

COOKING With CoCo



PART I

In which we gather together the ingredients and utensils and explore the possibilities of CoCo's Disk Operating System.

By Colin J. Stearman

I know I do not need to tell you that CoCo is a powerful computer. You have probably spent as much time as I arguing its merits over those "fruity" and "big blue" machines. So while we are in agreement thus far, you'll surely also agree that even the "best" can be improved.

In this series of articles over the next few months we will explore how to incorporate many improvements, some of which are often only found on systems costing 10 times as much. I hasten to add that these improvements will be completely incorporated into the operating system and will be there when you want them. There have been other articles detailing enhancements, but they always involve loading programs into memory and they never seem to be there when you need them. Not so with the enhancements we are going to cook up here!

What exactly are we going to enhance and what is it going to take to do it? These articles are aimed at the standard 32K Disk CoCo system running with version 1.1 of Color BASIC, 1.0 of Extended Color BASIC and 1.0 of Disk Extended Color BASIC. The earlier 1.0 version of Color BASIC will probably work also, but the 1.1 version of Disk BASIC will not without modifying the programs presented here.

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

Although I will give you every assistance, it is going to take some skill on your part. Even the best written recipe requires the cook to add his skill. Some of the enhancements require hardware construction and some electronic construction skills. Others will involve the assembly of machine language programs. But none of it is really difficult and if you have a 64K system you can have almost all of the enhancements without even lifting a screwdriver.

Required Utensils

If you're going to attempt the hardware projects, you will need the normal set of screwdrivers, pliers, cutters and a soldering iron. If you are about to embark on a "hardware hacking" career, your local Radio Shack can accommodate you.

The assembly language programs will require an assembler. *EDTASM+* will do the job, but I much prefer *MAC* from Computerware. This is what I use and I will attempt to point out the differences when necessary. For typing in the source code, a good editor is a must.

The Glossy Photo

Every good cookery book has glossy photos of the finished dish to whet your appetite. Our photo is by way of a list of the more tasty features:

- *FAST and SLOW to control CoCo's clock speed
- *XEQM to load and execute a machine code program
- *DATES\$ to return a string containing the date
- *Directory pause
- *File creation date in the directory
- *Confirmation of the Kill request
- *WPEEK/WPOKE 16-bit word PEEK and POKE
- *Error trapping in BASIC programs
- *Error code, error line and error name functions
- *Auto execution of a BASIC file on start-up
- *AUTO line numbering, with start line and increment
- *Flexible keyboard entry (FLEXIKEY)
- *Fully spelled-out error messages
- *SCAN\$, "INKEY\$" with built-in wait for key press
- *40-track versions of *DSKINI*, *BACKUP* and *DSKIS/DSKOS*
- *Fixes to the *FILES* and *PCLEAR* bugs
- *Up to *PCLEAR 16* allowed
- *BAUD command to set Baud rate
- *Parallel printer port
- *LDIR to send the directory to the printer
- *And more . . .

By now your mouth should be thoroughly watering so let's start cooking!

Underlying Principles

When Microsoft wrote the BASIC operating system for Radio Shack they planned ahead and left numerous "hooks" in the code to allow modifications and changes. These hooks take the form of jump instructions located in the lower RAM (random access memory) area of the map. Many of the useful subroutines in BASIC first jump to these hooks, making it very easy to intercept their function and modify or completely change.

Fortunately for us, when Microsoft was contracted to write Disk Extended Color BASIC (DECB), something odd happened. Color BASIC (CB) and Extended Color BASIC (ECB) fully occupied their 8K ROMs (read only memory). But DECB did not come close to filling the 8K. In fact, some 2000 or so bytes were unused. Maybe money or time ran out, but this available space can be put to good use for all those

added functions mentioned earlier. The only requirement is to come up with a way to permanently insert the new instructions in this available place.

There are two ways to do this. We can either replace the ROM with a similar EPROM (Eraseable Programmable ROM) containing our additional code, or we can make use of the 64K RAM capability of our CoCo (if we have it). The EPROM approach requires the design of an EPROM programmer and that will be the subject for next month. But the 64K method requires no hardware and does nearly as good a job, so for the remainder of this installment I'll detail what I mean.

Disk Resident DOS

If you have installed 64K memory chips and the now famous "Frank Hogg Modification," you know that CoCo can run in an "all RAM" mode in which the three BASIC ROMs play no part. Using this technique it is possible to store the entire BASIC operating system on a specially prepared disk and then boot it into the all RAM system and start it up. In fact, for many computers (the IBM PC, for example) this is the only way of loading the DOS (disk operating system) and is the normal procedure for getting things started at turn on.

If we give CoCo the ability to boot or load its DOS from disk, there is nothing to say that we cannot modify the contents as we desire. As a result we can have the original DOS in the internal ROMs and our enhanced DOS on a special "system disk."

I said this approach is nearly as good as "burning" EPROMs with the modified code. There are some limitations. If you press the Reset button while running the disk-resident DOS, you will be summarily returned to the ROM version. Also, if you run some application programs which make use of CoCo's 64K capabilities, you will probably be returned to the ROM DOS when you exit them. But disk-resident DOS (let's call it DRDOS) can be booted and running in about 10 seconds, so this is not much of a penalty. Further, there are not just 2,000 or so bytes available for enhancements, but using all the address space from \$D7DD to \$FFEF, there are some 10,000 bytes. This is because we are not limited to the 8K increments and socket space of the ROMs.

Two machine code programs are needed here — one to get DRDOS saved onto disk and the other to retrieve it and start it up. The first I called *SYSSAVE* and the second *SYSTEM*. As a result, the currently running DOS, modified as desired, can be saved to disk by *SYSSAVE* and recovered by *SYSTEM*.

Running BASIC In RAM

BASIC cannot run in a 64K RAM environment without two slight modifications. When it goes through its start-up procedure it switches back to the regular RAM/ROM configuration and we would rather it did not. Second, it goes through a sizing procedure to find out exactly how much RAM is available (remember the days of 4K and 16K?). This testing plays havoc in the 64K RAM mode and must be removed. We already know that BASIC has 32K to work with, so we can skip the testing and report this number immediately. This savings in bytes provides just the room we need to fix the first problem.

So, the first thing we must do is copy an image of BASIC from the three ROMs into the RAM, slightly modify it and then start it up. This is done by a program called *BAS-LOAD*, shown in Listing 1. This is entirely a BASIC program.

but it does load a simple machine code routine and the source for this I have included as REM statements at the end. The program is singularly anticlimatic! After a few seconds a tone sounds and the start-up credits are issued. Nothing seems to have changed. But, in fact, you are in a 64K RAM environment. Don't believe me? Try `POKE&HE000,55` and then `PRINT PEEK (&HE000)`. You'll get the 55 back because RAM is at \$E000. In the ROM system you will POKE to no avail.

By the way, I'll be using the assembler convention throughout these articles which says that a "\$" in front of a number says it's hexadecimal; a "%" means binary and nothing in front means it's decimal. But in BASIC statements I will use "\$H" to signify hexadecimal.

Type in the program in Listing 1, save it to a convenient disk and then run it. If you get the tone and new credits everything ran fine and we're ready to save the slightly modified system to a disk. To make absolutely sure your RAM version is okay, type `POKE113,0:EXEC$HA027`. This will do a cold start of the BASIC in RAM and should clear the screen and display the credits. After you're sure it works, get back to the ROM version by typing `POKE113,0` and then pressing Reset. I'll hang around here till you get back!

Saving DOS To Disk

The currently running operating system is saved to disk with a program called `SYSSAVE.BIN`. Once the assembler has created the binary file it is just run by the `LOADM` and `EXEC` commands.

`SYSSAVE` will request which drive (0 or 1) you wish to save the system to. This drive should contain a blank, formatted disk. The program will then save the contents of memory from \$8000 where ECB starts, up to \$FF00. This is one more than the highest useable memory. From here to \$FFFF are system addresses and vectors. It does not matter whether you have anything in the saved range, it just stores what is there on the disk. As DECB starts at \$C000 we could extend it up to \$FFEF and be able to run the system in RAM. (That would be a lot of capability, as all the enhancements I listed earlier will fit into the 8K space allotted to the DECB ROM from \$C000 to \$DFFF.

The bytes are stored on disk on tracks 0 through 6, plus the first sector of track 7. This means that granules 0 through 14 are used and unavailable to BASIC. The granule map on track 17 sector 2 is modified to reflect this. Therefore, once a system has been saved to a new disk, the `FREE` function will return a value of 53, even though the directory shows no files.

Sector 1 on track 17 is not used by BASIC, so the first byte is set to \$55 to indicate that this is a system disk. When `SYSSAVE` is run it first checks that this \$55 is there. If it is, then a system can safely be stored on the disk. If not, then this disk has never had a system saved on it before. In this case, `SYSSAVE` checks that the first 15 granules are free. If so, the system can be saved. If not, a "DISK NOT FREE FOR SYSTEM STORAGE" message is returned and `SYSSAVE` gives up. As a result it should not be possible for `SYSSAVE` to overwrite valuable data on a disk.

To run `SYSSAVE` it must first be entered as shown in Listing 2 and then assembled. If you're using `EDTASM+`, leave out the lines with mnemonics "NAM" and "OPT" in them; these are just directives to my `MAC` assembler. This will be true for all future assembly language programs. Another mnemonic `MAC` has which must be converted for

`EDTASM+` is the FCS instruction. This forms a constant string and allows embedded hexadecimal control codes and automatically adds a terminating zero byte. So the line in `SYSSAVE` which I have as:

```
FCS / <0D>DRIVE NUMBER (0 OR 1) ? /
```

would become:

```
FCB $0D A CARRIAGE RETURN
```

```
FCC /DRIVE NUMBER (0 OR 1) ? /
```

```
FCB 0 TERMINATING ZERO
```

You can convert all other FCS instructions into these groupings and `EDTASM+` will like them just fine.

When the code assembles correctly and you have checked it carefully, the only thing left is to try and run it! The best technique is to first load and run `BASLOAD`. This gets the system running in RAM and suitably modified for this environment. Now `LOADM"SYSSAVE"` but don't execute

"If you're going to attempt the hardware projects, you will need the normal set of screwdrivers, pliers, cutters and a soldering iron. If you are about to embark on a 'hardware hacking' career, your local Radio Shack can accommodate you."

it yet. Then remove all important disks from all your drives as chaos may be about to reign. Load a blank, formatted disk in drive 0 and type in `EXEC`.

The screen will clear and a request will appear asking which drive to save to. Enter a zero. Drive 0 should whir for a few moments and the OK prompt appear. If not, it's back to the editor and look for that inevitable typo!

The system has now been saved to disk. A couple of checks will help confirm this. Type in `PRINT FREE(0)` and a value of 53 should be returned. Another check would be to type the following commands:

```
CL:EAR 500
```

```
DSK1$ 0,17,1,A,$,B$
```

```
PRINT HEX$(ASC(A$))
```

This last line should print the value 55. But the ultimate test is to try to retrieve and run the saved system.

Booting From Disk

If you study the code of `SYSTEM` you will find it very similar to `SYSSAVE`, and it is hardly surprising. Type in and assemble the program in Listing 3. When you've thoroughly checked it for typing errors and are certain it is right, put a write-protect tab on your system disk anyway. Then when the impossible happens, your saved system won't be erased.

Now `LOADM` the binary file called `SYSTEM`, remove the disk and place the system disk in drive 0. `SYSTEM`

always boots from drive 0. Then *EXEC* the program. Once again the screen will clear and a message will announce what is happening. Drive 0 will run and you will hear the head moving. When it is finished you will be requested to input which "flavor" of BASIC you want, CB, ECB or DECB. Pressing the appropriate letter will cold start that version. This feature is a convenient way of disabling DECB should you want to run one of the other configurations.

If everything worked as expected, you can copy the *SYS-SAVE* and *SYSTEM* source and machine code files to your system disk. Then they will all be in the right place. I also wrote a simple BASIC program to start up *SYSTEM* which you might want to include. Then, if you call it BASIC you can just type *RUN BASIC*. It is:

```
10 DISK OPERATING SYSTEM LOADER
20 LOADM"SYSTEM"
30 EXEC
```

To remove the system from a system disk and make the full 68 granules available, the simplest way is to reformat with the *DSK/NI* command. Don't have anything else valuable on the disk though, as it will be erased.

Wrapping It Up

You now have the first tool to be used later in the DOS enhancements. When these have been installed and saved to a system disk, they can be booted at power-up and all the features will be there without absorbing any RAM space. Even if you intend taking the EPROM route, it's still a good idea to have these programs as it makes testing quicker and easier.

Which brings me to next month. Putting the enhanced version of the DOS in an EPROM is certainly a nice way to go. Then everything is there just as soon as the power is turned on. So, next month we will start the EPROM programmer. This is a very simple hardware project using only three chips! Most of the work is done by the software. So, if you've ever had a soldering iron in your hand, give it a try.

Throughout this series I will be happy to try to answer related questions which might arise. Please address them to me at 143 Ash Street, Hopkinton, Mass. 01748 and enclose a S.A.S.E. Be as precise as you can and give me a few weeks to get back to you. You can also send me EMAIL on CompuServe to 71036,256.

See you next month!

Listing 1

```
SYSSAVE          COMPUTERWARE MACRO ASSEMBLER
COOKING WITH COCO PART 1/LISTING 2 (C)1984 COLIN J. STEARMAN
```

```
0004      OPT  NOS,LIS
0005 *
0006 * THIS LOADS BASIC FROM 00000
0007 * UP TO $FF00 ONTO A BLANK
0008 * FORMATTED DISK. IT USES
0009 * THE FIRST 15 GRANULES.
0010 * 14 gran * 9 sectors * 256 bytes = 32256
0011 * plus
0012 * 1 sector = 32512 byte which cover from
0013 * $0000 to $FF00. All of accessible upper
0014 * memory
0015 *
0016 *****
0017 *SOME EQUATES
```

RETIRE EARLY? WHY NOT!

**HOW? PRACTICE THRIFT AND
PLAN WISELY. THE THRIFT IS UP TO
YOU, BUT FOR PLANNING . . .**

YOU NEED THE RETIREMENT PLANNING MODEL



ABOUT RETIREMENT PLANNING

By the year 2010, today's \$8800 auto will cost \$40,000 if inflation averages 6%. Inflation makes retirement planning essential. Proper retirement planning requires a complex year-by-year analysis which must consider these factors:

- * Your investment program
- * Tax-deferred savings
- * Social Security
- * Inflation
- * Pension
- * Taxes

START NOW

Start your planning now. Try different retirement ages and vary your investment program goals. The objective is to develop a plan for early retirement which eases doubt regarding your future financial security.

WHAT THE MODEL DOES

First, the model helps you organize your present assets. The model then projects these assets, along with estimated pension and social security, to the retirement age you select. Based on this projection, a detailed cash flow analysis is conducted for each year of your retirement.

The factors listed above are considered in all calculations. Each analysis stops when your funds deplete or when the analysis carries to the age of 100. The model is designed for "what if" analysis and optional printer output.

AN ESSENTIAL TOOL FOR COMPREHENSIVE
RETIREMENT PLANNING

FULLY DOCUMENTED

ABOUT THE AUTHOR

From the author of "Real Estate Investment," "Bond Analysis," "Owner Financed Real Estate" and "Homeowner Selling Analysis" as featured by Petrocci Freelance Associates.

REQUIRES 16K EXTENDED
COLOR BASIC

TAPE \$34.95
DISK \$39.95

ILL. RESIDENTS ADD
8% SALES TAX

SEE RAINBOW REVIEW
JULY 1984

A&P SOFTWARE
P.O. Box 202
Glenview, IL
60025

"... RPM does exactly what it
says it will do in fine style."



```

C002      0018 RETURN EQU $C002
0000      0019 BASIC EQU $0000
C004      0020 DSKCON EQU $C004
C006      0021 DCOPC EQU $C006
A002      0022 CHR0UT EQU $A002
A000      0023 POLCAT EQU $A000
A928      0024 CLEAR EQU $A928      DIRECT JUMP TO CLEAR ROUTINE
0025 *
0E00      0026      ORG $E00
0027 *
0E00 7F0F17 0028 SYSSAV CLR TRACK RESET TRACK POINTER
0E03 7F0F18 0029 CLR SECTOR CLEAR SECTOR POINTER
0E06 7C0F18 0030 INC SECTOR SET TO 1
0031 *
0E09 BDA928 0032 JSR CLEAR CLEAR SCREEN
0E0C 3000209 0033 LEAX TITLE.PCR LOAD TITLE MESSAGE POINTER INTO X
0E10 1700E3 0034 LBSR DISPLY DISPLAY IT
0035 *
0E13 30800224 0036 ASKDNO LEAX DRIVNO.PCR ASK FOR DRIVE NUMBER
0E17 1700DC 0037 LBSR DISPLY
0E1A AD9FA000 0038 REPET JSR [POLCAT] GET DRIVE NUMBER
0E1E 27FA 0039 BEQ REPET
0E20 AD9FA002 0040 JSR [CHR0UT] ECHO ENTRY
0E24 B130 0041 CMPA #0 IS IT LOWER THAN ASCII ZERO?
0E26 25E8 0042 BLO ASKDNO YES
0E28 B131 0043 CMPA #1 IS IT HIGHER THAN ASCII 1?
0E2A 22E7 0044 BHI ASKDNO YES
0E2C 10BEC006 0045 LDY DCOPC POINT Y AT PARAMETERS
0E30 8030 0046 SUBA #0 REDUCE TO A NUMBER
0E32 A721 0047 STA 1,Y SELECT DRIVE
0048 *
0049 *
0050 *GET SECTOR! TRACK 17 TO SEE IF
0051 *THIS WAS A SYSTEM DISK
0E34 B611 0052 LDA #17 TRACK

```

```

0E36 A722 0053 STA 2,Y
0E38 B601 0054 LDA #1 SECTOR
0E3A A723 0055 STA 3,Y
0E3C CC0F19 0056 LDD #BUFFER
0E3F ED24 0057 STD 4,Y
0E41 B602 0058 LDA #2 READ CODE
0E43 A7A4 0059 STA ,Y
0E45 AD9FC004 0060 JSR [DSKCON]
0E49 6D26 0061 TST 6,Y ERRORS?
0E4B 10260091 0062 LBNE ERRORS
0E4F F60F19 0063 LDB BUFFER TEST FOR #55
0064 *GET EXISTING DISK MAP INTO BUFFER
0065 *
0E52 1700A6 0066 LBSR GETMAP
0E55 6D26 0067 TST 6,Y ANY ERRORS
0E57 10260005 0068 LBNE ERRORS
0E5B C155 0069 CMPB #55
0E5D 2605 0070 BNE NEWSYS
0E5F BE0F28 0071 LDY #BUFFER+15
0E62 200E 0072 BRA OUTMAP
0073 *
0074 *CHECK FOR 255 IN FIRST 15 BYTES
0075 *IF NOT ALL 255 THEN DISK NOT AVAILABLE.
0076 *
0E64 BE0F19 0077 NEWSYS LDX #BUFFER POINT X TO BUFFER
0E67 A600 0078 NXTBYT LDA ,X+ GET BYTE
0E69 B1FF 0079 CMPA #0FF IS IT 255?
0E6B 267A 0080 BNE NOTAV OUTPUT NOT AVAILABLE MESSAGE
0E6D BC0F28 0081 CMPX #BUFFER+15 DONE ALL 15?
0E70 25F5 0082 BLO NXTBYT
0083 *
0084 *SET UP MAP AND WRITE OUT
0085 *
0E72 B6C6 0086 OUTMAP LDA #C6 LAST GRANULE POINTER
0E74 A7B2 0087 STA ,-X
0E76 B60F 0088 LDA #15 15 AT 14 ETC.
0E78 4A 0089 DONEXT DECA
0E79 A7B2 0090 STA ,-X
0E7B BC0F19 0091 CMPX #BUFFER DONE ALL 15?
0E7E 22F8 0092 BHI DONEXT
0093 *
0094 *PUT IT ONTO DISK
0E80 17007E 0095 LBSR PUTMAP
0E83 6D26 0096 TST 6,Y ANY ERRORS?
0E85 10260057 0097 LBNE ERRORS
0098 *****
0099 *MARK DISK AS A SYSTEM DISK BY
0100 *SETTING BYTE1 IN SECTOR 1 TO #55 IN TRACK 17
0E89 B601 0101 LDA #1 SECTOR
0E8B A723 0102 STA 3,Y
0103 *SET UP DRIVE OP CODE
0E8D B655 0104 LDA #55 MARKER
0E8F B70F19 0105 STA BUFFER
0E92 AD9FC004 0106 JSR [DSKCON]
0E96 6D26 0107 TST 6,Y
0E98 2646 0108 BNE ERRORS
0109 *****
0E9A B603 0110 LDA #3 WRITE CODE
0E9C A7A4 0111 STA ,Y
0112 *POINT X AT START OF BASIC
0E9E 8E8000 0113 LDX #BASIC
0114 *
0115 * START TRANSFER
0116 *
0117 NXTSCT LDA TRACK GET TRACK NUMBER
0EA1 B60F17 0118 STA 2,Y
0EA4 A722 0119 LDA SECTOR BET SECTOR NUMBER
0EA6 B60F18 0120 STA 3,Y
0EA9 A723 0121 STX 4,Y BUFFER ADDRESS
0EAB AF24 0122 *
0EAD AD9FC004 0123 JSR [DSKCON] WRITE BLOCK
0EB1 6D26 0124 TST 6,Y CHECK FOR ERRORS
0EB3 262B 0125 BNE ERRORS REPORT THEM
0126 *
0127 *INCREMENT VALUES
0EB5 30890100 0128 LEAX 256,X MOVE BUFFER POINTER
0EB9 B60F17 0129 LDA TRACK IS IT LAST TRACK?
0EBC B106 0130 CMPA #6
0EBE 2509 0131 BLO NOTLST

```

HARDWARE PRODUCTS FOR THE TRS-80 COLOR COMPUTER®

SERIAL SWITCHERS

These bi-directional switchers allow you to expand your serial port to two or three peripherals or to connect one peripheral to two or three computers. They are a compact 2 x 3 x 1 1/2 inches and are available with a mounted pilot light.

2 Ports \$25.00
 3 Ports \$30.00
 Add \$5.00 for Pilot Light

ROMs

BASIC ROM 1.1 \$45.00
 BASIC ROM 1.2 \$35.00
 E.C.B. ROM 1.1 \$60.00
 D.E.C.B. ROM 1.1 \$35.00

RAMs

4164-64K RAM \$6.00
 Set of Eight \$50.00
 4116-16K RAM \$1.10
 Set of Eight \$8.00

I.C.s

6809E-1 MHz MPU \$25.00
 68B09E-2 MHz MPU \$30.00
 6821-1 MHz PIA \$8.00
 68B21-2 MHz PIA \$10.00
 6883-SAM \$25.00
 6847-VDG \$20.00
 1 MHz Set of Four \$65.00
 2 MHz Set of Four \$70.00
 6822-H.D. PIA \$15.00

64K FOR \$75.00

Price includes expert installation, a 64K RAM Button, 64K Software (Specify disk or cass.), a 64K User Sheet, Return Shipping, and a 90-DAY UNCONDITIONAL WARRANTY. Requires 1.1 or newer BASIC ROM. Send your operating 285 (F) Series Color Computer, 10P-100, or Color Computer 2 with a Cashier's Check or Money Order for fastest return. For D, or E Series boards, add \$20.00. If necessary, add \$35.00 for new ROM.

MISC

VT-8302 Pilot Light Kit \$7.00
 VT-8401 Cooling Fan Kit \$25.00
 6' T.V. Cable w/R.F.I. Filter \$15.00
 40-Pin, Clip-on Heatsink \$1.00
 16K, 32K, or 64K RAM Button \$3.00
 16 to 24 Pin I.C. Extractor \$3.00
 4, 5, or 6 Pin, M or F, Cable DIN \$1.00
 4, 5, or 6 Pin, F, Chassis DIN \$2.00

TERMS: Cashier's checks and money orders for immediate delivery • Personal checks allow 2 weeks • Orders \$100 to \$199 save 10% • \$200 and over save 15% • California residents add 6% • Orders under \$25 add \$2 shipping • C.O.D. add \$4

4418 E. Chapman Ave., Suite 284
 Orange, CA 92669
 (714) 639-4070

VIDTRON

FREE
 CATALOG

```

0132 *WE GOT HERE BECAUSE THIS IS THE LAST TRACK(7)
0EC0 B60F18 0133 LDA SECTOR
0EC3 8102 0134 CMPA #2 LAST SECTOR IN TRACK 6
0EC5 2727 0135 BEQ CLOSE
0EC7 2007 0136 BRA INCMT GO TO INCREMENT
0137 *
0EC9 B60F18 0138 NOTLST LDA SECTOR
0ECC 8112 0139 CMPA #18
0ECE 2705 0140 BEQ NXXTRK
0141 *GET HERE BECAUSE NOT ALL SECTORS DONE YET
0ED0 7C0F18 0142 INCMT INC SECTOR
0ED3 20CC 0143 BRA NXSCT DO NEXT SECTOR
0144 *
0145 *NOT HERE BECAUSE LAST SECTOR
0ED5 7F0F18 0146 NXXTRK CLR SECTOR
0EDB 7C0F18 0147 INC SECTOR
0EDB 7C0F17 0148 INC TRACK
0EDE 20C1 0149 BRA NXSCT
0150 *****
0EE0 308D0170 0151 ERRORS LEAX ERR,PCR
0EE4 8D10 0152 BSR DISPLY
0EE6 39 0153 RTS
0154 *****
0EE7 308D017D 0155 NOTAV LEAX NTAV,PCR
0EEB 8D09 0156 BSR DISPLY
0EED 39 0157 RTS
0158 *****
0EEE 7FFF40 0159 CLOSE CLR $FF40 TURN OFF MOTOR
0EF1 39 0160 RTS
0161 *****
0EF2 AD9FA002 0163 DISPL1 JSR [CHROUT] DISPLAY ON SCREEN
0EF6 A6B0 0164 DISPLY LDA ,X+ GET CHARACTER
0EF8 26F8 0165 BNE DISPL1
0EFA 39 0166 RTS
0167 *
0EFB 8602 0168 BETMAP LDA #2 READ OP CODE
0EFD A7A4 0169 STORE STA ,Y
0EFF 2004 0170 BRA CONT
0F01 8603 0171 PUTMAP LDA #3 WRITE OPCODE
0F03 20F8 0172 BRA STORE
0F05 8611 0173 CONT LDA #17 SELECT TRACK
0F07 A722 0174 STA 2,Y
0F09 8602 0175 LDA #2 SELECT SECTOR
0F0B A723 0176 STA 3,Y
0F0D 8E0F19 0177 LDX #BUFFER BUFFER ADDRESS
0F10 AF24 0178 STX 4,Y
0F12 AD9FC004 0179 JSR [DSKCON]
0F16 39 0180 RTS
0181 *****
0182 *
0183 * VARIABLES AND STRINGS
0F17 0184 TRACK RMB 1
0F18 0185 SECTOR RMB 1
0F19 0186 BUFFER RMB 256
1019 20 0187 TITLE FCS / BASIC TO DISK<0D> STORAGE PROGRAM<0D><0D>/
1038 0D 0188 DRIVNO FCS /<0D>DRIVE NUMBER (0 OR 1)? /
1054 0D 0189 ERR FCS *<0D><0D>READ/WRITE ERROR<0D>*
1068 0D 0190 NTAV FCS /<0D>DISK NOT FREE FOR SYSTEM STORAGE<0D>/
0191 *
0E00 0192 END SYSSAV
NO ERROR(S) DETECTED

```

SYMBOL TABLE:

```

ASKNO 0E13 BASIC 0000 BUFFER 0F19 CHROUT A002
CLEAR A928 CLOSE 0EEE CONT 0F05 DCOPC C006
DISPL1 0EF2 DISPLY 0EF6 DONEAT 0E78 DRIVNO 103B
DSKCON C004 ERR 1054 ERRORS 0EE0 GETMAP 0EF8
INCMT 0ED0 NARG 0000 NEWSYS 0E64 NOTAV 0EE7
NOTLST 0EC9 NTAV 1068 NXXBYT 0E67 NXSCT 0EA1
NXXTRK 0ED5 OUTMAP 0E72 POLCAT A000 PUTMAP 0F01
REPET 0E1A RETURN C002 SECTOR 0F18 STORE 0EFD
SYSSAV 0E00 TITLE 1019 TRACK 0F17

```

CMD=SYSSAVE /P

Listing 2

SYSTEM COMPUTERWARE MACRO ASSEMBLER PAGE 1
COOKING WITH COCO PART 1/LISTING 3 (C)1984 COLIN J. STEARMAN

```

0004 OPT NOG.LIS
0005 *
0006 *THIS WILL LOAD A SYSTEM DISK
0007 *IN DRIVE 0 INTO 64K RAM AND
0008 *START IT UP
0009 *THE SYSTEM SHOULD HAVE BEEN SAVED
0010 *BY "SYSSAVE" AND OCCUPY THE FIRST 15
0011 *GRANULES ON THE DISK. A FLAG IN THE
0012 *FIRST BYTE OF TRACK 17 SECTOR 1 TELLS
0013 *IF THE DISK CONTAINS A SYSTEM
0014 *THIS WILL RESTORE FROM $0000 TO $FF00
0015 *****
0016 *
0017 *
0018 ORG $E00
0019 *
0020 *SOME EQUATES
0021 CHROUT EQU $A002
0022 POLCAT EQU $A000
0023 BASIC EQU $8000
0024 DSKCON EQU $C004
0025 DCOPC EQU $C006
0026 CLEAR EQU $A928
0027 ROM EQU $FFDE
0028 RAM EQU $FFDF

```

DIRECT JUMP TO CLEAR ROUTINE

SOFTWARE PRODUCTS FOR THE TRS-80 COLOR COMPUTER®

EDITTRON™
Full-Screen BASIC Program Editor SAVES YOU TIME!

Let EDITTRON cut your programming time in half! You will appreciate the absolute ease at which this Full-Screen Editor allows you to **INPUT, EDIT, and DEBUG** your BASIC programs. EDITTRON performs these functions:

CURSOR-CONTROL

- Directional Movement
- Screen Scrolling
- Home the Cursor
- Limit the Cursor
- Down Page
- Up Page
- Search a Line
- Call a Line
- Find a String
- Repeat Find

SCREEN-EDITING

- Change Characters
- Extend a Line
- Kill a Line
- Insert Characters
- Delete Characters
- Move a Line
- Split a Line
- Copy a Line
- Merge Two Lines
- Auto-Numbering

Other features include: Auto-Repeating keys, Key Tone, user-friendly Prompts and Error Messages, and 24 pages of comprehensive, easy-to-read Documentation.

EDITTRON is a 3K, fully position-independent Machine Language program that requires a minimum 16K of RAM, and Extended Color BASIC.

CASSETTE.....\$ 35 DISKETTE.....\$ 40

4418 E. Chapman Ave., Suite 284
Orange, CA 92669
(714) 639-4070



VIDTRON

***FREE* CATALOG**

```

A027      0029 COLD EQU %A027
          0030 *
          0031 *SET UP FOR DRIVE 0
0E00 10DEC006 0032 SYSTEM LDY DCOPC
0E04 6F21     0033 CLR 1,Y DRIVE NUMBER
          0034 *
          0035 *CLEAR SCREEN
0E06 8DA928  0036 JSR CLEAR
          0037 *
          0038 *PUT UP TITLE
0E09 BE100D  0039 LDY #TITLE
0E0C 170060  0040 LBSR DISPLY
          0041 *
          0042 *
          0043 *CHECK FOR SYSTEM DISK
0E0F 8D3D    0044 BSR SYSCHK
          0045 *RETURN "A" AS $55 IF SYSTEM DISK
0E11 8155    0046 CMPA #$55
0E13 2707    0047 BEQ DISKOK
0E15 0E0FEE  0048 LDY #NOSYS POINT X TO NO SYSTEM DISK MESSAGE
0E18 160054  0049 LBRA DISPLY OUTPUT IT
0E1B 39      0050 RTS
          0051 *
0E1C 8D56    0052 DISKOK BSR GETSYS
          0053 *
0E1E 7FFF40  0054 CLR %FF40 TURN OFF DRIVE
0E21 B7FFDF  0055 STA RAM SWITCH TO RAM
0E24 0F71    0056 CLR %71 CLEAR TO COLD START
          0057 *ASK FOR WHICH SYSTEM TO BOOT
0E26 8E1022  0058 LDY #000
0E29 170043  0059 LBSR DISPLY
0E2C AD9FA000 0060 POLAGN JSR [POLCAT] GET RESPONSE
0E30 27FA    0061 BEQ POLAGN NONE YET?
0E32 8142    0062 CMPA #'B IS IT BASIC?
0E34 2606    0063 BNE EORD
0E36 7F8000  0064 CLR %8000 SET UP COLOR BASIC
0E39 7EA027  0065 JMP COLD GO TO IT
0E3C 8145    0066 EORD CMPA #'E IS IT EXTENDED BASIC
0E3E 2606    0067 BNE ISITD
0E40 7FC000  0068 CLR %C000 SET UP FOR EXTENDED BASIC
0E43 7EA027  0069 JMP COLD GO TO IT
0E46 8144    0070 ISITD CMPA #'D
0E48 102791DB 0071 LBQR COLD GO TO DISK BASIC
0E4C 20DE    0072 BRA POLAGN
          0073 *****
          0074 *SYSTEM DISK CHECK.
0E4E 8611    0075 SYSCHK LDA #17 TRACK.
0E50 A722    0076 STA 2,Y
0E52 8601    0077 LDA #1 SECTOR
0E54 A723    0078 STA 3,Y
0E56 CC0ED8  0079 LDD #BUFFER
0E59 ED24    0080 STD 4,Y
0E5B 8602    0081 LDA #2 READ OP CODE
0E5D A7A4    0082 STA ,Y
0E5F AD9FC004 0083 JSR [DSKCON]
0E63 6D26    0084 TST 6,Y
0E65 2653    0085 BNE ERRORS
          0086 *SEE IF FIRST BYTE IS $55
0E67 860ED8  0087 LDA BUFFER
0E6A 39      0088 RTS
          0089 *****
          0090 *DISPLAY ROUTINE
0E6B AD9FA002 0091 DISPLI JSR [CHRROUT]
0E6F A6B0    0092 DISPLY LDA ,X+
0E71 26F8    0093 BNE DISPLI
0E73 39      0094 RTS
          0095 *****
          0096 * SYSTEM LOAD ROUTINE
0E74 BE0000  0097 GETSYS LDY #BASIC POINT X AS START OF BASIC
          0098 *SET UP DRIVE
0E77 7F0FDB  0099 CLR TRACK
0E7A 7F0FD9  0100 CLR SECTOR
0E7D 7C0FD9  0101 INC SECTOR TO SECTOR 1
          0102
          0103 *
0E80 B60FDB  0104 DOTFR LDA TRACK SET UP TRACK
0E83 A722    0105 STA 2,Y
0E85 B60FD9  0106 LDA SECTOR GET TO SECTOR
          0107 STA 3,Y
          0108 *READ SECTOR
          0109 JSR [DSKCON]
          0110 TST 6,Y
          0111 BNE ERRORS
          0112 *
          0113 *MOVE BUFFER INTO RAM AREA
          0114 BSR BUFMOV
          0115 *
          0116 *INCREMENT VALUES
          0117 LDA TRACK IS IT LAST TRACK?
          0118 CMPA #6 HIGHEST FULL TRACK
          0119 BLO NOTLST
          0120 *WE GOT HERE BECAUSE THIS IS ON TRACK 7
          0121 LDA SECTOR LAST SECTOR
          0122 CMPA #2 ONLY NEED ONE SECTOR ON TRACK 7
          0123 BNE INCMT GO TO INCREMENT
          0124 RTS
          0125 *
          0126 NOTLST LDA SECTOR LAST SECTOR IN OTHER TRACKS?
          0127 CMPA #18
          0128 BEQ NXTTRK
          0129 *
          0130 *GOT HERE BECAUSE NOT ALL SECTORS READ YET
          0131 INCMT INC SECTOR
          0132 BRA DOTFR CONTINUE TRANSFER
          0133 *
          0134 *GOT HERE BECAUSE LAST SECTOR
          0135 NXTTRK CLR SECTOR
          0136 INC SECTOR
          0137 INC TRACK
          0138 BRA DOTFR CONTINUE TRANSFER
          0139 *****
          0140 ERRORS' LDY #ERR
          0141 BSR DISPLY
          0142 RTS
          0143 *****
          0144 * THIS MOVES 256 BYTES FROM BUFFER
          0145 *TO LOCATION POINTED TO BY REG X
          0146 BUFMOV LDU #BUFFER POINT U TO BUFFER
          0147 ORCC #850 DISABLE INTERRUPTS
          0148 STA RAM SWITCH TO RAM
          0149 STDR LDA ,U+ GET BYTE AND INCR U
          0150 STA ,X+ STORE & INCR X
          0151 CMPU #BUFFER+256 ALL DONE
          0152 BNE STORE CONTINUE MOVING
          0153 STA ROM SWITCH BACK TO ROM
          0154 ANDCC #AF ENABLE INTERRUPTS
          0155 RTS
          0156 *****
          0157 *STORAGE
          0158 BUFFER RMB 256
          0159 TRACK RMB 1
          0160 SECTOR RMB 1
          0161 ERR FCS /*<0D>READ/WRITE ERROR<0D><0D>*
          0162 NOSYS FCS /<0D>NO SYSTEM ON DISK IN DRIVE 0<0D>/
          0163 TITLE FCS / DISK BASIC LOADER<0D><0D>/
          0164 BOOT FCS /BASIC, EXTENDED OR DISK(B,E,D)?/
          0165 *
          0166 END SYSTEM
          NO ERROR(S) DETECTED

SYMBOL TABLE:
BASIC 8000 BOOT 1022 BUFFER 0ED8 BUFMOV 0EC0
CHR0UT A002 CLEAR A928 COLD A027 DCOPC C006
DISKOK 0E1C DISPLI 0E6B DISPLY 0E6F DOTFR 0E80
DSKCON C004 EORD 0E3C ERR 0FDA ERRORS 0E8A
GETSYS 0E7A INCMT 0EAA ISITD 0E46 NARG 0000
NOSYS 0FEE NOTLST 0EA3 NXTTRK 0EAF POLAGN 0E2C
POLCAT A000 RAM FDFD ROM FFDE SECTOR 0FD9

STORE 0EC8 SYSCHK 0E4E SYSTEM 0E00 TITLE 100D
TRACK 0FDB

CMD=SYSTEM /P

```



COOKING With CoCo



PART II

In which we construct a simple plug-in cartridge programmer for the 2764 8K EPROM.

By Colin J. Stearman

This month we continue to build the tools needed to enhance the CoCo Disk Operating System (DOS). Last month we developed a means to store the complete BASIC operating system on a special system floppy disk. Now I will describe a simple construction project to build a plug-in programmer for one of the most popular (and hence cheapest!) 8K EPROM's currently available - the 2764. The primary purpose of this project is to allow us to put the modifications into an EPROM which will replace the ROM containing the original DOS. But once built, the programmer can be used to put any code you wish into a 2764.

Design Philosophy

I'm a firm believer in the "KISS" principle I learned many years ago ("Keep It Simple, Stupid!"). So this programmer uses three integrated circuits, a transistor and a few resistors and capacitors. The bulk of the work is done by the driving software. This means there are no timing circuits or other complex logic to worry about. The result is a simple project to build and get working.

Circuit Description

I do not propose to provide a long description of **how** a

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

2764 is programmed. In general, it is programmed by presenting the address and data to the chip, then pulsing the program input pin while supplying 21 volts to another.

If you look at the schematic in Figure 1, you will see that the key to the programmer's simplicity lies in the two 6821 peripheral interface adapters (PIA). These are the same chips used inside the CoCo for interfacing with the outside world. These two chips provide the 2764 with all address and data information along with other control lines. The only other chip is an inverting buffer to decode the address information to the PIAs.

One of the PIA outputs drives a transistor which activates a relay to control the 21-volt source. It's not the most elegant way of doing it, but certainly the simplest. A light emitting diode (LED) tells you when the programmer is programming. The diode around the relay suppresses transients during switching and the other two stop currents flowing to the wrong places. The capacitors are all for power supply filtering. These are not shown in Figure 1 for clarity. Locate a 0.1uF disk capacitor from +5V to 0V at each integrated circuit (polarity is not important) and one 10uF 12V electrolytic capacitor anywhere on the board across these same lines (polarity is important here, the wire labelled "+" goes to the +5V line).

The PIA is a programmable device and its external connections may be programmed as inputs or outputs. This makes it possible for the software to both program the 2764 and then read back the resulting data.

The 21 volt source is easily obtained from three nine-volt batteries and a few other components as shown in Figure 1. This circuit is not built on the board and may not be needed if you already have an adjustable power supply.

Finally, the two sockets shown at U4 and U5 have nothing to do with the programmer itself, but provide a convenient method of putting two programmed 2764s into the CoCo memory map. One socket is wired to fill the address space from \$C000 to \$DFFF and the other from \$E000 to \$FFFF. (The last 256 bytes are not accessible in the latter because the addresses \$FF00 to \$FFFF are used internally as system input/output and vector addresses.)

Construction Hints

Radio Shack sells a printed circuit breadboard with the correct 40-pin edge connect for the CoCo expansion port. Check the parts list in Figure 1 for the number. This board is ideal for the project. The photograph shows the construction method I used. The components were conveniently laid out and then hooked up using a combination of the copper tracks on the board and solid hook-up wire. Maybe it's not the most elegant, but it's serviceable and functional. You could lay out and etch a custom printed circuit board and make a more professional job if you wished.

Take your time during the construction! The finished project will be plugged into your precious CoCo and could cause some nasty problems if you make an error. Use a meter or continuity tester to make sure you have wired correctly and that there are no shorts. The most likely cause of damage is having the power supply voltages coming out of CoCo going to the wrong places. The only internal supply used is the five-volt source from pin nine of the connector so check this line carefully. Pins one and two, which supply -12 volts and +12 volts are not used, so make sure they do not go anywhere on your board. Check Figure 2 for the edge connector pin numbering.

Source For Components

Those parts available from Radio Shack have been listed

in Figure 1. The PIAs and 2764s are not available from them, nor is the Zero Insertion Force (ZIF) socket. The ZIF socket is not essential but is a good idea as it saves wear and tear on the 2764s. Most mail order houses can supply these components and I can recommend ACTIVE Electronics (800-343-0874) in Westboro, Mass. as a reputable firm. When ordering the 2764 ask for the 2764-3 which has an access time of 300nS. This is fast enough to work in the familiar "speed-up mode" that some CoCo programs use. An enclosure for the board can be obtained from The Microworks or Colorware who both advertise in RAINBOW.

The only other major item you might want to consider is an EPROM eraser. EPROMs are erased by exposure to ultraviolet light and can usually be programmed and erased many times. It is probable that you will wish to erase an EPROM you have programmed at some point and will need an eraser. If you live in the Sun Belt you might try leaving them outside in the sun for a week or two. But if you live in the north like me and forget what the sun looks like you'll have to buy an eraser. Hobby models are available for around \$60 (also from ACTIVE). They do the job in about 15 minutes and can erase 15 chips at once. UV is dangerous to the eyes and skin and these inexpensive models have no safety interlocks, so if you get one treat it with respect and NEVER look into the lighted lamp.

"Take your time during the construction! The finished project will be plugged into your precious CoCo and could cause some nasty problems if you make an error."

Software

Listing 1 shows the source code for the EPROM programmer. It is fully position independent and is an ideal candidate for loading into an EPROM. I put such a programmed EPROM into one of the sockets on the board so that the cartridge had both the hardware and software ready to go.

The program is menu driven and provides a variety of functions. Menu selection one will verify that all locations in the EPROM are erased. A colored bar shrinks as the EPROM is checked and if fully erased, this is reported. If not, the first unerased memory location is reported and the checking process stops. An EPROM is fully erased when all memory bits are a one. The programming process can only convert 1s to 0s, not the reverse. You can program a partially erased EPROM however, as long as the memory locations you do wish to program are erased.

Menu item two allows the data stored in any section of CoCo's memory to be programmed into the EPROM. This does not have to be the whole 8K and can be as little as one byte. All memory addresses are entered as hexadecimal and the EPROM memory locations are numbered from \$0000 to

\$FFFF. As the programming proceeds, the cell being programmed is indicated and also automatically verified. If a cell does not return the same data as was programmed in it, the address is shown and a "BAD EPROM" message issued. If it is just not erased, this will be reported as such. In either case the programming stops.

The third menu item allows the contents of the EPROM to be dumped as a hexadecimal and ASCII character table. This is useful for inspecting the contents of the EPROM. The EPROM start and stop addresses are supplied and the output can be directed to the screen or printer. If the screen is chosen, the output will pause and wait for any key-press after each screen is filled. In either case the BREAK key will stop output and return to the request for dump range. Pressing the ENTER key for this returns to the main menu.

Menu item four permits individual inspection and programming of EPROM memory locations. The up and down arrows scan through consecutive memory locations displaying their contents. If a new value is entered an attempt is made to program that cell. This is done by pressing the 'P' key at the appropriate address and then entering the data. Sometimes it is possible to correct minor errors in a programmed EPROM this way. A new address may be selected by pressing 'N' and entering the desired address. 'X' will return to the main menu.

The fifth menu item will return the load start and end addresses of a cassette binary file, along with the execution address. This is used to find out where a binary file from tape went in memory so that it can be transferred to the EPROM. This display does not take into account any load offset you might have used in the *CLOAD* command.

Menu item six simply returns you back to BASIC.

"When all 8K have been checked the 2764 will be declared fully erased. Pressing ENTER will return you to the menu. If you get this far, things are looking pretty good."

Assembling the Program

As I mentioned in the previous installment, I use *MAC* by Computerware as my assembler. However, many of you may have *EDTASM+* or some other brand. Generally they are compatible, but there are some differences. For example, *MAC* allows binary numbers in the operand field. These are preceded by a percent sign. For other assemblers simply figure out what the number is in hexadecimal and enter it with a dollar sign in front instead.

MAC also has an *FCS* (Form Constant String) mnemonic. This is similar to *FCC* (Form Constant Characters), but allows hexadecimal codes to be imbedded in the string by enclosing them in angle brackets. Also it automatically adds a zero byte at the end of the string. Every *FCS* instruc-

tion can be replaced by a series of *FCC* and *FCB* (Form Constant Byte) mnemonics. For example, this line:

```
FCS /<OD> Sample program <OD>Enter?/
```

would become:

```
FCB $0D return  
FCC /Sample program/  
FCB $0D return  
FCC /Enter?/  
FCB 0 terminating zero byte
```

You may also see mnemonics *OPT*, *NAM* and *TTL* in the listings. These are just directives to *MAC* and can be omitted.

Once you have entered the source code and it assembles without error, save a copy of the machine code binary file to a cassette. This will be needed to first "fire up" the programmer as the disk system will be disconnected.

Testing the Project

After you have thoroughly checked the circuit board for errors there is nothing else but to plug it in and try it. If you have a meter you might monitor a five-volt point somewhere on the board before powering up. Owners of the Multi-Pak Interface should plug the programmer into slot one and select this on the front switch. If you do not own one, remove the disk controller and plug the programmer directly into the computer.

Now cross your fingers and power up. (If you have the Multi-pak, just power that up and verify the five-volt line with your meter first.) Now power up CoCo. If the screen does not clear and the copyright notice does not appear in the normal time, power down immediately and further check your construction.

If everything is alright so far, *CLOADM* and *EXEC* the programmer driver software from your cassette. The title and menu should appear. If not, recheck your typing of the source code.

Without a 2764 in the ZIF socket, select menu item one. If the programmer is working you will see a purple horizontal bar which shrinks from the right as each of the 1024 bytes are verified. (If there is no 2764 chip in the socket, it looks like a fully erased chip to the programmer.) When all 8K have been checked the 2764 will be declared fully erased. Pressing ENTER will return you to the menu. If you get this far, things are looking pretty good.

Now try menu item five and verify that the start, end and execute addresses of the programmer software just loaded from cassette are returned. Make a note of these numbers.

Next is a dry run at programming. Connect the 21.5 volt external source using clip leads. Still without a 2764 in the ZIF socket, select menu item two. For the start and end address in RAM use the start and end address from the previous steps. For the EPROM target address use 0. As soon as you enter the zero, the program will announce the attempt to program the EPROM at address zero and then indicates you have a bad EPROM at the location. As you have not plugged in an EPROM, this is to be expected. You should have heard the relay actuate briefly and the LED may have flashed on momentarily. Press ENTER twice to return to the main menu. Things are still looking good.

Now plug in an erased 2764 into the ZIF socket. Use menu item one to verify it is erased. If so, return to menu item two and reenter the RAM start and end values as before. Target the code to begin at EPROM address \$0000. When you press ENTER the relay should "click" in and the LED come

on. As each address is programmed its EPROM address is shown on the screen. Remember that data for each address is being verified as it goes along, so there is little likelihood of wrong data being programmed in, unless it was wrong in the first place. It takes 50mS to program each location, so an entire 8K takes a little over six minutes. This is not a limitation of the software but rather a requirement of the EPROM. The programmer software is not 8K long so will not take that long.

When the last byte of the block has been programmed, the addresses of the range of bytes programmed is displayed. Pressing ENTER once would allow you to program another part of this EPROM or another one. (You could put some other program in the unused portion of the EPROM just programmed, if you wish.) Pressing ENTER again returns you to the main menu.

It would be a good idea to dump the data just programmed to double check it. This is done with menu item three. Dump the range programmed and spot check the data for errors. It should be alright.

Now power down the system and remove the 2764 from the ZIF socket and put it into the spare socket on the programmer labelled \$E000 - \$FEFF. Power up again and type in EXEC&HE000. The EPROM programmer software should immediately start up.

If you got this far without problems I think you can breathe a sigh of relief. . . the unit seems to be working fine. If not, check and double check everything and after all else fails, drop me a line and a SASE and I'll try to figure out what went wrong.

Using the Programmer with the Disk

It is a good idea to get a copy of the unmodified Disk BASIC on to a cassette and if you have the Multi-Pak to also put it into an erased EPROM. The latter is the case because the Multi-Pak Interface allows you to use the programmer with the disk system. Put the disk controller in slot four and the programmer in slot one. Initially select slot four.

To save disk BASIC to cassette, with the disk system running and a blank cassette in the tape drive, type: CSAVEM"DBASIC",&HC000,&HDFFF,&HA027.

If you have the Multi-Pak interface, the next few steps will put Disk BASIC into an EPROM so that it can be put into the other socket on the programmer. If you don't have this interface there is little point in doing this as the CoCo cannot have the programmer and disk controller available to it at the same time. However, Disk BASIC on a cassette will come in useful later.

For those with the interface, continue by powering down and selecting slot one. Then power up to Extended Color BASIC. Type in the following commands:

```
CLEAR 200,&H3FFF
CLOADM"DBASIC",&H4000-&HC000+65536
EXEC &HE000
```

Assuming you have the programmer software in an EPROM in the socket as \$E000, it should start up and you can program a fully erased 2764 with the data stored in RAM at \$4000 though \$5FFF. This, of course, is Disk BASIC.

When the EPROM is programmed, power down and put the EPROM in the other socket on the programmer (\$C000 to \$EFFF, the normal addresses for Disk BASIC). With the selector still in position one, power up the system. You should get the normal Disk Extended Color BASIC banner. You are now running Disk BASIC from the EPROM. How-

ever, it will not work properly because the secondary chip select signal is going to slot one (because of the position of the switch) and it needs to go to the controller in slot four. This is accomplished by entering POKE 65407,3. Now the system will act normally until you press Reset. Then you'll have to do this POKE again.

You can now load machine code files from disk and then activate the programmer code. This is done by redirecting the secondary chip select to slot one with a POKE65407,0, then EXEC&HE000 to start up the programmer code. Menu item six returns to Disk BASIC where the secondary chip select can once again be directed to slot four.

Wrapping It Up

If this was your first construction project and you got here with no problem, congratulations — you are now a qualified "hardware hacker." For those "old hands" this should have made a simple but rewarding project.

We now have all the necessary tools to enhance the DOS, so next month we will start that in earnest by revising some commands and maybe adding one or two new ones. Until then!

Listing 1:

```

EPROM.MAC                                COMPUTERWARE MACRO ASSEMBLER PAGE 1
2764 EPROM PROGRAMMER By C.J.STEARMAN (C)1984

0001 *****
0002 *   E P R O M   P R O G R A M M E R   *
0003 *           B Y                       *
0004 *   C O L I N   S T E A R M A N   *
0005 *           *                         *
0006 *   (C)1984 C.J.Stearman             *
0007 *****
0008 *
0009 * THIS IS POSITION INDEPENDENT
0010 *
0011 *
0012 *
0013 *****
0E07 0014   DRG  9E00
0015 *
0016 *
0017 *****
0018 * SOME EQUATES
0019 *
A928 0020 CLEAR EQU  9A928   BASIC CLEAR SCREEN ROUTINE
B1DA 0021 BUFFER EQU  91DA   USES THE CASSETTE BUFFER

```

THE ROMPACK COPIER

- Copy and run rompacks from cassette or disk.
- Works even on so called "problem packs"
64K required. cassette \$16.95

TRIVIAL CHASE

This is the one! The game that has become a cult phenomena finally comes to the Co Co. The board you play on is represented by graphics. 2000 trivia questions included. Not an imitation! ECB req. 16,32,64K all included. cassette \$24.95

EXTERNAL EVENTS SOFTWARE CO.

P.O. BOX 892 • MADISON, TN 37116

0000	0022 NUMK EQU	NUMBER OF K IN EPROM	0EF3 55414C2043
1FFF	0023 TOPADD EQU	(NUMK*1024)-1 TOP EPROM ADDRESS	0EF8 454C4C53
	0024 *		0EFC 00 0078 FCB 000
	0025 *		0EFD 20 0079 FCC / 5 - CASSETTE FILE DATA/
	0026 *****		0EFE 2035202020
	0027 *		0EF3 4341535345
	0028 * MAINLINE OF PROGRAM		0EF8 5454452046
	0029 *		0EF0 494C452044
	0030 *		0F12 415441
0E00 170130	0031 EPROM LBSR INIT	SET UP THE PIAS	0F15 00 0080 FCB 000
	0032 * NORMAL EPROM MODE IS	TO READ THE EPROM	0F16 20 0081 FCC / 6 - RETURN TO BASIC/
0E03 0DA920	0033 MENU JSR CLEAR	CLEAR SCREEN	0F17 2036202020
0E06 308D0043	0034 LEA1 MENU,PCR	POINT TO MENU TEXT	0F1C 5245455552
0E0A 17094E	0035 LBSR OUTST0	OUTPUT THE MENU	0F21 4E20544F20
	0036 *		0F26 4241534943
	0037 *GET RESPONSE		0F2B 0000 0082 FDB 00000
0E0D 1700FA	0038 LBSR INSTR0	GET RESPONSE INTO BUFFER	0F2D 20 0083 FCC / SELECTION? /
	0039 * FIRST SEE IF ONLY 1 CHARACTER ENTERED		0F2E 2053454C45
0E10 06010C	0040 LDA BUFFER+2	SHOULD BE ZERO	0F33 4354494F4E
0E13 26EE	0041 BNE MENU	IT WASNT	0F38 3F20
0E15 06010A	0042 LDA BUFFER	GET FIRST CHARACTER IN BUFFER	0F3A 00 0084 FCB # MESSAGE TERMINATOR
0E18 0131	0043 CMPA #'1	VERIFY ERASE	0085 *
0E1A 2605	0044 BNE .COPY		0086 *
0E1C 17021F	0045 LBSR ERASE		0087 *****
0E1F 20E2	0046 BRA MENU		0088 *****
0E21 0132	0047 .COPY CMPA #'2	COPY RAM	0089 * EPROM ACCESS ROUTINES
0E23 2605	0048 BNE .DUMP		0090 *****
0E25 170318	0049 LBSR COPY		0091 *
0E28 20D9	0050 BRA MENU		0092 *
0E2A 0133	0051 .DUMP CMPA #'3	DUMP EPROM	FF41 0093 COMREG EQU %FF41 LOWEST CONTROL REGISTER
0E2C 2605	0052 BNE .CELL		FF44 0094 LOWADD EQU %FF44 LOW ADDRESS OUTPUT
0E2E 170543	0053 LBSR DUMP		FF46 0095 HIADD EQU %FF46 HIGH ADDRESS OUTPUT
0E31 20D0	0054 BRA MENU		FF40 0096 DATARG EQU %FF40 DATA REGISTER
0E33 0134	0055 .CELL CMPA #'4	INDIVIDUAL CELL PROGRAM	FF42 0097 CLINES EQU %FF42 CONTROL LINES REGISTER
0E35 2605	0056 BNE .FILE		FF43 0098 VOLTS EQU %FF43 RELAY CONTROL REGISTER
0E37 170608	0057 LBSR CELL		0099 *
0E3A 20C7	0058 BRA MENU		0100 *
0E3C 0135	0059 .FILE CMPA #'5	CASSETTE FILE DATA RETURN	0101 *****
0E3E 2605	0060 BNE .BASIC		0102 * INITIALIZING ROUTINE
0E40 170818	0061 LBSR CFILE		0103 *
0E43 20BE	0062 BRA MENU		0F3B 4F 0104 INIT CLRA EXPOSE ALL THE DDRS
0E45 0136	0063 .BASIC CMPA #'6	IS IT EXIT TO BASIC?	0F3C 8D1F 0105 BSR DDRSET
0E47 260A	0064 BNE MENU	NO SO DO MENU AGAIN	0106 *
0E49 0DA920	0065 JSR CLEAR	BEFORE GOING TO BASIC	0107 * NOW ALL DATA DIRECTION REGISTERS ARE EXPOSED
0E4C 39	0066 RTS	EXIT FOR CHECK SO FAR	0108 *
	0067 *		0F3E C0FF 0109 LDB 00FF SET ALL ADDRESS LINES TO OUTPUTS
0E4D 45	0068 MENUT FCC	/ E P R O M P R O G R A M M E R /	0F40 F7FF44 0110 STB LOWADD
0E4E 205A205220			0F43 F7FF46 0111 STB HIADD
0E53 4F204D2020			0112 *
0E58 2020502052			0F46 C007 0113 LDB 07 SE CONTROL LINES TO OUTPUTS
0E5D 204F204720			0F4B F7FF42 0114 STB CLINES
0E62 522041204D			0115 *
0E67 204D204520			0F4B 7FFF40 0116 CLR DATARG TO MAKE IN INPUTS
0E6C 52			0117 *
0E6D 3D	0069 FCC	/*****	0F4E 0604 0118 LDA 04 RESET THE CONTROL REGISTERS
0E6E 3D3D3D3D3D			0F50 8D0B 0119 BSR DDRSET TO OUTPUTS
0E73 3D3D3D3D3D			0120 *
0E78 3D3D3D3D3D			0F52 0634 0121 LDA 0034 SET CONTROL REG FOR RELAY OUTPUT
0E7D 3D3D3D3D3D			0F54 07FF43 0122 STA VOLTS ENABLES CB2 AS OUTPUT AT ZERO
0E82 3D3D3D3D3D			0123 *
0E87 3D3D3D3D3D			0F57 0601 0124 LDA 01 SET UP CONTROL LINES FOR READ
0E8C 3D			0F59 07FF42 0125 STA CLINES OE,CS=0 PGM=1
0E8D 0000	0070 FDB 0000	TWO (CR)	0126 *
0E8F 20	0071 FCC	/ I - VERIFY ERASURE/	0F5C 39 0127 RTS
0E90 2031202020			0128 *
0E95 5645524946			0129 **
0E9A 5920455241			0130 ***** SUBROUTINES *****
0E9F 53555245			0131 * SET CONTROL REGISTERS TO CONTENTS OF A
0EA3 00	0072 FCB 000		0132 DDRSET LDB 04 # OF CONTROL REGISTERS
0EA4 20	0073 FCC	/ 2 - PROGRAM EPROM FROM MEMORY/	0133 LDX 0CONREG POINT I TO CONTROL REGISTERS
0EA5 2032202D20			0134 CLRREG STA ,I++ CLEAR AND DOUBLE INCREMENT
0EAA 50524F4752			0135 DECB DECREASE COUNTER
0EAF 414D204550			0136 BNE CLRREG DO NEXT REGISTER
0EBA 524F4D2046			0137 RTS
0EB9 524F4D204D			0138 *****
0EBE 454D4F5259			0139 *
0EC3 00	0074 FCB 000		0140 *
0ECA 20	0075 FCC	/ 3 - DUMP EPROM CONTENTS/	0141 *****
0EC5 2033202D20			0142 * PROGRAM EPROM ROUTINE
0ECA 44554D5020			0143 *****
0ECF 4550524F4D			0144 *
0EDA 20434F4E54			0145 * THIS PROGRAMS THE PROM FROM DATA STARTING
0ED9 454E5453			0146 * AT ADDRESS IN "START", FOR THE NUMBER OF
0EDD 00	0076 FCB 000		0147 * BYTES IN "COUNT", AT EPROM ADDRESS "TARGET"
0EDE 20	0077 FCC	/ 4 - PROGRAM INDIVIDUAL CELLS/	0148 * THESE LOCATIONS ARE RESERVED IN THIS ROUTINE
0EDF 2034202D20			0149 * START ENDS UP WITH LAST ADDRESS DATA WAS
0EE4 50524F4752			0150 * TAKEN FROM RAM. TARGET HAS LAST ADDRESS
0EE9 414D20494E			0151 * WRITTEN TO IN EPROM.
0EEE 4449564944			0152 * B HAS ERROR CODE

0153 *	1= MCT ERASED			0FF6 C602	0242	LDB #2	VERIFY ERROR CODE
0154 *	2= BAD EPROM LOCATION			0FFB 2010	0243	BRA PEJ1T	
0155 *	0= NO PROBLEM				0244 *		
0156 *	*****				0245 *	END PROGRAMMING LOOP	
0157 *					0246 *	*****	
01D1	0150 START EQU #1D1	USED CASSETTE NAME AREA		0FFA AD9FA000	0247 VERIDK JSR {POLCAT}	BREAK PRESSED?	
01D3	0159 COUNT EQU START+2			0FFE 2705	0248 BEQ DOWNCT	NO SO DECREASE COUNT	
01D5	0160 TARGET EQU START+4			1000 5F	0249 CLR#	READY FOR RETURN CODE	
	0161 *			1001 0103	0250 CMP# 03	BREAK VALUE	
0F58 B6FF43	0162 PROGRM LDA VOLTS	GET REG VALUE		1003 2705	0251 BEQ PEXIT	YES SO EXIT	
0F60 9A00	0163 OR# 0200001000	SET BIT 0 TO APPLY 21V		1005 301F	0252 DOWNCT LEAX -1,X	REDUCE COUNT	
0F6D B7FF43	0164 STA VOLTS			1007 2600	0253 BNE PLOOP	NOT DONE YET	
	0165 * 21V IS NOW APPLIED			1009 5F	0254 CLR#	NO ERROR CODE	
	0166 * WAIT A WHILE FOR RELAY TO CLOSE			100A 335F	0255 PEXIT LEAU -1,U	DECREASE TO LAST LOADED ADDRESS	
0F70 BEFFFF	0167 LDX #FFFF			100C FF01D5	0256 STU TARGET		
0F73 301F	0168 RLYDLY LEAX -1,X	DECREMENT X		100F 313F	0257 LEAY -1,Y	DO SAME FOR RAM ADDRESS	
0F75 26FC	0169 BNE RLYDLY			1011 10BF01D1	0258 STY START	SAVE LAST RAM ADDRESS	
	0170 *			1015 B6FF43	0259 PEXIT LDA VOLTS	GET VOLTS REGISTER	
0F77 3460	0171 * PRESERVE Y AND U REGISTERS			1018 04F7	0260 AND# 0111110111	TURN OFF 21V	
	0172 PSHS U,Y			101A B7FF43	0261 STA VOLTS		
	0173 *			101D 35E0	0262 PULS U,Y,PC	RECOVER REGISTERS & RETURN	
	0174 *			0263 *			
0F79 10BE01D1	0175 LDY START	POINT Y TO RAM START		0264 *			
0F7D FE01D5	0176 LDU TARGET	POINT U TO TARGET		0265 *			
0F80 5F	0177 CLR#			0266 *	THIS PULSES THE PGM LINE LOW FOR 50MS		
0F81 BE01D3	0178 LDX COUNT	GET BYTE COUNT		0267 *			
0F84 1027000D	0179 LBEQ PEXIT	ALL DONE PROGRAMMING		101F B6FF42	0268 PULSE LDA CLINES	GET LINES	
	0180 *			1022 04FE	0269 AND# 0111111110	MAGE PGM LOW	
	0181 * MOVE CURSOR AHEAD 4			1024 B7FF42	0270 STA CLINES		
0F88 FC0000	0182 LDD CURLOC			0271 *			
0F8B C30004	0183 ADD# #4			1027 3410	0272 PSHS X	FOR DELAY COUNT	
0F8E F00000	0184 STD CURLOC			1029 1A50	0273 ORCC 0201010000	PREVENT INTERRUPTS	
	0185 *****			102B 0E1600	0274 LDX 001600	FOR 50 MS	
	0186 * PROGRAMMING LOOP			102E 301F	0275 DLOOP LEAX -1,X	REDUCE COUNT	
0F91 1F30	0187 PLOOP TFR U,D	SET UP EPROM ADDRESS		1030 26FC	0276 BNE DLOOP	KEEP LOOPING	
0F93 3341	0188 LEAU 1,U	INCREMENT EPROM ADDRESS		1032 1CAF	0277 ANDCC 0110101111	ALLOW INTERRUPTS	
0F95 B7FF46	0189 STA HIADD			0278 *			
0F96 F7FF44	0190 STB LGWADD			1034 B6FF42	0279 LDA CLINES	GET LINES	
	0191 *			1037 B001	0280 OR# 0100000001	SET PGM HI	
	0192 * DISPLAY WORKING ADDRESS			1039 B7FF42	0281 STA CLINES		
0F9B 3436	0193 PSHS Y,X,D			103C 3590	0282 PULS X,PC	RECOVER X AND RETURN	
0F9D FC01D3	0194 LDD COUNT	DO NOT IF IT IS 1		0283 *****			
0FA0 10030001	0195 CMP# 01			0284 *****			
0FA4 270F	0196 BEQ NODISP	NO DISPLAY		0285 *	VERIFY ROUTINE *		
0FA6 ECE4	0197 LDD ,S	RECOVER VALUE IN D		0286 *****			
0FAB BE0000	0198 LDX CURLOC	MOVE CURSOR BACK 4		0287 *			
0FAB 301C	0199 LEAX -4,X			0288 *	THIS VERIFIES ERASURE OF THE EPROM		
0FAD BF0000	0200 STX CURLOC			0289 *	PROVIDES A 60/NOGO RESPONSE		
0FB0 1F01	0201 TFR D,X	MOVE VALUE TO X		0290 *	OF ERASURE OF ENTIRE EPROM		
0FB2 170916	0202 LBSR HE1OUT	DISPLAY IT		0291 *			
0FB5 3536	0203 NODISP PULS Y,X,D	RECOVER VALUES		103E 00A920	0292 ERASE JSR CLEAR	CLEAR SCREEN	
	0204 * GET DATA TO BE LOADED INTO REG B			1041 3420	0293 PSHS Y	PRESERVE REGISTER Y	
0FB7 E6A0	0205 LDB ,Y+	AND INCREMENT ADDRESS		1043 300D007B	0294 LEAX ERANS6,PCR	PUT UP TITLE	
	0206 *			1047 170711	0295 LBSR OUTST0		
0FB9 B6FF40	0207 LDA DATARG	GET DATA AT THIS ADDRESS		0296 *			
0FBC 01FF	0208 CMP# #0FF	SHOULD BE THIS		0297 *	PUT UP PROGRESS MONITOR		
0FBE 2704	0209 BEQ EMPTY			104A BE0000	0298 LDX CURLOC	GET CURSOR LOCATION	
0FC0 C601	0210 LDB 01	NOT ERASED CODE		104D 3000	0299 LEAX NUMK,X	MOVE OVER NUMBER OF K IN EPROM	
0FC2 2046	0211 BRA PEXIT			104F BF0000	0300 STX CURLOC	AND SAVE IT	
	0212 *			1052 CC9FBF	0301 LDB 00BF0F	2 RED SQUARES	
0FC4 B6FF42	0213 EMPTY LDA CLINES	GET CONTROL LINES		1055 10EE0000	0302 LDX #NUMK	COUNTER	
0FC7 0A02	0214 OR# 0100000010	RAISE OE		1059 ED01	0303 PUTMN STD ,X++	STORE ON SCREEN	
0FC9 B7FF42	0215 STA CLINES			105B 313F	0304 LEAY -1,Y	DECREASE COUNT	
	0216 *			105D 26FA	0305 BNE PUTMN		
0FCC 7FFF41	0217 CLR CONREG	MAKE DATA LINES OUTPUTS		0306 *			
0FCF B6FF	0218 LDA #0FF			105F 109E0000	0307 LDY #0	START ADDRESS	
0FD1 B7FF40	0219 STA DATARG			1063 AD9FA000	0308 VLOOP JSR {POLCAT}	TEST FOR BREAK	
0FD4 0604	0220 LDA #4	RESET TO OUTPUT REG		1067 2706	0309 BEQ NOBRK	HE KEY PRESSED	
0FD6 B7FF41	0221 STA CONREG			1069 0103	0310 CMP# 03	BREAK?	
	0222 *			106B 2602	0311 BNE NOBRK		
0FD9 F7FF40	0223 * SAVE DATA IN B IN EPROM			106D 35A0	0312 PULS Y,PC	RETURN	
	0224 STB DATARG	PUT ON DATA LINES		0313 *			
	0225 *			106F 1F20	0314 NOBRK TFR Y,D		
0FDC 0D41	0226 BSP PULSE	THE PGM LINE LOW		1071 B7FF46	0315 STA HIADD	SET UP ADDRESS ON PIA	
	0227 *			1074 F7FF44	0316 STB LGWADD		
	0228 *NOW VERIFY			1077 B6FF40	0317 LDA DATARG	GET DATA	
0FDE 7FFF41	0229 CLR CONREG	MAKE DATA LINE INPUTS		107A 01FF	0318 CMP# #0FF	IS IT ERASED	
0FEL 7FFF40	0230 CLR DATARG			107C 261C	0319 BNE NOTMTY	NOT ERASED	
0FE4 0604	0231 LDA #4			107E 3121	0320 LEAY 1,Y	INCREASE	
0FE6 B7FF41	0232 STA CONREG			0321 *	ADJUST PROGRESS COUNTER IF NEEDED		
	0233 *			1080 1F20	0322 TFR Y,D		
	0234 * ENABLE CHIP			1082 5D	0323 TST0	IF NOT ZERO CONTINUE	
0FE9 B6FF42	0235 LDA CLINES			1083 2609	0324 BNE DONEYT	DONE YET	
0FEC 04FD	0236 AND# 0111111101	OE LOW		1085 0403	0325 AND# 0200000011	SEEE IF THESE ARE ZERO	
0FEE B7FF42	0237 STA CLINES			1087 2605	0326 BNE DONEYT	NO SO SKIP	
	0238 *			1089 CC0FBF	0327 LDD 00BF0F	GREEN SQUARES	
	0239 * NOW COMPARE DATA ON DATARG WITH CONTENTS AT Y			109C ED03	0328 STD ,--X	DECREASE MONITOR FROM RIGHT	
0FF1 F1FF40	0240 CNP# DATARG	DATA WAS LEFT IN B FROM LOAD		0329 *			
0FF4 2704	0241 BEQ VERIDK	IT WAS THE SAME		108E 108C2000	0330 DONEYT CNPY #TOPADD+1	ADDRESS LIMIT	

```

1092 26CF #0331 BNE VLOOP
1094 308D0079 #0332 LEAX GOOD,PCR IS FULLY ERASED
1098 2019 #0333 BRA VEIIT
#0334 *
109A FC000B #0335 NOTMTY LDD CURLOC
109D C30020 #0336 ADD #32 MOVE TO NEXT LINE
10A0 FD000B #0337 STD CURLOC
10A3 308D0059 #0338 LEAX ADDNMT,PCR GET ADDRESS MESSAGE
10A7 170601 #0339 LBSR OUTST#
10AA 1F21 #0340 TFR Y,X
10AC 17071C #0341 LBSR HEXOUT PUT LAST ADDRESS UP
10AF 308D005B #0342 LEAX BAD,PCR
10B3 1706A5 #0343 VEIIT LBSR OUTST#
10B6 308D0063 #0344 LEAX VERIFY,PCR
10BA 17069E #0345 LBSR OUTST#
#0346 *
10BD 17064A #0347 LBSR INSTR# GET KEYBOARD RESPONSE
10C0 35A0 #0348 PULS Y,PC RECOVER Y AND RETURN
#0349 *
10C2 20 #0350 ERAMSG FCC # EPROM ERASURE VERIFICATION/
10C3 2020453052
10CB 4F4D204552
10CD 415355245
10D2 2050455249
10D7 4049434154
10DC 494F4E
10DF 00 #0351 FCB #0D (CR)
10E0 20 #0352 FCS # *****<0D><0D>/
10E1 20203D3D3D
10E6 3D3D3D3D3D
10E8 3D3D3D3D3D
10F0 3D3D3D3D3D
10F5 3D3D3D3D3D
10FA 3D3D3D0D0D
10FF 00
#0353 *
1100 00 #0354 ADDNMT FCS #<0D><0D>ADDRESS /
1101 0D41444452
1106 4553532000
110B 20 #0355 BAD FCS # NOT /
110C 4E4F542000
1111 00 #0356 GOOD FCS #<0D><0D> FULLY /
1112 0D20202046
1117 554C4C5920
111C 00
111D 45 #0357 VERIFY FCS #ERASED<0D><0D>PRESS 'ENTER' TO CONTINUE /
111E 5241534544
1123 0D0D0505245
1128 5353202245
112D 4E54455222
1132 20544F2043
1137 4F4E54494E
113C 55452000
#0358 *****
#0359 *****
#0360 # EPROM PROGRAMMING #
#0361 *****
#0362 *
#0363 *
#0364 * THIS GETS START ADDRESS AND END ADDRESS IN
#0365 * RAM AND START TARGET ADDRESS IN EPROM
#0366 * CHECKS FOR ERRORS THEN TRANSFERS DATA
#0367 *****
#0368 *
1140 B0A920 #0369 COPY JSR CLEAR SCREEN
1143 308D00CC #0370 LEAX CPYITL,PCR GET HEADER
1147 170611 #0371 LBSR OUTST# PUT IT UP
#0372 *
#0373 * GET START ADDRESS IF NULL THEN RETURN
114A 308D0171 #0374 LEAX STRITX,PCR
114E 17066E #0375 LBSR INPUT#
#0376 * DID WE BET A NULL
1151 B001DA #0377 LDA BUFFER GET FIRST BYTE
1154 8100 #0378 CMPA #00D IS IT CR?
1156 2601 #0379 BNE GETST
1158 39 #0380 CEIIT RTS
#0381 *
1159 170604 #0382 GETST LBSR HEYINT CONVERT INTO REG X
115C 8F01D1 #0383 STX START SET START ADDRESS
115F 5D #0384 TSTB CHECK FOR ERRORS
1160 26DE #0385 BNE COPY
#0386 *
1162 308D0170 #0387 LEAX ENDMG,PCR GET ENDING RAM ADDRESS
1166 170656 #0388 LBSR INPUT# GET ENDING ADDRESS
1169 B001DA #0389 LDA BUFFER TEST FOR NULL
116C 8100 #0390 CMPA #00D IS IT CR?
116E 27E8 #0391 BEQ CEIIT
#0392 *
1170 1705ED #0393 LBSR HEYINT GET VALUE ENTERED
1173 5D #0394 TSTB DID WE GET ERROR?
1174 26CA #0395 BNE COPY RESTART
#0396 * X NOW HAS ENDING
1176 1F10 #0397 TFR X,D PUT INTO ACC D
1178 0301D1 #0398 SUBD START FIND DIFFERENCE
117B 2542 #0399 BLO DERROR DATA ERROR MESSAGE
117D C30001 #0400 ADD #1 TO MAKE IT ACTUAL COUNT
1180 FD01D3 #0401 STD COUNT SAVE IT
#0402 *
#0403 #NOW GET TARGET ADDRESS
1183 308D0166 #0404 LEAX T8TMSG,PCR
1187 170635 #0405 LBSR INPUT#
118A B001DA #0406 LDA BUFFER NULL ENTRY?
118D 8100 #0407 CMPA #00D (CR)
118F 27C7 #0408 BEQ CEIIT SO EXIT ROUTINE
#0409 *
1191 1705CC #0410 LBSR HEXINT GET VALUE IN X
1194 5D #0411 TSTB ERROR?
1195 26A9 #0412 BNE COPY RESTART
#0413 *
#0414 * X NOW HAS START ADDRESS
1197 8F01D5 #0415 STX TARGET
119A 8C1FFF #0416 CMPX #TOPADD HIGHEST ALLOWED VALUE
119D 2229 #0417 0HI TOOHI GO TO ERROR MESSAGE
119F 3410 #0418 PSHS # PUT TARGET ONTO STACK
11A1 CC2000 #0419 LDD #NUMK*1024 EPROM SIZE
11A4 A3E1 #0420 SUBD #S+ SUBTRACT TARBET & CLEAN STACK
#0421 * D NOW HAS AVAILABLE BYTES ABOVE TARBET
11A6 00301D3 #0422 CMPD COUNT
11AA 2525 #0423 BLO NOROOM NOT ENOUGH ROOM
#0424 * ALL SEEMS OK GO PRGBRAM
#0425 * FIRST DISPLAY WORKING ADDRESS TEXT
11AC 308D01AC #0426 LEAX WRKADD,PCR
11B0 1705A0 #0427 LBSR OUTST#
#0428 *
11B3 17F0B2 #0429 LBSR PROGRAM
#0430 *
11B6 5D #0431 TSTB FOR ERROR CODE
11B7 2729 #0432 BEQ GOODPR GOOD PROGRAM
11B9 C101 #0433 CMPB #1 NOT ERASED
11BB 2741 #0434 BEQ UNERAS
11BD 204E #0435 BRA BADLOC BAD PROM LOCATION
#0436 *****
#0437 **
#0438 *
11BF 308D008B #0439 DERROR LEAX DIFF,PCR START ABOVE END MSG
11C3 170595 #0440 LBSR OUTST#
11C5 2010 #0441 BRA .KEY
#0442 *
11CB 308D00A0 #0443 TOOHI LEAX HIGH,PCR TARGET TOO HIGH
11CC 1705BC #0444 LBSR OUTST#
11CF 2007 #0445 BRA .KEY
#0446 *
11D1 308D00B2 #0447 NOROOM LEAX NROOM,PCR NOT ENOUGH ROOM IN EPROM
11D5 1705B3 #0448 LBSR OUTST#
#0449 *
11DB 308D00C7 #0450 .KEY LEAX EKEY,PCR WAIT FOR ENTER
11DC 1705E0 #0451 LBSR INPUT#
11DF 16FF5E #0452 LBRA COPY
#0453 *
11E2 308D011E #0454 GOODPR LEAX GOODP1,PCR GOOD PROGRAM
11E6 170572 #0455 LBSR OUTST#
11E9 BE01D1 #0456 LDX START GET LAST RAM ADDRESS
11EC 17050C #0457 LBSR HEXOUT OUTPUT IT
11EF 308D012D #0458 LEAX GOODP2,PCR
11F3 170565 #0459 LBSR OUTST#
11F6 BE01D5 #0460 LDX TARGET GET LAST PROM ADDRESS
11F9 1705CF #0461 LBSR HEXOUT OUTPUT IT
11FC 20DA #0462 BRA .KEY
#0463 *
11FE 308D0139 #0464 UNERAS LEAX UNERSD,PCR NOT ERASED
1202 170556 #0465 .LEAVE LBSR OUTST#
1205 BE01D5 #0466 LDX TARGET GET LAST EPROM ADDRESS
1208 1705C0 #0467 LBSR HEXOUT OUTPUT IT
120B 20CB #0468 BRA .KEY
#0469 *
120D 308D0139 #0470 BADLOC LEAX BADPRM,PCR BAD PROM LOCATION
1211 20EF #0471 BRA .LEAVE
#0472 *
#0473 *****
#0474 *
1213 20 #0475 CPYITL FCC # RAM TO EPROM TRANSFER/
1214 2020202052
1219 414D20544F
121E 204550524F
1223 4D20545241
1228 4E53464552
12D0 00 #0476 FCS #<0D> *****<0D><0D>/
122E 2020203020
1233 3D3D3D3D3D

```

PROGRAMMER'S SKETCH PAD

 *Saves Time & is Easy to Use*

 *Durable & Attractive*

 *Have Fun & Learn*



 *Would You Like To Design:*

- a) BUDGETS
- b) INVENTORY LISTS
- c) GAMES, GRAPHICS

The Kit includes: Two thick mylar coated **graphs** of the color computer's screen; step by step **instructions** for the beginner; two **demo programs**, and easy to follow "how to personalize" **budgets** that *you* write.

Each Sketch Pad has **print locations** on one side and **set screen locations** on the other, along with their corresponding **commands** and **color codes**.

This Month's Special
1 FREE SYNTACTICS' DISKETTE with each order.
Offer expires September 5, 1984

NEW!
Syntactics Single-Sided Double-Density Soft Sector Diskettes
10 Diskettes with case \$18.00
Plus \$2.00 Shipping and Handling
5 Year Guarantee

Don't delay, order yours today. . .
Write for catalog of other fine products

ONLY \$

CANADA—\$13.50
 EUROPE—\$14.50

Add an additional \$1.50 for Hi-Res

12

TO ORDER:
 CALL (707) 722-4280
 or WRITE TO:

Calif. residents add 6% sales tax.
 (Postage paid.)



P.O. Box 257
REDCREST, CALIFORNIA 95569

DEALER INQUIRIES INVITED

ORDER FORM SPI


Quantity

Name _____

Address _____

City _____ State _____

Country _____ Zip _____

Charge: MasterCard 

Acct. No. _____

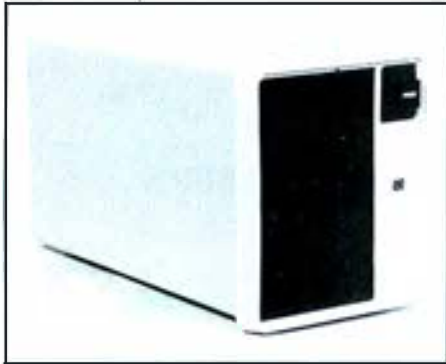
Expiration Date _____

Signature _____

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives



Introducing **MEGADISK**

5 to 20 Megabyte, ready to run on the TRS 80 Model I/III/IV/4P, color computer, I.B.M. PC, Apple, Franklin

DRIVE A HARD BARGAIN™ Complete Systems Starting at \$999.95

Call Toll Free Ordering 1-800-343-8841



High Quality Lowest Price

Drive 0, 1, 2, 3

for the

Color Computer

Starting at \$199.95



Disk Drive Upgrade

for model III/IV easy to install system

Starting at ~~\$369.95~~

Call for new lower price

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090

Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 am to 4:30 pm

DEALER INQUIRIES INVITED.

TERMS:

M.C./Visa/Amex and personal checks accepted at no extra charge. C.O.D., please add \$3.00. Shipping: Please call for amount. Not responsible for typographical errors.

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular Tel. (514) 845-1534
Canadian Toll Free 800-361-5155

Service! Service!

All in stock products are shipped within 24 hours of order. Repair/Warranty service is performed within 24 hours of receipt unless otherwise noted. We accept C.O.D., foreign and APO orders. School and D&B corporate P.O.s accepted.

TRS/80 Registered Trademark Tandy Corp. IBM-PC Registered IBM Corp. Apple Registered Trademark Apple Computer Corp. Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES


```

1238 3030303030
123D 3030303030
1242 3030303030
1247 30000000

0477 *
1248 00 #478 DIFF FCS /(<0D><0D>START HIGHER THAN END ADDRESS<0D>/
124C 0D53544152
1251 5420484947
1256 4845522054
1258 48414E2045
1260 4E44204144
1265 4452455353
126A 0000
126C 0D #479 HIGH FCS /(<0D><0D>TARGET ADDRESS TOO HIGH<0D>/
126D 0D54415247
1272 4554204144
1277 4452455353
127C 20544F4F20
1281 484947480D
1286 00
1287 0D #480 NROOM FCS /(<0D><0D>NOT ENOUGH ROOM IN EPROM<0D>/
1288 0D4E4F5420
128D 454E4F5547
1292 4820524F4F
1297 4D20494E20
129C 4550524F4D
12A1 0000
12A3 0D #481 EKEY FCS /(<0D>PRESS "ENTER" TO CONTINUE /
12A4 5052455353
12A9 2022454E54
12AE 455222054
12B3 4F20434F4E
12B8 54494E5545
12BD 2000
12BF 20 #482 STRTXT FCS / RAM START ADDRESS: /
12C0 202052414D
12C5 2053544152
12CA 5420414444
12CF 524553533A
12D4 2000
12D6 20 #483 ENDMSG FCS / RAM END ADDRESS: /
12D7 2020202052

12DC 414D20454E
12E1 4420414444
12E6 524553533A
12EB 2000
12ED 45 #484 TGMSG FCS /EPROM TARGET ADDRESS: /
12EE 50524F4D20
12F3 5441524745
12F8 5420414444
12FD 524553533A
1302 2000
1304 0D #485 GOODPI FCS /(<0D><0D> LAST RAM ADDRESS USED: /
1305 0D20204C41
130A 5354205241
130F 4D20414444
1314 5245535320
1319 555345443A
131E 2000
1320 0D #486 GOODP2 FCS /(<0D>LAST EPROM ADDRESS USED: /
1321 4C41535420
1326 4550524F4D
132B 2041444452
1330 4553532055
1335 5345443A20
133A 00
133B 0D #487 UNERSD FCS /(<0D><0D>NOT ERASED AT /
133C 0D4E4F5420
1341 4552415345
1346 4420415420
134B 00
134C 0D #488 BADPRN FCS /(<0D><0D>BAD EPROM AT /
134D 0D42414420
1352 4550524F4D
1357 2041542000
135C 0D #489 WRKADD FCS /(<0D><0D>PROGRAMMING EPROM AT /
135D 0D50524F47
1362 52414D4D49
1367 4E47204550
136C 524F4D2041
1371 542000

0490 *****
0491 *****
0492 * DUMPS EPROM CONTENTS TO SCREEN OR PRINTER *

```

The KEY-264K is here!!

DO YOU HAVE A 32K SYSTEM WITH 64K MEMORY CHIPS ?? ARE YOU STILL BEING TOLD YOU CAN ONLY USE 32K FROM BASIC ??

DON'T BELIEVE IT !! - KEY COLOR SOFTWARE brings you the KEY-264K. An exciting NEW SOFTWARE utility that allows any STANDARD 32K COLOR COMPUTER TO ACCESS 64K RAM FROM BASIC, and with NO HARDWARE MODIFICATIONS REQUIRED!!!

*** Works with CASSETTE based systems! ***

*** Works with DISK based systems! ***

The KEY-264K divides the 64K ram memory into two 32K banks or sides, each of which can be utilized independently by the BASIC interpreter, with the ability to switch instantly from one side to the other. IT'S LIKE HAVING TWO COMPUTERS IN ONE!! Have your BASIC program on one side and keep your variables on the other side, or have your main program on one side and your subroutines on the other side, or have your program on one side and use the other side for 4 additional HI-RES pages, or any combination you like. The possibilities are endless because the KEY-264K allows full communication between sides plus the ability to switch back and forth at will, all from within BASIC. You could also have different programs in each side and switch back and forth between them using simple keystrokes, even while the programs are running!! Or run them both at the same time in the FOREGROUND/BACKGROUND MULTI-TASKING mode. Don't buy that printer buffer yet! With the KEY-264K you can be printing in the background side while utilizing your computer normally in the foreground side AT THE SAME TIME!!! Debugging a program? Use either a BASIC command or simple keystrokes to instantly duplicate your program, in it's present status, on the opposite side. Switch to the opposite side later and pick up exactly where you were before!

For DISK users, the KEY-264K allows you to alternate between DISK and EXTENDED BASIC on the same side with simple keystrokes. No need to pull your controller or power down. You can be in EXTENDED BASIC on one side and in DISK BASIC on the other side and still switch back and forth and have full communications between the two sides.

The KEY-264K does this and MORE thru extensions to BASIC. No need to learn a new language! The KEY-264K adds 15 NEW COMMANDS and 1 function to BASIC, including powerful new BLOCK MEMORY MOVE and GRAPHICS VIEWING commands.

The KEY-264K works on 32K systems with "E", "F", or even modified "D" boards and requires EXTENDED or DISK BASIC with GOOD 64K MEMORY CHIPS! Systems with piggy-back 32K or half-good 64K memory chips WILL NOT WORK!!

ORDER YOUR KEY-264K CASSETTE TODAY by sending check or money order for \$39.95 plus \$2.00 postage U.S.A. (\$5.00 outside U.S.A.) Mass. residents add 5% sales tax.

MASTERCARD, VISA, OR COD
CALL (617) 263-1737

KEY COLOR SOFTWARE
P.O. BOX 360
HARVARD, MA. 01451



WORKS WITH THE NEW 64K
COLOR COMPUTER TOO!!

```

0493 *****
0494 *
1374 B0A928 0495 DUMP JSR CLEAR SCREEN
1377 308D0121 0496 LEAX DMPTTL,PCR DUMP TITLE
1378 1703DD 0497 LBSR OUTST0
0498 *
137E 308D0140 0499 LEAX DSTR1,PCR GET START ADDRESS
1382 17043A 0500 LBSR INPUT0
0501 * DID WE GET A NULL?
1385 B601DA 0502 LDA BUFFER
138B 810D 0503 CMPA #00D <CR>
138A 2601 0504 BNE DCONT CONTINUE ROUTINE
0505 *
138C 39 0506 RTS RETURN TO MENU
0507 *
138D 1703D0 0508 DCONT LBSR HEIINT INTO X REG
1390 5D 0509 TSTB AN ERROR
1391 26E1 0510 BNE DUMP RESTART IF SO
1393 8C1FFF 0511 CMPX #TOPADD CHECK RANGE
1396 22DC 0512 BHI DUMP RESTART IF OVER
1398 3410 0513 PSHS X PRESERVE START
0514 *
139A 308D013C 0515 LEAX ESTR1,PCR GET END ADDRESS
139E 17041E 0516 LBSR INPUT0
13A1 1703BC 0517 LBSR HEIINT INTO X REG
13A4 5D 0518 TSTB FOR ERROR
13A5 2605 0519 BNE RSTART RESTART IF SO
0520 * CHECK FOR OVER RANGE
13A7 8C1FFF 0521 CMPX #TOPADD
13AA 2384 0522 BLS EDUMP RANGE OK
13AC 3262 0523 RSTART LEAS 2,S CLEAN STACK
13AE 20C4 0524 BRA DUMP RESTART
13B0 1F10 0525 GDUMP TFR X,D TO SEE IF START IS AFTER END
13B2 43E4 0526 SUBD ,S START ON STACK
13B4 20F6 0527 BHI RSTART NOT SO RESTART
0528 *
13B6 3410 0529 PSHS X PRESERVE END ADDRESS
13B8 308D012E 0530 LEAX DEV,PCR WHICH DEVICE?
13BC 170400 0531 LBSR INPUT0 S OR P
13BF B601DA 0532 LDA BUFFER GET FIRST LETTER
13C2 C604 0533 LDB #0 FOR SCREEN DUMP WIDTH
13C4 B150 0534 CMPA #?P IS IT PRINTER?
13C6 2607 0535 BNE SCR NO SO LEAVE DEVNUM
13CB 86FE 0536 LDA #-2 PRINTER DEVICE CODE
13CA B7006F 0537 STA DEVNUM
13CD C610 0538 LDB #16 FOR ITEM COUNT
13CF 3530 0539 SCR PULS X,Y X HAS END, Y START
13D1 50 0540 NEGB FOR MASK
13D2 3404 0541 PSHS B SAVE ON STACK
13D4 1F20 0542 TFR Y,D ROUND DOWN START
13D5 E4E4 0543 ANDB ,S ROUNDED DOWN NOW
13D8 1F02 0544 TFR D,Y PUT IT BACK IN Y
13DA 3410 0545 PSHS X SAVE END ON STACK
0546 * SET LINE COUNT FOR SCREEN
13DC 8610 0547 LDA #16 # OF LINES
13DE B701D3 0548 STA COUNT
0549 *
13E1 1F21 0550 DNLOOP TFR Y,I OUTPUT ADDRESS
13E3 3420 0551 PSHS Y SAVE Y
13E5 B60D 0552 LDA #10D
13E7 AD9FA002 0553 JSR [CHROUT] START NEW LINE
13E6 1703DD 0554 LBSR HEXOUT OUTPUT ADDRESS
13EE 3520 0555 PULS Y RECOVER Y
13F0 C606 0556 LDB #0 SPACES COUNT
13F2 170090 0557 LBSR SPACES OUTPUT THEM
13F5 1F20 0558 INLOOP TFR Y,D GET START ADDRESS
13F7 B7FF46 0559 STA HIADD
13FA F7FF44 0560 STB LOWADD SET UP EPROM ADDRESS
13FD 3121 0561 LEAY 1,Y INCREMENT ADDRESS
0562 **** OUTPUT THE HEX CHARACTERS
13FF F6FF40 0563 LDB DATARG GET FROM EPROM
1402 3420 0564 PSHS Y PRESERVE VALUE
1404 170390 0565 LBSR H1PAIR PUT IN BUFFER
1407 8E01DA 0566 LDX #BUFFER POINT TO IT
140A 17034E 0567 LBSR OUTST0
1400 3520 0568 PULS Y RECOVER Y
140F C601 0569 LDB #1
1411 907D 0570 BSR SPACES
0571 *
1413 1F20 0572 TFR Y,D RECOVER COUNT IN D
1415 6362 0573 COM 2,S FOR LOOK AT LOWER BITS
1417 E462 0574 ANDB 2,S COUNT MASK
1419 3401 0575 PSHS CC PRESERVE TEST RESULT
141B 5363 0576 COM 3,S PUT IT BACK AS IT WAS
141D 3501 0577 PULS CC RECOVER TEST RESULT
141F 26D4 0578 BNE INLOOP NOT AT END OF LINE YET
0579 *
0580 *
1421 C602 0581 LDB #2 SPACES

```

```

1423 8D6B 0582 BSR SPACES OUTPUT THEM
1425 E662 0583 LDB 2,S GET COUNT AS NEG
0584 * B NGM HAS -16 IN IT
1427 30A5 0585 CHLOOP LEAX B,Y GET FIRST ADDR. IN GROUP
1429 3404 0586 PSHS B SAVE COUNT
142B 1F10 0587 TFR X,D PUT IT TO EPROM
142D B7FF46 0588 STA HIADD
1430 F7FF44 0589 STB LOWADD
1433 3504 0590 PULS B RECOVER COUNT
1435 B6FF40 0591 LDA DATARG GET EPROM DATA
1438 8120 0592 CMPA #32 HIGHER TO PRINT
143A 2509 0593 BLO DOT CHANGE TO DOT LESS THAN 3
143C 7D006F 0594 TST DEVNUM TO PRINTER?
143F 2706 0595 BEQ OKPRNT TO SCREEN SO OK
1441 B100 0596 CMPA #000 HIGHEST PRINTABLE?
1443 2502 0597 BLO OKPRNT
1445 862E 0598 DOT LDA #, REPLACE WITH DOT
1447 AD9FA002 0599 OKPRNT JSR [CHROUT]
1449 5C 0600 INCB
144C 2DD9 0601 BLT CHLOOP GOES ZERO WHEN DONE
0602 ****
0603 *ARE WE AT END YET?
144E 10ACE4 0604 CMPY ,S END ON STACK
1451 2228 0605 BHI DMPXT YES SO EXIT LOOP
0606 *
1453 7D006F 0607 TST DEVNUM TO PRINTER?
1456 2700 0608 BEQ NXLIN DELETE SCREEN LINE COUNT
145B AD9FA000 0609 JSR [POLCAT] BREAK PRESSED?
145C B103 0610 CMPA #3 BREAK
145E 2710 0611 BEQ DMPXT EXIT ROUTINE
1460 16FF7E 0612 LBRA DNLOOP CONTINUE OUTPUT
0613 *
1463 7A01D3 0614 NXLIN DEC COUNT LINE COUNTER
1466 1026FF77 0615 LBNE DNLOOP NOT DONE YET
146A B510 0616 LDA #16 RESE LINE COUNT
146C B701D3 0617 STA COUNT
146F AD9FA000 0618 DWAIT JSR [POLCAT] WAIT FOR KEY
1473 27FA 0619 BEQ DWAIT NO KEY YET
1475 B103 0620 CMPA #3 IS IT BREAK
1477 1025FF66 0621 LBNE DNLOOP NOSO CONTINUE
0622 *
147B 3263 0623 DMPXT LEAS 3,S CLEAN STACK

```

Marymac
INDUSTRIES

Pan American
ELECTRONICS

800-231-3680

800-531-7466

Radio Shack TRS-80's®

People you Trust to give you the very best!



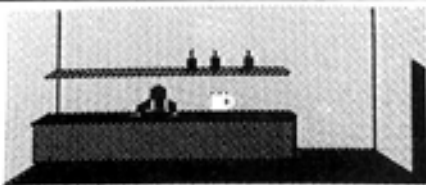
- Lowest Price
- Reliable Service
- Quality Products

22511 Katy Fwy., Katy (Houston) Texas 77450
(713) 392-0747 Telex 774132

HI-RES GRAPHIC ADVENTURES

DISC NOT REQUIRED

Cassettes—\$24.95/Disc—\$27.95



You are inside a small pub.

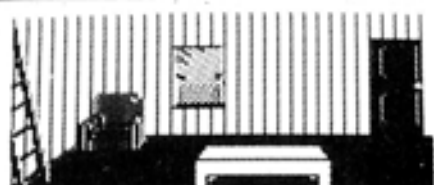
Obvious exits are West.

You see: a sign on the bar, the barkeep, small groups of customers, a glass of beer.

OK.

SHENANIGANS

Countless legends tell of a magnificent Pot of Gold hidden at the end of the rainbow. Many have attempted to find the marvelous treasure but success has eluded them and it remains hidden to this day. You, as a dedicated adventurer, have determined to search for the fabled gold and succeed where others have failed. This one is great fun! 32K required.



You are in a beach house.

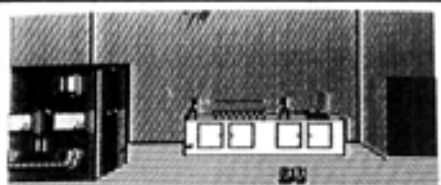
You see: a table, a chair, a ladder, a broken window.

OK.

SEA SEARCH

Get your shark repellent and scuba tanks ready! The graphics in this adventure are truly outstanding and the under water scenes are unforgettable. You'll run into a pirate, a mermaid and some hungry sharks in this colorful and unique treasure hunt. 32K required.

Hot CoCo—April, '84. "The fine graphics accent your imagination..."



I'm in the Professor's secret laboratory filled with complex machinery and test equipment.

I see: an unusual looking device, a passageway, a pair of hiking boots.

OK.

CALIXTO ISLAND

A valuable museum treasure has been stolen, can you recover it??? This is a challenging puzzle with an occasional twist of humor. You'll visit a secret laboratory, a Mayan pyramid and you'll meet crazy Trader Jack—all in living color and exciting detail. You will really love this hi-res graphic version of the classic Calixto Island Adventure. 32K required.

Rainbow—April, '84. "It was enough to keep my wife and 8 year old son glued to the computer for an entire weekend and two week nights..."



I'm in rugged mountain country. Snow is falling.

Obvious directions: North, South, West.

I see: pine trees, a cabin in the distance.

OK.

BLACK SANCTUM

Encounter the forces of black magic as you roam around an old 18th century monastery. You'll see all the evil locations in this spooky adventure, you'll love searching out and destroying the evil in this classic tale. A MUST for every adventure game fan! 32K required.

Rainbow—May, '84—"It's the graphic screens that are the shining stars..." "Some of the best I've seen."

MD

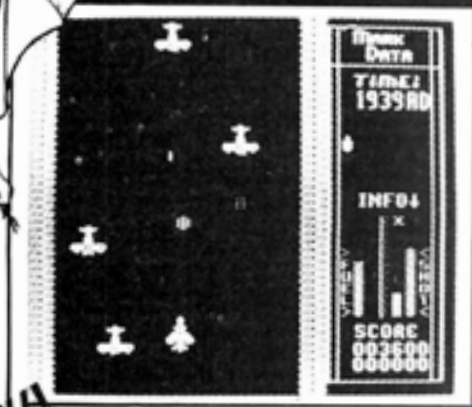
Mark Data Products

24001 ALICIA PKWY., NO. 207 • MISSION VIEJO, CA 92691 • (714) 768-1551

FREE—Send for our new, 24 page catalog.

NEW! ARCADE GAMES

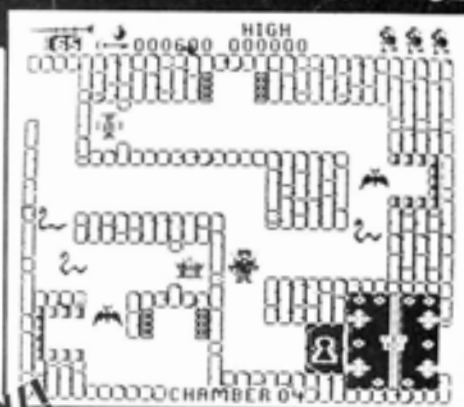
Cassettes—\$24.95/Disc—\$27.95



TIME FIGHTER

Pilot your MD-64 fighter through a hazardous time tunnel. Your mission is to destroy the dreaded Time Guardian who threatens the natural order of the universe. In order to reach this menace you must fight aerial dangers from strange and different time zones. If you like fast action, this one's for you! 16K required.

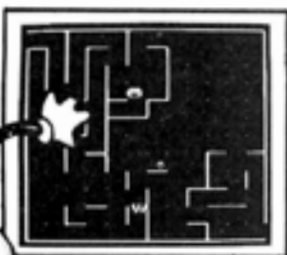
Rainbow—March, '84. "One of the best in your library of computer games. It is a real gem."



TUT'S TOMB

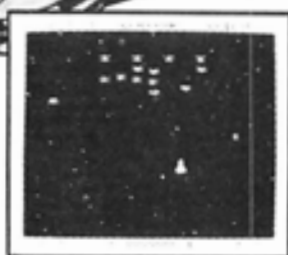
Explore the ancient, mystical tomb of the great Pharaoh. Find the magical keys which lead you to unbelievable treasures as you out maneuver the creatures that slither and swarm about you. Super fast arcade action—this one will knock your socks off with 16 screens of incredible color and sound. Fabulous! 32K required.

Hot CoCo—April, '84. "State-of-the-art CoCo graphics. A first rate game."



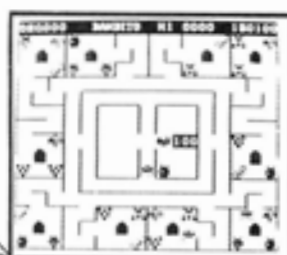
BUMPERS

A truly great maze game. Especially exciting when two players compete simultaneously. Tension mounts as you wildly race through a hidden obstacle course. Barrier walls are invisible until you bump into them and you must proceed cautiously as each dead end has a hidden booby trap. 16K required.



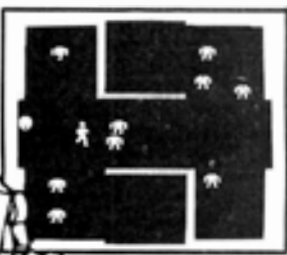
GLAXXON

Pit your playing skill against squadrons of swooping, diving spacecraft. Fast and furious with seven selectable skill levels and automatic game acceleration... guaranteed to blister your joystick finger. The object of the game is to achieve the highest score by eliminating as many attacking spacecraft as possible while avoiding your own destruction. Dynamite! 16K required.



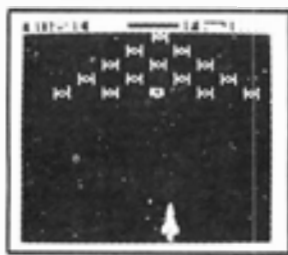
EL BANDITO

El Bandito has to be a crafty little hombre to stay alive as he loots the local countryside. Escape into a tunnel to avoid that angry spider. race around the corner towards your lair. Two players may compete simultaneously in this unusual game. Selectable skill levels provide a challenge for beginners as well as experts. 16K required.



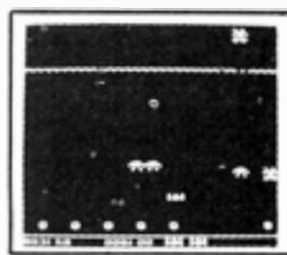
HAYWIRE

The best Color Computer version of the Berzerk arcade game you can get. A real hi-res classic! This challenging combination of angry robots and the evil menace will provide many hours of fun and excitement. Haywire combines joystick and fire button action and is great as a two player game. 16K required.



ASTRO BLAST

Wave after wave of alien attackers—each one different and unique. A great space "shoot-em-up" with hi-res graphics, lots of color and dramatic sound effects. Three selectable skill levels coupled with automatic game acceleration provide a challenge for novice and expert alike. One of our all time best sellers! 16K required.



COSMIC CLONES

Clonial Warriors, Super Klones, Double Bombs and "the Death Layer" relentlessly challenge the most skillful player in this unique, fast action game. Your goal is to achieve the highest score by eliminating the Clonial invasion forces thus protecting your starbase fuel cells. Fast Fun! One of our favorites. 16K required.

Mark Data Products

All Orders: Please add \$2.00 shipping and handling in the continental U.S. All others, add air shipping and \$3.00 handling. California residents add 6% sales tax. Foreign orders please remit U.S. funds. Software authors—Contact us for exciting program marketing details. We accept MasterCard and VISA. Distributed in Canada by Kelly Software.

```

147D 060D      0624      LDA      #00D      CR AT END
147E AD9FA002  0625      JSR      [CHROUT]  ALSO CLRS BUFFER IN PRINTER
1483 7F006F     0626      CLR      DEVNUM    RESET TO SCREEN
1486 30BDFE19  0627      LEAX    EKEY,PCR  ENTER MESSAGE
148A 178332     0628      LBSR   INPUT#     *
148D 15FEE4     0629      LBRA   DUMP        RESTART
                    0630 *****
                    0631 *****
                    0632 * OUTPUT SPACES BY COUNT IN 0
1490 0620      0633 SPACES LDA #32 SPACE
1492 AD9FA002  0634 OLOOP JSR [CHROUT]
1495 5A         0635      DECB          COUNT
1497 1026FFF7  0636      LBNE   OLOOP
1498 39         0637      RTS
                    0638 *****
149C 20        0639 DMPTTL FCC /      EPROM DUMP#
149D 20202020  0640 *****
14A2 20202020  0641 *****
14A7 4550524F  0642 *****
14AC 204454D5  0643 *****
14B1 0D        0644 *****
14B2 20        0645 *****
14B3 20202020  0646 *****
14B8 20202020  0647 *****
14BD 30303030  0648 *****
14C2 30303030  0649 *****
14C7 0D0D0D   0650 *****
14CA 53        0642 DSTRT FCS /START ADDRESS: /
14CB 5441525420 0643 ESTRT FCS / END ADDRESS: /
14D0 4144445245 0644 DEV FCS /(P)rinter or (S)creen? /
14D5 53533A2000 0645 *****
14DA 20        0646 *****
14DB 20454E4420 0647 * INSPECTS AND PROGRAMS INDIVIDUAL CELL *
14E0 4144445245 0648 *****
14E5 53533A2000 0649 *
14EA 20        0650 *
14EB 502972596E 0651 CELL JSR CLEAR SCREEN
14F0 746572206F 0652 LEAX CELMSG,PCR BET TITLE
14F5 7220285329 0653 LBSR OUTST#
14FA 637265656E 0654 *
14FF 3F2000    0655 * SET COUNT TO 1 AND START TO RAM LOCATION
                    0656 * TO STORE ENTERED DATA
0107          0657 TEMP EQU TARGET+2
                    0658 *
150C CC0001    0659      LDD #1 SET COUNT
150F F001D3    0660      STD COUNT
1512 CC01D7    0661      LDD #TEMP GET TEMPORARY ADDRESS
1515 F001D1    0662      STD START PUT IT AS START
                    0663 *
1518 7F01D5    0664 * CLEAR TARGET TO ZERO
151B 7F01D6    0665      CLR TARGET
151D 7F01D6    0666      CLR TARGET+1
151E 7F01D6    0667 *
                    0668 * DISPLAY DATA AT TARGET LOCATION
151E B60D      0669 DISDAT LDA #0D CR
1520 AD9FA002  0670 JSR [CHROUT] MOVE DOWN A LINE
1524 FC01D5    0671 LDD TARGET GET EPROM ADDRESS
1527 07FF46    0672 STA HIADD
152A F7FF44    0673 STB LOWADD
                    0674 *
152D 1F01      0675 TFR D,X DISPLAY ADDRESS
152F 170299    0676 LBSR HEXOUT
1532 1700BB    0677 LBSR MOVCRS PLACE TO RIGHT
                    0678 *
1535 F6FF40    0679 LDB DATARG GET EPROM DATA
                    0680 *
1538 3404      0681 PSHS # SAVE VALUE
153A 170262    0682 LBSR HXPAIR MAKE IT A HEX STRING
153D BE01DA    0683 LDX #BUFFER
1540 170218    0684 LBSR OUTST# OUTPUT CONTENTS
                    0685 *
1543 1700AA    0686 LBSR MOVCRS MAKE A SPACE
1546 3502      0687 PULS A RECOVER CHARACTER IN A
1548 B120      0688 CMPA #32 LOWEST PRINTABLE CHARACTER
154A 2202      0689 BHI CHARCT OUTPUT AS A CHARACTER
154C 862E      0690 LDA #. REPLACE BY A DOT
154E AD9FA002  0691 CHARCT JSR [CHROUT]
1552 17009B    0692 LBSR MOVCRS OVER A PLACE
1555 863F      0693 LDA #? PROMPT
1557 AD9FA002  0694 JSR [CHROUT] DISPLAY IT
                    0695 *
155B AD9FA000  0696 GTKEY JSR [POLCAT] BET RESPONSE
155F 27FA      0697 BEQ GTKEY WAIT FOR KEY
                    0698 *
                    0699 * VALID RESPONSES ARE:
                    0700 * ^ PREVIOUS ADDRESS (WRAPS AROUND)
                    0701 * DOWN ARROW NEXT ADDRESS DITTO
                    0702 * N ENTER NEW ADDRESS
                    0703 * P PROGRAM THIS ADDRESS
                    0704 * X EXIT TO MENU
                    0705 *****
                    0706 *
1561 815E      0707 CMPA #95E UP ARROW
1563 2610      0708 BNE .DARROW
1565 FC01D5    0709 LDD TARGET GET TARGET VALUE
1568 B30001    0710 SUBD #1 REDUCE BY ONE
156B 2A03      0711 BPL NOTNEG NO NEED TO WRAP
156D CC1FFF    0712 LDD #TOPADD TO WRAP ADDRESS
1570 FD01D5    0713 NOTNEG STD TARGET
1573 20A9      0714 BRA DISDAT DISPLAY IT
                    0715 *****
1575 B10A      0716 .DARROW CMPA #00A DOWN ARROW
1577 2613      0717 BNE NEWADD GET NEW ADDRESS FOR TARGET
1579 FC01D5    0718 LDD TARGET
157C C30001    0719 ADDD #1 INCREASE IT
157F 10032000 0720 CNPD #NUMX+1024 WRAPPED AROUND?
1583 2602      0721 BNE NTOVER
1585 4F        0722 CLRA
1586 5F        0723 CLR0
1587 FD01D5    0724 NTOVER STD TARGET
158A 2092      0725 BRA DISDAT
                    0726 *****
158C 014E      0727 NEWADD CMPA #0N ENTER A NEW ADDRESS
158E 261A      0728 BNE NEWDAT
1590 30B000AC 0729 TOOHIGH LEAX NADDRS,PCR GET NEW ADDRESS MESSAGE
1594 17022B    0730 LBSR INPUT# GET NEW VALUE
1597 1701C6    0731 LBSR HEXINT GET VALUE IN X
                    0732 * CHECK 0 FOR ERROR CODE B<0> FOR ERROR
159A 5D        0733 TSTB
159B 1026FF7F 0734 LBNE DISDAT BAD SO DO NOTHING
159F BC1FFF    0735 CMPI #TOPADD MUST NOT BE HIGHER THAN THIS
15A2 22EC      0736 BHI TOOHIGH
15A4 BF01D5    0737 STX TARBET
15A7 16FF74    0738 LBRA DISDAT 00 DISPLAY IT
                    0739 *****
15AA 0150      0740 NEWDAT CMPA #0P PROGRAM THE LOCATION
15AC 2630      0741 BNE DEXIT EXIT ROUTINE
15AE 30B0009D 0742 TOBIG LEAX NDATA,PCR HEW DATA MESSAGE
15B2 17020A    0743 LBSR INPUT# GET DATA
15B5 1701AB    0744 LBSR HEXINT GET VALUE IN X
                    0745 * TEST 0 FOR ERROR CODE B<0> FOR ERROR
15B8 5D        0746 TSTB
15B9 1026FF61 0747 LBNE DISDAT DO NOTHING
15BD BC00FF    0748 CMPI #0FF HIGHEST ALLOWED DATA
15C0 22EC      0749 BHI TOBIG
15C2 1F10      0750 TFR X,D
15C4 F701D7    0751 STB TEMP FOR PROGRAMMING
15C7 17F99E    0752 LBSR PROGRAM TRY TO PROGRAM IT
15CA 5D        0753 TSTB
15CC 1027FF4F 0754 LBNE DISDAT ALL OK
15CF C101      0755 CMPB #1 NOT ERASED
15D1 2710      0756 BEQ NOERSD NOT ERASED
                    0757 ***
                    0758 * BAD EPROM
15D3 30B0FD75 0759 LEAX BADPRM,PCR
15D7 1701B1    0760 .WRITE LBSR OUTST#
15DA BE01D5    0761 LDX TARGET
15DD 1701EB    0762 LBSR HEXOUT
15E0 16FF30    0763 LBRA DISDAT
                    0764 *
15E3 30B0FD54 0765 NOERSD BRA UNERSD,PCR UNERASED MESSAGE
15E7 20EE      0766 LBSR .WRITE
                    0767 *****
15E9 0150      0768 DEXIT CMPA #0X IS IT EXIT?
15EB 1026FF2F 0769 LBNE DISDAT NO SO REDISPLAY
15EF 39        0770 RTS RETURN TO MENU
                    0771 *****
                    0772 *****
                    0773 * THIS MOVES CURSOR 1 RIGHT IF NOT AT END OF SCREEN
15F0 FC0000    0774 MOVCRS LDD CURLOC
15F3 100305FF 0775 CNPD #05FF AT END?
15F7 2706      0776 BEQ ATEND
15F9 C30001    0777 ADDD #1 MAKE A SPACE
15FC F00000    0778 STD CURLOC
15FF 39        0779 ATEND RTS
                    0780 *****
1600 20        0781 CELMSG FCC / INDIVIDUAL CELL PROGRAMMING/

```

```

1601 2020494E44
1606 4956494455
1608 414C204345
1610 4C4C205052
1615 4F4752414D
161A 4D494E47
161E 00 0702 FCB 00D
161F 20 0703 FCS / *****<0D><0D>/
1620 202043D3D3D
1625 3D3D3D3D3D
162A 3D3D3D3D3D
162F 3D3D3D3D3D
1634 3D3D3D3D3D
1639 3D3D3D3D0D
163E 0D00
1640 0D 0704 NADDRS FCS /<0D>NEW ADDRESS? /
1641 4E45572041
1646 4444524551
1648 533F2000
164F 0D 0705 NDATA FCS /<0D>NEW DATA? /
1650 4E45572044
1655 4154413F20
165A 00
0706 *****
0707 * RETURNS CASSETTE FILE DATA
0708 *****
0709 *
0790 * THIS RETURNS THE ADDRESSES OF THE LAST CLOADM
0791 *
0792 *
01E7 0793 STADD EQU 407 START ADDRESS
007E 0794 ENDADD EQU 126 END ADDRESS
01E5 0795 EXECAD EQU 405 EXEC ADDRESS
0796 *
1650 0DA920 0797 CFILE JSR CLEAR SCREEN
165E 300D0035 0798 LEAX FILMSG,PCR HEADING
1662 1700F6 0799 LBSR OUTST0
1665 0E01E7 0800 LDX STADD GET START ADDRESS
1668 170100 0801 LBSR HEXOUT OUTPUT IT
0802 *
1668 300D0071 0803 LEAX ENDTIT,PCR GET END MESSAGE
166F 1700E9 0804 LBSR OUTST0
1672 9E7E 0805 LDX ENDADD
1674 301F 0806 LEAX -1,X MOVE TO ACTUAL END
1676 170152 0807 LBSR HEXOUT
0808 *
1679 300D0078 0809 LEAX EXEMSG,PCR GET EXE MESSAGE
167D 170000 0810 LBSR OUTST0
1680 0E01E5 0811 LDX EXECAD
1683 170145 0812 LBSR HEXOUT
0813 *
0814 * MOVE CURSOR DOWN 2 LINES
1686 FC0000 0815 LDD CURLOC
1689 C30020 0816 ADD #32
169C FD0000 0817 STD CURLOC
0818 *
168F 3000FC10 0819 LEAX EKEY,PCR GET ENTER MESSAGE
1693 170129 0820 LBSR INPUT0
0821 *
1696 39 0822 RTS
0823 *****
1697 20 0824 FILMSG FCC / CASSETTE FILE DATA/
1698 2020202020
169D 2043415353
16A2 4554544520
16A7 46494C4520
16AC 44415441
16B0 0D 0825 FCB 00D
16B1 20 0826 FCC / *****/
16B2 2020202020
16B7 203D3D3D3D3D
16BC 3D3D3D3D3D3D
16C1 3D3D3D3D3D3D
16C4 3D3D3D3D3D
16CA 000D 0827 FDB 00D0D
16CC 20 0828 FCS / START ADDRESS: /
16CD 2020205354
16D2 4152542041
16D7 4444524553
16DC 533A2000
16E0 0D 0829 ENDTIT FCS /<0D> END ADDRESS: /
16E1 2020202020
16E6 20454E4420
16EB 4144445245
16F0 53533A2000
16F5 0D 0830 EXEMSG FCS /<0D> EXECUTE ADDRESS: /
16F6 2020455045
16FB 4355544520
1700 4144445245
1705 53533A2000

```

```

0831 *****
0832 *****
0833 * UTILITY LIBRARY #
0834 *****
0835 *****
0836 *INSTR* GETS A STRING FROM KEYBOARD AND PUTS*
0837 *IT INTO "BUFFER" TERMINATED BY A ZERO BYTE.*
0838 *****
0839 * BASIC POINTERS
0840 CURLOC SET 08B CURSOR LOCATION
A000 0841 POLCAT SET 0A000 KEYBOARD POLL
A002 0842 CHRPUT SET 0A002 CHARACTER OUTPUT
085F 0843 DEVNUM SET 06F # FOR SCREEN, -2 FOR PRINTER
0844 *****
0845 *****
170A 100E01DA 0846 INSTR0 LDY 08BUFFER POINT Y TO BUFFER START
170E 0D40 0847 CRSR BSR CURSOR PUT BLACK SQUARE UP
1710 AD9FA000 0848 GETKEY JSR [POLCAT] LOOK FOR KEY
1714 27FA 0849 BEQ GETKEY NOTHING ENTERED YET
1716 0100 0850 CMPA 00B BACKSPACE
1719 2417 0851 BNE CHKRET
171A 100C01DA 0852 CMPY 08BUFFER AT START OF BUFFER
171E 27EE 0853 BEQ CRSR NO BACKSPACE POSSIBLE
1720 0560 0854 LDA 0560 BLANK
1722 277F0000 0855 STA [CURLOC] STORE AT CURRENT LOCATION
1726 313F 0856 LEAY -1,Y DECREASE CURSOR LOCATION
1728 0C00 0857 LDD CURLOC GET CURSOR LOCATION
172A 030001 0858 SUBD #1 REDUCE D BY ONE
172D 0D00 0859 STD CURLOC RESET CURSOR LOCATION
172F 200D 0860 BRA CRSR
0861 *
0862 * IF CR THEN PUT INTO BUFFER, WITH A ZERO BYTE
0863 * THEN EXIT
1731 0101 0864 CHKRET CMPA 000D CARRIAGE RETURN
1733 2609 0865 BNE INKEY NO SO PUT INTO BUFFER
1735 A7A0 0866 STA ,Y+ PUT CR INTO BUFFER
1737 AD9FA002 0867 JSR [CHRPUT] PUT RETURN ON SCREEN
1739 6FA4 0868 .EXIT CLR # SET LAST BYTE TO ZERO
173D 39 0869 RTS
0870 *
0871 * PUT CHARACTER INTO BUFFER, CHECK FOR
0872 * SPACE FIRST. IF BUFFER HAS 254 PUT IT
0873 * THEN SET 256 BYTE TO ZERO AND EXIT
0874 *
173E 0120 0875 INKEY CMPA 032 FIRST PRINTABLE CHARACTER
1740 25CC 0876 BLD CRSR NOT PRINTABLE SO LOOP
1742 A7A0 0877 STA ,Y+ PQ INTO BUFFER
1744 AD9FA002 0878 JSR [CHRPUT] OUTPUT ENTERED CHARACTER
1749 100C0200 0879 CMPY 08BUFFER+254 BUFFER FULL?
174C 25CC 0880 BLD CRSR NOT FULL
174E 20E0 0881 BRA .EXIT
0882 *
0883 * CURSOR ROUTINE
1750 0600 0884 CURSOR LDA #120 BLACK SQUARE
1752 A79F0000 0885 STA [CURLOC]
1756 39 0886 RTS
0887
0888 *****
0889 *****
0890 *OUTST0 TAKES A STRING POINTED TO BY REG X *
0891 *AND PUTS IT TO OUTPUT DEVICE. TERMINATED *
0892 *BY A ZERO BYTE IN BUFFER #
0893 *****
0894 * BASIC POINTER
A002 0895 CHRPUT SET 0A002 OUTPUT ROUTINE
0896 *
1757 AD9FA002 0897 .DSPLY JSR [CHRPUT] OUTPUT CHARACTER
1758 A600 0898 OUTST0 LDA ,X+ GET CHARACTER
175D 26F0 0899 BNE .DSPLY DISPLAY IF NOT ZERO
175F 39 0900 RTS
0901 *****
0902 *****
0903 *HEXINT GETS A HEX NUMBER FROM BUFFER AND *
0904 *PUTS IT IN REG X. REG B IS ZERO IF NO *
0905 *ERROR. WILL GET FIRST 4 CHARACTERS IN #
0906 *BUFFER OR TO <CR> OR ZERO BYTE #
0907 *****
0908 *****
1760 109E01DA 0909 HEXINT LDY 08BUFFER POINT Y TO BUFFER
1764 BE0000 0910 LDX 00 CLEAR X FOR NUMBER
1767 060A 0911 LDA #4 CHARACTER COUNTER
1769 E5A0 0912 GTHEX LDB ,Y+ GET CHARACTER FROM BUFFER
176B 271E 0913 BEQ HEXIT AT END OF BUFFER
176D C100 0914 CNPB 000D IS IT A <CR>?
176F 271A 0915 BEQ HEXIT YES SO AT END
1771 C130 0916 CNPB 010 IS IT LESS THAN 0?
1773 252A 0917 BLD HEXERR NO SO ERROR
1775 C139 0918 CNPB #9 GREATER THAN 9
1777 2214 0919 BHI ALPHA MAY BE A - F
1779 C030 0920 SUBS 010 MAKE A NUMBER

```

1778 1E01	0921 *				179F 10BE01DA	0954 HXPAIR LDY	0BUFFER	POINT TO BUFFER
	0922 *B NOW HAS VALUE ENTERED					0955 * GET HIGH NIBBLE FROM B		
	0923 HEX EXG D,X	SWAP REGISTERS FOR SHIFT			17A3 1F9S	0956 TFR B,A	INTO A	
0004	0924 * SHIFT D LEFT 4 PLACES				0004	0957 RPT 4	MOVE DOWN 4 PLACES	
	0925 RPT 4					0958 LSRA		
	0926 ASLB					0959 ENDR		
	0927 ROLA				17A5 44	LSRA		
	0928 ENDR				17A6 44	LSRA		
177D 58	+ ASLB				17A7 44	LSRA		
177E 49	+ ROLA				17A8 44	LSRA		
177F 58	+ ASLB					0960 *		
1780 49	+ ROLA				17A9 8D09	0961 BSR HEXASC		
1781 58	+ ASLB				17AB 1F98	0962 TFR B,A	GET LOW NIBBLE	
1782 49	+ ROLA				17AD 840F	0963 ANDA #0F	GET LOW 4 BITS	
1783 58	+ ASLB				17AF 8D03	0964 BSR HEXASC	CONVERT AND STORE	
1784 49	+ ROLA				1781 6FA4	0965 CLR Y	SET NEXT BUFFER LOCN TO 0	
1785 1E01	0929 EXG D,X	PUT IT BACK INTO X			1783 39	0966 RTS		
1787 3A	0930 ABX	ADD VALUE INTO REGISTER X				0967 *		
1788 4A	0931 DECA					0968 *		
1789 26DE	0932 BNE GTHX					0969 * HEX TO ASCII CONVERSION ROUTINE		
178B 5F	0933 HEXIT CLR				1784 8109	0970 HEXASC CMPA 09	IS DATA 9 OF LESS?	
178C 39	0934 RTS				1786 2302	0971 BLS ASCZ		
	0935 **				1788 8B07	0972 ADDA #'A-'9-'1	NO,ADD OFFSET FOR LETTERS	
178D C141	0936 ALPHA CMPB #'A	LESS THAN 'A'			178A 8B30	0973 ASCZ ADDA #'0	CONVERT DATA TO ASCII	
178F 2508	0937 BLD HEXERR	YES 50 ERROR			178C A7A0	0974 STA ,Y+	PUT INTO BUFFER	
1791 C146	0938 CMPB #'F	HIGHER THAN 'F'			178E 39	0975 RTS		
1793 2204	0939 BHI HEXERR	YES 50 ERROR				0976 *		
1795 C037	0940 SUBB #'A-10	SET TO VALUE				0977 *		
1797 20E2	0941 BRA HEX					0978 *		
	0942 **					0979 * INPUT* OUTPUTS A STRING POINTED TO BY REG *		
1799 C601	0943 HEXERR LDB 01					0980 * X, THEN RECEIVES A STRING FROM KEYBOARD *		
179B 8E0000	0944 LDX 00					0981 * AND PUTS IT INTO *BUFFER* TERMINATED WITH *		
179E 39	0945 RTS					0982 * A ZERO. IF X IS ZERO NO STRING IS OUTPUT.*		
	0946 *****					0983 * MAX. CHARACTERS IN BUFFER IS 255. #		
	0947 *****					0984 *****		
	0948 * HXPAIR CONVERTS CONTENTS OF REG B INTO A *					0985 * BASIC POINTERS		
	0949 * STRING IN BUFFER TERMINATED BY A ZERO #				0098	0986 CURLOC SET 99B	CURSOR LOCATION	
	0950 * BYTE. NO <CR> IS ADDED TO THE STRING #				A000	0987 POLCAT SET 9A000	KEYBOARD POLL	
	0951 *****				A002	0988 CHRUT SET 9A002	CHARACTER OUTPUT	
	0952				006F	0989 DEVNUM SET 96F	# FOR SCREEN, -2 FOR PRINTER	
	0953 *					0990 *****		

One Stop Shopping For The Color Computer

Reitz Super Disk Charger

Now you can run R.S. DOS at up to 6 times the speed of normal DOS. Diskcharger will work with single or double sided disk drives at step rates up to 6 ms. and in both DOS 1.0 and 1.1 with no equipment modification. Step up to first class color computer operation today with the Reitz Super Disk Charger.

64K COLOR COMPUTER \$21⁹⁵

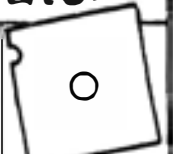
New From Reitz

- Reitz Serial to Parallel Interface **\$59.95**
- Three Way RS-232 Switcher with power light **\$34.95**
- Serial to Parallel Interface/Switcher with indicator lights **\$79.95**

New From Sugar Software

COCO CALLIGRAPHER...Prints Old English, Gay Nineties, and Cartoon fonts on your EPSON, GEMINI, OKIDATA, LPVII, OR DMP-100 PRINTER.

32KECB...TAPE...\$24.95 DISK...\$29.95



FLIP IT!
Punch
your
Disks
for
double
the
storage

\$9.95

1-(800)-242-2626 (Outside Ohio)

1-(419)-537-8937 (Computer Order Line)



3170 W. Central
Westgate Meadows Shopping Center
Toledo, Ohio 43606
1-419-537-1432



Please include phone number with all orders. Include \$5.00 for all hardware orders and \$2.00 for all software orders. Ohio residents please add 6% state sales tax.


```

0991 *
0992 *
17BF BC0000 0993 INPUT# CHPY 00 ANY TEXT TO OUTPUT
17C2 2703 0994 BEQ NOTEXT
17C4 17FF94 0995 LBSR OUTST# OUTPUT TEXT STRING
17C7 17FFA0 0996 NOTEXT LBSR INSTR# GET INPUT STRING
17CA 39 0997 RTS
0998 *
0999 *
1000 *
1001 *****
1002 *****
1003 * HEXOUT TAKES CONTENTS OF X AND PUTS IT ON *
1004 * SCREEN. USES HXPAIR TO DO IT IN 2 PARTS *
1005 * OUTST# IS ALSO USED #
1006 *****
1007
17CB 1F10 1008 HEXOUT TFR X,C PUT DATA INTO REG D
17CD 1E89 1009 EXG A,B PUT HIGH BYTE IN B
17CF 17FFC0 1010 LBSR HXPAIR PUT INTO SCREEN
17D2 3410 1011 PSHS X PRESERVE VALUE
17D4 BE010A 1012 LDX #BUFFER POINT TO START OF STRING
17D7 17FFB1 1013 LBSR OUTST# PUT OUT THE STRING
17DA 3506 1014 PULS D RECOVER VALUE IN D
17DC 17FFC0 1015 LBSR HXPAIR PUT LOW BYTE ON SCREEN
17DF 8E01DA 1016 LDX #BUFFER POINT TO START OF STRING
17E2 17FF76 1017 LBSR OUTST# PUT OUT THE STRING

```

```

EPR0M.MAC COMPUTERWARE MACRO ASSEMBLER PAGE 22
2764 EPR0M PROGRAMMER BY C.J.STEARMAN (C)1984

```

```

17E3 39 1018 RTS
1019 *
1020 *****
1021 *
1022 TTL 2764 EPR0M PROGRAMMER BY C.J.STEARMAN (C)1984
1023 NAM EPR0M.MAC
1024 *
0E00 1025 END EPR0M
NO ERROR(S) DETECTED

```

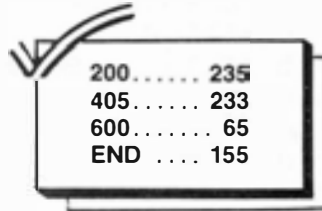
SYMBOL TABLE:

.BASIC 0E45	.CELL 0E33	.COPY 0E21	.DARRD 1575
.DSPLY 1757	.DUMP 0E2A	.EXIT 173B	.FILE 0E3C
.KEY 110B	.LEAVE 1202	.WRITE 15D7	ADDHMT 1100
ALPHA 178D	ASC2 178A	ATEND 15FF	BAD 1107
BADLOC 120D	BADPRM 134C	BUFFER 010A	CELL 1502
CELM5G 1600	CEXIT 115B	CFILE 165B	CHARCT 154E
CHKRET 1731	CHLOOP 1427	CHRQUT A002	CLEAR A928
CLINES FF42	CLRREG 0F42	CONREG FF41	COPY 1140
COUNT 0103	CPYTTL 1213	CRSR 170E	CURLOC 009B
CURSOR 1750	DATARB FF40	DCOMT 13BD	DDRSET 0F5D
DERROR 11B7	DEV 14EA	DEVNUM 006F	DEIT 15E9
DIFF 124B	DISDAT 151E	DLOOP 102E	DNLOOP 13E1
DMPCTL 149C	DMPIT 147B	DONEYT 10BE	DOT 1445
DOWNCT 1005	DSTRT 14CA	DUMP 1374	DWAIT 146F
EKEY 12A3	EMPTY 0FC4	ENDADD 007E	ENDMSB 12D6
ENDTIT 16E0	EPR0M 0E00	ERANSB 10C2	ERASE 103E
ESTRT 14DA	EXECA0 01E5	EXEM5G 16F5	FILMSG 1697
GDUMP 13B0	GETKEY 1710	GETST 1159	GOOD 1111
GOODP1 1304	GOODP2 1320	GOODPR 11E2	GTHEX 1769
GTKEY 155B	HEX 177B	HEXASC 17B4	HEXERR 1799
HEXINT 1760	HEXIT 178B	HEXOUT 17CB	HIADD FF46
HIGH 126C	HXPAIR 179F	INIT 0F3B	INKEY 173E
INLOOP 13F5	INPUT# 17BF	INSTR# 170A	LOWADD FF44
MENU 0E03	MENUT 0E4D	MOVCRS 15F0	NADDRS 1640
NDATA 164F	NEWADD 158C	NEWDAT 15AA	NBRK 106F
NODISP 0FB5	NDERSD 15E3	NOROOM 11D1	NOTEXT 17C7
NOTMTY 109A	NOTNEB 1570	NROOM 12B7	NTOVER 1587
NUMK 0000	NXLINE 1463	OKPRNT 1447	OLOOP 1492
OUTST# 175B	PEXIT 100A	PLOOP 0F91	POLCAT A000
PREXIT 1015	PR0GM 0F6B	PULSE 101F	PUTNM 1059
RLYDLY 0F73	RSTART 13AC	SCR 13CF	SPACES 1490
STADD 01E7	START 01D1	STARTX 12BF	TARGET 0105
TEMP 01D7	TGTMSG 12ED	T0BIG 15AE	TOOH1 11CB
TOOH1G 1590	TOPADD 1FFA	UNERAS 11FE	UNERSD 133B
VERFY 111D	VERIOX 0FFA	VEKIT 10B3	VLOOP 1063
VOLTS FF43	WRKADD 135C	WARG 0000	

CMD#3: EPR0MSRC.SRC /P

Correction for Cooking With Coco:

In the July installment, Listing 1 (BASLOAD) was inadvertently left out. Listing 2 and 3 were labeled 1 and 2. Here is last month's Listing 1 (which is also on the August RAINBOW on Tape):



Listing 1 (BASLOAD):

```

10 ' THIS WILL TRANSFER BASIC
20 ' EXTENDED BASIC AND DISK
30 ' BASIC TO ROM
40 ' CORRECT IT, THEN
50 ' COLD START IT.
60 ' IT WILL WORK WITH OR WITHOUT
T
70 ' EXTENDED BASIC OR DISK BASIC
C
80 ' IN ROM
90 ' NOTE: For Color Basic 1.1 on ly.
100 ' Revs of Ext. and Disk not important
110 CLEAR 200,32511
120 DATA 32512,41044,41092
130 ' RELOCATION PROGRAM
140 DATA 26,80,142,128,0,166,132,183,255,223,167,128,140,224,0,39,5,183,255,222,32,239,28,175,57
150 ' PATCH #1
160 DATA 198,13,189,160,137,18,18
170 ' PATCH #2
180 DATA 142,127,254,32,10,167,193,90,38,251,206,255,224,57
190 READ S1,S2,S3
195 TT=S1+S2+S3
200 ' LOAD RELOCATION PROGRAM
210 FOR A=S1 TO S1+24
220 READ CODE
225 TT=TT+CODE
230 POKE A, CODE
240 NEXT A
245 IF TT<>117877 THEN PRINT"PROGRAM ERROR, PLEASE CHECK":STOP
250 ' *SUBROUTINE IS NOW IN
260 ' GO EXECUTE IT
270 EXEC 32512
280 SOUND 120,1 ' ANNOUNCE COMPLETION
290 ' OVERLAY PATCH #1 PREVENTS MEMORY TYPE

```

```

300 ' FROM BEING SWITCHED BACK T
O ROM/RAM
310 FOR A=S2 TO S2+6
320 READ CODE
325 TT=TT+CODE
330 POKEA, CODE
340 NEXT A
345 IF TT<>118610 THEN PRINT "ER
ROR IN PATCH #1, PRESS RESET, RE
LOAD 'BASLOAD' AND CHECK":POKE11
3,0:STOP
350 ' PATCH #2
360 ' INITIALIZE PARALLEL PIA
370 FOR A=S3 TO S3+13
380 READ CODE
390 POKE A, CODE
395 TT=TT+CODE
400 NEXT A
405 IF TT<>120656 THEN PRINT "ER
ROR IN PATCH #2, PRESS RESET, RE
LOAD 'BASLOAD' AND CHECK":POKE11
3,0:STOP
410 ' CLEAR COLD START FLAG
420 POKE 113,0
430 ' START UP BASIC
440 EXEC40999
450 ' THIS IS THE ASSMEBLY SOURC
E FOR THE
460 ' ABOVE CODE SEGMENTS
470 ' *****
*
480 '* BASIC RELOCATOR
490 ' ORCC #$50 DISABLE
INTERRUPTS
500 ' LDX #$8000 BASIC
START ADDRESS
510 ' LOOP LDA ,X GET A BYTE
520 ' STA $FFDF SWITCH
TO RAM MAP
530 ' STA ,X+ PUT BYTE
IN RAM
540 ' CMPX #$E000 END O
F BASIC

```

```

550 ' BEQ DONE ALL MOVE
D LEAVE IN RAM MAP
560 ' STA $FFDE SWITCH
BACK TO ROM MAP
570 ' BRA LOOP CONTINUE
MOVING
580 ' DONE ANDCC #$AF ENABLE
INTERRUPTS
590 ' RTS RUNNING IN AL
L RAM SYSTEM
600 ' *****
*****
610 '*PATCH 1 PREVENTS SAM FROM
BEING SWITCHED
620 '*BACK TO ROM MAP TYPE DURIN
G BASIC STARTUP
630 ' ORG $A054
640 ' LDB #0D ADDRESSES
TO SET IN SAM
650 ' JSR $A089 JUMP TO
NEW SETUP CODE
660 '* SPACE FOR THIS NEW ROUTIN
E IS MADE
670 '* AVAILABLE BY THE REMOVAL
OF THE MEMORY
680 '* SIZING ROUTINE IN PATCH #
2. MEMORY MUST
690 '* BY 32K TO EVEN BE DOING T
HIS.
700 ' *****
*****
710 '*REMOVE MEMORY SIZE ROUTINE
AND INSTALL
720 '*SAM SETUP ROUTINE FOR PATC
H #1
730 ' ORG $A084
740 ' LDX #$7FFE MEMORY
SIZE
750 ' BRA CONT DO REST
OF ORIGINAL CODE
760 ' *****
770 '* INITIALIZE SAM
780 ' INIT STA ,U++ WRITE TO
SAM
790 ' DECB COUNTER
DOWN
800 ' BNE INIT DONE ALL
ADDRESSES?
810 ' LDU #FFE0 RESET U
FOR REST OF CODE
820 ' RTS TO CODE AFTER
PATCH #1
830 ' NOP FILLER BYTE
840 ' CONT EQU * FIRST BYTE
OF OLD CODE
850 ' END
860 ' *****
*****

```

**NEW FOR YOUR COCO !!!
CHROMART!
A NEW HI-RES ART PROGRAM.**

- No knowledge of BASIC required!
- For a 32K **ECH** CoCo. Joystick optional
- Ultra fast M.L. routine
- Lets you: autoloading, box, circle, save, load, fill, get, put, joystick or keyboard control, line, move, change mode and color set, copy, clear - all using single letter commands!

FOR ONLY \$19.95 + \$1 S&H (Add \$3 for DISK version)

CHROMATIC COMPUTER COMPANY
801 Eldridge Rd., Fairless Hills, PA 19030
(215) 946-0263 or 493-5423
Check or Money Order Only!

COOKING
With
CoCo



PART III

Having built the utensils, we now start on the recipe to enhance CoCo's Disk Operating System.

By Colin J. Stearman

Editor's Note:

Due to the considerable interest in this article from users of the new Disk BASIC 1.1, Colin Stearman has done some more "cooking" and has come up with the patch addresses needed. You will find this month's listing indicates the lines which are unique to each revision. The actual assembly shown is for version 1.0, so if you have 1.1 your assembly will look a little different. Next month, the author will explain the differences for you 1.1 owners. (This month's RAINBOW ON TAPE has the patch programs for both 1.0 and 1.1.)

Also, the patched "DIR" command as it stands at the end of this month's revision will give some "garbage" on the screen. This is normal and the real file creation date will appear after Part 5 of this series.

We are now at the point where we can start in earnest modifying CoCo's disk operating system (DOS). We have the capability of saving to disk and reloading a modified DOS (on a 64K CoCo) and we can also save it in an EPROM. Starting this month and for the remainder of this series, I will be presenting an assembly language program to modify or "patch" the DOS to add the desired features described earlier.

The Ground Rules

Before I start on this month's details I think we had better discuss the rules for building each layer of the assembly language "cake." This may be a little tedious but if we all understand the approach now, it'll stop problems from cropping up later.

At the end of the series you will have a complete patch program called *DOSPATCH* which will add all the commands and functions. This program generates a binary file which overlays Disk BASIC, modifying what is already there and adding new code. This month we will develop the foundation of this program and each month add a new section until it is complete. Each month you will be able to assemble the composition so far and use it to patch the DOS to check

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

the functions implemented.

However, it is inevitable that each month we will add some code which is not fully functional because it requires code not destined to be added until a future installment. When this happens we will use a technique called "commenting out," which makes a "comment" of the line of code which cannot yet be made functional. Then later, when the required code is there, we can remove the comment and reassemble to fully activate the feature. In assembly language an asterisk at the start of the line signifies a comment line and the assembler simply ignores the entire line, no matter what its contents.

As you look through Listing 1 you will see lines marked with a reference number in square brackets (for example, [REF 12]). Later in the series we will make some modification to the associated line (most likely remove the asterisk) and I will refer to it by the reference number.

So the best approach is to use your editor to enter the listing exactly as shown. Then each month add the new listing to it, modify the reference lines as described in the text of the article, and reassemble.

The Parallel Port

A final "housekeeping" note before we begin. In a later installment I will be describing a "Centronics" parallel printer port. This month's code contains lines for this purpose. My assembler (*MACRO* by Computerware) allows conditional assembly. This simply means that I can control which lines get assembled and which do not. I use this feature to control the assembly of all the code associated with the parallel port. You will notice a section of code bounded by the following assembler directive lines:

```
IFDF PARPRT
.
.
.
(lines of code)
.
.
.
ENDC
```

This simply means that if a label called *PARPRT* has been defined, then assemble all the bounded lines; otherwise, do not. At the very beginning of the listing the variable *PARPRT* is equated to one, thus defining it and causing the lines to be assembled. If this line were "commented out," the label would not be defined and the lines would not be assembled. If your assembler does not have this feature and you will be building the parallel port, type in the bounded lines of code and leave out the "IFDF" and "ENDC" lines. If you do not intend building it, leave the whole lot out.

Enough of all this mundane detail and on to the assembly language program.

A Strong Foundation

Listing 1 is the base we will build on over the months. It consists of these primary parts:

- 1) Equates to memory locations and BASIC routines
- 2) Overlay lines to "hook in" the new code
- 3) Revisions to existing commands
- 4) New commands and functions look-up table
- 5) Installation code for the new commands
- 6) Parallel port initialization
- 7) Automatic file startup
- 8) Dummy commands and functions

Overlays

By using the *ORG* (origin) statement in this section of the code I have patched in various jumps and subroutine calls right into the existing DOS code. This is one of the main techniques for modifying existing commands. The call jumps to our new code and this usually completes the operation replaced by the jump code, then performs the revisions and returns to the original code.

You will also notice two small patches to *DSKI\$* and *DSKO\$*. These allow a track value up to 40 instead of 35, for use with the revised functions below.

Revisions to Existing Commands

I am sure you have encountered the "bug" in the *PCLEAR* command when used in a program. Maybe you have not come across a similar one in the *FILES* command. Each stem from the same type of error. Both commands have to relocate the BASIC program in memory but they forget to update the parse pointer so that BASIC can continue interpreting your program. The parse pointer points to the next item in your program to be interpreted by BASIC.

The revised code for these functions partly replaces the original code, duplicating much of it. At the crucial point the new pointer is calculated and stored at *\$A6*. Then the old code is used to complete the command. As an added bonus, the revisions to *PCLEAR* allow values of up to 16 instead of the customary eight. No changes have been made to the operation of *FILES* command.

OPEN

The five lines at the label *FILDAT* complete what was happening before the jump and then add the value in the *DATE\$* variable to the directory entry. This results in a creation date being stored in the directory every time a new file is created. The date is stored in the first two bytes of the directory entry reserved for future use by Radio Shack. These are bytes 16 and 17, counting from zero. The date is compressed into two bytes by a particular coding method as follows:

```
! FIRST BYTE ! SECOND BYTE !
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
!<---YEAR---->!<MONTH>!<--DAY-->!
```

The year value is stored as the last two digits only. Besides the obvious advantage of saving storage space, this compression technique allows the resulting 16-bit word to be sorted correctly, if this is desired.

When the directory command revisions are complete, the directory will show the creation date along with the usual information. It is very useful to know when a file was created, especially if you have the same file on another disk. Which is the most recent? This modification will tell you.

DIR

There are two revisions to this command. First, the creation date of each file is now displayed and second, the listing pauses after each screen is full, giving time to read it.

The date is displayed as *MM/DD/YY* as part of the directory line. At this time the date will not be displayed correctly because of a missing subroutine called *DATOUT*. The call to it has been commented out in line [REF 5].

When the screen is full the display will halt and wait for any key press. All keys will continue the display, except

BREAK, which will terminate the command immediately. The pause will only occur if the output is to the screen. The new *LDIR* command (described in a future installment) uses the *DIR* command but redirects it to the printer. As a result, no pause occurs.

DSKINI

Many of you have disk drives capable of accessing 40 tracks. Even the 35-track Radio Shack drives can usually access 37 tracks. Although the DOS cannot use the tracks above 35, BASIC could make use of them via the *DSKIS* and *DSKOS* commands (suitably modified, of course).

However, to do this, the extra tracks must be formatted and thus the revisions to *DSKINI*. The syntax of the command is now:

DSKINI drive, number of tracks, skip factor

“Drive” is the drive number as usual. “Number of tracks” is any value from 35 to 40. If no value is given, 35 is assumed. “Skip factor” is as described in the DOS manual. If omitted, a skip factor of four is used. Because of the slight revision to this command, if you specify a skip factor you must also specify the number of tracks.

Some acceptable calls include:

DSKINI1 — A normal initialization
DSKINIO,37 — Initialize 37 tracks with skip = 4
DSKINI3,40,2 — Initialize 40 tracks with skip = 2

BACKUP

Similarly, the *BACKUP* command has been modified to include any of the additional tracks from 36 to 40. The new syntax is:

BACKUP source drive [TO destination drive],[tracks]

Therefore, acceptable commands include:

BACKUP0 — backup to a second disk in 0, 35 tracks
BACKUP0,40 — ditto, but all 40 tracks
BACKUP1TO0,37 — backup disk in 1 to disk in 0, 37 tracks

The only requirement for backing up more than 35 tracks is that both disks be previously initialized for at least the number of tracks specified in the command.

KILL

The final command revision is to the file *KILL* command. If this is issued as a direct command then CoCo will check that you are sure you wish to erase it. An uppercase ‘Y’ is the only response which will result in the file being deleted. All others will cancel the kill. If the disk should have a write protect tab on it, this command will indicate the file was deleted and then return a “Write Protected” error (?WP). The file will still be there.

If the *KILL* command is used from within a BASIC program then no verification is performed. The assumption is that you have thoroughly debugged your program first!

New Commands and Functions

Next comes the command table and its dispatch address

table. You will find all the new commands here. These tables are in standard BASIC format with the last character of each command having bit seven set to indicate its end. It is important that the order of the command words and the dispatch table be the same, otherwise you will issue one command and get another! The first command (*COLD*) is tokenized as \$E1 with the remainder sequentially from there. The *PARALLEL* command is last because some of you will not need it and this keeps the tokens for all other commands consistent.

Immediately following the command tables are those for the new functions. These start at \$A8 and when tokenized are preceded by \$FF.

Because all the new functions and commands are established here but the code has not yet been implemented, I have put dummy calls at the end of the listing for each. As a result, BASIC will accept the new words but do nothing. This way you can check the operation of the tables and installing code. When each function is added, these dummy calls will be deleted.

Installation Code

The section of code starting at the label *ADDCOM* is run whenever the CoCo does a cold start (described in a future installment). This code sets up a table in low memory which is used to search for each BASIC command and function as the interpreter encounters them. Microsoft (who wrote this BASIC) kindly set things up so one more table can be added above and beyond the Disk BASIC commands.

At the end of this section is a revision to the “hook” in memory which gets taken when an error is encountered. For now this revision has been “commented out,” but later it will allow us to both trap errors and prevent BASIC from halting program execution and also return more meaningful error messages.

Parallel Port Initialization

Continuing the code, which is executed during a cold start, we encounter the parallel port “hook” patch and the initialization routine for the new peripheral interface adapter (PIA) which will run it. If you are not going to use the parallel port, leave this entire section out.

Auto File Execution

Just prior to this, I have put a small reminder indicating who brought you these useful revisions. Then comes a feature which is more powerful than you might at first imagine.

Before completing start-up and giving you the OK prompt, the revised BASIC tries to find and run a BASIC file called *AUTOEXEC.BAS* on drive 0. If successful, this program is automatically run. If a disk is present but with no such file on it, then an NF Error is returned. If no disk is in the drive then an I/O Error results.

The power of this feature lies in the fact that you write the *AUTOEXEC.BAS* file and you can put in it anything you want. For example, it could simply be line calling for the running of some other program on the disk. Or perhaps an automatic backup scheme. Listing 2 is designed to request the date and store it in the new memory location for this purpose. I suggest that at the very least you have such a file on your disks.

The power up sequence I have used successfully is:

- 1) Power up the video monitor
- 2) Power the Multi-Pak Interface, if you have one

- 3) Then switch on each disk drive
- 4) Load the disk with the *AUTOEXEC.BAS* file in drive 0
- 5) Power up CoCo

I have used this hundreds of times with no problem. After a few seconds the banner will display and drive 0 will turn on. If the file exists it will automatically run.

Now you can get your favorite program running without even touching the keyboard!

The Final Odds and Ends

The code at *COMCOD* and *FUNCOD* is executed during BASIC interpretation to get the address of the code needed to execute the command or function. Then immediately following you will see the dummy calls mentioned earlier.

Testing The Program

64K COMPUTER OWNERS

Testing is very easy for these people. If you did as I suggested last month, you should have a bootable disk with unmodified disk BASIC on it. If so, load it and start.

Once you have BASIC running in the all RAM mode, the procedure is to disable the interrupts, then overlay the patch file and cold start the new BASIC. As all interrupts are generated through one of the PIAs, they can be simply disabled by disabling the PIA. The steps are as follows, once the all RAM BASIC is running.

- 1) POKE &HFF03,&H34:'stop interrupts
- 2) LOADM"DOSPATCH":POKE&H71,0:

EXEC&HA027

These two lines should be entered as direct commands to BASIC. When complete, a new start-up banner with the revisions copyright notice should be displayed. You should now be able to test all the revised commands implemented so far. Also, all the new commands and functions should be acceptable to BASIC (no SN Error), but of course, they will do nothing.

You could save the revised DOS back to disk, but I recommend you save this until all revisions are completed.

NON-64K COMPUTER OWNERS

For you the testing is a little more difficult. We really do not want to go replacing the DOS ROM (Read only memory) in the disk controller cartridge quite yet. We can however, put the revised code in an EPROM and load it into the socket on the EPROM programmer addressed at \$C000.

If you have the Multi-Pak Interface you can fully test the result; if not, then basic functionality can be tested by plugging the EPROM programmer with programmed EPROM inserted into the expansion socket and then trying the commands. Of course, those accessing the disk drives will not work because the controller is not plugged in.

Without the Multi-Pak

From last month, you should already have Disk BASIC saved on tape under filename *DBASIC*. With the disk system operational, do this:

CLEAR 200,&H3FFF



**112 W. WISCONSIN AV.
KAUKAUNA, WI 54130
(414) 766-1851**

STOCK ITEMS SHIPPED SAME DAY!

THE COMPLETE TRS-80® LINE

- ELITE CALC \$54.95
- ELITE WORD \$54.95
- ELITE FILE \$67.00
- TOM MIX CALL FOR PRICE

THE COSMOS CONNECTION IS A COMPLETE SERIAL TO PARALLEL INTERFACE FOR THE COLOR COMPUTER TO THE GEMINI — 10X and 15X PRINTERS.



\$289.00

**10X - \$289.00
15X - \$445.00
Delta 10 - \$484.00
Delta 15 - \$597.00**

THE GEMINI-10X PACKAGE READY TO PLUG IN TO YOUR COLOR COMPUTER ONLY*

\$325.00
GEMINI-10X PACKAGE**

- NO AC REQUIRED
- SWITCHABLE BAUD RATE
- AT: 600
1200
2400
- HIGH QUALITY CONSTRUCTION
- COMPACT
- 90 DAY WARRANTY

\$60.00

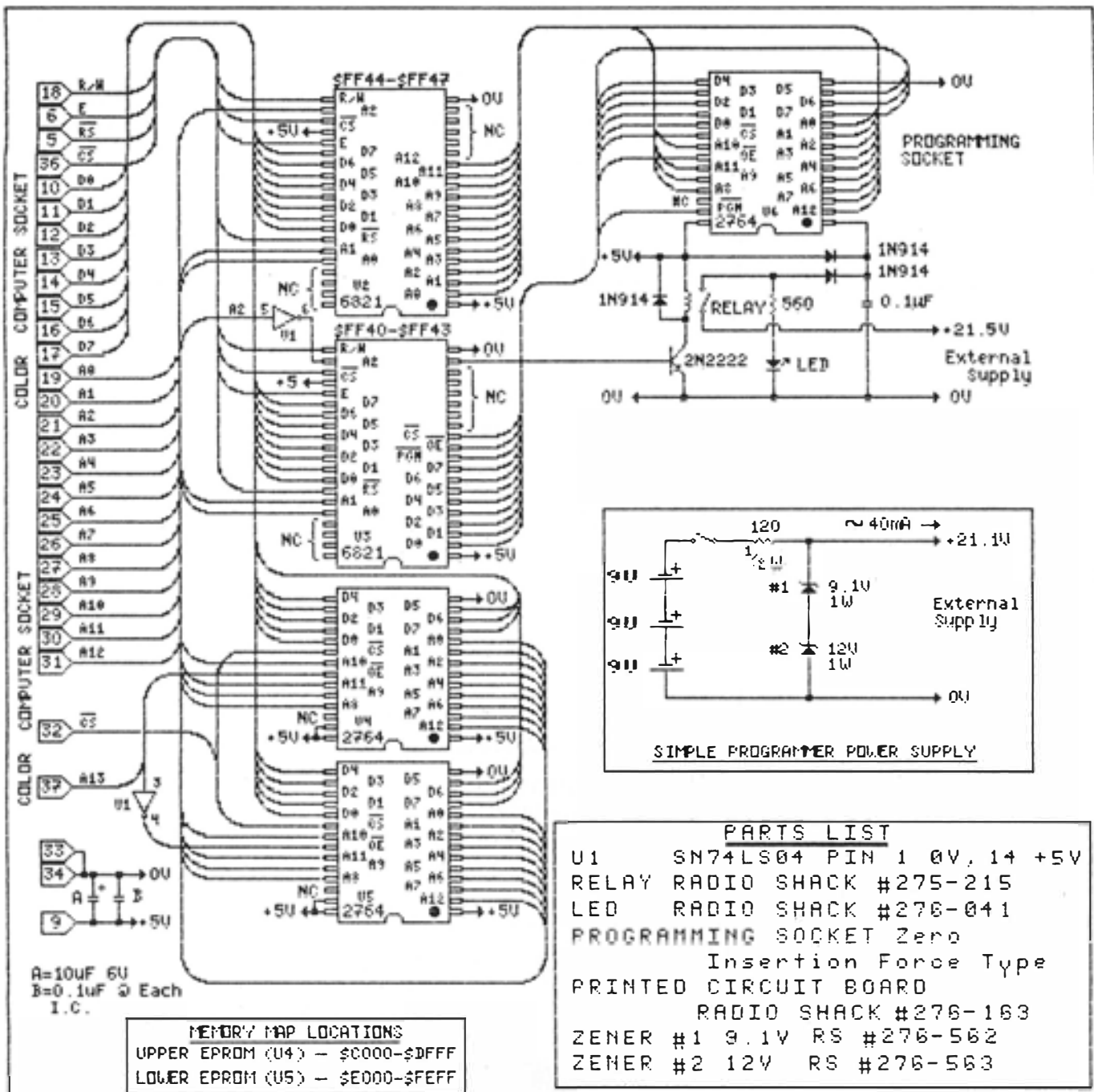


*Plus - U.S. ORDERS Add \$10.00 Shipping & Handling
*Plus - CANADIAN ORDERS Add \$25.00 for Shipping & Handling
**FREE - Shipping & Handling in U.S. with Package Orders
**Plus - CANADIAN ORDERS Add \$15.00 for Shipping & Handling with Package Orders



THE POWER BEHIND THE PRINTED WORD.

TRS-80 IS A TRADEMARK OF TANDY CORP. PRICES AND SPECIFICATIONS SUBJECT TO CHANGE

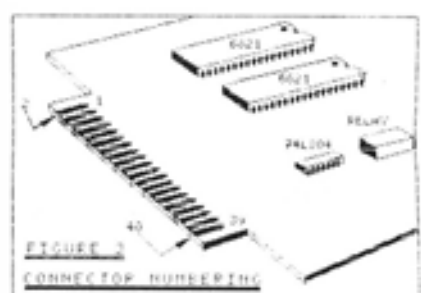


**COLOR COMPUTER
 2764 EPROM PROGRAMMER**

FIGURE 1
 Copyright 1984
 Colin J. Stearns

(Drawn with GRAPHICOM)

Editor's Note:
 Because of an error in production, two figures were left out of the last installment of "Cooking With CoCo." Here they are.



CLOADM"DBASIC",&H4000-&HC000+65536
 LOADM"DOSPATCH",&H4000-&HC000+65536
 CSAVEM"DBASIC#1",&H4000,&H5FFF,&HA027

Then power down, plug in the EPROM programmer, and do this:

CLEAR 200,&H3FFF
 CLOADM"DBASIC#1"
 CLOADM"EPROM"
 EXEC

Then transfer the memory contents from \$4000 to \$5FFF to a completely erased EPROM.

With Multi-Pak

Program the EPROM following the steps given last month under the subtitle "Using the Programmer with the Disk," but just before doing the EXEC, enter:

LOADM"DOSPATCH",&H4000-&HC000+65536

To test, use the procedure in the same section. But after doing the POKE65407,3 also enter POKE&H71,0 and EXEC&HA027. This will cold start the new system and allow you to see the automatic file execution feature.

Next Month

We will fill in some of the code for those commands and functions we just added. Also we will add FLEXIKEY. This

is a keyboard utility which is so useful (even though I say it myself!) that you'll wonder how you ever survived without it!

Finally, if you would like the entire DOSPATCH program source (with all future installments), along with binary files with and without the parallel port driver, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

Address this request or any questions to Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

Looking forward to your company next month.

Listing 1:

```
DOSPATCH - PATCH#001      OPT  C:\PROGRAMS\MACRO ASSEMBLER PAGE 1
Patch to RSDOS P#0021*****
0000      0003 REV EQU 0
0004 * Set REV = 0 for DOS 1.0, =1 for DOS 1.1
0005 *****
0006 * RADIO SHACK COLOR COMPUTER DOS *
0007 * IMPROVEMENTS AND MODIFICATIONS *
0008 * *
0009 * (C) 1984 COLIN J. STEARMAN *
0010 *****
0011 * Patch #1
0012 OPT NOG
0013 *****
0014 * COMMENT OUT THE NEXT LINE FOR A SERIAL PORT VERSION
0015 *(Controls conditional assembly)
0016 PARPRT EQU 1
0017 *****
0018 ****DOS 1.0 PATCH ADDRESSES*****
0019 IFEQ REV
C020 0020 A001 EQU %C020
C0D1 0021 A001 EQU %C0D1
C106 0022 A002 EQU %C106
```

It's time we put our chips on the table

... and showed you our best deals on computer hardware.

HARDWARE SPECIALS

Extended Basic w/bk	39.95
64k (DEI) Memory Upg	59.95
Amdek Disk Drives	99.95
26-3029 CoCo Drive 0	29.95
26-3023 CoCo Drive 1	239.95
HJL Keyboard (D,E,F,2)	79.95
Super Pro Keybd. (D,E)	64.95
26-3127 64k Extended CoCo 2	229.95
26-3801 Model 100 8k	549.95
26-3816P 8k Upgrade Model 100	99.95
26-1192 CGP-115 Printer/Plotter	169.95
26-1271 DMP-110 Printer 50 cps	349.95
C. Itoh 8510 AP Printer 120 cps	399.95
Gorilla/NAP Video Monitor (Grn)	109.95
Video Monitor Adapters	CALL
26-3024 RS Multi-Pac Interface	\$ 135.95
Botek Ser/Par Interface	\$ 69.95

ACCESSORIES

RS D.C. Modem IB	\$ 89.95
Novation J-Cat Modem	\$129.95
RS D.C. Modem II	\$179.95
Signalman MK X Modem	\$179.95
Hayes SM 300 Modem	\$239.95
USR Password 1200/300	\$429.00
CoCo Switcher	\$ 39.95
Elephant Disks ssdd	\$ 22.95
26-3030 OS-9 (64k)	\$ 64.95 (disk)
Basic-09 (req. OS-9)	\$ 87.95 (disk)
"C" Compiler (OS-9)	\$ 87.95 (disk)
FHL O-Pak (req. OS-9)	\$ 34.95 (disk)
Elite Word	\$ 59.95 (d&c)
Elite Calc	\$ 59.95 (d&c)
Color Term Plus	\$ 29.95 (cass)

MSI SOFTWARE

MSI DISKUTIL	NEW	\$19.95
COLOR FINANCE 1		\$49.95
COLOR FINANCE 11	NEW	\$69.95
MSI NAMEFILE		\$24.95
MSI CALENDAR	NEW	\$19.95

Call for prices and availability of your favorite software and hardware. All advertised items subject to availability. Prices do not include shipping and handling. All of the above units are covered by our 120 day carry-in warranty.

TRS-80 Trademark Tandy Corporation. Prices subject to change without notice.

Write for our FREE newsletter.

TOLL FREE TENNESSEE 1-800-545-2502 | TOLL FREE 1-800-251-5008



DELKER ELECTRONICS, INC.
 P.O. Box 897
 Dept. R
 408C Nissan Blvd.
 Smyrna, TN 37167



800-251-5008
800-251-2502 (TENNESSEE)
615-459-2636 (TENNESSEE)
615-254-0088 (NASHVILLE)

C11B 0023 A0003 EQU %C11B
 C124 0024 A0004 EQU %C124
 C17D 0025 A0005 EQU %C17D
 C570 0026 A0006 EQU %C570
 C575 0027 A0007 EQU %C575
 C58F 0028 A0008 EQU %C58F
 C65F 0029 A0009 EQU %C65F
 C68B 0030 A0010 EQU %C68B
 C6CB 0031 A0011 EQU %C6CB
 C6CF 0032 A0012 EQU %C6CF
 C990 0033 A0013 EQU %C990
 CA3B 0034 A0014 EQU %CA3B
 CB4A 0035 A0015 EQU %CB4A
 CB8F 0036 A0016 EQU %CB8F
 CB85 0037 A0017 EQU %CB85
 CC26 0038 A0018 EQU %CC26
 CC41 0039 A0019 EQU %CC41
 CC44 0040 A0019S EQU %CC44
 CE2E 0041 A0020 EQU %CE2E
 CEE5 0042 A0021 EQU %CEE5
 D169 0043 A0022 EQU %D169
 D182 0044 A0023 EQU %D182
 D18E 0045 A0024 EQU %D18E
 D1AF 0046 A0025 EQU %D1AF
 D1E5 0047 A0026 EQU %D1E5
 D446 0048 A0027 EQU %D446
 D4AB 0049 A0028 EQU %D4AB
 D4B2 0050 A0029 EQU %D4B2
 D571 0051 A0030 EQU %D571
 D594 0052 A0031 EQU %D594
 D670 0053 A0032 EQU %D670
 D6CD 0054 A0033 EQU %D6CD
 D723 0055 A0034 EQU %D723
 D7DD 0056 A0035 EQU %D7DD
 00E0 0057 HITOKN EQU %E0
 0058 * Highest command token in DOS 1.0
 0059 ENDC
 0060 *****
 0061 ****DOS 1.1 PATCH ADDRESSES*****
 0062 IFBT REV
 0063 A001 EQU %C02C
 0064 A0001 EQU %C0E4
 0065 A0002 EQU %C11B
 0066 A0003 EQU %C12B
 0067 A0004 EQU %C137

Quality Christian Software

MONEY BACK GUARANTEE

If for any reason you are not fully satisfied with any program you purchase from Quality Christian Software just return the original program (Cassette or Diskette) and we will refund the purchase price of the program.

★ ★ ★ ★ ★ 4 NEW PROGRAMS ★ ★ ★ ★ ★

PILGRIM'S PROGRESS: An interactive adaptation of Pilgrim's Progress in the form of an adventure game. Your progress is directed away from the city of destruction and towards the Celestial City. Important Biblical Doctrines are grasped as the player proceeds. Requires 16k E.C.B. — \$17.99 Cassette.

CHURCH TIME: A light hearted non-theological adventure for the whole family. You're almost late for church and to top it off you forgot your Bible. Rushing back into your house you find that the sticky front door has bolted behind you. The object is to find your Bible and get outside so that you won't be late for church. 32k E.C.B.—\$10.99 Cassette.

BIBLE REFERENCE PROGRAM: Topographical Bible Reference Program covering 27 Topics with 60 Biblical References. 16k E.C.B. not required—\$10.99 Cassette.

3-GAME PACK #3: Reversed Sword Drill game #2, "Who Did That" Game #2 & "Who Said That" Bible Quote game #2. — 16k E.C.B. — \$10.99 Cassette.

JUDE: A full text commentary and reference study on the Epistle of St. Jude. See the review in the December 1983 Issue of RAINBOW. Page 286. Requires 32k E.C.B. Cassette \$13.99 Disk \$16.99

3-GAME PACK #1: Books of the Bible Game, Bible Character Word Scramble game & "Who Said That" Bible quote game. Requires 16k E.C.B. — Cassette \$10.99

3-GAME PACK #2: Reversed Sword Drill game, "Who Did That" game & Bible Places Word Scramble game. Req. 16k E.C.B.—Cassette Version \$10.99.

Please Add \$2.00 for freight.
 C.O.D.'s add \$4.00
 Overseas add \$6.00

QCS
 P. O. Box 1899
 Duncan, OK 73534
 405/255-5696

24 Hour Phone Service



0068 A0005 EQU %C190
 0069 A0006 EQU %C59D
 0070 A0007 EQU %C5A2
 0071 A0008 EQU %C5BC
 0072 A0009 EQU %C68C
 0073 A0010 EQU %C6E5
 0074 A0011 EQU %C6FB
 0075 A0012 EQU %C6FC
 0076 A0013 EQU %CA3E
 0077 A0014 EQU %CAE9
 0078 A0015 EQU %CC1C
 0079 A0016 EQU %CCA9
 0080 A0017 EQU %CCAF
 0081 A0018 EQU %CD00
 0082 A0019 EQU %CD1B
 0083 A0019S EQU %CD1E
 0084 A0020 EQU %CF0A
 0085 A0021 EQU %CFC1
 0086 A0022 EQU %D256
 0087 A0023 EQU %D26F
 0088 A0024 EQU %D27B
 0089 A0025 EQU %D29C
 0090 A0026 EQU %D2D2
 0091 A0027 EQU %D534
 0092 A0028 EQU %D599
 0093 A0029 EQU %D5A0
 0094 A0030 EQU %D65E
 0095 A0031 EQU %D6B1
 0096 A0032 EQU %D761
 0097 A0033 EQU %D7C0
 0098 A0034 EQU %D816
 0099 A0035 EQU %D8D0
 0100 HITOKN EQU %E1
 0101 * Highest command token in DOS 1.1
 0102 ENDC
 0103 *****
 0104 *
 0105 CHRVRT EQU %008 OLD VECTOR JUMP
 0106 NTRACK EQU %00 USE CASSETTE TEMP STORE
 0107 *****
 0108 * USES UNUSED(?) LOW RAM LOCATIONS
 0109 ELINE EQU %76 LINE # CAUSING ERROR
 0110 JLINE EQU %DC LINE TO JUMP TO ON ERROR
 0111 ECODE EQU %5A ERROR CODE
 0112 *****
 0113 ZERO EQU %8A ZERO CONSTANT 16 BITS
 0114 DATA EQU %FF26 PIA DATA REGISTER
 0115 BETKEY EQU %A1B1 BASIC'S CURSOR/KEY ROUTINE
 0116 RETURN EQU %B95B OUTPUTS A CARRIAGE RETURN
 0117 SPACE EQU %A019 OUTPUT A SPACE
 0118 CHROUT EQU %A2B2 OUTPUTS CHARACTER IN A
 0119 STROUT EQU %B9A2 BASICS STRING OUTPUT X POINTS
 0120 * TO STRING, B HAS CHAR COUNT
 0121 DEVNUM EQU %6F OUTPUT DEVICE NUMBER
 0122 HLD8BFR EQU %1DA CASSETTE BUFFER FOR HOLD
 0123 BAS8BFR EQU %2DD BASIC BUFFER
 0124 HLDPTR EQU %1D7 IN CASSETTE FILE NAME BFR
 0125 INSERT EQU %1DB DITTO
 0126 WHLINE EQU %1D9 DITTO
 0127 BDFLAG EQU %95 BAUD RATE LOCATION USED AS
 0128 * SERIAL/PARALLEL FLAG
 0129 BAUDRT EQU %96 NORMAL SERIAL BAUD RATE LSB
 0130 * NEXT 3 WORDS ARE IN CASSETTE FILE NAME
 0131 LINNUM EQU %1D1 AUTO CURRENT LINE NUMBER
 0132 INCNUM EQU %1D3 AUTO LINE INCREMENT
 0133 LCOUNT EQU %1D5 USED IN DIR DELAY
 0134 * there are 4 empty ram locations in the command
 0135 *dispatch table terminator. they are %149/4A and
 0136 * %14E/F.
 0137 AUTOFB EQU %149
 0138 INTFLG EQU %14A RAM FLAG FOR REISSUED LINE
 0139 DATUM EQU %14E USES TWO BYTES TO STORE DATE
 0140 *
 0141 * This section contains the overlays to patch in
 0142 * the new commands, functions and revisions
 0143 *
 0144 * REMOVE <CR> AFTER BANNER
 C17D 0145 ORG %0005
 C17D 00 0146 FCB 0
 0147 *
 0148 **** PCLEAR PATCH ****
 C028 0149 ORG %001 SETS TABLE TO %0020 ORIGINALLY
 C028 CCD7DD 0150 LDD %PCLEAR REPOINT TO NEW ROUTINE
 0151 *
 0000 0152 IFEQ REV DOS 1.0
 0153 **** FILES PATCH ****
 D0E4 0154 ORB %D0E4 PATCH OVER EXISTING CODE
 D0E4 7EDB24 0155 JMP FILES DO EXTRA CODE
 0156 ENDC
 0157 *
 0158 *** PATCH FOR NEW KEYBOARD ROUTINE ****

```

C100      0159      DRG  A0002      SETUP FOR JMP AT $16A
          0160 *      FDB KEYBRD GOES TO NEW KEYBOARD RTN [REF 1]
          0161 * [REF 1: Uncomment when FLEXIKEY code is installed]
          0162 * DID HAVE A0000, JUMP TO THIS IF DEV CODE(>0)
          0163 *
          0164 **** ADD COMMANDS PATCH ****
C0D1      0165      DRG  A0001
C0D1 7ED991 0166      JMP  ADDCOM
          0167 *
C124      0168      DRG  A0004
          0169 *      FDB ERCNCL      [REF 2]
          0170 * [REF 2: Uncomment when ERRORS code is installed]
          0171 * PATCH INTO RUN COMMAND TO CANCEL ERROR JUMP
          0172 * A0004 ORIGINALLY HAD A0013
          0173 *****
          0174 * PATCH IN FOR AUTO INPUT
C11B      0175      DRG  A0003
          0176 *      FDB INPUT      [REF 3]
          0177 * [REF 3: Uncomment when AUTO code is installed]
          0178 * A0003 DID HAVE %C687 WHICH JUST RETURNED
          0179 *****
          0180 ** DO A PAUSE AFTER EACH 15 LINES IN DIR
C8D5      0181      DRB  A0017
C8D5 8DD886 0182 * INITIALIZE COUNTER
          0183      JSR  NOTBRK
          0184 *
CC26      0185      DRB  A0018
          0186 * DO PAUSE IN DIR
CC26 8DD849 0187      JSR  LINHLD
          0188 *
          0189 *
          0190 *****
          0191 * PATCH TO ADD DATE TO FILE WHEN OPENING
C570      0192      DRG  A0006
C570 7ED83C 0193      JMP  FILDAT      PUT DATE INTO FILE
          0194 *****
          0195 * PATCH FOR DSKINI EXTRA TRACKS
          0196 *****
D571      0197      DRG  A0030
D571 9180  0198      CMPA  <NTRACK
          0199 *
D594      0200      DRG  A0031
D594 9180  0201      CMPA  <NTRACK
          0202 *
D446      0203      DRG  A0027
          0204 * FIX DSKIN/DSKIN% TO ALLOW UP TO 40 TRACKS
D446 27  0205      FCB  39      TOP TRACK NUMBER
          0206 *
D4AB      0207      DRG  A0028      FIRST LINE OF DSKINI
D4AB 1603DE 0208      LBRA  DSKINI      GOTO NEW CODE
          0209 * DID HAVE LBEQ %A61F
          0210 *
          0211 * PATCH BACKUP
D182      0212      DRB  A0023
D182 7EDBAC 0213      JMP  BCKPAT      BACKUP PATCH
          0214 * RETURN TO A0024
          0215 *
          0216 * THIS PATCHES BACKUP SYNTAX CHANGES
          0217 * MAKE TRACK COUNT A VARIABLE
D1AF      0218      DRG  A0025
D1AF 9600  0219      LDA  <NTRACK      WAS LDA #23
          0220 *
          0221 * THIS PATCHES KILL TO CHECK FOR ERASING FILE
C6CB      0222      DRG  A0011
C6CB 7EDBDA 0223      JMP  KILLCK      DO KILL CHECK CODE
          0224 *
          0225 *****Following patches set the drive step rate
          0226 *Affects all drives, select rate of slowest drive
          0227 *
D6CD      0228      DRG  A0033      RESTORE step rate
D6CD 02  0229      FCB  2      =20mS;3=30mS;1=12mS;0=6mS
          0230 *
D723      0231      DRG  A0034      SEEK step rate
D723 16  0232      FCB  #16      =20mS;#17=30mS;#15=12mS;#14=6mS
          0233 *****
          0234 * Patch code to existing commands
          0235 *
          0236 * ALL NEW CODE RESIDES IN THE UPPER
          0237 * AREA OF DISK ROM NOT USED
          0238 * BY DISK BASIC, STARTING AT
          0239 * A0035.
D7DD      0240      DRG  A0035
          0241 *
          0242 *
          0243 *****
          0244 * PATCH FIXES THE BUG IN PCLEAR
          0245 *
          0246 *
          0247 *
          0248 * DO ROUTINE, FIX IS TO REVISE PARSER POINTER
          0249 * AT %A6 FOR CHANGE IN LOCATION

```



Megamunk

32K tape \$21.95

32K disk \$23.95

100% machine language fast action game. As a soldier/monkey you must save the forest of Ledonia from the evil mammoth spiders, avoid the falling coconuts, save the sacred birds and recover Ledonia's treasure. Megamunk has 11 different screens with multiple colors and "four voice" music. A REAL challenge. (Joy-stick required)

10KEY

16K \$17.95

A numeric keypad for your COCO for only \$17.95? Impossible? 10KEY is 100% position independent machine language software that turns a portion of your keyboard into a numeric keypad. 10KEY is useful when typing in those long DATA statements with lots of numbers or when entering numeric data with any BASIC program. (Note: 10KEY does not function with INKEYS statements). The 10KEY package contains the following: 1-10KEY a machine language program that loads at the top of 16K. 2-GEN a program to generate your own custom version of 10KEY. 3-DEMO a simple graphing program with which to practice with the 10KEY program.

ORDERING INFO

- Add \$2 for shipping and handling
- Utah residents add 5.75% sales tax
- We accept checks, money orders, VISA and MASTER CHARGE. Order by phone: 801-571-5023 (call 6:30 to 10 pm MDT for technical info.)
- We carry many other line programs, please call or write for our flyer.

* COLOR *
**CONNECTION
 SOFTWARE**

1060 Buddlea Drive
 Sandy, Utah 84070



DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives



Introducing **MEGADISK**

5 to 20 Megabyte, ready to run on the TRS 80 Model I/III/IV/4P, color computer, I.B.M. PC, Apple, Franklin

DRIVE A HARD BARGAIN™ Complete Systems Starting at \$999.95

Call Toll Free Ordering 1-800-343-8841



High Quality Lowest Price
Drive 0, 1, 2, 3
for the
Color Computer
Starting at \$199.95



Disk Drive Upgrade
for model III/IV easy to install system
Starting at ~~\$369.95~~
Call for new lower price

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 am to 4:30 pm

DEALER INQUIRIES INVITED.

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge. C.O.D., please add \$3.00. Shipping: Please call for amount. Not responsible for typographical errors.

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular Tel. (514) 845-1534
Canadian Toll Free 800-361-5155

Service! Service!
All in stock products are shipped within 24 hours of order. Repair/Warranty service is performed within 24 hours of receipt unless otherwise noted. We accept C.O.D., foreign and APO orders. School and D&B corporate P.O.s accepted.

TRS/80 Registered Trademark Tandy Corp. IBM-PC Registered IBM Corp. Apple Registered Trademark Apple Computer Corp. Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES



Super Sale on New Disk Drives

Starting at \$169.00!

Tandon — Siemens — Remex — MPI — Teac — Shugart — Tabor

40 or 80 Tracks — Single or Dual Head — New 3 1/2" Drivette™
Our Disk Drives are Capable of Single and Dual Density Operation

The NEWEST Technology Capable of Operating on Most Popular Computers

Drive a Hard Bargain!!™ For your TRS/80, Color Computer, IBM, Apple, Franklin
5 M.B.-20 M.B. Complete Systems from \$999.95

Diskette Breakthrough — 10 Pack in Library Case — ~~\$18.95~~ ^{NEW} LOW PRICE

SAVE!! PLEASE CALL FOR OUR MOST CURRENT PRICE REDUCTIONS.

TOLL FREE ORDERING
1-800-343-8841

GENERAL AND TECHNICAL
1-617-872-9090

- Disk Drives (0123) TRS/80-IBM-Apple - TI Franklin-Max/80-LNW
- Model I/III/IV Upgrade (Disk Drives - Memory)
- Printers — Daisywheel/Dot Matrix
- Percom Double Density Controller (Model I)
- Color Computer Printer Interfaces
- Disk Drive Operating Systems
- Repair Services Now Offered — FAST Turn-a-Round
- Apple/Franklin Compatible Add-On Drives with Case & Cable
- Diskettes in Library Cases
- DISK DRIVE CASES AND POWER SUPPLIES**
- Printer Buffers 8K to 512K starting at \$143.95
- Holmes Model I/III Speed-up Mod starting at \$90.00
- Cables — Printer/Disk Drive starting at \$23.00

**CALL
TOLL
FREE
FOR
NEW
PRICES**

Warranty on Disk Drives — 6 Months to 1 Year

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 am to 4:30 pm

DEALER INQUIRIES INVITED.

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D., please add \$3.00.
Shipping: Please call for amount.
Not responsible for typographical errors.

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular Tel. (514) 845-1534
Canadian Toll Free 800-361-5155

Service! Service!
All in stock products are shipped within 24 hours of order. Repair/Warranty service is performed within 24 hours of receipt unless otherwise noted. We accept C.O.D., foreign and APO orders. School and D&B corporate P.O.s accepted.

TRS/80 Registered Trademark Tandy Corp. IBM-PC Registered IBM Corp. Apple Registered Trademark Apple Computer Corp.
Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

```

0250 * OF BASIC
0251 *
D7DD 01C0 0252 PCLEAR CMPA 00C0 IS IT PCLEAR?
D7DF 1026F64B 0253 LBNE A0020 NO, EXIT TO PREVIOUS HOOK
D7E3 9D9F 0254 JSR 09F PARSE OVER PCLEAR TOKEN
D7E5 0D070B 0255 JSR 0B70B GET & EVAL. 1ST ARG.
D7EB 5D 0256 TST0 IS IT ZERO?
D7E9 274E 0257 BEQ FCERR YES, SO ERROR
D7EB C111 0258 CMPB 017 IT IS >16?
D7ED 244A 0259 BHS FCERR YES, ERROR
D7EF 0606 0260 LDA 06 MULTIPLY BY 1536(1 SCREEN)
D7F1 3D 0261 MUL 6*256=1536
D7F2 0B0C 0262 ADD0 0BC ADD TO START OF
D7F4 1F9F 0263 TFR B,A ISRT GRAPHIC SCREEN
D7F6 C601 0264 LDB 01
D7F8 1F02 0265 TFR D,Y COPY THIS+1 TO Y
D7FA 109307 0266 CMPD 0B7 IS THIS PAGE RESERVED?
D7FD 253A 0267 BLO FCERR YES, SO ERROR
D7FF 9319 0268 SUBD 019 SUB. START OF BASIC
DB01 1F03 0269 TFR D,U SAVE VALUE TEMPORARILY
DB03 D310 0270 ADD0 010 ADD END OF BASIC
DB05 1F01 0271 TFR D,X SAVE NEW END ADDRESS
DB07 C300CB 0272 ADD0 0200 STACK SIZE
DB0A 9321 0273 SUBD 021 STACK TOP ADDRESS
DB0C 2420 0274 BHS FCERR NO ROOM, ERROR
DB0E 0D00 0275 BSR DIRECT CHECK IF DIRECT
DB10 2706 0276 BEQ LI YES SO DIRECT NO FIX
DB12 1F30 0277 TFR U,D RECOVER OFFSET
DB14 03A6 0278 ADD0 0A6 REVISE PARSE POINTER
DB16 DDA6 0279 STD 0A6 AND SAVE IT
DB18 7E9604 0280 LI JMP 09604 CONTINUE PCLEAR ROUTINE
0281 *
0282 *THIS CHECKS IF IN DIRECT MODE, Z=1 IF SO
DB10 3410 0283 DIRECT PSHS X PRESERVE A AND B
DB1D 9E60 0284 LDX 060 GET LINE NUMBER
DB1F 3001 0285 LEAX 1,X IS IT $FFFF?
DB21 3510 0286 PULS X RECOVER D
DB23 39 0287 RTS
0288 *****
0000 0289 IFEQ REV DOS 1.0
0290 * PATCH FIXES A SIMILAR BUG
0291 * IN THE FILES COMMAND
DB24 9310 0292 FILES SUBD 010 END OF BASIC ADDRESS
0293 * D HAS OFFSET DUE TO MOVE OF BASIC
DB26 3406 0294 PSHS A,B SAVE RESULTS

```

```

DB2B 0DF1 0295 BSR DIRECT CHECK IF DIRECT MODE
DB2A 2706 0296 BEQ SKIP YES SO DIRECT COMMAND
DB2C ECE4 0297 LDD ,S GET D OFF STACK FIX OFFSET
DB2E D3A6 0298 ADD0 0A6 ADD TO PARSE POINTER
DB30 DDA6 0299 STD 0A6 SAVE IT
DB32 3506 0300 SKIP PULS A,B RECOVER OFFSET
DB34 D319 0301 ADD0 019 ADD BASIC START ADDRESS
DB36 7ED0E0 0302 JMP 0D0E0 CONTINUE FILES CODE
0303 ENDC
0304 *
DB39 7E844A 0305 FCERR JMP 0B44A ?FC ERROR
0306 *
0307 *****
0308 *FILE DATE TO DIRECTORY
DB3C B70976 0309 FILDAT STA 0976 FINISH WHAT WAS DOING
DB3F A742 0310 STA 2,U DITTO
DB41 FC014E 0311 LDD DATUM GET DATE
DB44 ED45 0312 STD 5,U PUT INTO BUFFER
DB46 7EC575 0313 JMP A0007 CONTINUE OPENING FILE
0314 *****
0315 * DIR command revisions
0316 *
0317 *
0318 *****
0319 * directory output of file creation date
DB49 3404 0320 LINHLD PSHS B SAVE GRANULE COUNT
DB4B 0BDDCC 0321 JSR 0BDDC OUTPUT IT TO SCREEN
DB4E BDCC41 0322 JSR SPACE OUTPUT 1 SPACE
DB51 3504 0323 PULS B RECOVER GRANULE COUNT
DB53 C109 0324 CMPB 09 HOW MANY DIGITS?
DB55 2203 0325 BHI ATCLM DONT NEED EXTRA SPACE
DB57 BDCC41 0326 JSR SPACE OUTPUT A SPACE
DB5A AE62 0327 ATCLM LDX 2,S GET DIRECTORY PNTR
DB5C ECBB10 0328 LDD 16,X GET DATE FROM DIRECTORY
DB5F 3406 0329 PSHS D SAVE VALUE
DB61 C600 0330 LDB 00 SEE IF ROOM FOR STRING
DB63 0B050F 0331 JSR 0B50F DONT RETURN IF NOT
0332 *X POINTS TO STRING SPACE
DB66 3506 0333 PULS D GET DATE AGAIN
0334 * JSR DATOUT PUT DATE IN IT (REF 5)
DB6B C6F0 0335 LDB 0-B CHARACTERS TO FIX
DB6A A605 0336 OUTCHR LDA B,X GET CHARACTER
DB6C BDA202 0337 JSR CHR0UT OUTPUT IT
DB6F 5C 0338 INCB REDUCE COUNTER
DB70 26F0 0339 BNE OUTCHR DO SOME MORE
0340 *****
0341 * DIRECTORY PAUSE TO SCREEN ONLY
0342 *
DB72 006F 0343 TST DEVNUM CHECK IF TO SCREEN
DB74 2615 0344 BNE CR DONT PAUSE IF DIR NOT TO SCREEN
DB76 7A01D5 0345 DEC LCOUNT DECREASE CURRENT LINE COUNT
DB79 2610 0346 BNE CR OUTPUT NEXT LINE
DB7B 0DA1B1 0347 WAIT JSR GETKEY GET KEYBOARD ENTRY
DB7E 27F0 0348 BEQ WAIT IF NONE YET
0349 *
DB80 0103 0350 CMPA 03 IS IT BREAK?
DB82 2602 0351 BNE NOTBRK NO
DB84 3264 0352 LEAS 4,S REMOVE OLD RETURN
0353 * AND X LEFT ON STACK
0354 *
DB86 C610 0355 NOTBRK LDB 016 REST LCOUNT
DB8B F701D5 0356 ST0 LCOUNT
DB80 39 0357 CR RTS
0358 *****
0359 ** PATCH DSKINI0 TO FORMAT UP TO 40 TRACKS
0360 ** SYNTAX IS DSKINI drive,number of tracks,skip factor
0361 ** NUMBER OF TRACKS IS 35 - 40, DEFAULTS TO 35
DB8C 1027CDBF 0362 DSKINI LBEQ 0A61F DN ERROR
DB90 BDD169 0363 JSR A0022 CHECK FOR 0-3 DEVICE #
DB93 C623 0364 LDB 035 DEFAULT # OF TRACKS
DB95 9DA5 0365 JSR 0A5 ANY MORE ON INPUT LINE?
DB97 2703 0366 BEQ NOVALS NO MORE VALUES
DB99 0B0730 0367 JSR 0B730 GET TRACK VALUE
DB9C 0D03 0368 NOVALS BSR TRKCH: CHECK FOR VALID DRIVE #
0369 *
DB9E 7ED4B2 0370 JMP A0029 RETURN TO REGULAR CODE
0371 *
DBA1 C123 0372 TRKCH: CMPB 035 LOWEST LEGAL VALUE
DBA3 2594 0373 BLO FCERR ?FC ERRDR
DBA5 C120 0374 CMPB 040 HIGHEST LEGAL VALUE
DBA7 2290 0375 BHI FCERR ?FC ERROR
DBA9 0700 0376 STB NTRACK SAVE IN TEMP BUFFER
DBAB 39 0377 RTS
0378 *
0379 *****
DBAC 3404 0382 BCKPAT PSHS B SAVE SOURCE DRIVE NO.
DBAE C623 0383 LDB 035 DEFAULT TRACKS
DBB0 0700 0384 STB NTRACK SAVE DEFAULT VALUE
DBB2 3504 0385 PULS B RECOVER SOURCE DRIVE NO.

```



- * Pass, Run, or Kick — You call the Plays!
- * Compete with friends or challenge the computer.
- * Contains extended basic and non-extended basic versions for 16K cassette color computers.

Send \$16.95 (check or money order) for each game (Colorado residents add 3 1/2% sales tax). Allow four weeks for delivery.

Big B Software
P. O. Box 91
Broomfield, Colorado 80020

Please send me _____ game(s) @ \$16.95 each.

Name _____

Address _____

City, State, Zip _____

From the programmer that brought **ZAXXON**
to the Color Computer,**

Moreton Bay Software proudly presents
BJORK BLOCKS.



An incredible graphic utility! Now you can design graphics just like the masters. You can even animate! User friendly. Precision drawing. Precision color selection. Fully menu driven. Only one joystick needed for menu selection and graphic creation. Compressed data storage or load and save 6K binary files. Almost impossible to crash. Create your own graphic adventure screens. Limitless applications in communication, education and program development.



Picture created with BJORK BLOCKS

Requires 32K Extended Basic
(64K for animation)

\$34.95 Tape or Disk

DOUBLE DRIVER

The BEST monitor driver available, unlike some monitor drivers the Double Driver provides TRUE monochrome and color composite output. Audio Output. Solderless installation. \$24.95



64K UPGRADES

Instantly access 64K via M/L Totally solderless kit to upgrade E Boards. Kit includes eight 4164 prime chips and chips U29 and U11 already soldered. E Board Kit \$69.95

Color Computer II kit requires soldering. \$64.95



MORETON BAY SOFTWARE

A Division of Moreton Bay Laboratory



316 CASTILLO STREET
SANTA BARBARA,
CALIFORNIA 93101
(805) 962-3127

Ordering information

Add \$2.00 shipping and handling per order. We ship within 24 hours on receipt of order. Blue Label Service available. California residents add 6% sales tax.



GRAPHICOM

The perfect line drawing companion to **BJORK BLOCKS**. You must see this program to believe it! Create pictures and text on the same screen. Now you can create pictures as good as any graphic you have seen on the color computer. Write graphic adventures or educational programs. Requires 64K EXB, Disk Drive and Joy Sticks \$29.95

SPECIAL: Bjork Blocks and Graphicom \$55.00

MORE BUSINESS -Ver 3.12 The *preferred* business package. Completely interactive. General Ledger. Accounts Receivable. Accounts Payable. Customer Statements. Mailing Labels. Profit/Loss. Balance Sheet Statements. Our most powerful business package. Buy the best!

32K Disk R/S DOS \$99.95

TRIVIA AND SOME SIGNIFICA

Get 40% more question at 66% the cost!

Great family or party game. More than 1900 questions in nine categories.

HISTORY ENTERTAINMENT ANIMALS
SCIENCE AND TECHNOLOGY
SPORTS AND GAMES ART AND MUSIC
LITERATURE AND LANGUAGE
POLITICS AND PLACES
MATHEMATICS AND COMPUTERS

Challenging, educational and even funny at times. Best of all, you get the utility the programmer used to create these questions. All ready for you to create your own challenges. Make up questions about family history, high school basketball scores or your favorite TV series. Parents and teachers can use this to develop their own educational files.

16K EXTENDED BASIC CASSETTE \$19.95
32K EXTENDED BASIC DISK \$21.95

*Zaxxon Reg TM Sega Corp.

**Color Computer Reg TM Tandy Corp.

DYNAMITE+™

"THE CODE BUSTER"

disassembles any 6809 or 6800 machine code program into beautiful source

- Learn to program like the experts!
- Adapt existing programs to your needs!
- Convert your 6800 programs to 6809!
- Automatic LABEL generation.
- Allows specifying FCB's, FCC's, FDB's, etc.
- Constants input from DISK or CONSOLE.
- Automatically uses system variable NAMES.
- Output to console, printer, or disk file.
- Available for all popular 6809 operating systems.



FLEX™ \$100 per copy; specify 5" or 8" diskette.
OS-9™ \$150 per copy; specify 5" or 8" diskette.
UniFLEX™ \$300 per copy; 8" diskette only.

For a free sample disassembly that'll convince you DYNAMITE+ is the world's best disassembler, send us your name, address, and the name of your operating system.

NEW

CoCo OS-9 VERSION

\$59.95

DISASSEMBLES OS-9, FLEX, DOS FILES

Order your **DYNAMITE+** today!

See your local DYNAMITE+ dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Orders outside North America add \$5 per copy. Please specify diskette size for FLEX or OS-9 versions.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020



UniFLEX software prices include maintenance for the first year.

DYNAMITE+ is a trademark of Computer Systems Center.

FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

Dealer Inquiries welcome.



```

DBB4 9DA5 0386 JSR #A5 ANY MORE ON LINE?
DBB6 2719 0387 BEQ BUPOUT NO SO EXIT
DBBB 812C 0388 CMPA #', LOOK FOR A COMMA
DBBA 2708 0389 BEQ GTTRK YES SO GET NO OF TRACKS
0390 * LOOK FOR '0' TOKEN
DBBC C6A5 0391 LDB #A5 '0' TOKEN
DBBE BDB26F 0392 JSR #B26F CHECK FOR IT SN ERROR IF NOT
DBC1 BDD169 0393 JSR A#022 get second drive and check it
0394 * now we have second drive in b
0395 *****
0396 * NOW GET NO OF TRACKS
DBC4 3404 0397 GTTRK PSHS B PRESERVE SECOND DRIVE #
DBC6 9DA5 0398 JSR #A5 ANY MORE ON LINE?
DBC8 2785 0399 BEQ BUPEXT NO SO CONTINUE OLD CODE
DBCA BDB73B 0400 JSR #B73B PARSE , GET VALUE
DBCD BDD2 0401 BSR TRKCHK FOR VALID DRIVE #
DBCF 3584 0402 BUPEXT PULS B RECOVER SECOND DRIVE VALUE
DBD1 7ED1BE 0403 BUPOUT JMP A#024 CONTINUE OLD CODE
0404 *****
0405 * REVISE KILL ROUTINE TO CHECK FOR ERASURE
0406 *
DBD4 BDC65F 0407 KILLCK JSR A#009 CHECK FOR FILE
DBD7 BDC68B 0408 JSR A#010 DID WE GET A MATCH?
0409 * WDN'T RETURN HERE IF WE DIDN'T
DBDA 3416 0410 PSHS X,A,B SAVE REGISTERS
DBDC BDD81B 0411 JSR DIRECT only confirm in direct mode
DBDF 2638 0412 BNE NDCNF Dont confirm delete
DBE1 C60A 0413 LDB #10 CHARACTER COUNT
DBE3 BEDBFD 0414 LDX #CHKMSG POINT TO MESSAGE
DBE6 BDB9A2 0415 JSR STROUT OUTPUT THIS
DBE9 BDA1B1 0416 JSR GETKEY GET ANSWER
DBEC BDA2B2 0417 JSR CHRPUT OUTPUT IT
DBEF 3402 0418 PSHS A SAVE IT
DBF1 BDB95B 0419 JSR RETURN OUTPUT A CR
DBF4 3502 0420 PULS A GET RESPONSE
DBF6 B159 0421 CMPA #'Y IS IT YES
DBFB 2714 0422 BEQ CONFRM CONFIRM DELETION
DBFA 3516 0423 PULS X,A,B
DBFC 39 0424 RTS EXIT AND DON'T DELETE
DBFD 53 0425 CHKMSG FCC %SURE (Y/N)?%
D907 44 0426 CNFMSG FCC /DELETED/
0427 *
D90E BED907 0428 CONFRM LDX #CNFMSG POINT TO CONFIRM MESSAGE
D911 C607 0429 LDB #7 CHARS IN IT
D913 BDB9A2 0430 JSR STROUT OUTPUT THIS
D916 BDB95B 0431 JSR RETURN PLUS A CR
D919 3516 0432 NDCNF PULS X,A,B RECOVER REGS
D91B 7EC6CF 0433 JMP A#012 CONTINUE KILL COMMAND
0434 *****
0435 * COMMAND TABLE AND JUMP CODE
0436 *
0437 *
0438 *****
0439 * ADDED BASIC COMMANDS AND FUNCTIONS #
0440 *****
0441 *
0442 *
0443 * COMMAND TABLE
0444 *
D91E 43 0445 COMBTL FCC /COL/
D921 C4 0446 FCB 'D+12B
D922 57 0447 FCC /WPOK/
D926 C5 0448 FCB 'E+12B
D927 46 0449 FCC /FAS/
D92A D4 0450 FCB 'T+12B
D92B 53 0451 FCC /SLD/
D92E D7 0452 FCB 'W+12B
D92F 58 0453 FCC /XE/
D931 D1 0454 FCB 'Q+12B
D932 41 0455 FCC /AUT/
D935 CF 0456 FCB 'D+12B
D936 53 0457 FCC /SWA/
D939 D0 0458 FCB 'P+12B
D93A 45 0459 FCC /ERROR/
D93F D3 0460 FCB 'S+12B
D940 42 0461 FCC /BAU/
D943 C4 0462 FCB 'D+12B
D944 4C 0463 FCC /LD1/
D947 D2 0464 FCB 'R+12B
0465 *
0466 IFDF PARPRT ASSEMBLE FOR PARALLEL PORT
0467 *KEEP THIS LAST IN LIST FOR TOKEN COMPATABILITY
D948 50 0468 FCC /PARALLE/ (REF 6)
D94F CC 0469 FCB 'L+12B (REF 7)
0470 ENDC
0471 * (REF 6 & 7: If no conditional assembler and
0472 * parallel port is used, delete IFDF and ENDC
0473 * lines. If not used, delete all 4 lines.)
0474 *
0475 *****
0476 * COMMAND JUMP TABLE
    
```

```

0477 * MUST BE IN SAME ORDER AS COMMANDS
0478 *
D950 0479 CTABLE EQU * TABLE START
D950 DA4E 0480 COMDSP FDB COLD COLD RESTART
D952 DA4F 0481 FDB WPOKE
D954 DA50 0482 FDB FAST
D956 DA51 0483 FDB SLDW
D958 DA52 0484 FDB XEQ
D95A DA53 0485 FDB AUTO
D95C DA55 0486 FDB SWAP
D95E DA54 0487 FDB ERRCMD
D960 DA56 0488 FDB BAUD
D962 DA57 0489 FDB LDIR PRINT DIRECTORY
0490 *
0491 *KEEP THIS LAST IN LIST FOR TOKEN COMPATABILITY
0492 IFDF PARPRT ASSEMBLE FOR PARALLEL PORT
D964 DA58 0493 FDB PARA (REF 6)
0494 ENDC
0495 * [REF B: If no conditional assembler and
0496 * parallel port is used, delete IFDF and ENDC
0497 * lines. If not used, delete all 3 lines.]*
D966 0498 CTBLEX EQU * TABLE END
0499 *****
0000 0500 NUMCMD EQU (CTBLEX-CTABLE)/2 NO. OF CMDS
0501 *****
0502 * FUNCTION TABLE
0503 *
D966 53 0504 FUNTBL FCC /SCAN/
D96A A4 0505 FCB '*+12B
D96B 44 0506 FCC /DATE/
D96F A4 0507 FCB '*+12B
D970 45 0508 FCC /ELIN/
D974 C5 0509 FCB 'E+12B
D975 45 0510 FCC /ECOD/
D979 C5 0511 FCB 'E+12B
0000 0512 IFEQ REV
D97A 45 0513 FCC /ENAME/
D97F A4 0514 FCB '*+12B
0515 ENDC
D980 57 0516 FCC /MPEE/
D984 CB 0517 FCB 'K+12B
0518 *****
0519 * FUNCTION JUMP TABLE
0520 *
D985 0521 NTABLE EQU * FUNCTION TABLE START

```

```

D985 DA59 0522 FUNDSP FDB SCAN
D987 DA5A 0523 FDB DATE
D989 DA5B 0524 FDB ERRLIN
D98B DA5C 0525 FDB ERRCOD
0000 0526 IFEQ REV
D98D DA5D 0527 FDB ERNAME
0528 ENDC
D98F 0529 ARGHRK EQU *
0530 * put all functions without an argument above
0531 * this equate
D98F DA5E 0532 FDB MPEEK
D991 0533 NTBLEX EQU * TABLE END
0534 *****
0006 0535 NUMFUN EQU (NTBLEX-NTABLE)/2 NO. OF FUNCTS
0536 *****
0537 * THIS IS EXECUTED DURING STARTUP
0538 *
0539 * Output revision banner
D991 BEDA0F 0540 ADDCGM LDX #BANNER-1 POINT TO BEFORE BANNER
D994 BDB99C 0541 JSR #B99C USE BASIC'S OUTPUT ROUTINE
0542 *****
D997 7F0149 0543 CLR AUTOFG SET UP FOR NO AUTO
D99A 7F014A 0544 CLR INTFLG OLD LINE REPEAT FLAG
D99D BED91E 0545 LDX #COMTBL POINT X TO COMMAND TABLE
D9A0 CE013E 0546 LDU #013E START OF COMMAND VECTOR TABLE
D9A3 AF41 0547 STX 1,U SAVE COMMAND TABLE ADDRESS
D9A5 B60B 0548 LDA #NUMCMD GET NUMBER OF COMMANDS
D9A7 A7C4 0549 STA ,U SET IT IN TABLE
D9A9 BEDA28 0550 LDX #COMCOD COMMAND CODE
D9AC AF43 0551 STX 3,U
0552 *****
D9AE B606 0553 LDA #NUMFUN GET NUMBER OF FUNCTIONS
D9B0 A745 0554 STA 5,U SAVE IT IN TABLE
D9B2 BED966 0555 LDX #FUNTBL GET FUNCTION TABLE ADDRESS
D9B5 AF46 0556 STX 6,U
D9B7 BEDA37 0557 LDX #FUNCOD GET FUNCTION CODE ADDRESS
D9BA AF48 0558 STX 8,U
D9BC 674A 0559 CLR 10,U SET END OF TABLES FLAG
D9BE 8EB277 0560 LDX #0B277 ?SN ERROR
D9C1 AFCB12 0561 STX 18,U STORE IN NEXT HOOK SLOTS
D9C4 AF4D 0562 STX 13,U FOR COMS & FUNCT.
D9C6 674F 0563 CLR 15,U SET TOKEN GROUP TO ZERO
D9C8 9EBA 0564 LDX ZERO
D9CA AFCB10 0565 STX 16,U CLEAR DATUM
0566 * JSR RESET ERROR TRAP VALUES (REF 9-1)

```

Forget Those Point Spread Blues!



With Pigskin Predictions!

Pigskin Predictions, the best-selling NFL Handicapper from Rainbow Connection Software, is now part of our library. And we're absolutely delighted! Why wrestle with those Sunday point spreads? Let your Color Computer do the work for you! And what a job it does:

- Menu-driven selection of schedules, ratings, division races, predictions or results by team or week. Seven different reports to screen or printer!
- Easy once-a-week entry of scores—no complex, meaningless stats!
- Predicts scores of all games for remainder of season each week!
- Calculates projected won-lost records for all weeks.
- Maintains home field advantage and power ratings for all teams!
- 1984 schedule data file included free. Or enter the schedule yourself.
- 32K enhanced version features dazzling Rainbow Writer screen display! Seeing is believing. 16K abridged version included too.

If you're a football fan, you'll be absolutely amazed at the power of this program. 16/32K Extended Basic required. Only \$35.95 on tape or disk. 1984 data tape or disk for previous owners only \$13.95.

Federal Hill Software 825 William St. Baltimore, Md. 21230 301-685-6254 VISA/MC Welcome


```

0567 * REDIRECT ERRORS TO ERRTRP BY CHANGING JUMP ADDRESS
0568 *AT $18F
0569 * LDD @ERRTRP [REF 9-2]
0570 * STD $18F [REF 9-3]
0571 * [REF 9: Uncomment when ERRORS code is installed]
0572 *
0573 *
0574 * IFDF PARPRT DO FOR PARALLEL [REF 10]
0575 * [REF 10 & 11: If no conditional assembler and
0576 * parallel port is used, delete IFDF and ENDC
0577 * lines. If not used, delete these and
0578 * all lines in between.]
0579 * REDIRECT CALLS FOR OUTPUT VIA [A002] A2B2
0580 * TO ALLOW PARALLEL PORT OPERATION.
0581 * LDD @PAROUT PARALLEL PORT ROUTINE
0582 * STD $168
0583 * NOW INITIALIZE PARALLEL PORT
0584 * *****
0585 * BASIC PATCH FOR PARALLEL OUTPUT
0586 * *****
0587 *
0588 *
0589 * THE UART BAUD RATE MSB ($95) IS SET TO 1 TO
0590 * ACTIVATE THE PARALLEL INTERFACE. SET TO ZERO
0591 * FOR THE SERIAL OUTPUT. THIS MEANS 300 BAUD AND
0592 * HIGHER WILL ACTIVATE THE SERIAL PORT, 110 OR LOWER
0593 * WILL ACTIVATE THE PARALLEL PORT.
0594 * THIS IS THE DEFAULT CONDITION.
0595 * *****
0596 * PIA LAYOUT
0597 * BIT 0 UNUSED INPUT
0598 * BIT 1 UNUSED INPUT
0599 * BIT 2 UNUSED INPUT
0600 * FF24 BIT 3 UNUSED INPUT
0601 * BIT 4 UNUSED INPUT
0602 * BIT 5 UNUSED INPUT
0603 * BIT 6 UNUSED INPUT
0604 * BIT 7 PRINTER BUSY=1
0605 *
0606 * FF25 SET TO $4 FOR ALL INPUTS
0607 *
0608 * BIT 0 PARALLEL OUTPUT
0609 * BIT 1 PARALLEL OUTPUT
0610 * BIT 2 PARALLEL OUTPUT
0611 * FF26 BIT 3 PARALLEL OUTPUT
0612 * BIT 4 PARALLEL OUTPUT
0613 * BIT 5 PARALLEL OUTPUT
0614 * BIT 6 PARALLEL OUTPUT
0615 * BIT 7 PARALLEL OUTPUT
0616 *
0617 * FF27 SET TO $2C FOR OUTPUTS & CB2
0618 *
0619 * BUSY IS ALSO CONNECTED TO CB1 BUT NOT USED
0620 * PIA DETECTS BUSY TO NOT BUSY. TRANSITION
0621 *
0622 * SET UP PIA FOR PARALLEL PORT
0623 * LDX @DATA POINT X TO PIA
0624 * LDA @0FF
0625 * STA ,X SET DATA DIRECTION REG TO $FF
0626 * LDA @2C SET FOR AUTO STROBE
0627 * STA 1,X CONTROL REGISTER
0628 * LDA @4 SET UP BUSY PIA
0629 * STA -1,X POINT FF24 TO DATA REG
0630 * SET UP OF PIA COMPLETE
0631 * SET UP DEFAULT BAUD RATE
0632 * LDD @1CA BASICS 120 BAUD
0633 * STD BDFLAG SET VALUE
0634 * ENDC END CONDITIONAL (REF 11)
0635 *
0636 * *****
0637 * RUN AUTOEXEC FILE
0638 *
0639 * LDX @AUTFIL POINT X TO COMMAND LINE
0640 * LDU @2DD BASIC INPUT BUFFER
0641 * LDB @FILEND-AUTFIL NUMBER OF CHARACTERS
0642 * PSHS B,U SAVE COUNT AND BUFFER PTRN
0643 * JSR @A59A MOVE X TO U B BYTES
0644 * LDA @635 WARM FLAG
0645 * STA @71 SET IT
0646 * JSR @B95C SET O/P PARAMETERS
0647 * PULS B,X CHAR COUNT * BUF PTR IN X
0648 * LEAX -1,X BACK OFF POINTER
0649 * JMP @AC7F STARTUP BASIC
0650 * RETURN TO BASIC ROM
0651 * AUTFIL FCS /RUN"AUTOEXEC"/ # BYTE ENDED
0652 * FILEND EQU *
0653 * BANNER FCS /REV(C)1984 C.STEARMAN@D<D>/
0654 * *****
0655 * COMMAND CODE
0656 * This is executed during token interpretation
0657 * to jump to correct code

```

```

0658 *
DA2B B1E8 0659 COMCOD CMPA @HITOKN+NUMCMD HIGHEST LEGAL CODE
DA2A 2303 0660 BLS GOODVL BOT A GOOD VALUE
0661 *
DA2C 7E277 0662 SNERR JMP >08277 ?SN ERROR JUMP
0663 *
DA2F BED950 0664 BODDVL LDX @COMDSP POINT TO DISPATCH TABLE
DA32 B0E1 0665 SUBA @HITOKN+1 LOWEST TOKEN IN RANGE
0666 * MAKES A HAVE OFFSET INTO DISPATCH TABLE
DA34 7EADD4 0667 JMP >@ADD4 CALCULATE AND EXECUTE IT
0668 * *****
0669 * FUNCTION CODE
0670 * This is executed during token interpretation
0671 * to jump to correct code
0672 *
DA37 C15A 0673 FUNCOD CMPB @4E+(2*NUMFUN)
DA39 22F1 0674 @HI SNERR BAD CODE
DA3B C050 0675 SUBB @50 LOWEST FUNCTION NUMBER
DA3D C10B 0676 CMPB @ARBMRK-NTABLE-2 Number of functions not
0677 * requiring an argument, X 2 +2
0678 *
0679 * ACTUAL TOKEN IS 50/2 + B0 = AB
DA3F 2F07 0680 BLE NOARB FIRST FUNCTIONS HAVE
0681 * NO ARGUMENT
0682 * ALL OTHERS DO AND ITS OBTAINED
0683 * FIRST HERE
DA41 3404 0684 PSHB B SAVE TOKEN OFFSET
DA43 BDB262 0685 JSR @B262 EVAL BRACKETTED ARGUMENT
DA46 3504 0686 PULS B RESTORE OFFSET
DA4B BDB9B5 0687 NOARB LDX @FUNDSP POINT TO FUNCT. DISPATCH TABLE
DA4D 7E82CE 0688 JMP @B2CE GO LOOKUP AND JUMP
0689 * *****
DA4E 39 0690 COLD RTS [REF 12]
DA4F 39 0691 WPOKE RTS [REF 13]
DA50 39 0692 FAST RTS [REF 14]
DA51 39 0693 SLOW RTS [REF 15]
DA52 39 0694 XEQ RTS [REF 16]
DA53 39 0695 AUTO RTS [REF 17]
DA54 39 0696 ERRCMD RTS [REF 18]
DA55 39 0697 SWAP RTS [REF 19]
DA56 39 0698 BAUD RTS [REF 20]
DA57 39 0699 LDIR RTS [REF 21]
DA58 39 0700 PARA RTS [REF 22]
DA59 39 0701 SCAN RTS [REF 23]
DA5A 39 0702 DATE RTS [REF 24]
DA5B 39 0703 ERRLIN RTS [REF 25]
DA5C 39 0704 ERRCOD RTS [REF 26]
DA5D 39 0705 ERNAME RTS [REF 27]
DA5E 39 0706 WPEEK RTS [REF 28]
DA5F 39 0707 PAROUT RTS [REF 29]
0708 *
0709 *
0710 *
0711 * ZLAST EQU *-1 last used address value
0712 *
0713 * ZLAST must not be greater than $DFFF for
0714 * DOS 1.0 and $DEFF for DOS 1.1. The latter
0715 * has the OS-9 Boot program and SWI set routines
0716 * from $DF00 to $DFAC
0717 *
0718 *
0727 * OPT LIS
0728 * END ADDCOM
0729 *
NO ERROR(S) DETECTED

```

Listing 2:

```

10 ' DATE LOADER
11 DIM DAYS(12)
12 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
13 FOR I=1 TO 12
14 READ DAYS(I)
15 NEXT
30 INPUT "DATE (MM, DD, YY) "; M, D, Y
50 IF M<0 OR M>12 THEN 1000
70 IF Y<0 THEN 1000
80 IF D<1 THEN 1000
90 IF M=2 THEN 120
100 IF D>DAYS(M) THEN 1000 ELSE 150
110 ' DO FEBRUARY
120 IF (INT(Y/4) (> Y/4) AND (D) DAYS (M)) THEN 1000
130 ' LEAP YEAR
140 IF D>29 THEN 1000
150 DATE =(Y*INT(2^9)) + (M*INT(2^5)) + D
160 WPOKE &H14E, DATE
170 END
1000 PRINT "ERROR": GOTO 30

```

COOKING With CoCo



Part IV

This month the chef serves up a tasty appetizer to make keyboard entry deliciously easy.

By Colin J. Stearman

If you were paying close attention last month, you might have noticed I included a couple of items in the patch listing which were not mentioned in the text of the article. These were put in at the last minute due to the overwhelming number of reader requests for them. Before we get started on this month's feature, I will describe what they were.

DECB 1.1

It seems more of you have the new revision of Disk BASIC than I imagined, and were frustrated by this series being based on the 1.0 revision. Well fret no more, as the part three listing contains patch addresses for both revisions. I have used MAC's conditional assembly to select which revision to assemble. If the label *REV* is zero then the 1.0 version is built and if it's one then 1.1 is built. The listing each month will be assembled for 1.0, but all information will be included regarding what to change for 1.1.

DECB 1.1 takes up more room in the ROM than does 1.0, so I have had to leave some features out. First to go is the fix to the *FILES* command. I haven't checked, but would like to think that 1.1 fixed that bug itself. Second, the fully spelled out error messages and return of the error message name in

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

ENAME\$ had to go. These seemed like the least important, but if you disagree, leave something else out and include them. But whatever you do, don't let the additions go beyond *\$DEFF*. The OS-9 boot routine resides at *\$DF00* through *\$DF4C*.

Finally, each month *RAINBOW ON TAPE* will have the machine code file for both revisions of *BASIC*. The name of the file will be built from the initials of the article, the part number and then *V10* for *DECBI.0* and *V11* for *DECBI.1*. So this month the files will be *CWC4V10* and *CWC4V11*.

Drive Step Rate

Many of you have disk drives that can step from track to track at a rate faster than the 30 ms (milliseconds) set by *BASIC*. Even my old *RS* drives can step at 20 ms.

If you look at last month's listing Lines 225 through 232, you will see that I adjusted the rate to 20 ms. That's why your drives sounded a little strange. If you had problems maybe you should set this back to 30 ms.

There are four possible settings: 30 ms, 20 ms, 12 ms and 6 ms. This patch will affect all your drives equally, so set the value to that of the slowest drive, if you have a mix. I have patched both the *RESTORE* rate and *SEEK* rate. The first sets the rate at which the drive is restored to track zero; the second, the rate at which each track is sought. I toyed with making a command to allow *BASIC* to change the rate "on the fly." But that takes up precious *ROM* space and you would always want the fastest rate your drives can handle. If you don't know how fast your drives are, keep reducing the rate until a *LOAD* command fails, then go back a notch.

Back to Business

Last month we ended the assembly code listing with a series of dummy functions. Next month we will add the code to make some of them functional. But this month we introduce *FLEXIKEY*.

Hands up all of you *CoCo* keyboard-pounders who have just entered a long direct command to *BASIC*, only to notice a "typo" in the second character. I guess I'm not alone! With *FLEXIKEY* you can instantly save the bad line, recall it for editing and re-execute it. You never have to type in the same thing twice. I must confess, the idea came from my *IBM PC* at work, which has similar functions.

FLEXIKEY

The *FLEXIKEY* routine completely replaces *BASIC*'s normal keyboard entry routine and places each entered *BASIC* line into a buffer when you press the *ENTER* key. This entry is then recallable for re-execution or modification by a set of simple commands.

The best way to describe how it works is by example. Let's say you have just typed in the command

```
COPY"OLP.PGM" TO "NEW.PGM"
```

and *ENTER*ed it. It returned an *?NE* error because you meant to type *OLD.PGM*. Instead of retyping the whole line, use the right arrow key to recall each letter from the buffer. Pressing it seven times will recall

```
COPY"OL
```

with the cursor just after the 'L'. Now type in the 'D'. This replaces the incorrect 'P'. You could get the rest of the line out by repeatedly pressing the right arrow, but if you press *SHIFT*/*right arrow* the remainder of the line appears, with

the cursor at the end. If you were to press *ENTER*, then this line would be put into the holding buffer and executed also.

But let's say that just as you were about to press *ENTER* you realized that the proper program name was *VERY OLD.PGM*. You could press *ENTER* anyway and get another error and then edit again, but if you press *SHIFT*/*@* the command line will be stored in the buffer without execution, ready for further editing. When you do this a '@' is displayed at the end of the line to remind you that the command was just stored and not executed.

So you do this and then press the right arrow five times to recall *COPY*". To insert the *VERY*, press the *SHIFT*/*up arrow*. This puts you into the insert mode and each character typed will be inserted in the command line, with the remaining characters in the buffer not overwritten. The overtype mode is returned whenever you press a left, right or down arrow key. Once *VERY* is typed, the *SHIFT*/*right arrow* key will recall the remainder of the line for entry.

But once again you get an *?NE* error because the name of the file was really *VERY.PGM* (will you ever get it right?). Press the right arrow key nine times until *COPY"VERY* is displayed. Now press the down arrow key three times, once for each letter in *OLD*. *SHIFT*/*right arrow* will then spit out the rest of the line which now reads

```
COPY"VERY.PGM" TO "NEW.PGM"
```

If you are editing a line and things get really scrambled, don't worry, just hit left arrow to delete the character to the left of the cursor. The original character at that position is still in the buffer and could be pulled out with right arrow. If the whole line is messed up, press *SHIFT*/*left arrow* and the whole thing will disappear. But the original line is still in the buffer so you can start all over.

Some of the arrow keys now used by *FLEXIKEY* previously created printable characters (square brackets, left arrow and the like). To get these now, press *SHIFT*/*CLEAR* and then the arrow key you want. The normal character will appear. To get the back slash which *SHIFT*/*CLEAR* normally produces, press *SHIFT*/*CLEAR* twice.

FLEXIKEY does not interfere with the normal operation of *BASIC*'s *EDIT* command. It works in the command mode and also within *BASIC* programs when entry is via an *INPUT* command. Also, some machine language programs use *BASIC*'s entry routine, and therefore *FLEXIKEY* is available for use within them also. (Computerware's *MACRO* assembler *MAC* falls into this category, for one.)

The buffer used by *FLEXIKEY* is the cassette buffer, so correct operation will not occur immediately after cassette input/output operations. It does not interfere with this *I/O*, it's just that they share a common buffer area.

As I said earlier, once you get used to remembering *FLEXIKEY* is there, you'll wonder how you ever managed without it.

Adding The New Functions

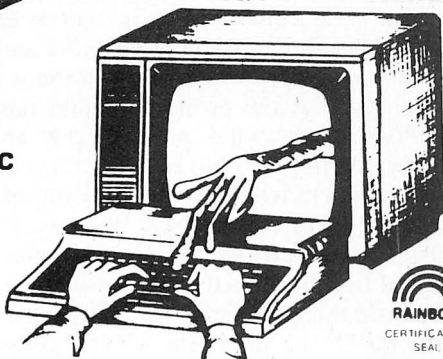
This is a simple process using your editor. Call in last month's listing and make the following changes using the [*REF#*] given as a locating guide. 'Uncomment' (remove the initial asterisk from) reference Line 1 and delete all lines after reference Line 29, as these are in this month's listing.

Type in the additional code in Listing 1 at the end of the existing code. Then reassemble the result and try it as you did last month's listing. You should find that *FLEXIKEY* works as described. If not then it's "hunt the typo" time, until it does.

NEW

E.T.T. ELECTRONIC TYPING TEACHER

by
CHERRYSoft



Learning to type the right way can save you hours of tedious work when entering programs into your CoCo, and this is just what ETT was designed to do. Devote a little time every day practicing with ETT and before you know it you will be typing with confidence. Entering those programs will no longer be the chore it used to be.

ETT's video keyboard lets you practice with all the keys labeled, all the keys blank or only the "home" keys labeled. The visual cues guide you while you learn to type without watching your fingers. ETT shows your accuracy, response time, and words per minute. You will quickly see that you are improving with practice.

With the sentences provided by ETT learning to type can be fun. Over 1000 variations chosen because they include every letter in the alphabet. You can also create your own practice sets. This outstanding program was written by a certified teacher and professional programmer and comes with a ten page student manual-study guide. Requires 16K Extended Basic.

Cassette **\$21.95**

ETT NOW AVAILABLE FOR COMMODORE 64
CASSETTE **\$24.95** DISK **\$29.95**

NEW

MASTER CONTROL II

The best doesn't always cost more and MASTER CONTROL II is a good example. What would you be willing to pay for a program that would cut your typing time by more than 50% and eliminate hours of debugging because you misspelled a command word? For example the command STRING\$ (requires nine strokes) with MASTER CONTROL II you only require two strokes, just hit the down arrow key twice and it's done, and no mistakes. That is just one of the 50 pre-programmed commands available to you. If that isn't enough you also have the ability to customize your own key to enter a statement or command correctly, automatically every time. But that's not all, how about automatic line numbering. Just enter the starting number and the increment you want and MASTER CONTROL II will do it for you. You also have direct control of MOTOR, AUDIO and TRACE plus a direct RUN key. Sounds great? Well, thousands of color computer owners have been enjoying these features for years. But now the new MASTER CONTROL II also has the following features:

- ::New plastic overlay that can be removed when you are not using MASTER CONTROL II.
- ::New documentation, to help you get the most from the program.
- ::New repeating keyboard.

Cassette **\$21.95**

Include \$2.50 Shipping and Handling in U.S.-\$5.00 Foreign

MasterCard **VISA** **CoCo**
Warehouse

Where Shopping By Mail is "USER FRIENDLY"
500 N. DOBSON - WESTLAND, MI 48185
Phone (313) 722-7957

**FREE
CATALOG**

**DEALER
INQUIRIES
INVITED**

EDTASM+ Bug

A bug in EDTASM+ can cause you problems. If your assembly creates *Multiply Defined Symbol* errors when you know there aren't any, then the bug bit you! It manifests itself when you use arithmetic in the operand field, and the math references a label.

For example, in the program SYSTEM from part one, EDTASM+ does not like the line *CMPU#BUFFER+256*, but if you change it to *CMPU#256+BUFFER* it likes it just fine. So look for lines like this before tearing all your hair out!

A Gentle Reminder

When you have transferred BASIC (unmodified or otherwise) to a disk or an EPROM using information in this series, the result is *still copyrighted* by RS and Microsoft. Giving the disk or EPROM away or selling it to others infringes on this and is illegal.

None of my patch code contains original RS BASIC code and is itself copyrighted. However, it may be freely distributed as long as my copyright notice remains intact, both in the source code and in the start-up banner. My revisions may not be sold for profit without my written consent.

Coming Next Month

We will add the code to make many of the new BASIC commands fully functional, including *COLD* and *AUTO* and *DATE\$*. So let's make it a date\$!

If you would like the entire DOSPATCH program source, along with binary files with and without the parallel port driver for DECB 1.0 and DECB 1.1, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

*Address this request or any questions to:
Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.*

The listing:

```

0700      OPT LIS
0709 *****
0710 * PATCH #2 to RSDOS (C)1984 Colin Stearman *
0711 *****
0712 *
0713 *****
0714 *          FLEXIKEY
0715 **      BASIC LAST LINE RE-ENTRY AND EDIT ROUTINE
0716 * This is not a callable command, but a set of
0717 * direct commands from the keyboard, to allow access
0718 * to the last command entered. It is designed to
0719 * work only when called from BASIC and does not
0720 * interfere with the EDIT commands.
0721
0722 *          COMMANDS ARE:
0723 *
0724 * LEFT ARROW -      output next character of old line
0725 * SHIFT/LEFT ARROW- output rest of old line
0726 * SHIFT/UP ARROW - insert, no old line increment
0727 * DOWN ARROW -     delete next character in buffer
0728 * SHIFT/E -        store line input so far.
0729 *                  No interpretation

```

```

0730 *
0731 *****
0732 * GENERAL PRINCIPLE OF OPERATION:
0733 *
0734 * To allow access to special keyboard entries the
0735 * RAM hook at $16A is modified to go to this routine.
0736 * If the device is 0, the keyboard,
0737 * the key and cursor are obtained and output from
0738 * here. The special keys interpreted and characters
0739 * are drawn from this as required. One permanent RAM
0740 * location is used to indicate the need to initialize
0741 * pointer.
0742 *
0743 * At the end the old return is removed from the stack
0744 * so it is not taken. This allows the input
0745 * handling routine to handle the character as normal.
0746 *
0747 * Because SHIFT/UP ARROW & SHIFT/RIGHT ARROW are also
0748 * used to create the left arrow and ), these are
0749 * now obtained by pressing SHIFT/CLEAR first.
0750 * As this is the backslash this can be obtained by
0751 * pressing SHIFT/CLEAR twice.
0752 *
0753 * FLAGS:
0754 * INTFLG 0 = line in BASIC buffer just stored
0755 *          FF = line in hold buffer in use
0756 * HLDPTR zero-based pointer into hold buffer
0757 * INSERT 0 = Insert mode off
0758 *          FF = Insert mode on
0759 * WHLINE 0 = SHIFT/RIGHT ARROW not previously pressed
0760 *          FF = SHIFT/RIGHT ARROW previously pressed
0761 *
0762 *****
0763 *
0764 KEYBRD LDA DEVNUM
0765 BEQ KEY DEVICE IS KEYBOARD

---

0766 * SEE IF CASSETTE I/O GOING ON
DA64 61FF 0767 CMPA #1 CASSETTE DEVICE CODE
DA66 2605 0768 BNE JMPOUT NOT CASSETTE SO DO NOTHING
DA68 8601 0769 LDA #1
DA6A 87014A 0770 STA INTFLG MAKE FLAG POSITIVE
DA6D 7EC5BF 0771 JMPOUT JMP CHRVRT CONTINUE OLD CODE
0772 *****
0773 *
0774 KEY PSHS B,X PRESERVE REG VALUES
DA72 AE67 0775 LDX 7,S SEE IF CALLED FROM IDLE LOOP
DA74 BCA39D 0776 CMPX #A39D IDLE LOOP CALL RETURN ADDRESS
DA77 2704 0777 BEQ INIDLE IN THE IDLE LOOP
DA79 3514 0778 PULS B,X FLAGS NOT AFFECTED
DA7B 20F0 0779 BRA JMPOUT IS NOT IDLE LOOP
0780 * THIS ENTRY LINE RECALL WILL ONLY FUNCTION
0781 * WHEN IN THE BASIC IDLE LOOP
0782 *
DA7D 0F70 0783 INIDLE CLR #70 FLAB BUFFER FLUSHED
DA7F 7D014A 0784 TST INTFLG HAVE WE BEEN HERE SINCE
0785 * LAST <CR>?
DA82 270A 0786 BEQ GETTKN NO CLEAR THE FLAGS
0787 * YES SEE IF CASSETTE I/O JUST DONE
DA84 2B2B 0788 BMI TESTWH NO SO CONTINUE
DA86 7F01DA 0789 CLR HLDPTR SET FIRST BYTE IN HOLD=0
DA89 7F014A 0790 CLR INTFLG READY FOR COMPLEMENTING
DA8C 2000 0791 BRA GETTKN GO CLEAR FLAGS
0792 *
0793 *
0794 * FIRST TIME THROUGH SINCE <CR> SO SET UP
DABE 73014A 0795 GETTKN COM INTFLG SET FLAG TO %FF
0796 * CLEAR FLAGS
DA91 7F01D7 0797 RENTER CLR HLDPTR
DA94 7F01DB 0798 CLR INSERT
DA97 7F01D9 0799 CLR WHLINE
0800 *
0801 * READ CHARACTER FROM KEYBOARD

DA9A BDA1B1 0802 *
0803 KYREAD JSR GETKEY RETURNS KEY IN A
0804 *
0805 * NOW SEE WHAT WE GOT
0806 *
DA9D B109 0807 CMPA #109 RIGHT ARROW next character
DA9F 2715 0808 BEQ GETCHR GO DO IT
DAA1 B15D 0809 CMPA #15D SHIFT/RT ARROW rest of line
DAA3 2605 0810 BNE J1 NOT THIS
DAA5 7301D9 0811 COM WHLINE SET WHOLE LINE FLAG
DAA8 200C 0812 BRA GETCHR BET NEXT BUFFER CHARACTER
DAAA B15F 0813 J1 CMPA #15F SHIFT/UP ARROW insert toggle
DAAC 261F 0814 BNE J2 NOT THIS
DAAE 7301DB 0815 COM INSERT TOGGLE INSERT FLAG
0816 *
0817 * SEE IF SHIFT/RT ARROW PREVIOUSLY PRESSED
0818 TESTWH TST WHLINE OUTPUT WHOLE LINE IF SET
0819 BEQ KYREAD NO SO READ KEYBOARD
0820 *****
0821 * GET CHARACTER FROM HOLDING BUFFER
0822 GETCHR CLR INSERT RESET INSERT FLAG
0823 LDB HLDPTR BET POINTER
0824 LDX #HLDPTR POINT X TO HOLDING BUFFER

---
0825 LDA B,X BET CHARACTER
0826 BNE GOODCH
0827 * ALL BUFFER IS OUT
0828 CLR WHLINE RESET POINTER
0829 BRA KYREAD IGNORE
0830 * GOT GOOD CHARACTER
0831 GOODCH INC HLDPTR MOVE PAST CHARACTER
0832 BRA EXIT AND RETURN WITH IT
0833 *****
0834 J2 CMPA #113 SHIFT/# close line
0835 BEQ LINCLS GO TO LINE CLOSE
0836 CMPA #10D RETURN enter
0837 BEQ ENTER
0838 CMPA #10B BACKSPACE delete last char
0839 BEQ J4
0840 CMPA #10A DOWN ARROW delete next char
0841 BNE J3
0842 JSR INCPTR INCREASE HOLD POINTER
0843 BRA KYREAD JUMP BACK TO KEY READING
0844 *
0845 * HANDLE BACKSPACE IF INSERT OFF
0846 * DECREASE HLDPTR
0847 J4 TST INSERT
0848 BNE CONXIT ON SO DON'T DECREMENT
0849 BSR DECPNT CONDITIONAL DECREMENT HLDPTR
0850 BRA CONXIT GO TO CONDITIONAL EXIT
0851 *****
0852 DECPNT TST HLDPTR
0853 BEQ ATZERO ALREADY ZERO
0854 DEC HLDPTR REDUCE HLDPTR BY ONE
0855 ATZERO RTS
0856 *****
0857 J3 CMPA #115 SHIFT/BCKSP clear to start
0858 BEQ CLRPNL GO CLEAR HLDPTR
0859 CMPA #10C CLEAR
0860 BEQ CLRPNL DITTO
0861 CMPA #103 BREAK
0862 BEQ CLRPNL YES SO RESET HLDPTR AND EXIT
0863 CMPA #15C SHIFT/CLEAR special insert
0864 BNE CONXIT NO SO CONDITIONALLY EXIT
0865 JSR BETKEY GET ANOTHER KEY
0866 BRA CONXIT AND CONDITIONALLY EXIT
0867 *****
0868 CLRPNL CLR HLDPTR CLEAR HLDPTR
0869 *****
0870 CONXIT CMPA #120 CHECK FOR CONTROL CHARACTER
0871 BLO EXIT EXIT FROM ROUTINE
0872 * PRINTABLE CHARACTER SO SEE IF INSERT ON
0873 TST INSERT
0874 BNE EXIT

```

```

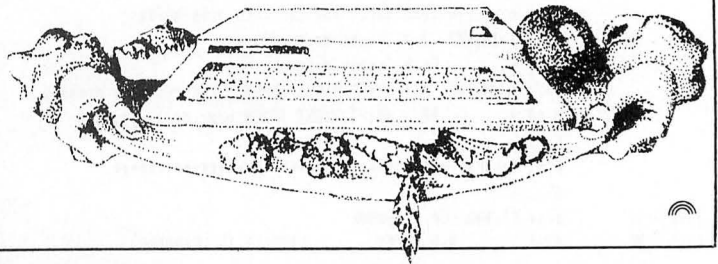
DB15 8D05 0875 BSR INCPTR INCREMENT HLDPTR
DB17 3514 0876 EXIT PULS B,X RECOVER INCOMING VALUES
DB19 3262 0877 LEAS 2,S CLEAN OLD RETURN OFF
DB1B 39 0878 RTS RETURN TO BASIC CALL
0879 *****
DB1C BE01DA 0880 INCPTR LDX #HLDDBFR POINT TO HOLDINGB BUFFER
DB1F F601D7 0881 LDB HLDPTR
DB22 6D85 0882 TST B,X BET CHARACTER IN HOLD
DB24 2703 0883 BEQ ZEROBT ZERO BYTE SO AT AT END
---
DB26 7C01D7 0884 INC HLDPTR
DB29 39 0885 ZEROBT RTS
0886 *****
0887 * DO SHIFT/0 LINE CLOSE
DB2A 6FF801 0888 LINCLS CLR [1,S] ZERO OUT LAST BYTE
0889 * 1,S IS X, THE PNTR IN THE BASIC INPUT BFR
0890 *
DB2D 8640 0891 LDA #*0 LOAD # SIGN
DB2F BDA282 0892 JSR CHR0UT OUTPUT IT
DB32 BDB95B 0893 JSR RETURN OUTPUT CARRIABE RETURN
DB35 C601 0894 LDB #1 RESET BASICS CHARACTER COUNT
DB37 E7E4 0895 STB ,S ON STACK
DB39 BE02DD 0896 LDX #BASBFR ALSO BUFFER POINTER
DB3C AF61 0897 STX 1,S ALSO ON STACK
DB3E 8D0E 0898 BSR MOVBLK TRANSFER INPUT BUFFER TO HOLD
DB40 7EDA91 0899 JMP RENTER RESET AND START OVER
0900 *****
0901 * DO ENTER
DB43 7F014A 0902 ENTER CLR INTFLG INDICATE BASIC BUFFER CHANGED
0903 *
0904 * CLEAR LAST BYTE IN BASIC INPUT BUFFER
0905 * FOR MOVE CODE TO DETECT IT
DB46 6FF801 0906 CLR [1,S]
DB49 8D03 0907 BSR MOVBLK TRANSFER INPUT BUFFER TO HOLD

```

```

DB4B 7EDB17 0908 JMP EXIT AND LEAVE
0909 *****
0910 * COPY BASIC INPUT BUFFER TO HOLD UNTOKENIZED
DB4E BE02DD 0911 MOVBLK LDX #BASBFR BET START OF BASIC BUFFER
DB51 10BE01DA 0912 LDY #HLDDBFR BET START OF HOLD BUFFER
DB55 E680 0913 DOMORE LDB ,X+
DB57 E7A0 0914 STB ,Y+
DB59 26FA 0915 BNE DOMORE NOT A ZERO BYTE YET
DB5B 39 0916 RTS
0917 *****
0918
0919
DB5B 0920 ZLAST EQU *-1 last used address value
0921 *
0922 * ZLAST must not be greater than $0FFF for
0923 * DOS 1.0 and $0DEF for DOS 1.1. The latter
0924 * has the OS-9 Boot program and SWI set routines
0925 * from $DF00 to $DFAC
0926 *
0927 *
0936 OPT LIS
D991 0937 END ADDCOM
NO ERROR(S) DETECTED

```



Educational Programs☆☆☆ for the TRS-80 Color Computer

All TRS-80 programs require Extended Basic.
Available for both tape and disk.

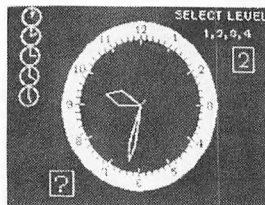
Used successfully in classrooms across the country on a daily basis, B-5 programs make learning fun! Each program can be geared to the individual needs of each student.

Instructive programs on:

- ☆ MATH FUNCTIONS
- ☆ LANGUAGE ARTS
- ☆ LEARNING TO COUNT MONEY
- ☆ LEARNING TO TELL TIME
- and more!

Priced from
\$9.95 to \$26.95

- ☆ Graphics
- ☆ Sound/Color
- ☆ Individualized Lessons
- ☆ Positive Feedback



Write today for a free catalog,
or ask for a dealer demonstration.

B-5 Software Co.
1024 Bainbridge Place
Columbus, Ohio 43228
Phone (614) 276-2752

Teachers: Have you written the "ultimate" program? We'd like to take a look . . .

Sonburst Software

233 S.E. ROGUE RIVER HWY.
GRANTS PASS, OR 97527 1-503-476-5977

The very best utilities for the 64K Disk Color Computer, featuring . . .

- Full use of 64K RAM
- 100% Machine Language
- Parameters easily changeable in basic loader
- No ROM calls
- "Cold start" exit to basic
- Easy-to-read, informative documentation
- Keyklik
- Selectable drive stepping rate
- Supports 1-4 drives



1. **EDT** — Professional programmers! Why struggle to make a word processor handle your Assembly files when you can let EDT's built-in features make life easy? Imagine tracking sub-routines 10 deep and then returning with a single key press!! Imagine 4 scroll speeds . . . all the way from a slow slide to the fastest hi-res text scroll ever on a CoCo — bar none! Imagine reviewing a disk file without disturbing the file in memory and then appending a line (or a block!) at the exact point you need it . . . not just at the end of the file! Plus a 51x24 screen, 2-way cursor, easy disk access, text files to 48K+, copy/save/move/delete blocks, optional type-ahead and more yet! Just remember this: If you're NOT using EDT — you're WORKING TOO HARD!! Special . **\$35.95**
2. **The Sector Inspector** — "VERY user friendly" — the Rainbow (Aug '84). 212 sectors in memory! Still the best . . . **\$29.95**
3. **The Deputy Inspector** — unique and still only . . . **\$21.95**
4. **The Archivist** — tape backup of your disks . . . **\$14.95**
5. **The Chief Inspector** — SAVE! All 3 disk utilities . . . **\$59.95**

• Please add \$1.50 for shipping, \$2.50 for C.O.D. •



PART V

By Colin J. Stearman

In which the CoCo kitchen will cook up something SLOW, FAST, and COLD.

It's time we got down to some BASIC cooking and add the code for many of the new commands.

New BASIC Commands

When you add the assembly language in Listing 1 to last month's listing (I will tell you how to do this shortly), it will add the following commands and functions:

COLD

This is a Reset command from the keyboard. When you issue it, any program in memory will be lost and BASIC will be "cold" started. This is useful if you have corrupted BASIC somehow and it performs exactly the same as entering the BASIC command `POKE &H71,0:EXEC&HA027`. The start-up banner will be displayed and the `AUTO-EXEC.BAS` file will be run.

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

WPOKE

This is like `POKE`, but is WORD oriented instead of byte. The syntax is the same as `POKE`, but the value can be anything from zero to 65535. This number is poked into the given address and the next address location.

FAST

Issuing this command puts CoCo into high gear and is exactly the same as `POKE65495,0`. You can run the disk system in the FAST mode if you remove capacitor C85 from the mother board. This is a 220pF capacitor on the "Cartridge Select Signal" at pin 32 socket and ground. A word of warning though: do not attempt any disk input/output while in the FAST mode, because it will surely fail!

SLOW

No prizes for guessing what this one does: it issues the equivalent of `POKE 65494,0` and should be performed whenever a FAST has been issued and disk input/output is required.

XEQ(M)

If you type in `XEQ"GAME"`, it is exactly the same as entering `RUN`

"GAME"; in other words the BASIC program "GAME.BAS" is retrieved from the disk and run. However, if you enter XEQM "GAME", then the machine code program "GAME.BIN" will be loaded from disk and started up. It's equivalent to entering LOADM "GAME":EXEC.

AUTO

This "direct only" command automatically generates BASIC program line numbers. If you just enter AUTO then the first line will be 10 and the increment will be 10. If you enter AUTO 100, for example, the first line number generated will be 100, with an increment of 10. If you enter AUTO 4,2 the first line number will be four with an increment of two. To exit the AUTO mode, either press BREAK or ENTER immediately after the line number.

SCANS

SCAN\$ is a function similar to INKEY\$. Its syntax is the same. However, SCAN\$ will wait for a key to be pressed rather than continuing on like INKEY\$. So, if you have a program Line 100 A\$=SCAN\$, the program will wait at Line 100 until a key is pressed, and the key value will be assigned to A\$.

DATES

This string function will return the current date stored in the computer. The format of the date is mm/dd/yy, for example 06/12/84. It is always eight

characters long. You can use DATES like any other string variable, including assigning it to another string variable with an "equals" statement, or manipulating it with MID\$, LEFT\$, etc. However, you cannot assign a new string value to it by having it on the left side of an equals sign.

Once this code has been added we can "uncomment" some lines from last month (details below), and the DIR command will now pause after the screen fills, awaiting any key to continue. Also, the creation date of each file will be displayed in the directory.

Listing 2 is a BASIC program called "DATESET.BAS" which sets the date and also dates any undated files on the disk. Files created before you patched BASIC can be dated this way and also any files created by machine language programs which do not use BASIC to open them. Files will be dated if their date fields in the directory contain \$0000 or \$FFFF. Files with legitimate dates will not be changed. I have this file on my main editor disk and renamed it "AUTOEXEC.BAS" so it runs every-time I start up.

WPEEK

This is the complement of WPOKE and will return the WORD stored at the given address and the next consecutive address. The value returned is in the range zero to 65535. The syntax is the same as for PEEK.

Adding The New Functions

Call in last month's listing and make the following changes using the [REF#] given as a locating guide. Remove the commenting asterisk from reference Lines 3 and 5. Then delete reference Lines 12 through 17, 23, 24 and 28. Also, delete the last four lines of last month's listing starting with the line "ZLAST EQU *-1", as these are in this month's listing.

Now type in the new assembly language code found in Listing 1. Finally, reassemble the result and try it as you did last month's listing. The commands and functions should all work as advertised. If not, double check all your typing or subscribe to RAINBOW ON TAPE!

Coming Next Month

The next installment will be devoted entirely to the construction of the parallel interface and the software to integrate it into BASIC. So clean up the CoCo kitchen and we'll go to it next month.

If you would like the entire DOS PATCH program source, along with binary files with and without the parallel port driver for DECB 1.0 and DECB 1.1, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly. Address this request or any questions to: Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

Listing 1:

```

>>>UNKNOWN MNEMONIC--
0917 OPT LIS
0918 *****
0919 * PATCH #3 to RSDOS (C)1984 Colin Stearman *
0920 *****
0921 *
0922 *****
0923 * "COLD" performs a cold restart
DB56 0F71 0924 COLD CLR 071 RESET COLD FLAG
DB58 7EA027 0925 JMP 0A027 RESTART BASIC
0926 *****
0927 * "WPOKE" COMMAND
DB58 BDB73D 0928 WPOKE JSR 0B73D GET 1ST ARGUMENT 0 TO FFFF
DB5E 9F2B 0929 STX 02B & SAVE TEMPORARILY
DB60 BDB26D 0930 JSR 0B26D PARSE OVER REQUIRED COMMA
DB63 BDB73D 0931 JSR 0B73D GET SECOND ARGUMENT
DB66 AF9F002B 0932 STX [02B] DO DOUBLE POKE
DB6A 39 0933 RTS RETURN TO BASIC
0934 *****
0935 * "FAST"
0936 *
DB68 B7FFD7 0937 FAST STA 65495 SPEED UP PROCESSOR
DB6E 39 0938 RTS
0939 *****
0940 * "SLOW"
0941 *
DB6F B7FFD6 0942 SLOW STA 65494 SLOW DOWN PROCESSOR

DB72 39 0943 RTS
0944 *****
0945 * "XEQ" COMMAND
DB73 B14D 0946 XEQ CMPA 0'M XEQM?
DB75 2703 0947 BEQ XEQM YES
DB77 7EAE75 0948 JMP 0AE75 NO - SAME AS RUN
DB7A BDCCE5 0949 XEQM JSR A0021 DO LOADM
DB7D 7FFF40 0950 CLR 0FF40 STOP DRIVE MOTOR
DB80 6E9F009D 0951 JMP [09D] EXEC
0952 *****
0953 * "AUTO n,i"
0954 *
0955 AUTO JSR DIRECT CURRENT BASIC LINE #
DB84 BDBB1B 0956 BNE SYNERR SYNTAX ERROR
DB87 266B 0957 LDD 000A DEFAULT LINE #
DB89 CC000A 0958 STD LINNUM SAVE IT
DB8C FD01D1 0959 STD INCNUM SAVE IT FOR INCREMENT TOO
DB8F FD01D3 0960 JSR <0A5 ANY MORE ON LINE?
DB92 9DA5 0961 BEQ NOMORE
DB94 271D 0962 JSR 0B73D EVALUATE ARGUMENT
DB96 BDB73D 0963 LDD <052 GET IT IN D
DB99 DC52 0964 STD LINNUM OVERRIDE DEFAULT LINE #
DB9B FD01D1 0965 JSR <0A5 ANY MORE VALUES?
DB9E 9DA5 0966 BEQ NOMORE
DBA0 2711 0967 JSR 0B26D PARSE COMMA
DBA2 BDB26D 0968 JSR 0B73D EVALUATE IT
DBA5 BDB73D 0969 LDD <052 GET IT IN D
DBA8 DC52 0970 BEQ SYNERR CANNOT BE ZERO
DBAA 2745 0971 STD INCNUM OVERRIDE DEFAULT
DBAC FD01D3 0972 JSR <0A5 ANY MORE ON LINE?
DBAF 9DA5 0973 BNE SYNERR ERROR IF SO
DBB1 263E

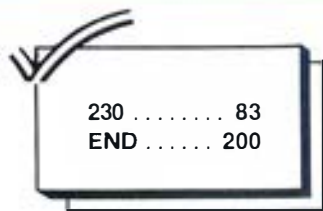
```



```

DBB3 86FF 0974 NOMORE LDA #0FF SET UP AUTO FLAG
DBB5 B70149 0975 STA AUTOFG
DBBB 39 0976 RTS ALL DONE
0977 *****
0978 * This is the trap routine to see if in
0979 * AUTO mode
0980 *
DBB9 7D0149 0981 INPUT TST AUTOFG AUTO MODE?
DBBC 270C 0982 BEQ INEXIT
0983 *****
DBBE FC01D1 0984 DOAUTO LDD LINNUM GET LAST LINE NUMBER
DBC1 1083F9FF 0985 CMPD #F9FF TOO HIGH?
DBC5 2304 0986 BLS NOTHI
DBC7 7F0149 0987 CLR AUTOFG RESET FLAG
DBCA 39 0988 INEXIT RTS RETURN
0989 *
0990 *****
DBC8 0F07 0991 NOTHI CLR #07 INKEY STORE
DBC9 0F08 0992 CLR #70 FLAG BUFFER FLUSHED
DBCF EDE4 0993 STD ,S D SAVE CURRENT VALUE OVER RETURN
DBD1 F301D3 0994 ADD INCNUM INCREMENT IT
DBD4 FD01D1 0995 STD LINNUM AND SAVE IT
DBD7 3506 0996 PULS D GET OLD VALUE OFF STACK
DBD9 BDBDCC 0997 JSR #BDDC DISPLAY NUMBER
DBDC 8620 0998 LDA #20 SPACE
DBDE BDA2B2 0999 JSR CHROUT DISPLAY IT
DBE1 CE03DA 1000 LDU #3DA WHERE CONVERTED # IS
DBE4 8E02DD 1001 LDX #BASBFR POINT TO BASIC BUFFER
DBE7 5F 1002 CLR# SET UP CHARACTER COUNTER
DBE8 A6C0 1003 ILOOP LDA ,U+ GET FIRST CHAR
DBEA 2708 1004 BEQ GOTNUM GET ALL NUMBERS
DBEC A780 1005 STA ,X+ MOVE TO BUFFER
DBEE 5C 1006 INCB COUNTER UP
DBEF 20F7 1007 BRA ILOOP CONTINUE
1008 * JUMP IS HERE SO EVERYONE CAN GET IT WITHOUT
1009 * LONG BRANCHING
DBF1 7EDA2F 1010 SYNERR JMP SNERR
1011 *
DBF4 8620 1012 BOTNUM LDA #20 SPACE
DBF6 A780 1013 STA ,X+ SAVE IT AT BUFFER END
DBF8 5C 1014 INCB COUNT IT
DBF9 BDA171 1015 JSR #A171 READ A CHARACTER
DBFC 810D 1016 CMPA #0D RETURN?
DBFE 2704 1017 BEQ ENDAUT END AUTO FUNCTION
DC00 8103 1018 CMPA #03 BREAK?
DC02 2609 1019 BNE INDONE NOT SPECIAL SO EXIT
DC04 7F0149 1020 ENDAUT CLR AUTOFG RESET FLAG
DC07 CC0D01 1021 LDD #0D01 GET A RETURN IN A, 1 CHR IN B
DC0A 8E02DD 1022 LDX #BASBFR POINT TO BUFFER START
DC0D 7EA39D 1023 INDONE JMP #A39D CONTINUE BASIC LOOP
1024 *****
1025 * "SCAN"
1026 *
DC10 9607 1027 SCAN LDA #07 HAS A KEY BEEN PRESSED?
DC12 2605 1028 BNE GOTKEY YES, RETURN WITH CODE
DC14 BDA1C1 1029 #SCAN JSR #A1C1 NO CALL KEY SCAN
DC17 27FB 1030 BEQ #SCAN KEEP LOOKING
DC19 7EA568 1031 GOTKEY JMP #A568 RETURN A 1 CHAR. STRING
1032 *****
1033 *
1034 * "DATE$"
1035 *
DC1C C608 1036 DATE LDB #0 CHARACTERS IN MM/DD/YY
DC1E BDB50F 1037 JSR #B50F VERIFY SPACE AVBLB, ALLOCATE
DC21 8D03 1038 * X IS RETURNED WITH ADDRESS OF STRING START
DC23 7EB69B 1039 BSR DATGET PUT CURRENT DATE AT B
1040 JMP #B69B EXIT VIA STRING# CODE
1041 *****
1042 * DATGET PUTS MM/DD/YY AT ADDRESS IN X BASED UPON
1043 * VALUE AT DATUM. DATE IS STORED AS FOLLOWS:
1044 * 15 - 9 8 - 5 4 - 0
1045 * YEAR (MOD1900) MONTH DAY
1046 DATGET LDD DATUM GET DATA FOR MONTH
1047 * ENTER BELOW WITH DATE ALREADY IN D
1048 DATOUT PSHS D SAVE ON STACK
1049 LSRA GET UPPER BIT IN CARRY
1050 RORB MOVE DOWN
1051 LSRB MOVE DOWN
1052 LSRB MOVE DOWN
1053 LSRB MOVE DOWN
1054 LSRB MOVE DOWN
1055 BSR DECODE PUT CHARACTERS IN BUFFER
1056 LDA #'/
1057 STA ,X+
1058 LDB 1,S GET DAY
1059 ANDB #X00011111 MASK OFF MONTH
1060 BSR DECODE
1061 LDA #'/
1062 STA ,X+
1063 LDB ,S GET UPPER BYTE
1064 LSRB POSITION YEAR DATA
1065 BSR DECODE GET CHARACTERS IN A,B
1066 LEAS 2,S REMOVE DATE FROM STACK
1067 RTS
1068 *
DC49 4F 1069 DECODE CLRA SET UP TENS COUNTER
DC4A C00A 1070 SUBTEN SUBB #10 REDUCE BY TEN
DC4C 2503 1071 BLO BOTTEN EXIT AS WENT NEG
DC4E 4C 1072 INCA INCREMENT TENS
DC4F 20F9 1073 BRA SUBTEN CONTINUE SUBTRACTING
1074 *
DC51 CB3A 1075 GOTTEN ADDB #10+0 RESTORE UNITS AND
DC53 8B30 1076 ADDA #0 TENS TO ASCII
DC55 ED81 1077 STD ,Y++ SAVE IN BUFFER
DC57 39 1078 RTS
1079 *****
1080 * "WPEEK"
1081 *
1082 *WPEEK RETURNS 2 BYTES
1083 WPEEK JSR #B740 INTEGRIZE PARSED VALUE
1084 LDD ,X DO DOUBLE PEEK
1085 UNSIGN STD #52
1086 JMP #B80E SEND UNSIGNED # TO VARIABLE
1087 *****
1088
1089
1090
DC61 1091 ?LAST EQU *-1 last used address value
1092 *
1093 * ?LAST must not be greater than $FFFF for
1094 * DOS 1.0 and $DEFF for DOS 1.1. The latter
1095 * has the OS-9 Boot program and SWI set routines
1096 * from $DF00 to $DF4C
1097 *
1098 *
1099 *
1100 *
1101 * OPT LIS
1102 * END ADDCOM
D994 NO ERROR(S) DETECTED

```



Listing 2:

5 * "DATESET.BAS" LISTING #2 COO
KING WITH COCO- PART 5

```

10 CLEAR 1000
20 * DATE LOADER
30 DIM DAYS(12)
40 DATA 31,28,31,30,31,30,31,31,
30,31,30,31
50 FOR I=1 TO 12
60 READ DAYS(I)
70 NEXT
80 IF WPEEK(&H14E)<>0 AND WPEEK(

```

```

&H14E)<>&HFFFF THEN 210
90 INPUT"DATE(MM,DD,YY)";M,D,Y
100 IF M<0 OR M>12 THEN 240
110 IF Y<0 THEN 240
120 IF D<1 THEN 240
130 IF M=2 THEN 160
140 IF D>DAYS(M) THEN 240 ELSE 1
90
150 ' DO FEBRUARY
160 IF (INT(Y/4)<>Y/4)AND(D>DAYS(
M))THEN 240
170 ' LEAP YEAR
180 IF D>29 THEN 240
190 DATE=(Y*INT(2^9))+(M*INT(2^
5))+D
200 WPOKE &H14E,DATE
210 INPUT"DATE FILES";A$
220 IF LEFT$(A$,1)="Y" OR LEFT$(
A$,1)="y" GOSUB 250
230 NEW
240 PRINT"ERROR":GOTO90
250 ' FILE REDATER
260 ' DATES ANY FILES WITH ZERO
OR 255
270 ' IN THE DATE FIELD WITH TOD
AYS DATE
280 INPUT"DRIVE NO";DR

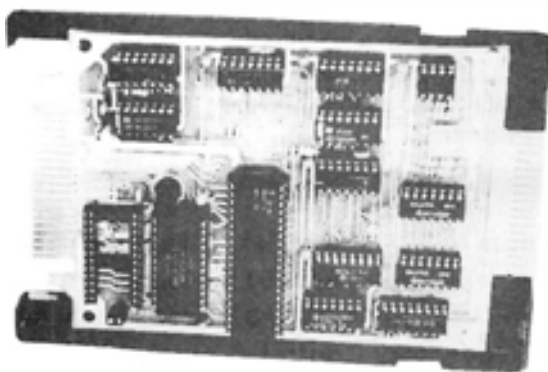
```

```

290 PRINT"THESE FILES REDATED WI
TH ";DATE$
300 IF DR<0 OR DR>1 THEN 280
310 FOR X= 3 TO 11
320 DSKI$ DR,17,X,A$,B$
330 A$=A$+LEFT$(B$,127)
340 FOR N=0 TO 7
350 FILE$=MID$(A$,N*32+1,8)
360 EXT$=MID$(A$,N*32+9,3)
370 IF ASC(FILE$)=0 THEN 450
380 IF FILE$=STRING$(8,255) THEN
FLAG=1:GOTO460
390 MSB=ASC(MID$(A$,N*32+17,1))
400 LSB=ASC(MID$(A$,N*32+18,1))
410 IF MSB=0 AND LSB=0 THEN 430
420 IF MSB<>255 OR LSB<>255 THEN
450
430 MID$(A$,N*32+17,2)=CHR$(PEEK
(&H14E))+CHR$(PEEK(&H14F))
440 PRINTFILE$+"."+EXT$
450 NEXT N
460 B$=RIGHT$(A$,127)
470 A$=LEFT$(A$,128)
480 DSKO$ DR,17,X,A$,B$
490 IF FLAG=1 THEN 510
500 NEXT X
510 RETURN

```

NEW! HDS FLOPPY DRIVE CONTROLLER



FEATURES:

- GOLD PLATED EDGE CARDS
- DUAL SELECTABLE ROM SOCKETS
- NO POTS TO ADJUST
- COMPATIBLE WITH COCO I & II
- 120 DAY WARRANTY
- DOUBLE AND SINGLE DENSITY
- FULLY SOCKETED BOARD

REDUCE YOUR I/O ERRORS WITH THE NEW HARD DRIVE SPECIALIST FLOPPY DRIVE CONTROLLER FOR THE COLOR COMPUTER. GOLD EDGE CARO CONNECTORS AND THE ABSENCE OF POTENTIOMETERS MAKE THIS THE BEST BOARD AVAILABLE TO DATE. SOLO WITH AND WITHOUT ROM (Read Only Memory)

COMPLETED & TESTED BOARD WITH ROM \$139.00
 (INCLUDES CASE, AND DOS INSTRUCTIONS)
 COMPLETED & TESTED BOARD WITHOUT ROM \$119.00
 (INCLUDES CASE)
 BARE BOARD WITH INSTRUCTION MANUAL \$39.95
 (ADD \$40. FOR COMPLETE PARTS KIT, ADD \$20. FOR ROM)

HARD DRIVE SPECIALIST

Ordering Information
 We accept Visa, Mastercard, Wire Transfers, and Certified checks for
 quickest shipping. Orders received on personal checks are held

Dealer inquiries invited
 16206D Hickory Knoll, Houston, Texas 77059

Order Line
 1-800-231-6671
 Local Sales and Service Line
 1-713-480-6000

cooking
with
CoCo



Part VI

By Colin J. Stearman

If you think CoCo is without parallel, this month we cook up something to prove you right and wrong!

I love my printer. It prints quickly, it prints letter quality, it draws pictures, I can send it my own character fonts . . . but the darn thing has a parallel port and CoCo has a serial printer output. Sure I can buy a serial interface for it but it's over 25 percent of the cost of the printer alone, and I hate to waste money. The only solution is to design a parallel port for CoCo.

The actual design is easy, but I wanted the software to fully integrate the port into BASIC, allowing me to direct printer output to either the parallel port or the existing serial port; and for good measure I wanted the BASIC to allow easy adjustment of the Baud rate on the serial port.

To achieve all this meant adding initialization code for the parallel port hardware, trapping output destined to go to the serial port and redirecting it to the desired printer port. This month's assembly language listing does all that as well as adding three new BASIC commands. If you do not need this parallel port and are thinking of turning to the next article, two of the new commands apply to the existing serial port also, so maybe you might want to stick around.

But before we get to the software, let's get the hardware built. If you didn't have trouble with the EPROM programmer, this project will be a snap.

Adding The Parallel Port

The object of the construction is to mount a new 6821 PIA (peripheral interface adapter) inside the computer, without making irreversible modifications to the circuit board. I did this by "piggybacking" the new PIA onto U4. The photos of my unit should give you an overall idea of the look of the finished unit.

U4 is an existing PIA used to drive the D/A converter and control the VDG chip. Please note that these modifications refer to the REVE-style motherboard. If you have a later model, your PIA may not be labeled U4 and will have to be identified by the function it performs.

To construct the unit, first gather the following components together:

- 1) 6821 PIA Peripheral Interface Adapter
- 2) Breadboard PCB Radio Shack #276-158

- 3) SN7404 Hex Inverter Radio Shack #276-1802
- 4) 40 Pin IC socket, wire-wrap type
- 5) Thin hook up wire
- 6) Flat ribbon cable, 36 conductor wide
- 7) Centronics-type female plug, ribbon mounting

Items 1, 4, 6 and 7 are not carried by Radio Shack but are available via mail order from Active Electronics, Westboro, Mass. and other sources. The IC socket must be the wire-wrap type.

To assemble the parts, first remove the cover from CoCo and also the RF shield lid inside. Locate U4 (REV E board #), the 6821 on the right as you face CoCo. Gently pry the IC out of its socket, using a small screwdriver or IC puller. Be careful not to damage the pins. Put CoCo to one side as we will now construct the "piggyback" board assembly.

Mount the 40-pin socket to the PCB (printed circuit board, item 3) anywhere convenient, but leave room for the SN7404 near pin 24. Solder all pins on the socket to the PCB, but *do not* cut off the excess.

Take the new 6821 and gently bend pin 24 outward a little so that when the IC is put into the socket, this pin will not enter it. Put the IC in the socket and press it home.

Mount the SN7404 alongside the 6821 near pin 24. Solder all pins to the PCB. Using the hookup wire, connect pins 1, 3, 5, 7, 9 and 11 together and also to pin 20 of the 40-pin socket. Connect pin 14 to pin 1 on the 40-pin socket. Connect pin 13 to pin 24 of the 6821. This is the bent pin not inserted into the socket. Also connect this pin to a length of wire about nine inches long. The other end will be connected later. Connect pin 12 to the 40-pin socket pin 24.

Turn the PCB upside down and cut off the wire-wrap pins from pins 2 through 19 only. Cut them as close as possible to the PCB. The next task is to mount the assembly on top of the 6821 removed from U4.

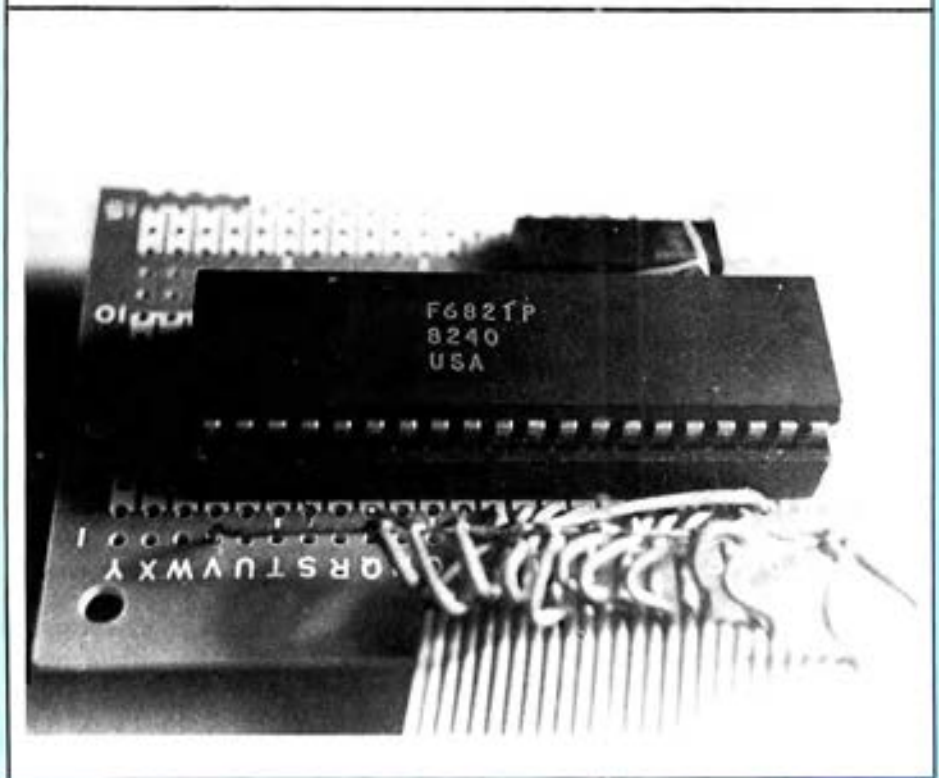
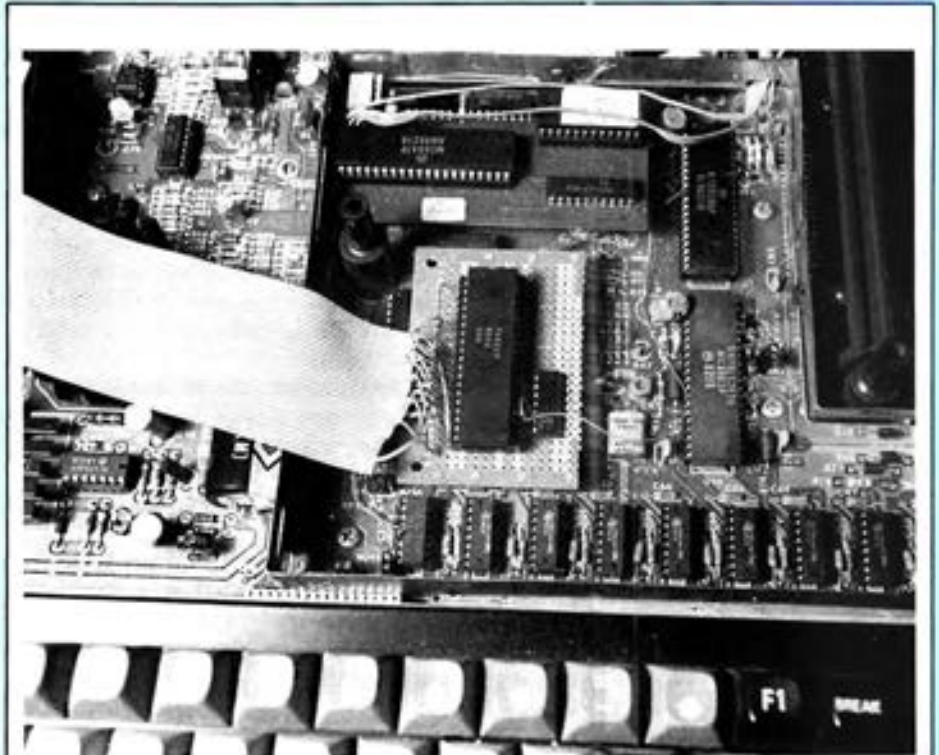
Locate the 6821 removed earlier from CoCo and carefully bend pin 24 so it points vertically upward. Position the assembly on top of this IC to test for fit. It may be necessary to splay the wire-wrap pins out a little. In order for the finished assembly to fit under the RF shield lid, the remaining wire-wrap pins must be trimmed as short as possible. Gauge how much you can cut from each pin and then trim all to this height.

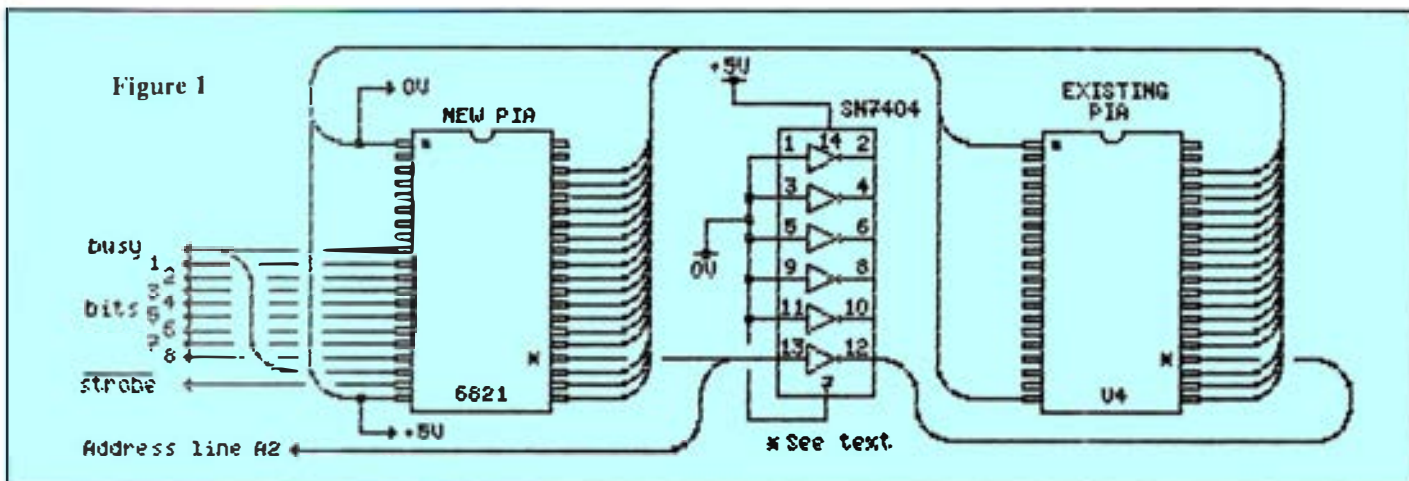
Now solder the assembly to the 6821,

soldering each wire-wrap pin to its respective pin on U4. You should be connecting to U4 pins 1, 20, and 21 through 40. The wire-wrap pin 24 will connect to the upturned pin 24 on U4. This pin will not connect to the socket when U4 is returned to the CoCo motherboard. When soldering the assembly to the back of U4, minimize the amount of solder used so that the IC will still fit

into its socket. Also position the solder joint high on the pins so that the lower part will still fit the socket.

Finally, the ribbon cable must be attached to the assembly. Consult your printer manual and Figure 1. The best approach is to fit the Centronics socket to the ribbon cable to aid in identifying the wire numbers. Most sockets have the numbers molded into them. On the





standard parallel interface the wires and their functions are:

Wire	Function
1	Data Strobe (bar)
2	Data Bit 1
3	Data Bit 2
4	Data Bit 3
5	Data Bit 4
6	Data Bit 5
7	Data Bit 6
8	Data Bit 7
9	Data Bit 8
11	Busy
14	Signal Ground

Connect the wires to the pins as indicated in Figure 1. Wire 14 should be connected to any 0V point on the assembly. Finally trim all the excess PCB from the assembly to minimize its size.

Now mount the finished assembly into CoCo. Press the lower IC gently but firmly into the U4 socket. All pins of the lower 6821 must enter the socket, except for pin 24 which was bent upwards. It's not easy to see that this happens, so inspect the results carefully. The assembly should be firm and quite rigid when installed.

The wire still left unattached must be soldered to the main computer board near the 6809. Cut this wire to a suitable length and attach to the solder point, as indicated in Figure 2. Use a light solder tack to minimize the possibility of damage to the board. This wire picks up address line 2 to allow the software to distinguish between the two PIAs.

The ribbon cable will head toward the left as you face CoCo. Take the RF shield lid and bend the fingers where the cable is, so the lid can be replaced without pinching the wire. If the assembly is too high to allow the lid to be replaced, either leave it off entirely, or extend the height of the RF shield using some shielding metal from an old TV.

The ribbon cable can be routed out of the computer by doubling it back on itself and running it under the main circuit board. A notch cut in the lower plastic shell underneath the serial and cassette ports will allow the cable to leave the case.

This completes the hardware construction. We now move on to this month's software additions to the Disk BASIC patch.

The New BASIC Commands

This month we add three new commands, all associated with the printer port. Two apply even if you do not intend building the parallel port, so stick with us.

PARALLEL

Issuing the BASIC command PARALLEL, either directly from the keyboard, or within a program will result in all data destined for the printer being routed out of the new parallel port. In other words, all *PRINT#-2* statements will output through the parallel port.

The code to drive the parallel port is conditional assembled based upon whether a variable called *PARPNT* is defined or not. Review the paragraph in September's issue for more details on how to include or exclude the code for the parallel port, as desired.

BAUD

This command applies whether or not you have the parallel port. Either way, it establishes the Baud rate of the serial port. If you have the parallel port, it also activates the serial port so that all *PRINT#-2* commands direct output through the standard serial port. The original serial driver code in the Color BASIC ROM is still used for the serial port.

The syntax for this command is:

BAUD(n)

where n = 300, 600, 1200, 2400, 4800 or 9600.

If you have the parallel port, then CoCo starts up with this activated. If you do not, then the serial port is activated and set at 600 Baud.

LDIR

A simple but useful command which does a normal directory but directs it to the currently active printer port. The directory contains the creation date enhancement, but, of course, does not pause after each 16 lines, as when directed to the screen.

Adding This Month's Code

As last month, use your editor to pull in the source code built up so far. Delete the lines identified with reference numbers 20, 21, 22, and 29. Read and follow the notes at reference Lines 6, 7, 8, 10 and 11 regarding including or excluding the parallel port code.

Go to the end of the listing and delete all the remaining lines from and including *ZZLAST EQU *-1*. Then add the assembly text in Listing 1. When all is set, re-assemble the resulting file and test as you have in previous months.

To test the parallel port, connect it to a printer and try *LLISTing* a BASIC program or run some other program which has printer output. If it does not work, but the computer works otherwise, double check your wiring on the new PIA, especially around the ribbon cable connection point. It's very easy to misconnect the wires.

A Final Point

All BASIC programs will have no trouble sending output to the parallel

port. However, you may have trouble with some machine language programs. If they use the serial port in the Color BASIC ROM and do not "mess" with the hooks in RAM, the port should work alright. If the program has Baud rate control, set it to 110 or 120 and this will activate the parallel port; 300 or higher will activate the serial port.

If you have FHL FLEX then you can use the parallel port driver routine described in the FLEX manual. The reason

that the BUSY line goes to both pins 9 and 19 on the new PIA is specifically to accommodate the approach these routines use to detect the printer busy condition. From a programmer's point of view, the PIA is addressed as follows:

FF24 Bit 0 - 6 unused
Bit 7 printer busy line

FF25 Control port for above (set to \$4)

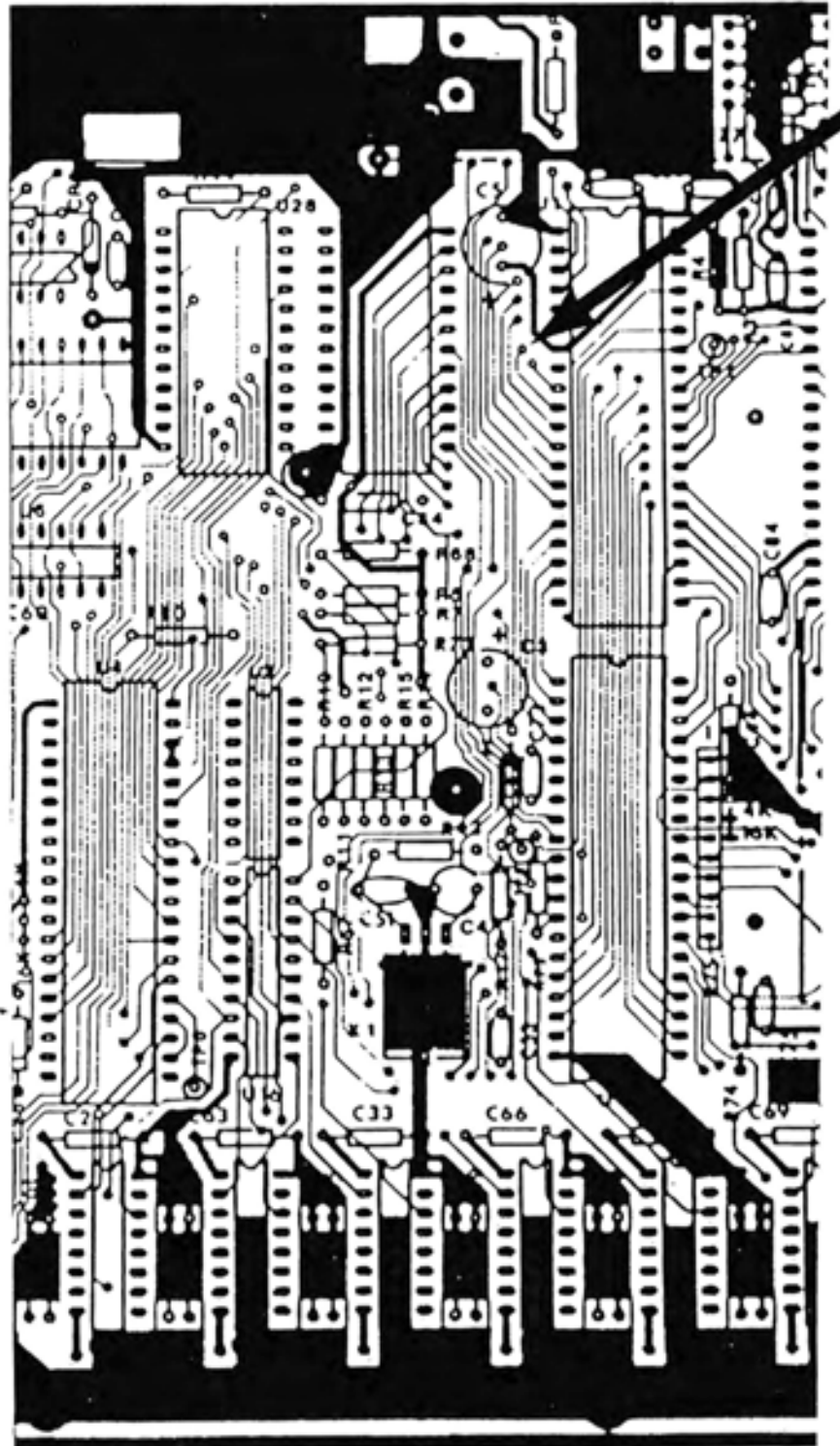
FF26 Bit 0 - Parallel port bit 1
Bit 1 - Parallel port bit 2
Bit 2 - Parallel port bit 3
Bit 3 - Parallel port bit 4
Bit 4 - Parallel port bit 5
Bit 5 - Parallel port bit 6
Bit 6 - Parallel port bit 7
Bit 7 - Parallel port bit 8

FF27 Bit 0 - 0
Bit 1 - 0
Bit 2 - 1

Figure 2

Attach address line 2 wire from parallel port to this point on Rev 'E' boards.

On other revision boards, locate the trace from Pin 10 on the 6809 micro-processor.



Bit 3 - STROBE (BAR)
 Bit 4 - 1
 Bit 5 - 1
 Bit 6 - not used
 Bit 7 - BUSY FLAG (1 when not busy)

This should provide the information you need to incorporate the parallel port into FLEX. Drop me a line if you have trouble.

Coming Attractions

One of the glaring omissions from BASIC is its ability to trap and deal with system errors in a graceful way. We will add this trapping, along with fully spelled out error messages, both on the screen as well as available in a string variable; plus variables identifying the type of error and the line number it occurred.

If you would like the entire DOS-

PATCH program source, along with binary files with and without the parallel port driver for DECBI.0 and DECBI.1, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly. Address this request or any questions to Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

The listing:

```

1087 OPT LIS
1088 *****
1089 * PATCH 04 to RSD09 (C) 1984 Colin Stearman *
1090 *****
1091 *
1092 * *BAUD* COMMAND CODE
1093 * SYNTAX IS BAUD(N) WHERE N =
1094 * 300,600,1200,2400,4800,9600
1095 *
DC9E 0E 1096 BDCMST FCB 0BE,057,029,012,06,01 300,600,1200,2400
1097 * 4800,9600 BAUD CONSTANTS
1098 *
1099 *
DC64 BDB262 1100 BAUD JSR 0B262 EVAL BRKT ARGUMENT
DC67 BDB740 1101 JSR 0B740 GET INTEGER IN X
DC6A 6FE2 1102 CLR ,S FOR COUNTER
DC6C IF10 1103 TFR X,D GET BAUD VALUE
DC6E 10032580 1104 CMPD 09600 HIGHEST LEGAL VALUE
DC72 1022FBC3 1105 LBHI FCERR ERROR IF HIGHER
DC76 6CE4 1106 CMTBD INC ,S COUNT SUBTRACTION
DC79 83012C 1107 SUBD 0300 DIVIDE BAUD BY 300
DC7D 1028F8BA 1108 LBHI FCERR NOT A VALID VALUE
DC7F 26F5 1109 BNE CNTBD CONTINUE SUBTRACTION
1110 * GOT A VALID MULTIPLE OF 300
DC81 3502 1111 PULS A GET RESULT
DC83 5F 1112 CLR B POWER COUNTER
DC84 BDC05E 1113 LDX 0BDCMST POINT X TO BAUD CONSTANTS
DC87 44 1114 SFTAGM LSRA BIT INTO CARRY
DC88 2503 1115 BCS GETCOM BOT BIT GET CONSTANT
DC8A 5C 1116 INCB COUNT SHIFT
DC8B 20FA 1117 BRA SFTAGM 80 SHIFT AGAIN
DC8D A685 1118 GETCOM LOA B,X GET BAUD RATE
DC8F 9796 1119 STA BAUDRT GET BAUD RATE
DC91 0F95 1120 CLR BDFLAG CLEAR TO ENABLE SERIAL PORT
1121 * AND SET LSB OF BAUD RATE
1122 *
DC93 39 1123 RTS ALL DONE
1124 *****
1125 * *LDIR* COMMAND, PRINT DIRECTORY
1126 *
DC94 C6FE 1127 LDIR LDB 0-2 POINT DEVNUM TO PRINTER
DC96 D76F 1128 STB DEVNUM
DC98 7ECBCF 1129 JMP A0016 DO DIR COMMAND
1130 *****
1131 IFDF PARPRT ASSEMBLE FOR PARALLEL PORT
1132 *
1133 * *PARALLEL* COMMAND CODE AND OUTPUT ROUTINE
DC9B CC01CA 1134 PARA LDD 001CA 120 BAUD DELAY
1135 * SET MBB TO 1 FOR PARALLEL PORT
DC9E DD95 1136 STB BDFLAG TO MAKE PARALLEL ACTIVE
DCA0 39 1137 RTS
1138 *****
1139 * Parallel port output routine
1140 * This is called by the modified jump at 1168
DCA1 0D95 1141 PAROUT TST BDFLAG IF NOT ZERO THEN PARALLEL
DCA3 1027EEA3 1142 LBDR A0015 DO SERIAL OUTPUT
DCA7 3402 1143 PSHS A SAVE VALUE
DCA9 966F 1144 LOA DEVNUM GOING TO DEVICE -2?
DCAB 01FE 1145 CMPA 0-2
DCAD 3502 1146 PULS A RECOVER CHAR, FLABS DONT CHANGE

```

```

DCAF 1026EE97 1147 LBNE A0015 NOT DOING DEVICE 0-2
1148 *
1149 * PARALLEL OUTPUT WANTED
DCB3 810D 1150 CMPA 000D WAS IT A CR?
DCB5 2703 1151 BEQ WASCRC
DCB7 0C9C 1152 INC <09C INCREMENT LINE PRINT POSITION
DCB9 8C 1153 FCB 08C SKIP NEXT 2 BYTES
DCBA 0F9C 1154 WASCRC CLR <09C LINE COUNTER
DCBC 3411 1155 PSHS CC,X PRESERVE BASIC VALUES
DCBE 8EFF26 1156 LDX 0DATA POINT X TO P1A
DCC1 8D1E 1157 CHKRDY TST -2,X BUSY IF LINE 7 HI
DCC3 2BFC 1158 BMT CHKRDY WAIT UNTIL LOW
DCC5 A784 1159 STA ,X DATA REGISTER
DCC7 3511 1160 PULS CC,X RECOVER VALUES
DCC9 3262 1161 LEAS 2,S OLD RETURN OFF STACK
DCCB 39 1162 RTS TO ORIGINAL CALLER
1163 *****
1164 ENDC
1165 OPT LIS
1166
DCCB 1168 ZLAST EQU *-1 last used address value
1169 *
1170 * ZLAST must not be greater than $DFFF for
1171 * DOS 1.0 and 0E0F for DOS 1.1. The latter
1172 * has the 08-9 boot program and 8MI set routines
1173 * from 0DF00 to 0DF4C
1174 *
1175 *
1184 OPT LIS
D994 1185 END ADDCOM
NO ERROR(S) DETECTED

```

Submitting Material To the Rainbow

Contributions to THE RAINBOW are welcome from everyone. We like to run a variety of programs which will be useful/helpful/fun for other CoCo owners.

Program submissions must be on tape or disk and it is best to make several saves, at least one of them in ASCII format. We're sorry, but we do not have time to key in programs. All programs should be supported by some editorial commentary, explaining how the program works. We're much more interested in how your submission works and runs than how you developed it. Programs should be learning experiences.

We do pay for submissions, based on a number of criteria. Those wishing remuneration should *so state* when making submissions.

For the benefit of those who wish more detailed information on making submissions, please send a SASE to: Submissions Editor, THE RAINBOW, P.O. Box 385, Prospect, KY 40059. We will send you some more comprehensive guidelines.

Please do not submit programs or articles currently submitted to another publication.

COOKING
WITH
CoCo



Part VII

By Colin J. Stearman

Teaching CoCo how to clean up after its errors and own up to mistakes

Probably the most frustrating limitation of the Microsoft BASIC in CoCo is its lack of ability to trap errors. Even the best written programs generate errors and when they do, it's infuriating to have CoCo tell you how you messed up and then tell you with a condescending smirk that it's OK! It isn't OK, so we must do something about it.

Error Trapping

Most flavors of BASIC have a statement similar to *ON ERROR GOTO nnn* which tells the interpreter that if an error occurs jump to line 'nnn' and continue running. Then at line 'nnn' we can write some lines which handle the error and continue the running of the program.

Because *ON* is already a BASIC keyword I decided to simplify the syntax. So here is a description of the error trapping command and some associated variables.

ERRORS

The syntax for the error directing line is *ERRORS GOTO nnn*, where 'nnn' is an existing line number or zero. When such a line is encountered in your program it simply tells the interpreter that, should an error occur, go to line 'nnn'. This command will stay in effect until another such line is encountered saying go to a different line on an error. Except if 'nnn' is a zero, error trapping is canceled and errors cause BASIC to stop the program and report just as before (or nearly as before, as you will see).

If line 'nnn' does not exist, then a 'No such line number' error will occur if the statement is entered in the direct mode. However, if it is in a program, it will create an error itself, but the error will have nowhere to go, and the program will lock up. Pressing Reset is the only option left.

Because the line number follows a normal *GOTO* statement, the *RENUM*

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

command will handle it correctly.

When any error occurs all *FOR . . . NEXT* loops and subroutine return addresses are canceled, allowing the error handling routine to jump to anywhere in the program without a problem.

ECODE

This numeric variable returns the current error code number. If no error has yet been encountered, it will have the value -1, so if a NO SUCH FILE error was the most recent error, then doing a PRINT ECODE would print 26, the code number for that error. ECODE may be used just as any other numeric variable, but it may not be assigned a value by putting it on the left of an equal sign.

ELINE

This is also a numeric variable and all comments about ECODE apply equally to it. This returns the BASIC line number on which the most recent error occurred. If no error has yet occurred this variable will have the value of -1.

ENAMES

This is a string variable which contains the name of the most recent error. If no errors have yet occurred, ENAMES\$ is a zero length string. All normal string manipulation functions may use it, but it too must not appear on the left of an equal sign.

The error code numbers returned by ECODE and the associated error strings are:

ECODE	ENAMES
0	NEXT without FOR
1	Syntax
2	Return without GOSUB
3	Out of Data
4	Function Call
5	Overflow
6	Out of Memory
7	No such line #
8	Subscript
9	Redimensioned Array
10	Divide by 0
11	Illegal Direct Command
12	Type Mismatch
13	Out of String Space
14	String too long
15	String too complex
16	Can't Continue
17	File Data
18	Already Open
19	Device Number
20	Read/Write
21	File Mode
22	File Not Open
23	Read past End of File

24	Direct Command in File
25	Undefined Function
26	No such File
27	Record #
28	Disk Full
29	Out of Buffer Space
30	Write Protect
31	File Name
32	Directory
33	File Exists
34	Field Overflow
35	Set to Non-Fielded String
36	Verify
37	Access past End of File

If no error trapping is set, BASIC will return these fully spelled out error messages followed by the word ERROR, instead of the cryptic question mark and two letter code.

Due to memory space limitations, ENAMES\$ and fully spelled out error messages are not included in the patch to *DECB 1.1*.

SWAP

The final BASIC command to be added is SWAP. This has no connection with error trapping but is useful to have around. The syntax is: *SWAP var1,var2*

“When any error occurs all FOR . . . NEXT loops and subroutine return addresses are canceled, allowing the error handling routine to jump to anywhere in the program without a problem . . . If no error trapping is set, BASIC will return these fully spelled out error messages followed by the word ERROR, instead of the cryptic question mark and two letter code.”

where 'var1' and 'var2' are like variables. This means that SWAP A\$,B\$ will cause the string associated with A\$ to be assigned to B\$ and vice versa. Similarly, SWAP DL,WP will cause the value assigned to DL to be assigned to WP and that of WP to be assigned to DL. If the two variables are not of the same type, (string or numeric) then a 'Type Mismatch' will occur.

The SWAP command saves the need for an intermediate holding variable when exchanging variable values and is considerably faster than this approach. The obvious application is in 'bubble

sorts' where elements must be swapped.

A Final Flourish

If you look at Listing I around the label RESET you will notice some additional start-up codes. This executes when CoCo does a cold start. The first thing this code does is restore all the drives to track 0. This eliminates that annoying search up and down the disk during the first disk access. The slight increase in start-up time is worth the subsequent savings in access time and reduction in wear and tear on the drive itself, not to mention your nerves!

This code restores all possible drives to track 0. If you do not have four drives you can improve the start-up time a little by only restoring the drives you do have. This is done by changing the '3' in the line immediately after the line defining RESET (which reads 'LDB #3 NUMBER OF DRIVES') to one less than the number of drives you do have.

Adding This Month's Code

Just as in previous months, pull the assembly file built up so far into your editor, then remove the commenting asterisks from the start of line with [REF #] of 2, 9-1, 9-2 and 9-3. Completely delete reference lines 18, 19, 25, 26 and 27. Also delete all lines at the end starting with 'ZZLAST EQU *-1'.

Now type in the new code found in Listing I and reassemble the result. As this month's addition is the last, rename the composite assembly language source as *DISKPTCH.ASM* and the binary file as *DISKPTCH.BIN*. Test the binary patch file just as you have for the past few months.

Wrapping It Up Next Month

The next issue of THE RAINBOW will see the last installment of this series. In it we will tie up a few loose ends; put the entire revised version of Disk BASIC in an EPROM and mount it in the disk controller, and make some suggestions for commands you could add yourself. I hope you'll plan on joining me then.

If you would like the entire *DOS-PATCH* program source, along with binary files with and without the parallel port driver for *DECB 1.0* and *DECB 1.1*, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

Address this request or any questions to: Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

The listing:

```

1087 OPT LIS
1088 *****
1089 * PATCH #4 to RSDOS (C)1984 Colin Stearnan *
1090 *****
1091 *
1092 * "BAUD" COMMAND CODE
1093 * SYNTAX IS BAUD(N) WHERE N =
1094 * 300,600,1200,2400,4800,9600
1095 *
DC62 BE 1096 BDCNST FCB $BE,$57,$29,$12,$6,$1 300,600,1200,2400
1097 * 4800,9600 BAUD CONSTANTS
1098 *
1099 *
DC60 BDB262 1100 BAUD JSR $B262 EVAL BRKT ARGUMENT
DC68 BDB740 1101 JSR $B740 GET INTEGER IN X
DC6E 6FE2 1102 CLR ,-S FOR COUNTER
DC70 1F10 1103 TFR X,D GET BAUD VALUE
DC72 108325B0 1104 CMPD $9600 HIGHEST LEGAL VALUE
DC76 1022F8BF 1105 LBHI FCERR ERROR IF HIGHER
DC7A 6CE4 1106 CNTBD INC ,S COUNT SUBTRACTION
DC7C 83012C 1107 SUBD $300 DIVIDE BAUD BY 300
DC7F 102BF8B6 1108 LBMI FCERR NOT A VALID VALUE
DC83 26F5 1109 BNE CNTBD CONTINUE SUBTRACTION
1110 * GOT A VALID MULTIPLE OF 300
DC85 3502 1111 PULS A GET RESULT
DC87 5F 1112 CLRB CLR# POWER COUNTER
DC88 BEDC62 1113 LDX #BDCNST POINT X TO BAUD CONSTANTS
DC8B 44 1114 SFTAGN LSRA BIT INTO CARRY
DC8C 2503 1115 BCS GETCON GOT BIT GET CONSTANT
DC8E 5C 1116 INCB COUNT SHIFT
DC8F 20FA 1117 BRA SFTAGN GO SHIFT AGAIN
DC91 A685 1118 GETCON LDA B,X GET BAUD RATE
DC93 9796 1119 STA BAUDRT SET BAUD RATE
DC95 0F95 1120 CLR BDFLAG CLEAR TO ENABLE SERIAL PORT
1121 * AND SET LSB OF BAUD RATE
1122 *
DC97 39 1123 RTS ALL DONE
1124 *****

```

```

1125 * "LDIR" COMMAND, PRINT DIRECTORY
1126 *
DC98 C6FE 1127 LDIR LDB #2 POINT DEVNUM TO PRINTER
DC9A D76F 1128 STB DEVNUM
DC9C 7ECBCF 1129 JMP A0016 DO DIR COMMAND
1130 *****
1131 IFDF PARPRT ASSEMBLE FOR PARALLEL PORT
1132 *
1133 * "PARALLEL" COMMAND CODE AND OUTPUT ROUTINE
DC9F CC01CA 1134 PARA LDD #1CA 120 BAUD DELAY
1135 * SET MSB TO 1 FOR PARALLEL PORT
DCA2 DD95 1136 STD BDFLAG TO MAKE PARALLEL ACTIVE
DCA4 39 1137 RTS
1138 *****
1139 * Parallel port output routine
1140 * This is called by the modified jump at $168
DCA5 0D95 1141 PAROUT TST BDFLAG IF NOT ZERO THEN PARALLEL
DCA7 102EE9F 1142 LBEQ A0015 DO SERIAL OUTPUT
DCAB 3402 1143 PSHS A SAVE VALUE
DCAD 966F 1144 LDA DEVNUM GOING TO DEVICE -2?
DCAF 81FE 1145 CMPA #-2
DCB1 3502 1146 PULS # RECOVER CHAR, FLAGS DONT CHANGE
DCB3 1026EE93 1147 LBNE A0015 NOT DOING DEVICE #-2
1148 *
1149 * PARALLEL OUTPUT WANTED
DCB7 810D 1150 CMPA #00 WAS IT A CR?
DCB9 2703 1151 BEQ WASCRL
DCBB 0C9C 1152 INC <#9C INCREMENT LINE PRINT POSITION
DCBD BC 1153 FCB #0C SKIP NEXT 2 BYTES
DCBE 0F9C 1154 WASCRL CLR <#9C LINE COUNTER
DCC0 3411 1155 PSHS CC,X PRESERVE BASIC VALUES
DCC2 8EFF26 1156 LDX #DATA POINT X TO PIA
DCC5 6D1E 1157 CHKRDY TST -2,X BUSY IF LINE 7 HI
DCC7 2BFC 1158 BMI CHKRDY WAIT UNTIL LOW
DCC9 A78A 1159 STA ,X DATA REGISTER
DCCB 3511 1160 PULS CC,X RECOVER VALUES
DCCD 3262 1161 LEAS 2,S OLD RETURN OFF STACK
DCCF 39 1162 RTS TO ORIGINAL CALLER
1163 *****
1164 ENDC
1165 OPT LIS
1166 *****
1167 * PATCH #5 to RSDOS (C)1984 Colin Stearnan *
1168 *****
1169 *
1170 *****
1171 * "SWAP"
1172 *
1173 * CODE FOR SWAP COMMAND SYNTAX IS SWAP V1,V2
1174 * WHERE V1 AND V2 ARE LIKE VARIABLE TYPES
1175 *
DCD0 BDB357 1176 SWAP JSR $B357 GET FIRST STRING POINTER
DCD3 9606 1177 LDA <6 TYPE #NUMBER -1=STRING
DCD5 3412 1178 PSHS X,A SAVE ON STACK
DCD7 BDB26D 1179 JSR $B26D PARSE REQUIRED COMMA
DCDA BDB357 1180 JSR $B357 GET 2ND STRING POINTER IN X
1181 * NOW TEST THAT BOTH VARIABLES ARE SAME TYPE
DCDD 3502 1182 PULS A RECOVER FIRST TYPE
DCDF 9106 1183 CMPA <6 CHECK FOR SAME AS SECOND
1184 * NOT SAME TYPE SO ISSUE ?TM ERROR
DCE1 1026D46C 1185 LBNE $B151 TYPE MISMATCH
1186 * SAME TYPE SO SWAP POINTER INFO
DCE5 3540 1187 PULS U ONE IN X, OTHER IN U
DCE7 C605 1188 LDB #5 COUNTER
DCE9 A684 1189 SWAP5 LDA ,X GET VALUE AT X
DCEB 3402 1190 PSHS A PRESERVE IT
DCED A6C4 1191 LDA ,U GET VALUE AT U
DCEF A780 1192 STA ,X+ PUT AT X
DCF1 3502 1193 PULS A GET ORIGINAL AT X
DCF3 A7C0 1194 STA ,U+ PUT AT U
DCF5 5A 1195 DECB REDUCE COUNTER
DCF6 26F1 1196 BNE SWAP5 CONTINUE SWAPPING
DCFB 39 1197 RTS
1198 *****
1199 * "ERRORS" Command
1200 * Executed when the ERRORS command is encountered
1201 *
DCF9 C6B1 1202 ERRCMD LDB #01 CHECK "G0"
DCFB BDB26F 1203 JSR $B26F NOT THEN SYNTAX ERROR
DCFE C6A5 1204 LDB #A5 CHECK "10"
DD00 BDB26F 1205 JSR $B26F NOT THEN SYNTAX ERROR
DD03 BDAF67 1206 JSR $AF67 PROCESS LINE # INTO #28
DD06 DC2B 1207 LDD <#28 GET THE LINE #

```

DATE BOOK & CALENDAR

- Tape or Disk files
- Index records by date, month, year, or day
- Prints date-to-date
- Encryption by password, Password is not stored

32k ECB \$25.95

RELATIONAL DATABASE

- Blistering fast sort, 1000 records in 10 sec OR LESS!
- Multikey sort
- Tape or Disk files
- Math ability

SASE for more info

32k ECB \$39.95

Butterfly Software

Rt 7 Box 565-A (806)
Lubbock, Tx 79401 762-1941

```

DD0B DDDC      120B   STD   JLINE   SAVE IT
1209 ** IF ZERO THEN CLEAR TRAPPING
DD0A 2773      1210   BEQ   ERRSET
1211 ** CHECK FOR VALID LINE NUMBER
DD0C DCA6      1212   LDD   #A6     GET PARSER POINTER
DD0E 3406      1213   PSHS  D       SAVE ON STACK
DD10 BDAEA9    1214   JSR   #AEA9   CHECK VALID NUMBER
1215 *IF WE GOT BACK HERE IT'S OK
DD13 3506      1216   PULS  D       RESET PARSER POINTER
DD15 DDA6      1217   STD   #A6
DD17 39        1218   RTS
1219 *****
1220 * ERROR TRAPPING AND HANDLING ROUTINE
1221 *
1222 * this code is executed when an error is
1223 * encountered by BASIC from jump at #18F
1224 *
DD1B BDD81B    1225  ERRTRP JSR   DIRECT   CURRENT LINE
DD1B 2724      1226   BEQ   NOTRAP   SO DONT TRAP IT
DD1D 9EDC      1227   LDX   JLINE   GET ERRLINE JUMP
DD1F 2720      1228   BEQ   NOTRAP   SO DONT TRAP IT
1229 *****
1230 * WE WANT TO TRAP ERROR NOW B HAS ERROR CODE #2
1231 * IF AN OD ERROR THEN THEN ADDRESS AT #2B NEEDS
1232 * PUTTING AT #A6 BECAUSE READ MOVED IT TO SCAN
1233 * THE DATA STATEMENTS
1234 *
DD21 C106      1235   CMPB  #B6     OD ERROR NUMBER
DD23 2604      1236   BNE  NOREAD  NOT A OD ERROR
DD25 9E2B      1237   LDX  #2B     GET POINTER
DD27 9FA6      1238   STX  #A6     PUT IT IN PARSER
DD29 54        1239  NOREAD LSRB   DIVIDE BY 2
DD2A D75A      1240   STB  ECODE   CODE ADDRESS
DD2C 9E6B      1241   LDX  <#6B   CURRENT LINE
DD2E 9F76      1242   STX  ELINE   ERRLINE ADDRESS
DD30 9EDC      1243   LDX  JLINE   GET ERROR GOTO LINE #
DD32 9F2B      1244   STX  <#2B   PREPARE TO GO TO IT
DD34 18DE21    1245   LDS  <#21   CLEAN STACK
DD37 CCADC4    1246   LDD  #ADC4   RETURN TO INTERPRET LOOP
DD3A 3406      1247   PSHS  D       PUT ONTO STACK
DD3C 0F6F      1248   CLR  DEVNUM  RESET DEVICE CODE
DD3E 7AEA9     1249   JMP  #AEA9   80 TO NEW LINE
1250 ****
1251 *PROCESS NO TRAP
DD41 BD3C      1252  NOTRAP BSR   ERRSET   RESET ERROR CODE
1253 *
1254   IFGT  REV     <---
1255   JMP   #AC49   ; DOS 1.1 onlv
1256   ENDC  <----
1257 *
####          1258   IFEQ  REV     <----
1259 * Process new error display ;
DD43 BDD1E5    1260   JSR  A0026   CLEAR DISK SYSTEM ;
DD46 3404      1261   PSHS  B       PRESERVE ERROR CODE ;
DD4B BDCA3B    1262   JSR  A0014   MORE DISK SHUTDOWN ;
DD4B 3504      1263   PULS  B       8ET ERROR CODE BACK ;
DD4D BDA7E9    1264   JSR  #A7E9   MOTOR OFF ;
DD50 BDAAD33   1265   JSR  #AD33   RESET STACK ETC. ;
DD53 0F6F      1266   CLR  DEVNUM  REST TO SCREEN ;
DD55 BDB95C    1267   JSR  #B95C   OUT RETURN IF NEEDED ;
DD58 54        1268   LSRB         DIVIDE ERROR CODE BY 2 ;
DD59 8D06      1269   BSR  ERFINO  FIND ERROR MESSAGE ;
1270 * OUTPUT NEW ERROR MESSAGE ;
DD5B BDB9A2    1271   JSR  STROUT  OUTPUT IT ;
DD5E 7EAC65    1272   JMP  #AC65   PRINT " ERROR" ETC. ; DOS 1.0 onlv
1273 ***** ;
1274 * error message finder ;
1275 * B has error count/2 coming in ;
1276 * HAS CHARACTER COUNT COMING OUT ;
1277 * X HAS POINTER TO FIRST CHAR ;
1278 ERFINO TFR  B,A   MOVE ERROR CODE TO A ;
1279 LDX #ERR0     POINT X TO MSG #0 ;
1280 CLRB        DONT AFFECT X FIRST TIME ;
1281 KPLOOK ABX      ADD COUNT TO ERROR ADDRESS ;
1282 LDB ,X+       GET CHARS IN MESSAGE ;
1283 DECA        DECREASE ERROR COUNT ;
1284 BPL KPLOOK    KEEP LOOKING ;
1285 RTS
1286 ENDC <----
1287 *****
1288 * CLEAR ERROR TRAPPING ON RUN
1289 ERCNCL BSR   ERRSET
DD6E BDF

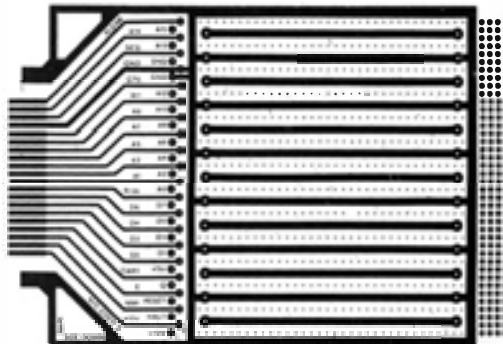
```

```

DD70 7EC990    1290   JMP  A0013
1291 *****
1292 * THIS CODE RUNS ON A COLD START AND RESETS ALL
1293 * DRIVES TO TRACK ZERO AND RESETS ERROR TRAPPING
1294 *
1295 * Reset drive 0-1 to track zero
DD73 0FEA      1296  RESET CLR  #EA   RESTORE OP CODE =0
DD75 C601      1297   LDB  #1     NUMBER OF DRIVES-1
DD77 D7EB      1298   STB  #EB     DRIVE NUMBER
DD79 8D14      1299  NXTDRV BSR   HOME   DO RESTORE TO TRACK 0 WITH 1 RETRY
DD7B 0AEB      1300   DEC  #EB     NEXT DRIVE
DD7D 2AFA      1301   BPL  NXTDRV
1302 *
1303 *
1304 * Clears ERROR trapping
1305 *
DD7F 3416      1306  ERRSET PSHS  D,X   SAVE REGS
DD81 9E8A      1307   LDX  ZERO
DD83 9FDC      1308   STX  JLINE
DD85 CCF5FF    1309   LDD  #FFFF
DD88 DD76      1310   STD  ELINE
DD8A 975A      1311   STA  ECODE
DD8C 3516      1312   PULS  D,X   RECOVER REGS
DD8E 39        1313   RTS
1314 *****
1315 * restore drive head with no retries
DD8F 3476      1316  HOME  PSHS  A,B,X,Y,U
DD91 8601      1317   LDA  #1     RETRY COUNT 1= NO RETRIES
DD93 7ED670    1318   JMP  A0032   RESTORE CODE ENDS WITH AN RTS
1319 *****
1320 *
1321 * "ELINE"
1322 *
DD96 DC76      1323  ERRLIN LDD  ELINE
DD98 10B3FFFF  1324   CMPD  #FFFF   IF #FFFF NOT SET?
DD9C 1026FEBD  1325   LBNE  UNSIGN  YES IT IS
DDA0 7E84F4    1326  SIGNED JMP  #84F4   RETURN AS SIGNED VALUE (-1)
1327 *****
1328 *

```

6809 SYSTEM DEVELOPMENT



EXPANSION HARDWARE FOR THE TRS-80 COLOR COMPUTER

XPNDR1™

SuperGuide™

We've added grounding tabs to the XPNDR1 and, on the out-board end, an array of plated-through solder pads. Shown is the bottom side of the card with the CoCo signals identified and the +5V and ground buses. The edge connector and tabs are gold plated; the 4.3x6.3 inch glass/epoxy card is drilled for standard .3 and .6 inch DIP sockets. Includes 8 page *Application Notes* to help you get started.

Precision molded plastic insert designed specifically to align and support printed circuit cards in the CoCo cartridge slot; an unbreakable removable card guide. Patent Pending.

\$3.95 each

Available now from:

ROBOTIC MICROSYSTEMS

\$19.95 each or 2 for \$36

BOX 30807 SEATTLE, WA 98103

```

1329 *      "ECODE"
1330 *
DDA3 4F      1331 ERRCD CLRA
DDA4 D65A    1332 LDB  ECODE
1333 *IF MINUS THEN IT IS -1 AND THEREFORE UNSET
DDA6 102AFEB3 1334 LBPL UNSIGN OUTPUT UNSIGNED # TO VARIABLE
DDAA 1D      1335 SEX      MAKE D HAVE VALUE IN B
DDAB 20F3    1336 BRA  SIGNED  OUTPUT TO VARIABLE(-1)
1337 *****
0000      1338 IFEQ REV      <----
1339 *      ENAME# ;
DDAD D65A    1340 ERNAME LDB  ECODE      GET ERROR CODE ;
DDAF 2A02    1341 BPL  GETNM      GET ERROR NAME STRING ;
DDBI 5F      1342 CLR#      FOR NULL STRING LENGTH ;
DDB2 A1      1343 FCB  #A1      SKIP NEXT INSTRUCTION ;
1344 * ;
DDB3 0DDD61  1345 GETNM JSR  ERFIND      RETURNS X AT ERROR NAME ;
1346 *      B WITH COUNT ;
DDB6 1F13    1347 TFR  X,U      SAVE ERROR STRING POINTER ;
DDBB DDB50F  1348 JSR  #B50F     CHECK FOR AVAILABLE SPACE ;
1349 *X NOW HAS STRING START ADDRESS ;
DDBB 2705    1350 BEQ  STREX     NULL LENGTH STRING ;
DDBD 1E13    1351 EXG  X,U      SWAP THE POINTERS ;
DDBF BDA59A  1352 JSR  #A59A     MOVE STRING ;
DDC2 7EB69B  1353 STREX JMP  #B69B     RETURN VIA STRING# CODE ;
1354 ***** ;
1355 * ;
1356 * ERROR MESSAGES ;
1357 * ;
1358 * FORMAT IS CHARACTER COUNT/CHARACTERS ;
1359 * ;
DDC5 10      1360 ERR# FCB  ERR1-(#+1) ;
DDC6 4E      1361 FCC  /NEXT WITHOUT FOR/ ;
DDD6 06      1362 ERR1 FCB  ERR2-(#+1) ;
DDD7 53      1363 FCC  /SYNTAX/ ;
DDDD 14      1364 ERR2 FCB  ERR3-(#+1) ;
DDDE 52      1365 FCC  /RETURN WITHOUT GOSUB/ ;
DDF2 0B      1366 ERR3 FCB  ERR4-(#+1) ;
DDF3 4F      1367 FCC  /OUT OF DATA/ ;

```

```

DDFE 0D      1368 ERR4 FCB  ERR5-(#+1) ;
DDFF 46      1369 FCC  /FUNCTION CALL/ ;
DE0C 0B      1370 ERR5 FCB  ERR6-(#+1) ;
DE0D 4F      1371 FCC  /OVERFLOW/ ;
DE15 0D      1372 ERR6 FCB  ERR7-(#+1) ;
DE16 4F      1373 FCC  /OUT OF MEMORY/ ;
DE23 0E      1374 ERR7 FCB  ERR8-(#+1) ;
DE24 4E      1375 FCC  /ND SUCH LINE #/ ;
DE32 09      1376 ERBB FCB  ERR9-(#+1) ;
DE33 53      1377 FCC  /SUBSCRIPT/ ;
DE3C 13      1378 ERR9 FCB  ERR10-(#+1) ;
DE3D 52      1379 FCC  /REDIMENSIONED ARRAY/ ;
DE50 0B      1380 ERR10 FCB  ERR11-(#+1) ;
DE51 44      1381 FCC  /DIVIDE BY 0/ ;
DESC 16      1382 ERR11 FCB  ERR12-(#+1) ;
DE5D 49      1383 FCC  /ILLEGAL DIRECT COMMAND/ ;
DE73 0D      1384 ERR12 FCB  ERR13-(#+1) ;
DE74 54      1385 FCC  /TYPE MISMATCH/ ;
DEB1 13      1386 ERR13 FCB  ERR14-(#+1) ;
DEB2 4F      1387 FCC  /OUT OF STRING SPACE/ ;
DE95 0F      1388 ERR14 FCB  ERR15-(#+1) ;
DE96 53      1389 FCC  /STRING TOO LONG/ ;
DEA5 12      1390 ERR15 FCB  ERR16-(#+1) ;
DEA6 53      1391 FCC  /STRING TOO COMPLEX/ ;
DEBB 0E      1392 ERR16 FCB  ERR17-(#+1) ;
DEB9 43      1393 FCC  /CAN'T CONTINUE/ ;
DEC7 09      1394 ERR17 FCB  ERR18-(#+1) ; DOS 1.0 only
DECB 46      1395 FCC  /FILE DATA/ ;
DED1 0C      1396 ERR18 FCB  ERR19-(#+1) ;
DED2 41      1397 FCC  /ALREADY OPEN/ ;
DEDE 0D      1398 ERR19 FCB  ERR20-(#+1) ;
DEF4 44      1399 FCC  /DEVICE NUMBER/ ;
DEEC 0A      1400 ERR20 FCB  ERR21-(#+1) ;
DEED 52      1401 FCC  /READ/WRITE# ;
DEF7 09      1402 ERR21 FCB  ERR22-(#+1) ;
DEFB 46      1403 FCC  /FILE MODE/ ;
DF01 0D      1404 ERR22 FCB  ERR23-(#+1) ;
DF02 46      1405 FCC  /FILE NOT OPEN/ ;
DF0F 15      1406 ERR23 FCB  ERR24-(#+1) ;
DF10 52      1407 FCC  /READ PAST END OF FILE/ ;
DF25 16      1408 ERR24 FCB  ERR25-(#+1) ;
DF26 44      1409 FCC  /DIRECT COMMAND IN FILE/ ;
DF3C 12      1410 ERR25 FCB  ERR26-(#+1) ;
DF3D 55      1411 FCC  /UNDEFINED FUNCTION/ ;
DF4F 0C      1412 ERR26 FCB  ERR27-(#+1) ;
DF50 4E      1413 FCC  /NO SUCH FILE/ ;
DF5C 08      1414 ERR27 FCB  ERR28-(#+1) ;
DF5D 52      1415 FCC  /RECORD #/ ;
DF65 09      1416 ERR28 FCB  ERR29-(#+1) ;
DF66 44      1417 FCC  /DISK FULL/ ;
DF6F 13      1418 ERR29 FCB  ERR30-(#+1) ;
DF70 4F      1419 FCC  /OUT OF BUFFER SPACE/ ;
DFB3 0D      1420 ERR30 FCB  ERR31-(#+1) ;
DFB4 57      1421 FCC  /WRITE PROTECT/ ;
DF91 09      1422 ERR31 FCB  ERR32-(#+1) ;
DF92 46      1423 FCC  /FILE NAME/ ;
DF9B 09      1424 ERR32 FCB  ERR33-(#+1) ;
DF9C 44      1425 FCC  /DIRECTORY/ ;
DFA5 0B      1426 ERR33 FCB  ERR34-(#+1) ;
DFA6 46      1427 FCC  /FILE EXISTS/ ;
DFB1 0E      1428 ERR34 FCB  ERR35-(#+1) ;
DFB2 46      1429 FCC  /FIELD OVERFLOW/ ;
DFC0 19      1430 ERR35 FCB  ERR36-(#+1) ;
DFC1 53      1431 FCC  /SET TO NON-FIELDED STRING/ ;
DFDA 06      1432 ERR36 FCB  ERR37-(#+1) ;
DFDB 56      1433 FCC  /VERIFY/ ;
DFE1 17      1434 ERR37 FCB  ENDERR-(#+1) ;
DFE2 41      1435 FCC  /ACCESS PAST END OF FILE/ ;
DFF9          1436 ENDERR EQU  *      <----
1437          1438 ENDC          <----
1438
1439
DFFB          1440 ZLAST EQU  *-1      last used address value
1441 *
1442 * ZLAST must not be greater than $DFFF for
1443 * DOS 1.0 and $DEFF for DOS 1.1. The latter
1444 * has the OS-9 Boot program and SWI set routines
1445 * from $DF00 to $DF4C
1446 *
1447 *
1456          OPT LIS
D994          1457          END ADDCOM

```

RAINBOW SCREEN MACHINE

The Rolls Royce of graphics text screen enhance 5-more features than all others combined.
Tape \$29.95; Disk \$32.95

SUPER SCREEN MACHINE

Revolutionary — Heralded as the most useful, powerful and versatile state-of-the-art utility ever developed for the Color Computer.
Tape \$44.95; Disk \$47.95

GRAPHICOM II

Rotate graphic image about on any Z axis • slide position graphic with wrap around • copy/enlarge with user-defined shapes • pan and zoom — "blow-up" or "zoom in" on image • font editor — create font styles or char sets • special effects — tunnel vision, fish eye etc • pixel blaster — widen lines color separation.
Disk \$24.95; Disk only

GRAPHCOM/Video Digitizer only \$199.95

- | | | | |
|------------|---------|-------------------|----------|
| 1. G/L | \$59.95 | 5. Mail Labels | \$ 49.95 |
| 2. A/P | \$59.95 | 6. Invoice Writer | \$ 49.95 |
| 3. A/R | \$59.95 | 7. Budget | \$ 49.95 |
| 4. Payroll | \$79.95 | 8. Master 1-7 | \$299.95 |

We carry DFS forms to run with our software. These forms are compatible with over 385 software companies.

Bluegrass Software
P.O. Box 573
Franklin, KY 42134

Send 3.00 for shipping and handling for free catalog and product information.

Postage paid on all orders. To receive **Free** catalogue & product information send \$3.00 to cover shipping & handling.



COOKING
WITH
CoCo



PART VIII

By Colin J. Stearman

The last of the series where we 'burn' the EPROM and savor the delicacies we have been cooking up.

(Colin J. Stearman is an electronics engineer educated in the U.K. He has worked with all kinds of computers and has been a CoCo enthusiast for over two years.)

This issue sees the closing of the CoCo kitchen. We have added all the commands and features, and turned a good DOS into one which I hope you agree is even better. We have filled all the available space in the Disk BASIC ROM, and the only task left is to permanently place the modified DOS into an EPROM and install it in the controller.

Loading the EPROM

I covered how to transfer the modified DOS into an EPROM in Part 3 of the series in the September 1984 issue. But, here we are into 1985, so maybe we had better recap the procedure.

There are several ways to load the EPROM, so I will describe the one which is applicable to all configurations of CoCo. Before starting, you should assemble the entire patch file to a binary file in disk and call it *DISKPTCH.BIN*. Also, you should have a reliable blank cassette in the recorder.

The first step is to save the original Disk BASIC to a file on the tape. This is done with

```
CSAVEM"DBASIC",&HC000,
&HDFFF,&HA027
```

Now transfer the patch file to cassette. We will relocate the file during this process. Enter the following direct commands.

```
CLEAR 200,&H3FFF
LOADM"DISKPTCH",&H4000-
&HC000+65536
```

```
CSAVEM"DISKPTCH",&H4000,
&H5FFF,&HA027
```

Now disconnect the disk system and plug in the EPROM programmer. Don't forget to connect your 21-volt supply to the programmer. Rewind the tape and enter the following commands.

```
CLEAR 200,&H3FFF
CLOADM"DBASIC",&H4000-
&HC000+65536
```

```
CLOADM"DISKPTCH"
EXEC &HE000
```

The last command will start up the EPROM driver code in the EPROM in the programmer socket. If you haven't put it in an EPROM yet, then load it from tape, but make sure it does not conflict with the revised version of Disk BASIC temporarily resident at \$4000 through \$5FFF.

When the EPROM programmer is started up, load a 2764 EPROM into the ZIF socket and check that it is erased. Then transfer the memory contents from \$4000 through \$5FFF into the EPROM starting at EPROM address 0. This completes the programming. You can check the EPROM by powering down and moving the EPROM to the socket at address space \$C000. When you power up, the revised Disk BASIC should start up, and CoCo will try to run *AUTOEXEC.BAS* from drive 0. As the disk controller is not plugged in, this will fail with a READ/WRITE ERROR. If you get this far the likelihood is that the EPROM is all right.

Loading The EPROM Into The Controller

Unfortunately, the 2764 does not have the same pin assignments as the ROM inside the disk controller. It doesn't even have the same number of pins. The ROM has 24, the EPROM has 28. To overcome this we must construct a conversion interface using a 28-pin IC socket.

The diagrams in Figure 1 show the overall approach. Obtain a good quality 28-pin IC socket, the solder type, not wire-wrap. Get the type with the pins

oriented in the same plane as the IC pins, as shown in the figure. These pins have to enter to original ROM socket so they need to be this way. Some brands of socket have the pins at 90 degrees to the normal plane.

Take the EPROM and gently bend out pins 20 and 23 so they will not enter the socket, then press the EPROM home in the socket. Now run hookup wire from IC pin 20 to socket pin 22;

“There are several ways to load the EPROM . . . Before starting, you should assemble the entire patch file to a binary file in disk and you should have a reliable blank cassette.”

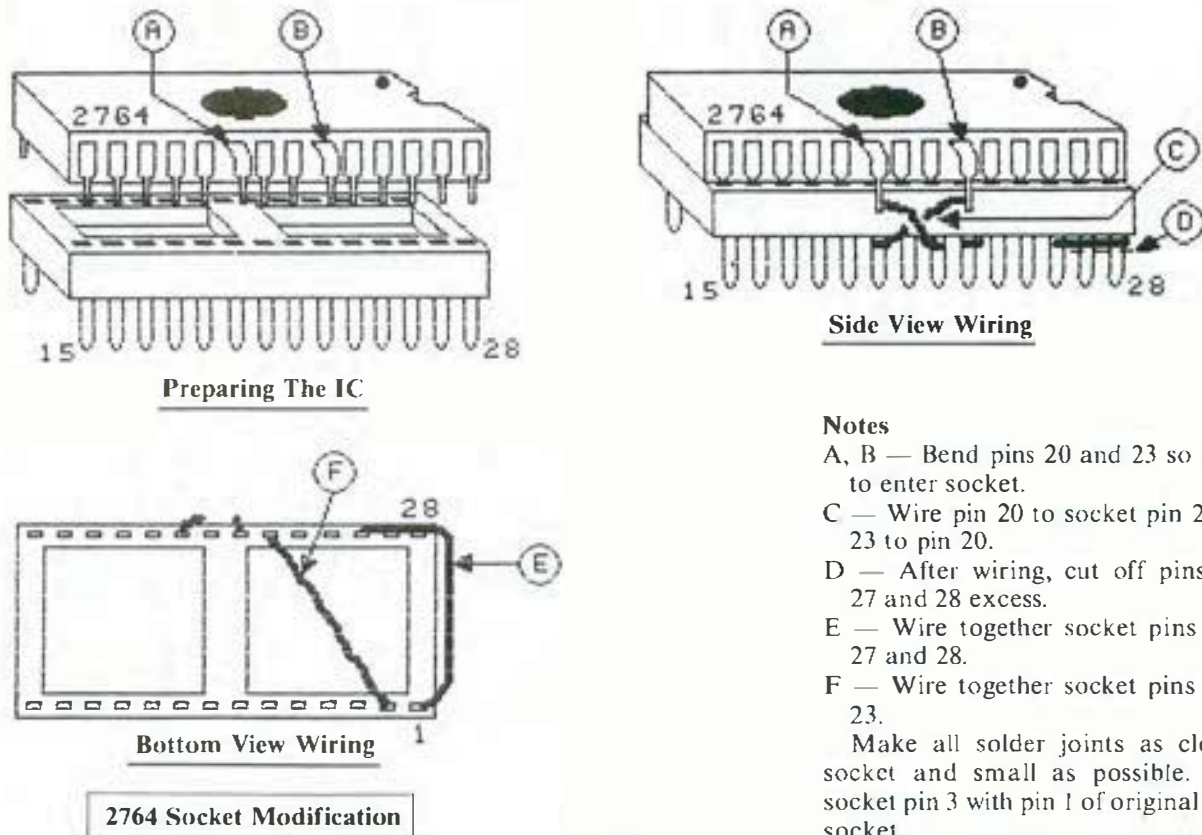
IC pin 23 to socket pin 20; socket pin 23 to socket pin 2; and also interconnect socket pins 1, 26, 27 and 28. This should be clear from Figure 1. I suggest you use wire-wrap wire available from

Radio Shack, as it is thin and strong. Make neat, small solder joints on the socket pins as these still have to go into the ROM socket in the controller. Cut off socket pins 1, 2, 27 and 28 close. Check all your connections carefully.

You should now have a 28-pin IC plugged into a 28-pin socket with only 24 pins on it. These pins now correspond exactly to the pin functions of the ROM in the disk controller cartridge. To make the swap, you must open the controller cartridge. To do this peel back the metallic label, exposing a retaining screw. Remove this, then gently pry apart the two box halves. The Disk BASIC ROM is the only 24-pin IC in the unit. Gently lever the ROM out of the socket and replace it with the prepared EPROM. Pin 3 of the 2764 EPROM should be lined up with pin 1 of the socket. There is a small capacitor near the end of the socket and this could interfere with the conversion socket where it overhangs; gently bend it out of the way. Press the EPROM down firmly, replace the cover and screw, and press back the label.

The above instruction is for the older disk controller designed for the CoCo. If you have the newer CoCo your controller is probably different. However, it will also have the 24-pin ROM

Figure 1



Notes

- A, B — Bend pins 20 and 23 so as not to enter socket.
- C — Wire pin 20 to socket pin 22, pin 23 to pin 20.
- D — After wiring, cut off pins 1, 2, 27 and 28 excess.
- E — Wire together socket pins 1, 26, 27 and 28.
- F — Wire together socket pins 2 and 23.

Make all solder joints as close to socket and small as possible. Align socket pin 3 with pin 1 of original ROM socket.

and should present no additional difficulty.

Now the acid test. Replace the controller cartridge and power up. The revised logo should appear, all drives should restore to track 0 and then drive 0 should whirl, looking for *AUTO-EXEC.BAS* to run. If you get that far you are "home and dry."

Fond Farewells

My enhancements have deliberately stayed within the 8K of the original Disk BASIC ROM, and if you have built the parallel port there are only a few bytes unused. There are many commands you might wish to add for yourself, and there is plenty of map space from \$E000 to \$FEFF available for this.

If you're running the 64K RAM version of the patch, you can use this space right now. If you went the EPROM route, maybe you could use the new 27128 EPROM or possibly piggyback two 2764s to receive the new commands. Either way, don't suffer with the limitations, do something to get rid of them!

If you intend transferring BASIC programs between a machine running

DECBI.0 and another running *DECBI.1*, some of the BASIC tokens will be different. This is due to the DOS command in *DECBI.1*. Therefore, save the BASIC file as an ASCII file (use the "A" after the *SAVE* command) and transfer will be successful. Of course, this is only needed if your program uses any of the new commands or functions.

"There are many commands you might wish to add for yourself, and there is plenty of map space from \$E000 to \$FEFF available for this."

I have greatly enjoyed cooking up this series and having you along to sample the treats these last eight months. I hope that you find my DOS enhancements useful and instructive, and they offer ways you can further personalize your CoCo.

If you would like the entire *DOS-*

PATCH program source, along with binary files with and without the parallel port driver for *DECBI.0* and *DECBI.1*, just send me a disk (no cassettes please) along with \$6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

I will program a 2764-250 EPROM with any reader-supplied code for \$25, if you furnish the EPROM, and \$35, if I do. The machine code to be programmed must be supplied in a CoCo binary file on disk. It can be put there with the *SAVEM* command. For example, to save the DOS use *SAVEM "DOS",&HC000,&HDFFF,0*. Indicate in a cover note the address range of memory saved this way. This file will be transferred to the EPROM starting at location 0 unless otherwise specified. Both disk and EPROM will be returned promptly. No other EPROM types will be programmed. EPROM contents are guaranteed to be the same as the file and nothing more.

Address this request or any questions to: Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

!!! FREE !!!



Learn
HOW to USE
YOUR Color Computer

**TRY ONE ON US
FREE
SAMPLE ISSUE
1-800-338 6800**

MON. FRI. 9-5 E.S.T.

Color Micro Journal™

5900 Cassandra Smith Rd.
Hixson, TN. 37343

TEL. (615) 842-4600 • TELEX 558 414 PVT BTH

Subscription Rates

12 Issues a Year

USA- \$12.50 per year.
Canada & Mexico- \$19.50 per year

Surface Foreign- \$24.50 per year.
Airmail Foreign- \$48.50 per year

™ Color Micro Journal is a trademark of Computer Publishing Inc.