sound conditions, though, so you'll have to make this subroutine a part of any programming that is to mix external input through SND.

Next: More time! A battery backed-up real-time clock using a brand new ten-year timer from National Semiconductor, plus an interface for the SP0256-AL2 allophone-based speech synthesizer to give a full range of vocabulary and inflection. I'll wrap this series up with a talking clock with battery back-up. *(end)*

Editor's Note: Dennis has been very busy these last few weeks on an exciting new project he'll call the Data Gatherer. It's a data acquisition and control system with, among other things, 12-bit A/D conversion; 12-bit D/A conversion; a real-time clock calendar with battery back-up; a 10-bit parallel port; and with operating system in ROM. We're going to take a break from the Real-Time Talking clock next month to present Part I of the Data Gatherer. As soon as Dennis gets the excitement out of his system, we'll go back to the Talking Clock!

PIN CONFIGURATION
16 LEAD DUAL IN LINE

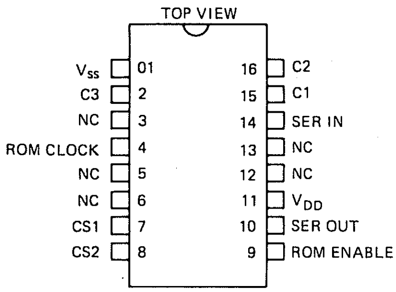


Figure 4. Pin-out of the SPR016 serial speech ROM. Parallel ROM interfacing and internal ROM specifications are not provided by General Instrument.

Hints

Printer—For different printer bauds, POKE 150,x - where x is one of these:

180 for 300 baud
87 for 600 baud
41 for 1200 baud
25 for 1800 baud
23 for 2000 baud
18 for 2400 baud
10 for 3600 baud
7 for 4800 baud
3 for 7200 baud
0 for 9600 baud

For lower bauds, type POKE 149,y:POKE 150,x, with these values:

Y	X	
4	88	for 50 baud
2	227	for 75 baud
1	246	for 110 baud
1	153	for 134.5 baud
1	110	for 150 baud

```
3E85 C6   0A          LDB   #$0A     * COUNT 10 SCREEN POSITIONS
3E87 A6   80    LOOP  LDA   ,X+      * GET CHARACTER FROM CLOCK
3E89 A7   A0          STA   ,Y+      * AND PLACE IT ON THE SCREEN
3E8B 5A               DECB           * DONE WITH IMAGE YET?
3E8C 26   F9          BNE   LOOP     * IF NOT, THEN GET NEXT CHAR.
                      *
3E8E 7E   894C        JMP   $894C    * AND TO BASIC TO DO RTI
                *
3E91 E7   84    DEC1  STB   ,X       * PLACE $30 (ASCII ZERO)
3E93 6C   82          INC   ,-X      * BACK UP ONE MEM. LOCATION
3E95 A6   84          LDA   ,X       * GET VALUE FROM IMAGE
3E97 39               RTS            * BACK TO MAIN PROGRAM
                *
3E98 E7   84    DEC2  STB   ,X       * PLACE $30 (ASCII ZERO)
3E9A 6C   83          INC   ,--X     * BACK UP TWO MEM. LOCATIONS
3E9C A6   84          LDA   ,X       * GET VALUE FROM IMAGE
3E9E 39               RTS            * BACK TO MAIN PROGRAM
                *
3E9F      31    IMAGE FCC   /11:59:59.00/
          31
          3A
          35
          39
          3A
          35
          39
          2E
          30
          30
                *
3EAA      00    AMFLAG FCB  $00      * 00 = AM, 01 = PM
3EAB      00    WHICH FCB   $00      * SAYING WHICH?
3EAC      00    CLKON FCB   $00      * 00 = MUTE, 01 = TALKING
                *
3EAD 34   02    CLKTST PSHS A        * SAVE A REGISTER
3EAF B6   3EAC        LDA   CLKON    * GET TALKING STATUS
3EB2 27   02          BEQ   NOTYET   * IF ZERO, NOT TALKING
3EB4 8D   03    DOTALK BSR  CLOXER   * READY TO SPEAK TIME
3EB6 35   02    NOTYET PULS A        * WHEN DONE RESTORE A
3EB8 39               RTS            * AND BACK TO INT. CLOCK
                *
3EB9 34   36    CLOXER PSHS A,B,X,Y  * SAVE ALL IN SIGHT
3EBB 86   01          LDA   #1       * SET CLOCK IN PROGRESS
3EBD B7   3EAC        STA   CLKON    * AND PUT INTO FLAG
3EC0 B6   FF50        LDA   VOICE-2  * POINT TO CONTROL PORT
3EC3 81   FB          CMPA  #$FB     * AND SEE IF READY
3EC5 27   03          BEQ   OKAY     * IF $FB, IT'S READY
3EC7 35   36          PULS  A,B,X,Y  * ELSE RESTORE EVERYTHING
3EC9 39               RTS            * AND BACK TO INT. CLOCK
                *
3ECA 108E 3ED7  OKAY  LDY   #SPEAK   * POINT TO ROUTINES
3ECE 8E   3E9F        LDX   #IMAGE   * POINT TO CLOCK DIGITS
3ED1 B6   3EAB        LDA   WHICH    * FIND WHICH ROUTINE
3ED4 48               ASLA           * 2-BYTE ADDRESS OFFSET
3ED5 6E   B6          JMP   [A,Y]    * AND GO TO THE ROUTINE
                *
3ED7      3EDF  SPEAK FDB   ITIS     * ROUTINE FOR "IT IS"
3ED9      3EEA        FDB   HOURS    * ROUTINE SAYS HOUR
3EDB      3F32        FDB   MINUTE   * ROUTINE SAYS MINUTE
3EDD      3F66        FDB   AMPM     * ROUTINE SAYS "AM" OR "PM"
                *
3EDF 86   18    ITIS  LDA   #24      * GET VOICE VALUE "IT IS"
3EE1 17   0098        LBSR  TALKER   * AND SPEAK IT
3EE4 7C   3EAB        INC   WHICH    * POINT TO NEXT ROUTINE
3EE7 35   36          PULS  A,B,X,Y  * RESTORE ALL REGISTERS
3EE9 39               RTS            * BACK TO INT. CLOCK
                *
3EEA A6   00    HOURS LDA   0,X      * GET FIRST HOUR DIGIT
3EEC 80   30          SUBA  #$30     * STRIP ASCII OFFSET
3EEE 27   2C          BEQ   HOUR0    * GO IF 00 - 09 HOURS
3EF0 4A               DECA           * DECREMENT TO TEST
3EF1 27   0D          BEQ   HOUR1    * GO IF 10 - 19 HOURS
3EF3 A6   01          LDA   1,X      * ELSE IS 2; GET NEXT
3EF5 80   28          SUBA  #40      * NUMBER JUGGLING *****
3EF7 17   0082        LBSR  TALKER   * AND SPEAK THE VALUE
3EFA 81   0C          CMPA  #12      * SEE IF 12 O'CLOCK
3EFC 27   24          BEQ   SETAM    * IF IT IS, THEN 12 AM
3EFE 20   27          BRA   SETPM    * ELSE THEN IS PM
3F00 A6   01    HOUR1 LDA   1,X      * HOUR IS 1-9; GET NEXT
3F02 80   30          SUBA  #$30     * STRIP ASCII OFFSET
3F04 81   02          CMPA  #2       * CHECK IF 12 O'CLOCK
```

```
3F06 22   0E              BHI    NIGHT    * IF HIGHER, THEN PM
3F08 27   06              BEQ    TWELVE   * ELSE IS EXACTLY 12
3F0A 8B   0A              ADDA   #10      * ELSE JUGGLE FOR VOICE
3F0C 8D   6E              BSR    TALKER   * AND SPEAK THE VALUE
3F0E 20   12              BRA    SETAM    * AND SET THE MORNING
3F10 86   0C       TWELVE LDA    #12      * IF 12 THEN GET IT
3F12 8D   68              BSR    TALKER   * AND SPEAK THE VALUE
3F14 20   11              BRA    SETPM    * AND SET IT TO BE PM
3F16 80   02       NIGHT  SUBA   #2       * ELSE IS NIGHT; JUGGLE
3F18 8D   62              BSR    TALKER   * AND SPEAK THE VALUE
3F1A 20   0B              BRA    SETPM    * AND SET IT AS PM
3F1C A6   01       HOUR0  LDA    1,X      * IF 0-9 HOURS, GET NEXT
3F1E 80   30              SUBA   #$30     * STRIP ASCII OFFSET
3F20 8D   5A              BSR    TALKER   * AND SPEAK THE VALUE 0
3F22 7F   3EAA     SETAM  CLR    AMFLAG   * ROUTINE SETS AM FLAG
3F25 20   05              BRA    GOHOUR   * AND GOES ON OUT
3F27 86   01       SETPM  LDA    #1       * ROUTINE SETS PM FLAG
3F29 B7   3EAA            STA    AMFLAG   * AND PUTS IN PLACE
3F2C 7C   3EAB     GOHOUR INC    WHICH    * POINT TO NEXT ROUTINE
3F2F 35   36              PULS   A,B,X,Y  * RESTORE ALL REGISTERS
3F31 39                   RTS             * BACK TO INT. CLOCK
                   *
3F32 A6   03       MINUTE LDA    3,X      * GET FIRST MINUTE DIGIT
3F34 80   30              SUBA   #$30     * STRIP ASCII OFFSET
3F36 27   19              BEQ    MIN1X0   * IF :00 TO :09, GO
3F38 4A                   DECA            * DECREMENT FOR TEST
3F39 27   0E              BEQ    MIN1X1   * IF :10 TO :19, GO
3F3B 8B   13              ADDA   #19      * ELSE GET VOICE OFFSET
3F3D 8D   3D              BSR    TALKER   * SPEAK :20 :30 :40 :50
3F3F A6   04              LDA    4,X      * GET LAST MINUTE
3F41 80   30              SUBA   #$30     * STRIP ASCII OFFSET
3F43 27   1B              BEQ    GOMIN    * IF :X0, THEN GO
3F45 8D   35              BSR    TALKER   * ELSE SPEAK THE MINUTE
3F47 20   17              BRA    GOMIN    * AND GO ON OUT
3F49 A6   04       MIN1X1 LDA    4,X      * IF :10 TO :19, GET NEXT
3F4B 80   26              SUBA   #38      * JUGGLE FOR VOICE
3F4D 8D   2D              BSR    TALKER   * AND SPEAK THE VALUE
3F4F 20   0F              BRA    GOMIN    * FINALLY GOING OUT
3F51 A6   04       MIN1X0 LDA    4,X      * IF :00 TO :09, GET NEXT
3F53 80   30              SUBA   #$30     * STRIP ASCII OFFSET
3F55 27   09              BEQ    GOMIN    * IF :00, GO OUT SILENTLY
3F57 34   02              PSHS   A        * STASH A VALUE
3F59 4F                   CLRA            * GET READY A ZERO
3F5A 8D   20              BSR    TALKER   * AND MAKE IT SAY "OH"
3F5C 35   02              PULS   A        * RESTORE LAST MINUTE
3F5E 8D   1C              BSR    TALKER   * AND SPEAK THE MINUTE
3F60 7C   3EAB     GOMIN  INC    WHICH    * POINT TO NEXT ROUTINE
3F63 35   36              PULS   A,B,X,Y  * RESTORE ALL REGISTERS
3F65 39                   RTS             * AND BACK TO INT. CLOCK
                   *
3F66 B6   3EAA     AMPM   LDA    AMFLAG   * GET AM-PM FLAG VALUE
3F69 27   04              BEQ    MORN     * IF 0, THEN IT'S AM
3F6B 86   1A              LDA    #26      * IF 1, THEN GET PM VALUE
3F6D 20   02              BRA    GOTIME   * AND GO OUT OF ROUTINE
3F6F 86   19       MORN   LDA    #25      * IF 0, THEN GET AM VALUE
3F71 8D   09       GOTIME BSR    TALKER   * SPEAK "AM" OR "PM"
3F73 7F   3EAB            CLR    WHICH    * CLEAR ROUTINE POINTER
3F76 7F   3EAC            CLR    CLKON    * CLEAR CLOCK ON POINTER
3F79 35   36              PULS   A,B,X,Y  * RESTORE ALL REGISTERS
3F7B 39                   RTS             * BACK TO INT. CLOCK
                   *
3F7C B7   FF52     TALKER STA    VOICE    * STORE VALUE TO SPEAK
3F7F 34   02              PSHS   A        * SAVE THE VALUE
3F81 86   38              LDA    #$38     * GET VALUE FOR LOW PULSE
3F83 B7   FF50            STA    VOICE-2  * PULSE VOICE TO ACCEPT
3F86 4C                   INCA            * GET VALUE FOR HI PULSE
3F87 B7   FF50            STA    VOICE-2  * PULSE VOICE TO READY
3F8A 35   02              PULS   A        * GET VOICE VALUE BACK
3F8C 39                   RTS             * BACK TO TALK ROUTINE
                   *
          3E00            END    INTOFF
00000 TOTAL ERRORS

AMFLAG  3EAA   GOHOUR  3F2C   ITIS    3EDF   NOTYET  3EB6   TWELVE  3F10
AMPM    3F66   GOMIN   3F60   LOOP    3E87   OKAY    3ECA   VOICE   FF52
CLKON   3EAC   GOTIME  3F71   MIN1X0  3F51   OUT     3E7E   WHICH   3EAB
CLKTST  3EAD   HOUR0   3F1C   MIN1X1  3F49   SETAM   3F22
CLOXER  3EB9   HOUR1   3F00   MINUTE  3F32   SETPM   3F27
DEC1    3E91   HOURS   3EEA   MORN    3F6F   SPEAK   3ED7
DEC2    3E98   IMAGE   3E9F   NIGHT   3F16   START   3E33
DOTALK  3EB4   INTOFF  3E00   NOT24   3E76   TALKER  3F7C
```

Program Listing 2.
Basic program to set the time for the program in Listing 1. Once the time is set, this program may be deleted.

```
1 REM * TIME SETTING PROGRAM
2 REM * FOR INTERRUPT-DRIVEN
3 REM * VOICE CLOCK (ONLY).
4 CLS
5 PRINT:PRINT
6 PRINT"ENTER THE TIME IN THE FORMAT:"
7 PRINT"00:00:00.0"
8 PRINT
9 PRINT"NOTE:  USE 24-HOUR TIME."
10 PRINT:PRINT
11 LINEINPUTA$
12 IFLEN(A$)<>10THENRUN
13 FORX=1TO10:A$(X)=MID$(A$,X,1):NEXT
14 IFA$(3)<>":"ORA$(6)<>":"ORA$(9)<>"
."THENRUN
15 Q$=A$(1):GOSUB30
16 Q$=A$(2):GOSUB30
17 Q$=A$(4):GOSUB30
18 Q$=A$(5):GOSUB30
19 Q$=A$(7):GOSUB30
20 Q$=A$(8):GOSUB30
21 Q$=A$(10):GOSUB30
22 FORX=1TO10
23 POKE16030+X,ASC(MID$(A$,X,1))
24 NEXT
25 PRINT:PRINT"TIME SET"
26 FORX=1TO1000:NEXT
27 CLS
28 END
29 STOP
30 IFQ$<"0"ORQ$>"9"THENRUNELSERETURN
```

Program Listing 3.
A Basic clock program to demonstrate the use of the voice synthesizer.

```
10 INPUT"HOUR";H
20 INPUT"MINUTE";M
30 A=&HFF50:B=A+1:C=B+1:D=C+1
40 POKEB,0:POKEA,&H39:POKEB,4
50 POKED,0:POKEC,&HFF:POKED,4
60 T=&H38:U=&H39
70 POKEA,&H20
80 TIMER=0
90 IFTIMER>55THEN100ELSE90
100 S=S+1:IFS=60THENS=0:M=M+1:GO
SUB120:IFM=60THENM=0:H=H+1:IFH=1
3THENH=1
110 GOTO80
120 POKEC,24
130 GOSUB200
140 POKEC,H:GOSUB200
150 IFM<10THENPOKEC,0:GOSUB200
160 IFM<21THENPOKEC,M:GOSUB200:G
OTO180
170 IFM>20THENPOKEC,18+INT(M/10)
:GOSUB200:IFM-(INT(M/10))*10=0TH
EN180ELSEPOKEC,M-(INT(M/10))*10:
GOSUB200
180 POKEC,26:GOSUB200
190 RETURN
200 POKEA,T:POKEA,U
210 IFPEEK(A)=253THEN210ELSERETU
RN
```

EPROM Erasers

By Martin Goodman, M.D.

For other than industrial use, probably the best bet in UV erasers is to buy a bottom of the line commercial EPROM eraser. Such items, which can erase one to three EPROMs in 10 to 20 minutes, are available from a number of suppliers for about $40. If you are a do-it-yourselfer, you can make your own out of a light-tight box (I used a cheap tool box) and a short-wave UV (germicidal) light bulb. I used a GE G15T8 bulb (18 inches long). Whatever bulb you use, it *must* have a rated max output wavelength around 2537 Angstrom units. Long-wave UV bulbs (the kind used for sun-tanning systems) will not work well at all. Do *not* use suntan UV bulbs to try to erase EPROMs! They will erase the EPROM eventually, but will take hours. The box I made ended up costing me about $80 total in parts (including an auto-resetting timer I got for it), and is a bit bulky. On the other hand, it is capable of erasing over 60 EPROMs at once.

Once you have an EPROM eraser, it is *essential* that you calibrate it for the particular EPROM type(s) you are going to erase. After extensively reviewing the literature on this subject, here is my advice: Take an EPROM and program it. Then put it in your eraser and expose it for 30 second intervals. After each exposure, check to see if it's fully erased. When you have determined the minimal exposure time needed to erase the EPROM, multiply that time by five. That is the amount of time needed to reliably and properly erase that brand of EPROM. Different EPROM types and different brands will have different erasure times, varying by up to a factor of four. Using this guideline you can be assured of complete and proper erasure. And you never have to worry about complicated equations involving "nominal erasing energy" in watt sec/cm squared!

Why a light-tight box? UV radiation is extremely damaging to the retina. Even the weaker tanning UV lights can, with long exposure, blind an unwary sun worshipper. The short-wave UV used for EPROM erasing is much more dangerous than tanning UV light. If you wear *glass* spectacles, your eyes are fully protected, as glass is opaque to short-wave UV. Plastic and quartz can't block short-wave UV. Even if you do wear glass lenses, you must make your EPROM eraser light-tight to protect others who may wander near it.

A brief note on tanning: I, like you, burn rather than tan. Actually, that is a blessing, as current medical knowledge strongly indicates that sun bathing and tanning is quite unhealthy. UV exposure of skin predisposes it to several forms of skin cancer and always results in skin aging much more rapidly.

Added Note: If the time to apparent erasure is less than one minute, it means your EPROM is too close the UV bulb and is getting too intense UV. Just arrange your eraser so the EPROM is further away from the bulb, and try again.

Editor's Note: CompuServe users will recognize this article from The Color Sig. Our thanks to Marty for letting us print it! (end)

MONITOR AUDIO

By Mark Haverstock

Do you have a silent monitor? The monitor you own may have a great picture, but probably no sound capabilities. Until a few days ago our 13 inch color tv was a fixture on an already cluttered computer table, providing audio for game and educational programs. Fortunately, there is an easier and smaller solution.

Let's look at two sets of plans to make your own audio amplifier. The first is for the electronic hobbyist who enjoys building circuits. The second is a "quick and dirty" method for the person who has a passing acquaintance with the soldering iron and doesn't care to spend much time on projects. Both can be put together for $10 or less, depending on the state of your parts bin.

All the components are common, and could be purchased at most electronics stores. (Radio Shack part numbers are included for your convenience.) The projects were designed with the Video Plus monitor driver in mind; other drivers may require different audio cables.

Project 1—Integrated Circuit Amplifier

This amplifier is built around the LM386 integrated circuit (IC). It's an expensive, low voltage audio amplifier that requires a minimum of external parts. Power requirements are not critical; it can be powered by voltages ranging from 4–12 volts. This makes it an ideal candidate to use existing power supplies inside the Color Computer or a monitor.

Here's the parts list:

		RS part number
R1	47 kohm resistor	271–042
R2	10K potentiometer switch	271–215
C1	4.7 uF electrolytic capacitor	272–1012
C2	10 uF electrolytic capacitor	272–1013
C3	47 uF electrolytic capacitor	272–1015
C4	220 uF electrolytic capacitor	272–1017
IC1	LM 386 audio amplifier	276–1731
	IC board	276–024
	Speaker 3 inch	40-248
	Audio cable	42-2370
	Battery clip	270-325
Optional		
	Plastic project case	270–222

Tools Needed: Soldering iron, needlenose pliers, wire cutters, screwdriver, and drill (if project case is used).

Assembly

The IC board was chosen for its small size and large solder pads. Note that there is space for a 16 pin IC; since the LM 386 only has eight pins, I decided to center it on the board to leave free space to connect other components. Following the schematic diagram in Figure 1, mount and solder the components together. Leave the volume control and speaker off the board, as shown in Photo 1.

Parts placement is not critical, but all the ground points on the schematic should be soldered together at one location to prevent a ground loop. This will discourage feedback and reception of local radio stations. R_1 acts as an attenuator to limit the maximum amount of volume. It may be removed if increased volume is desired.

The next steps depend on how and where you plan to mount your amplifier. The finished unit was designed to be placed inside a project case and operated from a 9V batter. This unit could be placed inside the case of the Color Computer, under the keyboard, or in the housing of the monitor.

Listed below are locations in the Color Computer and Color Computer 2 where supply voltages can be obtained. As for the monitor, you'll need a schematic for your particular monitor to determine where supply voltages between four and 12 volts can be found.

Color Computer		Color Computer 2	
TP9	+12V	U1 pin 3	+5V
TP12	+5V	R2	+5V
R59	+9V		

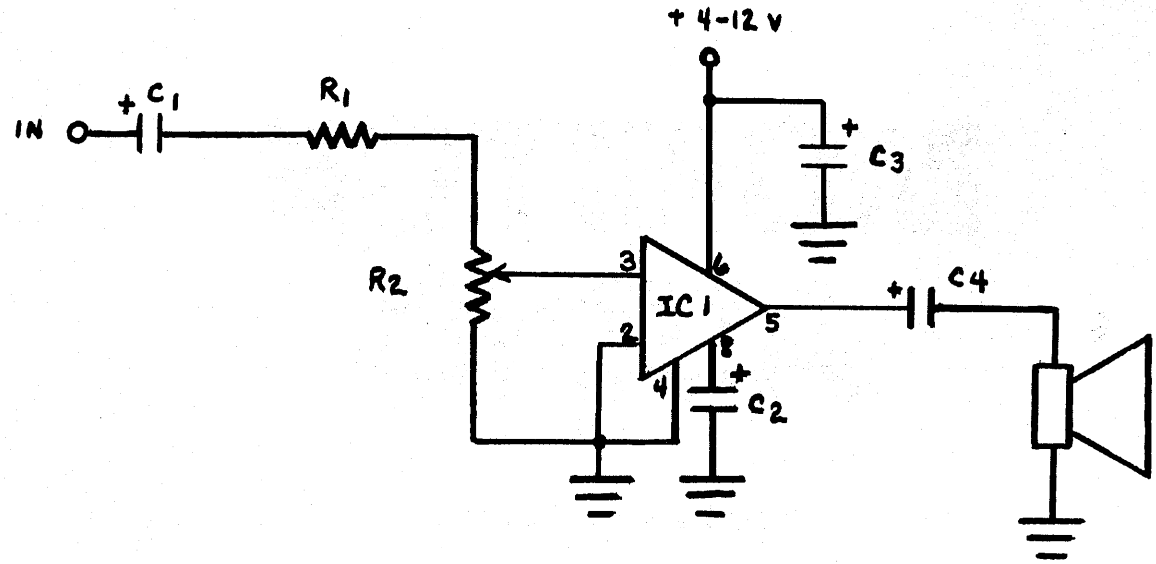


Figure 1. Schematic For Project 1

The Quick and Dirty Solution

If the first project didn't inspire you, maybe this one will. This amplifier is based on a transistor radio. A transistor radio? Well, think for a minute: A transistor radio is basically a tuner and an amplifier in a small plastic case. The trick is to disconnect the tuner so you'll hear Pac-Man munching instead of Boy George singing. Fortunately, this is a fairly simple task and requires only a few simple tools, a capacitor, and some wire.

Here's the parts list:

		RS part number
C1	.47 uF capacitor	272–1433 or 272–112
	Transistor radio	12–166
	Audio cable	40–2370

Tools needed: small phillips screwdriver, soldering iron, PVC tape, and wire cutters.

Procedure

Actually, any transistor radio would work, as long as the audio section is in good shape. I chose the Flavor Radio because it was the only transistor radio available in our house. The directions and photographs are for this radio, but the procedure is basically the same for others. The schematic is shown in Figure 2.

First remove the rear portion of the case to expose the innards of the radio. Be sure to remove the battery if there is one inside. Look at the circuit board: there are two screws holding this board to the plastic case (on the left and right center). Remove these carefully, using the phillips screwdriver. Lift the board carefully from the case and turn over. Don't pull too hard—the speaker wires are still attached.

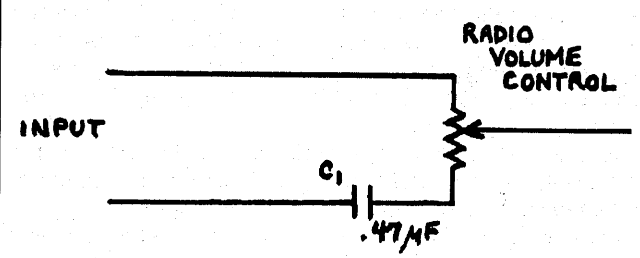


Figure 2. Schematic For Project 2

Next, locate the volume control opposite the tuning knob (the one without numbers). Remove the volume control knob with the phillips screwdriver; this will expose the volume control. After this we'll need to do some trace cutting and wire-tapping.

No, we aren't going to do any subversive or covert activities here! We'll disconnect the tuner from the amplifier and provide a means to connect the finished product to the computer.

The volume control has three leads coming from it which are soldered to the board. Orient the radio as shown in Figure 3. We'll number the leads clockwise from top to bottom—1,

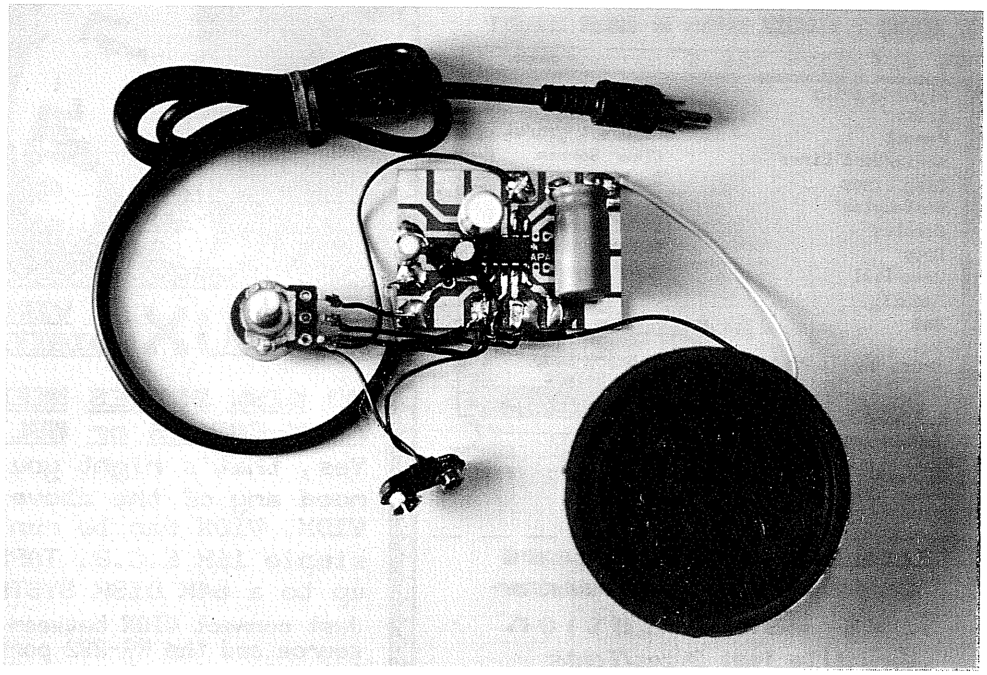


Photo 1. Component Arrangement

2, and 3. The center lead will not be used. Use a single-edge razor blade or small hobby knife to cut the uppermost circuit trace, which is connected to the volume control at **1**. Be careful not to cut adjoining traces, and to cut through completely until the plastic circuit board shows.

The next step is to prepare the wire for soldering. Solder the .47 uF capacitor to the inner conductor of the audio cable. This acts as a coupling capacitor between the computer's audio output and the amplifier in the radio. Solder the other end of the capacitor to lead **1**. The shield will be soldered to lead **3**. After soldering, wrap all exposed leads in PVC tape to prevent shorting.

Before assembling, test the unit. Install a battery and attach the cable to the computer. Turn on the radio. Load a program that generates sound, or test using this short program:

```
10 FOR X=1 TO 255
20 SOUND X,1:NEXT X
```

If you don't get sound, check for incorrect wiring, loose connections or cold solder joints. Most radios I've seen follow the same wiring pattern, but some have **1** and **3** reversed.

Your newly built amplifier could be left in its plastic radio case or removed and placed inside the monitor or computer; however, a suitable 9V supply will have to be furnished.

Now the game players in your family can enjoy sound, and the the tv can be put to other uses—maybe with a second Color Computer?

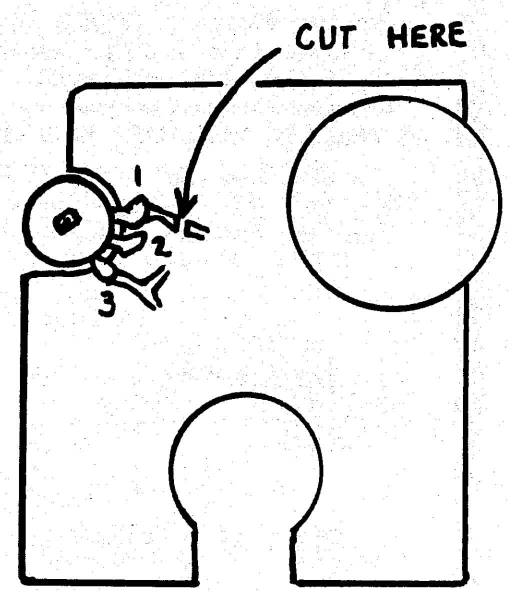


Figure 3. Lead Orientation

Off Color:
A Photo Essay

One of many useful ways
computers have really entered our lives . . .

Comparison Review: Video Digitizers

By Martin Goodman

GRAFX Digitizer	Digisector 69	Computize Digitizer
GRAFX	The Microworks	Computize Inc.
POB 254	POB 1110-A	POB 207
West Mifflin, PA 15122	Del Mar, CA 92014	Langhorne, PA 19047
$99.95	$149.95	$199.95

Recently I had the opportunity to compare the three video digitizers that are or will be available for the Color Computer.

The Grafx digitizer is the least expensive of the three. It communicates with the Color Computer via the RS-232 port, taking about three seconds to digitize a single frame. The subject must be stationary during this time. I've seen two disks of pictures made with this digitizer, and am impressed with its ability to digitize very contrasty objects, such as block letters and cartoons. When using it on a face, one runs into the problem of the long exposure needed to produce a picture, and also the problem that its ability to show gray levels is somewhat limited. Improvements in software may help this problem somewhat. I would recommend the Grafx digitizer for users who want a low cost way to transfer large letters or black and white drawings into the computer. It is the least suitable of three products for use with live subjects. To effectively use it, you really do need a video monitor dedicated to monitoring the video signal while you digitize. I'm told the device will be supplied coated in epoxy, a method meant to deter hardware pirates. The support software may also be supplied with John Yurek's easy to crack but difficult to clone "fingerprint" (scratch on the disk) protection scheme.

Digisector 69 plugs into the expansion port or the Multipak Interface. It will probably work on a PBJ multiport as well. This unit will be provided with rather sophisticated software, with more support software to follow. One program enables it to render five levels of gray on the screen, while digitizing two frames per second (0.5 seconds per frame). This allows rather pleasing images of faces to be made. With a second support program it can produce up to 16 levels of gray on hard copy, but at a great price: the image takes eight seconds to scan. (Editor's Note: Since hearing Dr. Goodman's comments, the Microworks has improved the software so that this mode will scan in two seconds. Please consider this when reading the rest of this review.) I saw a few pictures of faces made using this eight second scan mode, and was stunned by their clarity. They were unquestionably the finest digitized pictures I've ever seen generated by the Color Computer; in some respects superior to certain digitizations I've seen on the Apple and IBM pc. Remember, though, to get such gray scale production an eight second exposure is needed!

In the two frame per second mode, pictures were moderately superior to those of faces done with the Grafx product, and roughly the same in quality as those produced with the Computize product below. Digisector 69 is the best choice for users who demand the highest possible number of gray levels. It is the best choice for transferring still photographic images with many gray levels into memory. Its two frame per second mode is acceptable for live subjects, though a little sluggish. For the most part, a monitor to view the video signal while you are digitizing it needed to most effectively use this device. The current software support allows sophisticated adjustment of both vertical and horizontal contrast, and features a clever user interface. Further support software may later be released that will allow some degree of image processing and doing logical pixel operations on pictures (reverse, AND, XOR, etc.). The price of this digitizer is unlikely to drop in the forseeable future. It will be supplied on a well layed-out printed circuit board in a black case. The pins will not be gold plated (they probably don't need to be). Accompanying software will probably not be copy protected, in accordance with Micro Works long-standing policy. Micro Works may not make a schematic of its unit available for awhile, however. The unit uses a PIA and seven small-scale logic chips. An A to D resistor ladder under software control is the heart of its fine handling of gray levels.

The Computize digitizer is an improved version of the video digitizer designed by one of the authors of Graphicom. This unit is a true frame grabber, capable of digitizing a single video frame (1/30th of a second) at a time. Unfortunately, the digitized data must be transferred from an on-board RAM to the computer's internal memory. This transfer takes time. Therefore, while the device does digitize a single video frame at a time, it is capable of *displaying* only every sixth frame. This comes to five frames a second (0.2 seconds per frame). It is still by far the fastest of the three digitizers. When its controls are properly set, it can show up to five gray levels. Its ability to render faces is quite good . . . the equal of the two frame per second Micro Works product, and slightly superior to the Grafx product.

The strongest point of the Computize digitizer is that it lets the user view a moving subject (such as a TV show) on the digitized screen in something like real time. For this reason it should be preferred by video tape addicts. Although it is the most expensive of the three video digitizers its great speed lets you use it effectively without having a monitor dedicated to viewing the video image. This may, for some applications, offset its high cost. This product is the most electronically complex of the three, using an 8K by 8 static RAM and 15 smaller IC s. Although I have not seen its final packaging, I have been assured that it is professionally made. The software interface for this product is through existing copies of Graphicom (version 1.2). All copies of Graphicom sold through all distributors had code for operating this video digitizer buried in them. The Computize product will be sold with instructions on how to activate this "undocumented" code.

This digitizer plugs into the expansion port and will work with the computer alone or with the Tandy Multipak. It will probably also work with the PBJ Multiport device. *(end)*

DYNACALC®

CoCo's Best & Fastest Spreadsheet System

ACCLAIMED BY THE EXPERTS

"DYNACALC is my choice for a CoCo spreadsheet."
Dan Downard, RAINBOW, September, 1984.

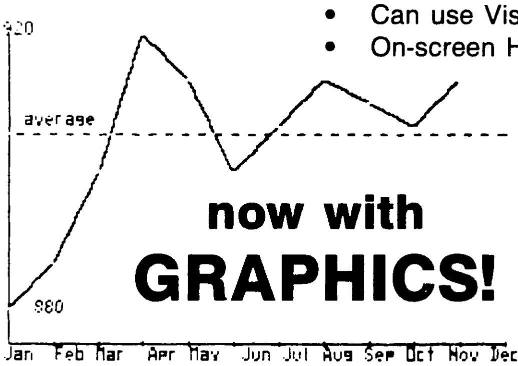
"Eat your heart out, Lotus 1-2-3!"
Scott Norman, HOT CoCo, October, 1984.

NOW ONLY $99.95

Built-in Features:

- 51 x 24 Display with Lower Case
- Super-fast Smart Screen Refresh
- Auto-Repeat Keyboard Driver
- Keysaver (TM) repeats last command x times
- Disk Operating System (works just like ROM DOS)
- Easy communication with BASIC/DOS programs
- Two-way communications with PRO-COLOR-FILE ★Enhanced★
- Outputs to ASCII Word Processors like Telewriter-64
- Fast 16-Digit Arithmetic with Scientific Functions
- Summation, Mean, and Standard Deviation Functions
- Logical Functions with String & Numeric Comparison
- String locate command to navigate large worksheets
- Sort full or partial worksheet by columns or rows
- Line, Bar, Hi/Lo/Close, Circle Graphs
- Full Graphics captioning and overlay facility
- Graphics Drivers for all popular Printers
- Joystick/Mouse Driver for Cursor Movement
- Works with any ROM versions, even JDOS
- 33k Available Worksheet Space
- Up to 256 Columns or 256 Rows
- Can use VisiCalc worksheets & training materials
- On-screen Help Messages

RAINBOW
CERTIFICATION SEAL



now with GRAPHICS!

FOR 64K DISK SYSTEMS
available from

COMPUTER SYSTEMS CENTER
NEW ADDRESS
42 Four Seasons Center #122
Chesterfield, MO 63017
(314) 576-5020
10 to 6 Mon.-Fri.

or your local DYNACALC dealer

Price US$99.95
Outside North America add $10 postage

CANADA- CDN $129.95
RGS MICRO INC.
759, VICTORIA SQUARE 405
MONTREAL H2Y 2J3
TEL.: (514) 287-1563
ORDER LINE ONLY ★ ★ ★
QUEBEC-ONTARIO-MARITIMES
800-361-5338
WESTERN CANADA 800-361-5155

MasterCard
VISA