For your TANDY Color Computer

Registered by Australia Post: Reg. No.QBG 4009

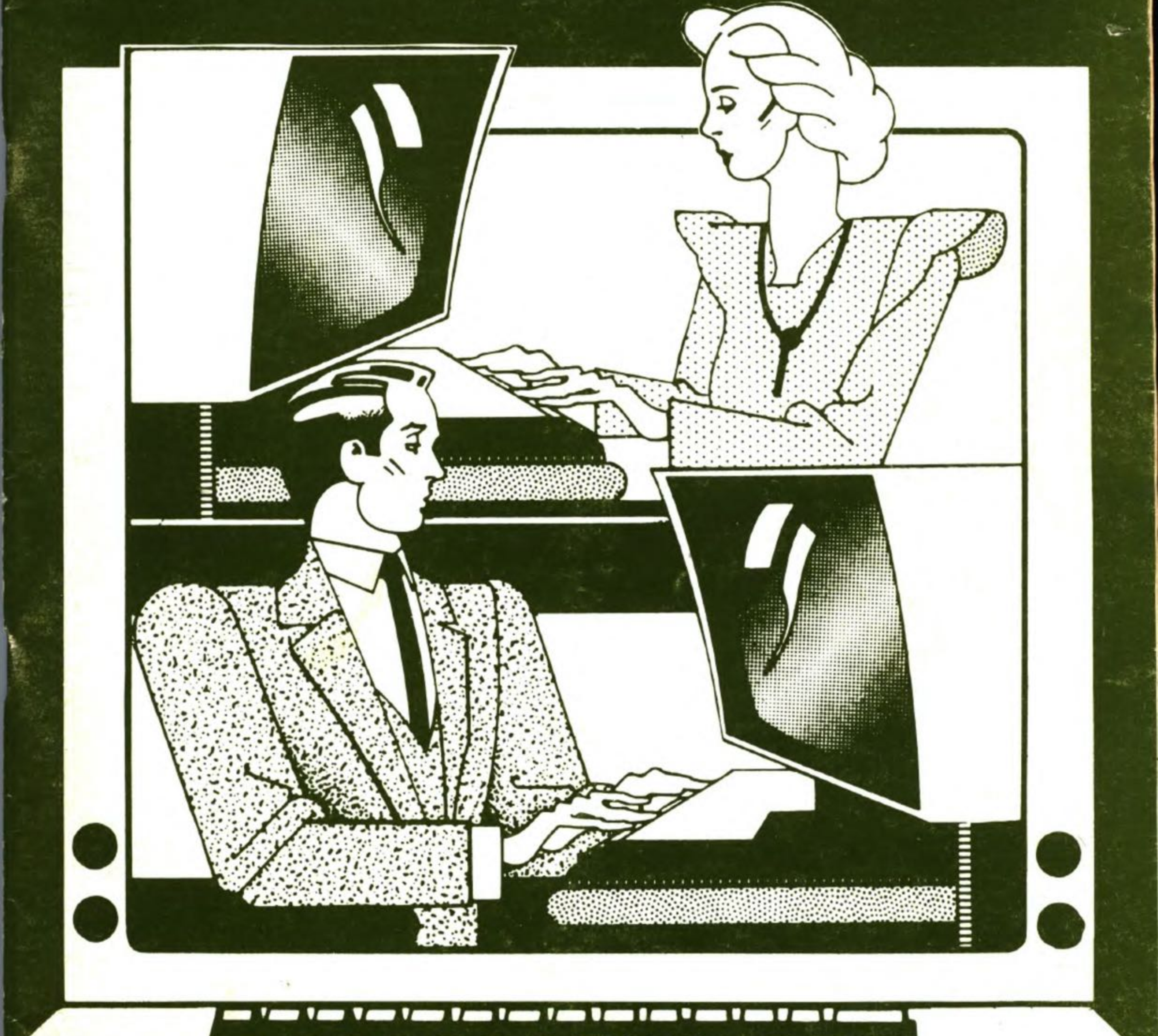
NEW GUINEA K5.95

$4.50

AUSTRALIAN RAINBOW

June, 1985

No.48



PARIS RADIO ELECTRONICS

CALL
(02) 344 9111 VOICE
(02) 344 9511 BBS
TO ORDER

CoCo OS-9™FLEX ™

Free Diskette with Each $50 Purchase

COMPILERS

K-BASIC — A BASIC language to MACHINE language Compiler; includes an Extended Macro Assembler. CCF — $292.00

PL/9 — by Graham Trott. A combination Editor/Compiler/Debugger; Structured Programming at the "almost Assembly Language" Level. CCF — $292.00

INTROL C Compiler — Full Featured C Compiler, Linking Loader includes full Library Manager. CCF — $549.00

•Forth — Forth language for CoCo
CCR — $99.95
CCF — $99.95

MICROWARE BASIC09 — for CoCo OS-9 systems. CCO — $149.95

MICROWARE C Compiler — for CoCo OS-9 Systems. CCO — $149.95

DYNA C — C Compiler from the authors of Dynastar and Dynaforms.
CCO — $99.95
CCF — $99.95

DATA BASE'S

XDMS — "Mainframe" User's say: "We don't have anything NEARLY as powerful as XDMS;" pure Assembly Language, F-A-S-T and small enough to operate on a single-sided 5" disk.
XDMS (V) I — CCF — $199.95 XDMS (V) II — CCF — $299.95
XDMS (V) III — CCF — $$399.95

RMS — Machine Language Data Base Manager — Super Fast.
CCF — $220.00
CCO — $180.00

OPERATING SYSTEMS

XEX — The latest Flex Operating System from Frank Hogg Laboratory USA. Includes hi res screen drivers, user definable keyboard, supports 128k upgrades, and 35, 40, and 80 track drives. CCF — $149.95

OS-9 for the CoCo. The hotest selling software to hit Australia. CCO — $99.95

DISASSEMBLERS

SUPER SLEUTH — Interactive; extremely POWERFUL!! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer," and Files of "Standard Label Names" for different Operating Systems included. CCF, Obj. Only $89.95
CCF, w/Source $145.00 CCO, Obj. Only $89.95

DYNAMITE + — Excellent standard "Batch Mode" Disassembler Includes XREF Generator and "Standard Label Names" Files.
CCF, Obj. Only $89.95 CCO, Obj. Only $89.95

WORD PROCESSING

STYLOGRAPH III — A full Screen-oriented WORD PROCESSOR (what you see is what you get); also supports Daisy Wheel proportional printers and WORDPAK
CCF — $165.00
CCO — $165.00

STYLO-SPELL Spelling checker to suit stylo.
CCF — $99.95
CCO — $99.95

STYLO-MAIL Mail merge program for Stylograph.
CCF — $89.95
CCO — $89.95

DYNASTAR & DYNAFORMS — Another full screen Editor and Word Processor for the CoCo.
CCF — $149.95
CCO — $149.95

XWORD — A powerful word processor that can be run under O-PAK, XSCREEN hi res drivers and WORD-PAK 80. CCO — $99.95

SPREAD SHEETS

DYNACALC — CoCo's best and fastest Spread Sheet system. CCR — $149.95

UTILITIES

UTILIX is a unix like utilities package for OS-9. It includes 15 different utilities to aid you in the manipulation of text files. CCO — $75.95

FILTERS KIT No. 1 — Eleven utilities used as filters for OS-9 CCO — $39.95

FILTERS KIT No. 2 — Ten more utilities, APPEND, CONFIRM, FF FORCERROR, MACGEN, NULDEVICE, REP, SIZE, TOUCH, and unload. CCO — $39.95

SDISK and BOOTFIX — Use 40 or 80 track, single or double sided drives with OS-9. Create a bootable double sided system disk. CCO — $59.95

HACKERS KIT No. 1 — This package contains a set of programs useful to anyone working with assembly language, trying to unravel the operating system or other assembly language programmes, or customizing their own. CCO — $59.95

XSCREEN — Hi Res Screen driver for OS-9 CCO — $39.95

O-PAK — Hi Res Screen driver for OS-9 CCO — $56.95

SEARCH AND RESCUE — disk utilities to search and retrieve lost data. CCO — $56.95

XTERM — A communications program for use with XSCREEN. CCO — $82.95

MEMORY MINDER — Disk diagnostics program. The most comprehensive program available to analyse your disk drives
DOUBLE SIDED CCR — $149.95
SINGLE SIDED CCR — $139.95

XMENU — This is a program that creates a menu driven environment for the coco under OS-9. CCO — $69.95

RAM-DISK — This program simulates a disk drive, utilizing spare RAM memory. Very fast execution of commands. For use with 64K or 128K memory, Flex, XEX, or OS-9.
CCF — $49.95
CCO — $49.95

MAGAZINES and BOOKS

AMERICAN RAINBOW — Direct from the USA the same month of publication for the same price as the newsagents. Why wait! $6.80

AMERICAN HOT COCO — Airfreighted direct each month. Subscription available. $5.80

THE OFFICIAL BASIC09 TOUR GUIDE $29.95

HARDWARE

PBJ WORKPAK 80 — an 80 column card to fit in the expansion port of the CoCo. Will run under Flex or OS-9. $199.95

PBJ WORDPAK II — The ultimate wordpak. Includes smooth scrolling, software controlled video switch, improved character set, hardware inverse video and up to 8K video RAM. $249.95

PBJ CC-BUS — A six slot software selectable expansion bus. Will take your disk controller, WORDPAK 80, GAMES Cards etc. $219.95

PBJ PC-PAK — This is a dual function cartridge which contains a centronics compatible parallel printer port and/or a battery backed real time clock.
w/ real time clock $199.95
w/o real time clock $ 99.95

PBJ 2SP-PAK — A 2 port RS-232 serial interface with programmable baud rates up to 19.2K. Each port can also be interrupt driven. $119.95

DISK CONTROLLERS

J&M Disk Controller with JDOS Disk Basic	$219.95
RADIO SHACK Disk Controller Ver 1.1	$199.95

MISC

CC-VID Monochrome video output board	$ 35.00
MC6809EP Processor chip	$ 24.00
MC6883 SAM chip	$ 24.00
4164-150 nSec, 64K chips	each $ 7.80

VISA

TM FLEX is a trademark of Technical Consultants
TM OS9 is a trademark of Microware
PRICES SUBJECT TO ALTERATION
WITHOUT NOTICE

PARIS RADIO ELECTRONICS

161 BUNNERONG RD, KINGSFORD 2032
P.O. BOX 380, DARLINGHURST 2010
Ph: 344 9111

Availability Legends —
CCF — Color Computer FLEX
CCO — Color Computer OS-9
CCR — Color Computer RS-DOS

PRICES DO NOT INCLUDE
POSTAGE AND PACKAGING

I HAVE SOMETHING TO SAY!

NEW • SOFTWARE • NEW

PICOSOFT STRATEGY GAMES

THE SPANISH ARMADA

Simulates the problems faced by the English Commanders in 1588 as they struggled to defeat the ARMADA. Cope with fickle winds. A relentless current. Difficulty of supply. $24.95

COCO - CASSETTE

Every month you will receive a cassette tape with 10 ready to run documented programs.

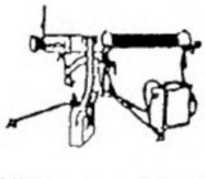
1 YR (12 ISSUES) .	$ 75 - 00
6 MO (6 ISSUES) .	$ 39 - 00
SINGLE COPIES .	$ 7 - 50

.·.Many programs use machine language!.·.

DEBACLE

Command an 18th century army in the opening campaign of the French and Indian War. Build a road through the wilderness. Establish supply lines. Cope with Indian attacks. Capture a frontier fort. $24.95

FEUER AND GASSE



Lead the American 2nd Division in a counter offensive against von Ludendorff's final drive on Paris in 1918. Recreates the battles of Belleau Wood and Chateau Thierry. Plays in real time. $24.95

All games require a 32K computer and are graphically portrayed using the semi-graphics 4 mode to depict the battle maps. Tape and Disk compatible. Games are shipped on tape.

available at
Independent Tandy Electronic Dealers

OR GIVE US A CALL OR WRITE TO

COMPUTER HUT SOFTWARE

21 Williams St., Bowen, Qld., 4805 - Phone (077) 862220

RAINBOW BITS

supplier of the latest hardware add-ons for CoCo

CoCo MODEM

AUTOMATIC $295 STANDARD $250
(P&P $5) (Reg. Mail $10)

1 **Telecom approved to 300 baud bi-directional modem.**

2 **Plugs into Rom port or expansion pack to work with disk.**
(Specify Tandy Controller or other)

3 **Inbuilt terminal programme on Eprom (no loading).**

4 **Automatic model includes push button phone with auto redial facility.**

5 Requires no power pack. Works with all models of CoCo.

6 Leaves printer port free for continuous down load to printer.

DISK CONTROLLER (TANDY Compatable)
Improved DOS, Gold Plated Contacts. Has Auto Num., OS9, Flex commands, inbuilt monitor etc. **$200**

4-SLOT EXPANSION PACK (MARK II) $135
Enables disk controller, modem, 80 column card, Eprom programmer, Rom packs etc, to be run/selected at one time. Now supports voice pack modules.

CASE CHANGER

LOWER CASE BOARD made and tested $70
(simple installation) gives true lower case (with descenders) on screen with reverse video.

PHONE (07) 302072, (07) 3690860

Write to 17 Penley St, The Gap. Q. 4061

RAINBOW BITS

AUSTRALIAN RAINBOW

index

lower case = article only .
UPPER CASE = PROGRAM + ARTICLE

All the programs listed in this magazine are available on Australian Rainbow on Tape.

Unfortunately due to the irregular arrival of the American source material we can only issue the tape as that source material is received.

Australian Rainbow on tape is available from us monthly or yearly, at $12 an issue . For those of you with FOOTSORE FINGERS.

AUSTRALIAN RAINBOW Publisher and Editor Graham Morphett. Co- editor Kevin Mischewski. Assistant Editor Sonya Young. With grateful assistance from Brian Dougan, Richard and Judy, Bob Thomson, Paul Humphries, Alex Hartmann, Michael Horn, Jim and Sheryl Bentick, Annette Morphett. Cover Art Jim Bentick.

ADVERTISING DEADLINES: The 7th of the preceeding month of publication. All advertising is arranged through ToToAdvertising, PO Box 5730, Gold Coast Mail Centre, Qld, 4217.

OS 9: Kevin Holmes is the contact for os 9 information. He also has access to OS 9 Software from the US. His address is: 39 Pearson St., Narara, NSW, 2250.

All programmes in this issue of AUSTRALIAN RAINBOW are available on Rainbow on Tape.

Telephone: 075 51 0015 Voice; 075 32 6370 CoCoLink.

Printed by: Australian Rainbow Magazine PO Box 1742, Southport, Qld, 4215.

Founder GREG WILSON.

PRINT #-2,

This month, for me, has been a most stimulating month.

At last we can say that the magazine is getting somewhere financially, which is great! Getting a new enterprise like this off the ground is always hard. Thank you for your help.

In addition to this good news, this month we have been a part of several promotions that have gone well. And there is news of more exciting software, which will ensure CoCo's popularity for some time to come.

Tandy have done some clever things this month too. The first was when they tied up distribution rights for STARS. Stars I have mentioned before, but for those of you who don't read the magazine, Stars is a 2,500,000 article library which you access with your modem.

Typically, Stars will be accessed by students who are researching topics for school projects. If the subject being studied is, say, Sulphuric Acid, the student punches "Sulphuric Acid" into Stars, and within 1.5 seconds, Stars searches all 2,500,000 articles for references to Sulphuric Acid, and places on screen, a list of the articles that are relevant.

The student reads the articles in turn, decides which ones he wants, saves the required ones to disk or tape, and exits Stars.

The article/s is/are then read into a word processor, and any changes that may be necessary, are made (depending on how much of a plagerist you are!).

I don't believe that people in remote areas have been delt a more important tool with which to assist in the education of their children! People in cities and towns will also find their children benefiting immensely. Its like having Mitchell Library in your home!

Stars is available now in all the Tandy stores, but presently only 30 stores plus all the computer centres are set up to demonstrate the product.

The second clever thing that Tandy did this month, was to include external suppliers in the launch of their Tandy 1000 in Queensland. It showed for the first time, the depth of external support for Tandy computers, thus providing buyers with additional confidence in the product.

Thirdly, Tandy's Education Sourcebook was finally released this month, and this also adds strength to the argument for the use of CoCos in schools. The Sourcebook demonstrates just what is available, not only for CoCo, but also for the other Tandy computers.

I suppose the big reason therefore, for my high this month is the growing use of CoCos for educational purposes. The ground- swell of support for the computer in this area is wonderful to see! The CoCoMax program released last month, coupled with Dynacalc, Pro Color, Telewriter 64 and the Library Borrowing System, provides the most exciting combination of educational software available at any price today.

The exciting news software-wise this month, is the program "Ears". Ears is a speech recognition package for CoCo which claims up to 95% reliability. 64 words can be kept in memory at any time, and others can be loaded quickly from disk.

Apart from Ears' obvious value in games playing, Ears makes Stars, and other information, available to the blind. For these people in particular, Ears is very good news.

If your User Group is in a position to work with this program and with local blind folk, could you contact me, and we'll discuss ways of integrating some of the work we have planned.

Please ensure that you take your magazine with you when you go shopping for the goods advertised in the Tandy ads. Despite several internal memos, some Tandy people still don't believe the offers!

I am starting a round of visits to the User Groups.

Now that the magazine is more or less on an even keel, I want to meet you, and get to understand how we can serve you better. The visits need to be conducted when the magazine is at the printer, otherwise we'll get behind - so I'm concentrating on the east coast for the first couple of months. As the system gets to cope with my excursions, I guess I'll be able to go further afield, and I'm looking forward to the trips into the more remote states.

The plan is that I'll be visiting the Northern Rivers of NSW in July, Toowoomba, the Darling Downs and the New England Region in August, and Melbourne in September. Sonya accuses me of changing my plans too often. I reckon I probably tell people too soon! Whatever, there could be some changes to these plans; I'll try to give as much warning as possible to those affected.

REVIEWS

VIDEO LEARNING SERIES

Bob Thompson

The ability to use the home video recorder as a medium for teaching various aspects of programming has probably not escaped many of our readers and yet only now do we have the first of such packages available to us. Bob Thompson has produced a package which takes the novice by the hand and begins to explore the mysteries of TANDY's OS9.

Many people are discouraged from using OS9 by it's alleged complexity and yet those who have delved into the operating system generally have nothing but high praise for its features. Most of the 'getting started' problems relate to the lack of suitable raw beginners type tutorials.

The first of Bob's Video Learning Series takes the absolute novice by the hand and SHOWS!! him how to firstly call OS9 from DISK BASIC and then step by step he is taken through the procedure required to create his own backup system disk complete with BASIC09 installed and ready to run. As an added bonus for those who have purchased the Frank Hogg OPAK hires screen driver you are shown how to install and use this package.

Throughout the entire course your T.V screen shows exactly what you should see on your own screen with Bob's commentary complementing what you see. We all make mistakes and Bob has included plenty of examples of the few more common errors along with the thought processes he uses to correct the problem as well as showing plenty of 'tricks of the trade' along the way.

The package is intended for the user groups to be presented at their meetings. This is an admirable way to introduce the novice to OS9 and certainly every user group should make the effort to present the package to its members.

The VIDEO LEARNING SERIES is available from:

BOB T Computer Services
4 Penelope Place
Cranebrook 2750.

Software Review

The Peeper Monitors Machine Language Program Operations

"*The Peeper* is an interrupt-based program tracer that lets you monitor operation of a machine language program while it is running." In other words, *The Peeper* runs at the same time as another machine language program. "*The Peeper* is designed to let you inspect any desired portion of the Color Computer's memory, even while BASIC or machine language programs are running. With *The Peeper*, you can display any part of memory, using any of the Color Computer's 26 documented display modes (test, graphics, or semigraphics). You can freeze the action at any time, or slow it down by practically any desired amount."

"*The Peeper* is capable of providing a running trace of the 6809's registers and stack, and supports single-stepping, breakpoints, and other useful program-monitoring functions." But *The Peeper* also caters to the non-machine language programmer. By looking "behind the scenes" at what BASIC is doing, you can get a better understanding of how BASIC winds its way through the computer and how various *POKE*s can change things. Also, *The Peeper* can be used to just watch machine language games running in the computer; seeing how the programs draw things on the screen, how some use hidden screens, etc., all by using *The Peeper* to control how the game is running.

To use and appreciate *The Peeper* to its fullest extent, I recommend some knowledge of machine language and the hexadecimal system, as the program is geared towards this. To get an idea of how powerful and extensive *The Peeper* is, here is a brief summary of *The Peeper's* commands.

Display Mode commands — change the display to the next mode in a cycle of 13 display modes and the two color sets.

Display Window commands — control what portion of memory is displayed on the screen. They allow you to scroll up and down through memory, go back to page zero, go back to the text screen and text mode, and go back and forth between *The Peeper's* status display page and the current display page you are looking at.

Speed commands — allow you to freeze the execution of the program being monitored, select a slow motion speed mode from one of seven (of which all can be altered, giving a very broad range of speed controls), and select slow motion speed after a 1/60 second delay so the ENTER key can be processed by BASIC.

Breakpoint command — allows the entry of up to three breakpoints. (Used by machine language programmers to halt the execution of a program at a specific address in memory.)

Trace commands — allow the trace of the program by either the address or the registers. You can display the register values and the current address being executed.

Examine Mode commands — can display any address' contents and change its value.

Other commands allow you to enable or disable output to the printer, exit *The Peeper*, and define your own "custom" functions. *The Peeper* is easy to load and comes on a cassette which can be transferred to disk. It will run in any system 16K or higher; it does not require Extended BASIC and will run with both versions of Disk BASIC. The cassette also includes two companion programs: "The first, *Rompeep*, supplements *Peeper* in that it allows you to inspect the contents of the BASIC ROMs and of ROM pack cartridges. The second program, *Find*, is a utility to aid in locating interrupt-disabling instructions that must be removed from some machine language programs in order to achieve *The Peeper* compatibility."

The manual supplied with *The Peeper* is one of the best I have seen so far for a utility program. It is long and detailed and describes every aspect of the program. It includes instructions for 64K operation and user modification as well. Also included with my review copy was a source code listing which can be purchased with the program for a couple of extra dollars -- well worth it.

One part of the manual I enjoyed was the section on using *The Peeper* with commercially available machine language games. It goes into great detail on interrupts and interrupt servicing routines, and explains how to make *The Peeper* work with quite a list of games from all companies. The ones not included are left up to the user to see if they will work with *The Peeper*, but after a thorough reading and understanding of the manual for *The Peeper*, one should be able to append the list of games provided by *The Peeper*.

If nothing else, the manual is an excellent learning and teaching tool for many aspects of the Color Computer and the 6809 processor. It answered a few questions I had come across while disassembling various programs and should become a highly used publication in your library.

For the price, buy it! Order the version with the source code because even if you're not a machine language programmer yet, you will be some day, and hopefully *The Peeper* will bring that day a little closer.

(Spectrosystems, 11111 N. Kendall Drive, Suite A108, Miami, Fl. 33176, $21.95, $24.95 with source, plus $2 S/H)

— Eldon Doucet

Software Review

Trivia Fever — A Fun Delirium

For over three years I have been fighting the invasion of games on my Color Computer. I have argued that the MC6809E should be saved from such meaningless tasks as saving the world from alien invaders and getting the frog back to the swamp. Then it happened; a game hit the street that finally grabbed me. *Trivia Fever* is not just the name of a game, it is also the name of a fun affliction.

Trivia Fever is one of the two games purchased for my Color Computer (*Zaxxon* was the first), and hopefully the first of hundreds of programs you will be seeing on the Color Computer to take advantage of a new licensing agreement with Tandy that allows a software house to include the OS-9 operating system on the program disk.

If you've been living in a cave for the past two years, *Trivia Fever* is one of the variations of the now famous game "Trivial Pursuit."

As is true of all of the trivia games, *Trivia Fever* is a competitive game to measure how much otherwise useless information you are carrying around in the memory banks of the original personal computer, your brain. What makes *Trivia Fever* different is that it comes on a disk as well as in the more conventional gameboard/scorepad format.

One of the advantages of *Trivia Fever* over the other trivia games is that you can handicap each player. There are a total of four ways to handicap more experienced "triviologists."

The first method is by requiring more experienced players to obtain more points in each category. The more experienced player may have to attain eight points per category, while a less experienced or younger player may only have to get one or two points.

Another method of handicapping is making the more experienced player answer more difficult questions. *Trivia Fever* comes with three different levels of questions. While Level 1 questions are relatively easy, the manufacturer tells us that Level 3 questions "almost require a Ph.D. in 'triviology'." My wife loves to make me play with all Level 2 and 3 questions. Level 1, as I said before, is relatively easy; I put a lot of emphasis on the word relatively. There are no easy questions in any trivia game; it is all a matter of your exposure to meaningless facts.

Next, you can handicap players by making more knowledgeable players have the computer select their five categories. There are a total of seven categories. Each player has to answer questions in only five of the seven categories. Having never paid much attention to professional sports, I find getting stuck with the sports category very humbling.

The last of the four handicapping methods is by varying the amount of time a player has to answer each question. Although this method is documented, I have found that players either know the answer within the first 10 to 15 seconds or they don't know the answer at all, and longer time limits are of little help in handicapping players. In five rounds of *Trivia Fever*, with a total of over 25 players, only one question was answered correctly after the 15 second mark.

If there is a downfall in *Trivia Fever*, it is the need for a game master. This person performs the task of operating the computer. The downfall is in that it is more fun to play *Trivia Fever* in a group than to operate the computer. The game master in *Trivia Fever* will soon know that feeling.

The game master is important though, and helps with some very important functions. Since there are a total of over 3,000 questions on the disk, there is no way to make sure the computer has not used a question unless there is a game master to keep track of whether a question is duplicated. The game master can also listen for anyone playing "from the sides," and disqualify any answers that are heard from the sidelines.

It seems that every possibility has been thought of when programming *Trivia Fever*. The game master is given the option to stop and/or restart the clock (someone ran to the bathroom), disqualify the questions (another player or a spectator exclaimed the answer), or get the computer's answer to the questions. There are no defaults, and the computer keeps a running total of everyone's standings.

The entire *Trivia Fever* package is professional and well-written. A player does not have to be a computer programmer to make the game run. Players don't have to even know what OS-9 is to use the program. (OS-9 has a start-up file that is automatically executed when OS-9 is booted. In the case of *Trivia Fever*, the game is automatically executed.) On the other hand, you must have some experience with life and trivia to effectively compete.

All in all, *Trivia Fever* is a welcomed addition to my Color Computer library, and if you have ever played a round of "does anyone know?" you will find *Trivia Fever* a welcomed addition to your software library as well.

(Available at Radio Shack stores nationwide, $29.95, manufactured by Professional Software, Inc.)

— Bruce N. Warner

EDUCATION PAGE

It was my pleasure to be invited to the launch of the Tandy 1000 in Brisbane recently.

Tandy put on their best show yet, involving 8 software support companies, ourselves included. I even had to give a speech - along with the other 7 suppliers, to the 100 specially invited guests, most of whom were from the Education Department and The Catholic Education Dept.

I think it took a show like this to help many realise the commitment to education that Tandy has made. In the US, Tandy holds 50% of the education market for computers and computer software. That's a fair slice of a very busy market!

We have not seen evidence of that commitment here in Australia till now. I think that is about to change - if only because those of us who support the Tandy range of computers will push Tandy!

There was a range of product available, but I want to mention several in particular.

John Brothers come especially from South Australia. He had his digitizer there, which we thought was damaged beyond repair two minutes before the show started, when a stand crashed down on it. The digitizer survived, and John went on to demonstrate the breadth of software he has available that is specifically made for the Education market.

The Digitizer creates a computer image of whatever is placed in front of it's lens. It is rather like a small TV camera. The images are used in CoCoMax and Graphicom to great effect.

Deloittes were there with the Maze program we have mentioned previously. I had time to delve more deeply into its timetableing abilities and found it as capable as we expected. This program offers the solution to the problem of administrating a school by computer. Maze runs on the Tandy 2000, and should also run on the T1000. We'll be checking this possibility out and will report!

There were two bulletin board style data bases at the launch. Agridata offers information on a very detailed variety of Agricultural topics. Whilst aimed at the man on the land, its resources are available to the student who needs to research agricultural topics.

And then there was Stars.

Stars should be thought of as a library. You can choose to read from a selection of two and a half million articles on a very broad range of topics.

Through Stars you have access to the first computerised dictionary, (The Macquarie Dictionary).

You can search for information which satisfies several parameters.

You can communicate for the cost of a local phone call plus $25/hour, with people in Australia, The U.S, Europe, Japan, Hong Kong and Singapore.

(You could do undertake a class project with a school, say, in Los Angeles or Tokyo.)

Stars brings the facilities of a total library to your home and makes private study a breeze. Stars is absolutely essential for children in remote areas, and a must for the city dweller too.

Ask your Tandy shop for a demonstration of Stars - it is too important a development for you to ignore!

Finally, in addition to our magazine, our stand featured several products, which we have been greatly impressed by - especially as they relate to the school environment.

We had the Jangen monitor there - both amber and green screens. This monitor is featured in the Centurion advertisment and is also supplied by Blaxland Computer Services.

Of course, we blew all minds with CoCoMAX - it was the first thing they saw as they came through the door, and the last as they left. We were loaned several youngsters from the Springwood South school, and they quickly mastered the MAX, creating a number of drawings which we stored on disk.

Thanks to Peter Collison for the supply of the MAX - he went to considerable pain to ensure its arrival in time.

We had Paris Radio's Word Pac II 80 column card on two computers. One was running a demonstration program which switches between the 80 columns and the standard graphics screen automatically. The program also demonstrates the very smooth scrolling that is a feature of this card.

Kevin had Stylograph III runing in conjunction with the Word Pac on the second computer, and it looked great. Stylo can work with O Pac - a 51 column graphics screen, but when latched to the 80 column card, Stylo would have you believing very quickly that you were dealing with a much larger computer.

Kevin also had Dynacalc and Telewriter going. Dynacalc is a spreadsheet we hope to review in the near future. It can do all the usual tricks that spreadsheets do, and produce several types of graphics to illustrate the spreadsheet. Finally, the information is transferable to Telewriter and to Pro Color - a database.

Speadsheets have the same value in the classroom that they have in business. That is, "what if" games can be played with information and comparitive performances can be ably illustrated. Later, the information can be transferred to a Data Base for other comparisons - perhaps multi relational. Then the information can be read into a word processor such as Telewriter, so that a report can be written.

We thank the advertisers who supplied the gear we used on the day, and a big thanks to Tandy, who once more broke new ground - it is the first time they have attempted anything like this - I have a feeling it will not be the last!

Our Education section this month reflects our continuing commitment to Education, both with programing assistance, and from the philosphical stand point.

We repeat our earlier statement that we need to hear from those of you who are teachers using computers in the classroom. It is one thing to have a computer in the classroom, it is another to make it effective. So we look forward to your reactions to the following articles.

Bob Horne provided a series of programs that he uses in his classroom. The second one "Antonyms" is presented this month.

ANTONYMS by Bob Horne

The Scene: You are being attacked by robots carrying swords. You are in charge of a swivelling gun. You can hold back the robots by destroying the word which is an antonym of the one beneath your gun position!

This program uses a machine language routine to print upper and lower case letters on the graphic screen. Rather than load this separately as an assembled program, I have included data statements and the routine is poked into upper memory. This routine will print 42 characters per line and 24 lines before scrolling. You can not use PRINT@. Instead, a POKE to 243 sets the horizontal position for printing. You may POKE a value from 0 to 41. To set the vertical position, Poke 244 with a value from 0 to 23. For example: POKE243,6:POKE244,10:PRINT"something" will start printing 7 spaces across and 11 lines down.

The program was written for a 32K CoCo, so if you have only 16K, there are quite a lot of changes to be made, but I am sure it could be done.

Readers may wish to change the DATA statements as the examples are about YEAR 5 level. Many text books contain examples of Antonym Exercises suitable for other year levels.

Note that the DATA pairs in lines 1320- must be given in this form: WORD,ANTONYM/ANOTHER WORD/ANOTHER WORD

THE LISTING:

32K

```
1 'ANTONYMS    BOB HORNE
2 GOTO10
3 SAVE"ANTONYMS":STOP
5 '*****************************
6 '*         ANTONYMS          *
7 '*       BY BOB HORNE        *
8 '*       IPSWICH, QLD.       *
9 '*****************************
10 CLS:PRINT@128,"NOW LOADING MA
CHINE LANGUAGE.   STAND BY."
20 CLEAR800,31918
30 N=104:DIMWO$(N),AN$(N),A1(N),
A(5),B(5),C(5),D(20),E(20):EX$="
T15001V31CDCGABV25GCABV20GCABV15
GDABV10GDABV5EDABV1EDAB":B$="V30
03T100L50CDEFG"
40 MO$="U2R12H5UL2D3LU7RD3R2UE5R
10F5DR2U3RD7LU3L2D65R12D2"
50 M1$="D8LD2RD6L5D2R15U2L5U6RUN
RULU8"
60 PMODE4,1:PCLS1
70 A$="BM20,20U5RU5RU9R3D9RD5RD5
L7"
80 DRAW"C0XA$;":PAINT(23,10):GET
(20,0)-(27,20),B,G
90 DRAW"A1XA$;":PAINT(30,23):GET
(20,20)-(40,27),C,G
100 DRAW"A3XA$;":PAINT(10,17):GE
T(20,20)-(0,13),A,G:DRAW"A0"
110 PCLS1:FORX=1TO50:PSET(RND(70
),RND(9),0):NEXTX
120 GET(1,1)-(69,9),D,G:GET(1,11
)-(69,19),E,G:PCLS1
130 FORX=31919TO32767
140 READA$:POKEX,VAL("&H"+A$)
150 NEXTX
160 FORA=1TO N:READWO$(A),AN$(A)
:NEXTA
170 GOTO340
180 NA=1
190 LS=INSTR(AN$,"/"):IF LS=0 TH
ENA$(NA)=AN$:GOTO210
200 A$(NA)=LEFT$(AN$,LS-1):AN$=R
IGHT$(AN$,LEN(AN$)-LS):NA=NA+1:G
OTO190
210 SL=0
220 FORC=3TO1 STEP-1
230 Z5=RND(C)
240 IF Z5=1 AND SL=0 THENSL=1:CA
=C
250 T$=A$(C):A$(C)=A$(Z5):A$(Z5)
=T$
260 NEXTC
270 RETURN
280 POKE243,H:POKE244,V:PRINTPR$
;:RETURN
290 F2=1:MI=MI+1:H=37:V=22:PR$=S
TR$(MI):GOSUB280:RETURN
300 COLOR0:IF C1=1 THEN LINE(103
,114)-(71,114),PSET ELSE IF C1=2
 THEN LINE(129,89)-(129,41),PSET
 ELSE IF C1=3 THEN LINE(154,114)
-(184,114),PSET
310 SOUND254,1
320 COLOR1:IF C1=1 THEN LINE(103
,114)-(71,114),PSET ELSE IF C1=2
 THEN LINE(129,89)-(129,41),PSET
 ELSE IF C1=3 THEN LINE(154,114)
-(184,114),PSET
330 RETURN
340 EXEC31919:Z=RND(-TIMER)
350 TI$="ANTONYMS":H=16:V=10
360 FORX=1TOLEN(TI$):PR$=MID$(TI
$,X,1):GOSUB280:PLAY"L100T2003AD
GL1P1":H=H+1:NEXTX
370 H=19:V=12:TI$="BY"
380 FORX=1TOLEN(TI$):PR$=MID$(TI
$,X,1):GOSUB280:PLAY"L100ADGL1P1
":H=H+1:NEXTX
390 H=15:V=14:TI$="BOB  HORNE"
400 FORX=1TOLEN(TI$):PR$=MID$(TI
$,X,1):IF PR$=CHR$(32) THEN H=H+
1:NEXTX ELSE GOSUB280:PLAY"L100A
DGL1P1":H=H+1:NEXTX
410 FORX=1TO500:NEXTX
420 EXEC31930:HI=0:MI=0:H=5:V=10
:PR$="Do you want instructions ?
":GOSUB280
430 H=10:V=12:PR$="Type 'Y' or '
N'":GOSUB280
440 IN$=INKEY$:IF IN$="" THEN440
 ELSE IF IN$="Y" THENGOSUB1010 E
LSE IF IN$="N" THEN450 ELSE440
450 Q=0
460 EXEC31930:COLOR0:LINE(125,10
9)-(132,119),PSET,BF
470 LINE(118,118)-(138,121),PSET
,BF:LINE(0,170)-(98,122),PSET:LI
NE-(165,122),PSET:LINE-(255,170)
,PSET
480 LINE(0,110)-(70,120),PSET,B:
LINE(98,30)-(168,40),PSET,B:LINE
(185,110)-(255,120),PSET,B
490 DRAW"BM18,110XM0$;":DRAW"BM1
16,30XM0$;":DRAW"BM203,110XM0$;"
:DRAW"BM30,100R10D3L10U3":DRAW"B
M128,20R10D3L10U3":DRAW"BM215,10
0R10D3L10U3"
500 DRAW"BM18,120XM1$;BR24XM1$;"
:DRAW"BM116,40XM1$;BR24XM1$;":DR
AW"BM203,120XM1$;BR24XM1$;"
510 PAINT(40,106):PAINT(135,28):
PAINT(225,106)
520 PAINT(20,122):PAINT(50,122):
PAINT(118,42):PAINT(148,42):PAIN
T(205,122):PAINT(235,122)
530 LINE(98,126)-(168,136),PSET,
B
540 H=0:V=22:PR$="HITS:"+STR$(HI
):GOSUB280:H=30:V=22:PR$="MISSES
:"+STR$(MI):GOSUB280
 550 FORTU=1TO20
 560 Z=RND(N):IF A1(Z)=1 THEN560
 ELSE A1(Z)=1:AN$=AN$(Z)
```

```
570 GOSUB180
580 H=5-LEN(A$(1))/2:V=14:PR$=A$
(1):GOSUB280
590 H=22-LEN(A$(2))/2:V=4:PR$=A$
(2):GOSUB280:H=36-LEN(A$(3))/2:V
=14:PR$=A$(3):GOSUB280:H=22-LEN(
WO$(Z))/2:V=16:PR$=WO$(Z):GOSUB2
80
600 FORX=1TO300:NEXTX
610 F1=0:FORC1=1TO4
620 IF C1=4 THEN C1=2:F1=1
630 IF C1=1 THEN PUT(124,111)-(1
04,118),A,PSET ELSE IF C1=2 THEN
 PUT(125,109)-(132,89),B,PSET EL
SE IF C1=3 THEN PUT(133,111)-(15
3,118),C,PSET
640 PLAYB$
650 FORX=1TO15
660 IN$=INKEY$
670 IF IN$=CHR$(32) THENGOSUB300
:IF C1=CA THEN X=15:NEXTX:C1=4:N
EXTC1:GOTO740
680 IF IN$=CHR$(32) AND C1<>CA T
HEN GOSUB290
690 NEXTX
700 COLOR1:IF C1=1 THEN LINE(124
,111)-(104,118),PSET,BF ELSE IF
C1=2 THEN LINE(125,109)-(132,89)
,PSET,BF ELSE IF C1=3 THEN LINE(
133,111)-(153,118),PSET,BF
710 IF C1=2 AND F1=1 THEN C1=4:F
1=0
720 NEXTC1
730 GOTO610
740 IF F2=0 THEN COLOR1:HI=HI+1:
H=5:V=22:PR$=STR$(HI):GOSUB280 E
LSE F2=0
750 IF CA=1 THENPUT(1,111)-(69,1
19),D,PSET ELSE IF CA=2 THEN PUT
(99,31)-(167,39),D,PSET ELSE IF
CA=3 THEN PUT(186,111)-(254,119)
,D,PSET
760 PLAYEX$:PLAY"P2"
770 X=RND(6):ONX GOTO780,790,800
,810,820,830
780 PR$="TERRIFIC!!!!":GOTO840
790 PR$="KEEP IT UP!!":GOTO840
800 PR$="GREAT WORK!!":GOTO840
810 PR$="!!!!WOW!!!!!":GOTO840
820 PR$="TOP EFFORT!!":GOTO840
830 PR$="REAL COOL!!!"
840 H=16:V=18:GOSUB280:PLAY"L255
V1003ADEBCDGGFAACDFBEDEBCDV5BCGG
DFEGDFACABACDBDGE":FORX=1TO200:N
EXTX:PR$=STRING$(12,32):GOSUB280
850 PUT(1,111)-(69,119),E,PSET:P
UT(99,31)-(167,39),E,PSET:PUT(18
6,111)-(254,119),E,PSET:PUT(99,1
27)-(167;135),E,PSET
860 IF CA=1THENLINE(124,111)-(10
4,118),PSET,BF ELSE IF CA=2 THEN
 LINE(125,109)-(132,89),PSET,BF
ELSE IF CA=3 THEN LINE(133,111)-
(153,118),PSET,BF
870 Q=Q+1:IF Q=N THEN Q=0:FORX=1
TO N:A1(X)=0:NEXTX
880 NEXTTU
890 EXEC31930
900 POKE244,9:PRINT"
 HITS         MISSES"
910 PRINTSTRING$(41,45)
920 PRINT"CURRENT       "HI"
     "MI
930 IF HI>HH THENHH=HI
940 IF MI>MH THEN MH=MI
950 PRINT"HIGH          "HH"
     "MH
960 PRINTSTRING$(41,45)
970 H=8:V=20:PR$="Press <ENTER>
to continue.":GOSUB280
980 IN$=INKEY$
990 IN$=INKEY$:IF IN$="" THEN990
 ELSE IF IN$=CHR$(13) THENHI=0:M
I=0:FORX=1TO104:A1(X)=0:NEXTX:GO
TO420
1000 GOTO990
1010 EXEC31930
1020 POKE243,16:PRINT"ANTONYMS"
1030 PRINT:PRINT" ANTONYMS are w
ords of opposite meaning.":PRINT
:PRINT" For example, NOISY has t
he opposite      meaning of QUIE
T."
1040 PRINT:PRINT" NOISY and QUIE
T therefore are ANTONYMS."
1050 H=7:V=20:PR$="Press <ENTER>
 to continue.":GOSUB280
1060 IN$=INKEY$:IF IN$="" THEN10
60 ELSE IF IN$<>CHR$(13) THEN106
0
1070 EXEC31930:POKE243,16:PRINT"
ANTONYMS"
1080 PRINT:PRINT" You are in cha
rge of the swinging gun on the t
op of the hill. Beneath the gun
is a word for which you have to
find the        ANTONYM. Around t
he screen are three word-carryin
g robots."
1090 PRINT:PRINT" When the gun i
s pointing to the ANTONYM  press
 the SPACEBAR. If you have chose
n thecorrect ANTONYM the word wi
ll disappear   and you will scor
e a hit."
1100 PRINT:PRINT" A hit is only
scored if you choose the   right
 word at the first try."
1110 PRINT:PRINT"
 GOOD LUCK"
1120 H=9:V=20:PR$="Press <ENTER>
 to start.":GOSUB280
1130 IN$=INKEY$:IF IN$="" THEN11
30 ELSE IF IN$<>CHR$(13) THEN113
0
1140 RETURN
1150 DATACC,06,00,97,F5,DD,F6,8B
,18,DD,F8,0F,F3,0F,F4,CC,FF,FF,9
E,F6,ED,81,ED,81,9C,F8,25,F8,30,
8C,0A,BF,01,6B,30,8C,39,BF,01,68
,39,0D,6F,27,03,7E,8C,F1,0F,70
1160 DATA34,14,CC,01,00,34,06,BD
,A1,C1,26,12,6A,E4,26,F7,63,61,2
7,03,86,80,8C,86,20,17,00,A7,20,
E9,A7,E4,86,20,17,00,9E,35,06,35
,14,32,62,39,0D,6F,27,01,39,34,1
6
1170 DATA86,F8,B7,FF,22,96,F5,44
,8A,6C,9F,FF,C6,A7,1A,A7,1D,A7,1
F,44,27,0C,25,04,A7,81,20,F7,A7,
01,30,02,20,F1,A6,E4,81,20,25,49
,8D,66,DC,F3,4C,81,2A,25,39,4F,9
7
1180 DATA89,5C,34,04,DB,F6,D0,F8
,35,04,25,2B,5A,34,66,9E,F6,33,8
9,01,00,20,0E,37,26,ED,81,10,AF,
81,37,26,ED,81,10,AF,81,11,93,F8
,25,ED,CC,FF,FF,ED,81,ED,81,9C
1190 DATAF8,26,F8,35,66,DD,F3,35
,16,32,62,39,81,0D,26,04,D6,F4,2
0,B8,81,08,26,EF,DC,F3,4A,2A,06,
86,2A,4A,5A,2B,E4,DD,F3,86,20,8D
,02,20,DC,34,36,34,06,96,F3,C6
1200 DATA06,3D,CB,02,54,49,54,49
,54,49,A7,60,96,F4,9B,F6,1F,01,A
6,60,31,8C,51,A6,A6,A7,60,43,A7,
61,31,8C,AF,A6,62,C6,05,D7,89,3D
,31,AB,CC,08,06,34,06,5F,20,06,8
6
1210 DATA08,A7,E4,E6,A0,59,A6,84
,24,04,A4,63,20,02,AA,62,A7,84,3
0,88,20,6A,E4,26,EC,30,89,FF,00,
6A,61,27,10,A6,62,44,24,04,30,01
,86,80,A7,62,43,A7,63,20,CE,32
1220 DATA64,35,B6,80,08,20,02,40
,04,10,01,00,00,00,00,00,00,00,F
A,00,00,00,E0,00,E0,00,28,FE,00,
FE,28,24,54,D6,54,48,C6,C8,10,26
,C6,6C,92,6A,04,0A,00,00,E0,E0,0
0
1230 DATA38,44,82,00,00,00,00,82
,44,38,10,38,7C,38,10,10,10,7C,1
0,10,1A,1C,00,00,00,10,10,10,10,
10,06,06,00,00,00,06,08,10,20,C0
,7C,8A,92,A2,7C,00,42,FE,02,00
1240 DATA4E,92,92,92,62,44,82,92
,92,6C,10,30,50,FE,10,E4,A2,A2,A
2,9C,7C,92,92,92,0C,86,88,90,A0,
C0,6C,92,92,92,6C,60,92,92,92,7C
,00,6C,6C,00,00,00,DA,DC,00,00,1
```

```
0
1250 DATA28,44,82,00,28,28,28,28
,28,00,82,44,28,10,40,80,9A,60,0
0,4C,92,9A,82,7C,3E,48,88,48,3E,
82,FE,92,92,6C,7C,82,82,82,44,82
,FE,82,82,7C,FE,92,92,92,82,FE
1260 DATA90,90,90,80,7C,82,82,92
,9E,FE,10,10,10,FE,00,82,FE,82,0
0,0C,02,02,02,FC,FE,10,28,44,82,
FE,02,02,02,02,FE,40,30,40,FE,FE
,40,20,10,FE,7C,82,82,82,7C,FE,9
0
1270 DATA90,90,60,7C,82,8A,84,7A
,FE,90,98,94,62,44,A2,92,8A,44,8
0,80,FE,80,80,FC,02,02,02,FC,E0,
18,06,18,E0,FE,04,18,04,FE,C6,28
,10,28,C6,C0,20,1E,20,C0,86,8A
1280 DATA92,A2,C2,FE,82,82,00,00
,C0,20,10,08,06,00,00,82,82,FE,2
0,40,80,40,20,01,01,01,01,01,00,
80,40,20,00,1C,22,22,3C,02,FE,12
,22,22,1C,1C,22,22,22,04,1C,22,2
2
1290 DATA12,FE,1C,2A,2A,2A,18,00
,10,7E,90,40,18,25,25,25,1E,FE,2
0,20,1E,00,00,00,BE,00,00,04,02,
22,BC,00,FE,08,14,22,00,00,00,FE
,00,00,1E,20,1E,20,1E,3E,10,20
1300 DATA20,1E,1C,22,22,22,1C,3F
,24,24,24,18,18,24,24,24,3F,3E,1
0,20,20,00,12,2A,2A,2A,04,20,7C,
22,04,00,3C,02,02,04,3E,38,04,02
,04,38,3C,02,0C,02,3C,22,24,3E,1
2
1310 DATA22,38,05,05,05,3E,22,26
,2A,32,22,10,6C,82,82,00,00,00,F
F,00,00,00,82,82,6C,10,30,40,20,
10,60,AA,55,AA,55,AA,FF,FF,FF,FF
,FF
1320 DATAWRITER,READER/STORE/EXP
LAIN
1330 DATACROOKED,STRAIGHT/PATHWA
Y/INSIDE
1340 DATAYOUNGSTER,ADULT/CHILD/C
OUSIN
1350 DATATAME,WILD/NARROW/YESTER
DAY
1360 DATASTRANGE,KNOWN/JOURNEY/B
ICYCLE
1370 DATASENSIBLE,FOOLISH/TOGETH
ER/AHEAD
1380 DATASPEEDY,SLOW/JOCKEY/LIBR
ARY
1390 DATAPRAISE,BLAME/PROBLEM/AT
TEND
1400 DATAWATCHMAN,INTRUDER/WATCH
/ARRIVAL
1410 DATATENANT,LANDLORD/ILLNESS
/CAUSEWAY
1420 DATAFORBIDDEN,LAWFUL/EARTH/
TRACTOR
1430 DATASWELL,SHRINK/GREATER/SM
ELL
1440 DATAFINISH,BEGIN/RAILWAY/ME
SSAGE
1450 DATAFLAT,HILLY/WATCH/ORANGE
1460 DATAINSULT,FLATTER/BANANA/H
OSPITAL
1470 DATALEADER,FOLLOWER/AMOUNT/
SALT
1480 DATAWORKER,MANAGER/CASTLE/A
LREADY
1490 DATAREMEMBER,FORGET/MINGLE/
SHOUT
1500 DATAPURE,FOUL/ALLOW/SPEEDY
1510 DATAWHOLE,FRAGMENT/LATELY/N
AUGHTY
1520 DATACALMNESS,FRENZY/BOXES/B
ALLOON
1530 DATASMILE,SCOWL/SAFETY/MUSI
CAL
1540 DATALOSS,PROFIT/AWFUL/PASS
1550 DATASPEND,HOARD/BIRTHDAY/CO
TTAGE
1560 DATAPRECISE,VAGUE/WEATHER/N
ARROW
1570 DATASORROWFUL,JOVIAL/SPADE/
SEW
1580 DATAEVIL,GOOD/HEIGHT/NEWSPA
PER
1590 DATARUDE,POLITE/CAPTAIN/MEA
DOW
1600 DATATHREATEN,DEFEND/BREAKFA
ST/PATH
1610 DATAINTRUDER,GUARD/BLAZE/BA
LANCE
1620 DATAHONEST,DISHONEST/CHIMNE
Y/ELECTRIC
1630 DATAINNOCENT,GUILTY/BLANKET
/ADMIRE
1640 DATADOUBLE,HALVE/SNAKE/DESE
RT
1650 DATAEASY,DIFFICULT/PEOPLE/Y
ELLOW
1660 DATAFRAIL,RUGGED/WEAKLY/FRO
STY
1670 DATADISCOURAGE,HEARTEN/BROA
DEN/UNDER
1680 DATALIGHT,WEIGHTY/FEMALE/OF
FICE
1690 DATALOW,LOFTY/FLOWER/STRANG
E
1700 DATAWASTE,SAVE/WAIST/SIGNAL
1710 DATAMISS,COLLIDE/COLLAR/BEC
AUSE
1720 DATARELEASE,CAPTURE/PARENT/
SNEEZE
1730 DATAINDIVIDUAL,CROWD/BROWN/
INSIDE
1740 DATACRUEL,HUMANE/CRUSTY/GAR
DEN
1750 DATAARROGANT,HUMBLE/SAUCER/
USEFUL
1760 DATADOUSE,IGNITE/FAULTY/SLU
MBER
1770 DATALAWFUL,ILLEGAL/VIRTUE/Y
AWN
1780 DATAHEALTHY,ILL/NOBODY/WEAL
THY
1790 DATAASSIST,OBSTRUCT/BENEATH
/UNTIL
1800 DATACERTAIN,UNLIKELY/PLUNDE
R/FIERCE
1810 DATAPOLITE,IMPUDENT/THIRSTY
/SECRET
1820 DATAFINISHED,INCOMPLETE/SEC
RET/HARMFUL
1830 DATATRUE,INCORRECT/MYSELF/C
ORRECT
1840 DATAASCEND,DESCEND/CLIMB/SH
ORT
1850 DATAWEALTHY,PENNILESS/TOWAR
DS/SIGNED
1860 DATAPLEASED,UNHAPPY/DAUGHTE
R/HEART
1870 DATASTRICT,LAX/FRUIT/STRENG
THEN
1880 DATAWORTHLESS,PRICELESS/STA
RE/GENERAL
1890 DATAMATURE,CHILDISH/GLASS/N
ATURE
1900 DATASAFE,DANGEROUS/LAUGHTER
/ARTIST
1910 DATASOBER,DRUNK/SARDINE/SCA
TTER
1920 DATAOBVIOUS,CONCEALED/WESTE
RN/KANGAROO
1930 DATAOUTSIDERS,RELATIONS/UNL
IKELY/MUSICIANS
1940 DATAMEAN,GENEROUS/PARCEL/PE
NCIL
1950 DATADEPRESSION,BULGE/KNOWN/
RECEIVE
1960 DATASUCCEED,FAIL/SUMMER/STA
RT
1970 DATAEXCESS,SHORTAGE/RABBIT/
SLIPPERS
1980 DATAREJOICE,LAMENT/NEARLY/V
ISITOR
1990 DATAAROUSE,QUIETEN/AROUND/M
ACHINE
2000 DATAPLUMP,LEAN/AUTUMN/MILKE
D
2010 DATAINSTRUCTOR,STUDENT/STRA
IGHT/BOUNDARY
2020 DATAMASTER,SERVANT/GARDEN/S
KILL
2030 DATAWORK,LEISURE/PRESSED/LA
BOUR
2040 DATAFORBID,PERMIT/MARKET/PO
```

```
LICE
2050 DATACONFINE,RELEASE/ENGINE/
CARRIER
2060 DATADARKNESS,LIGHT/STUDIO/H
UNGRY
2070 DATALISTLESS,ACTIVE/GHOST/P
UNISH
2080 DATAOPEN,FASTEN/WILDEST/BEL
IEVE
2090 DATAHURRY,DAWDLE/SORRY/FORE
ST
2100 DATAQUIET,DEAFENING/ROUND/J
OURNEY
2110 DATAUGLY,BEAUTIFUL/CROWD/FO
UGHT
2120 DATAHEAL,MAIM/FEWEST/COURT
2130 DATANEGLECT,PRESERVE/HIGHER
/POCKET
2140 DATADESTROY,CONSTRUCT/SPEAK
/THIEF
2150 DATAPERMANENT,TEMPORARY/GRO
CER/GIANT
2160 DATAFEW,MANY/FRIGHTEN/LOSS
2170 DATANOBLE,HUMBLE/MESSAGE/HO
IST
2180 DATASAFETY,THREAT/KETTLE/VA
RNISH
2190 DATAORDER,CONFUSION/FINAL/K
ITTEN
2200 DATAMIDNIGHT,NOON/MIDDLE/GR
UFF
2210 DATACHEERFUL,SOLEMN/INTERES
T/DIVIDE
2220 DATATALKATIVE,SILENT/GRASS/
COLOUR
2230 DATASHORTAGE,GLUT/SHORTEST/
FAMOUS
2240 DATACONCEAL,DISPLAY/STREAM/
BIDDEN
2250 DATAPUBLIC,PRIVATE/UNLESS/S
ATISFY
2260 DATATHIN,PORTLY/LEMON/THREA
D
2270 DATASILENCE,UPROAR/TOUGH/SI
NCERE
2280 DATASECURE,UNSAFE/MONEY/PRO
PERTY
2290 DATASOLUTION,PUZZLE/SEVENTY
/SHOCK/SHIELD
2300 DATAOLD,MODERN/TONIGHT/AGED
2310 DATASELDOM,FREQUENT/DROWN/S
ELFISH
2320 DATARARELY,OFTEN/EXPECT/SLO
WLY
2330 DATASHORTAGE,GLUT/POSTAGE/S
HORTEST
2340 DATACOMMON,RARE/PRAISE/SOME
THING
2350 DATASTRANGER,FRIEND/GENUINE
/SUPERIOR
```

16K
ECB

Does This Sentence Make Sense?

By Steve Blyn

There has been a recent trend back to stressing the "basics" in education. It was found that too many students who lacked basic skills were being promoted and even graduated from many schools. I feel this return to basics is essential to education in today's world.

The emphasis has been on reading, math and writing skills. We began to get away from these basics in the '60s and this trend continued into the '70s. A greater significance was placed on thought processes rather than on content. This kind of thinking was fine, but somehow the basics got lost in the shuffle. Average student achievement scores on standardized tests declined over the years.

This trend has been largely reversed in the '80s. The three R's are once again back in fashion. A few years ago, the New York City Board of Education instituted the "Gates Program." Every student is now given standardized tests in math and reading every few years. Minimum scores must be obtained on these tests in order to pass to the next grade. A student may pass all of his subjects, but must still pass the standardized exams to be promoted.

This type of program removes any capricious promotions of students who seem to have done well in their class work, but nevertheless have not achieved minimum competency in reading and math. The students who do not pass these tests are retained in their present grades and receive intensive small group instruction in the needed area the following year. Extra allowances, of course, are provided on these tests for learning disabled students.

There is also a third part of the Gates Program. This is a writing test. The test consists of three writing assignments. The student must show that s/he can properly communicate thoughts in writing. Since there is much more room for subjectivity in grading writing assignments, the tests are graded by teams of teachers. The student again must achieve a minimum passing grade.

I have participated on these grading teams and been astonished to discover some excellent, and some very poor, writing by junior high school students. There is surely a need to help a good number of our students to write using proper sentence structures. Therefore, it is the skill of writing properly that is the aim of our article this month.

Very common mistakes are in basic sentence structure. Some students have little idea when to stop their sentences with a period. This results in catastrophic run-on sentences. Other students end their sentences without finishing a complete thought. In this instance, either a subject or a predicate is left out.

This program contains four sentence parts in the DATA statements. The data is read on lines 80-110. The actual data is contained on lines 390-420. The sentence parts are subjects, verbs, prepositions and adverbs. Phrases or sentences are presented randomly. The user must decide if the words presented comprise a complete sentence.

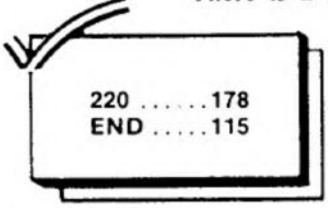
We will consider the words presented to be a sentence if both a subject and a predicate are present. The sentences 210. Line 150 picks random DATA statements from the four sentence parts A, B, C, and D. The variable 'E' will randomly choose to print one of the lines from 160-210. This will determine the phrase or sentence that is printed each time. The variable 'X' acts as a pointer to tell the computer whether a complete sentence has been selected. If X=1 then we have a sentence; if X= 1 then we have a phrase.

Four phrases or sentences are presented on each screen. After each appears, the child should type 'Y' or 'N' for a complete sentence. At the end of each set of four, the student may continue or end the program.

There is a final scoring at the end of the program. This will enable the student to check progress made in mastering this skill.

You may easily modify this program to suit individual needs. The data may be updated for your purposes. It is always best to include names that relate to the users. This requires no real programming changes.

If you change the amount of data within the DATA statements, be certain to also change the DIM statement on Line 30 and the READ statement on either lines 80, 90, 100 or 110. As always, Computer Island enjoys hearing from you about our ideas and programs. Write to 227 Hampton Green, Staten Island, NY 10312.



220 178
END 115

The listing:

```
10 REM"IS THIS A SENTENCE?"
20 REM" STEVE BLYN,COMPUTER ISLA
ND,NY,1985"
30 DIMA$(15),B$(5),C$(5),D$(5)
40 CLS
50 RESTORE
60 K=0
70 PRINT"          IS THIS A SENTENC
E ?":PRINTSTRING$(32,255);
80 FORT=1TO15:READA$(T):NEXTT
90 FORT=1TO5:READB$(T):NEXTT
100 FORT=1TO5:READC$(T):NEXTT
110 FORT=1TO5:READD$(T):NEXTT
120 FOR J=1TO4
130 PRINTSTRING$(32,239);::PRINT
" ";
140 REM"RANDOMLY CHOOSE THE KIND
 OF SENTENCE"
150 A=RND(15):B=RND(5):C=RND(5):
D=RND(5):E=RND(6):NN=NN+1
160 IF E=1 THEN PRINTA$(A)" "B$(
B)" "C$(C)".":X=1
170 IF E=2 THEN PRINTC$(C)" "A$(
A)" "B$(B)".":X=1
180 IF E=3 THEN PRINTA$(A)" "B$(
B)".":X=1
190 IF E=4 THEN PRINT B$(B)" "C$
(C)".":X=-1
200 IF E=5 THEN PRINT B$(B)" "D$
(D)" "C$(C)".":X=-1
210 IF E=6 THEN PRINT A$(B)" "C$
(C)".":X=-1
220 PRINT@157+K,"---";
230 PRINT@125+K,"?";
240 K=K+96
250 EN$=INKEY$
260 IF EN$="" THEN 250
270 PRINTEN$;
280 REM"  DECIDE WHETHER OR NOT
THE CORRECT CHOICE WAS MADE"
290 IF EN$="Y" AND X=1 THEN PRIN
T"      CORRECT":SOUND240,3:CC=CC
+1:GOTO 340
300 IF EN$="Y" AND X=-1 THEN PRI
NT"    SORRY,IT IS NOT A SENTENCE
":SOUND20,5:GOTO 340
310 IF EN$="N" AND X=-1 THEN PRI
NT"    CORRECT":SOUND240,3:CC=CC+
1:GOTO 340
320 IF EN$="N" AND X=1 THEN PRIN
T"    SORRY,IT IS A SENTENCE":SOU
ND20,3:GOTO 340
330 GOTO 250
340 NEXT J
350 PRINT@448,STRING$(32,224);
360 PRINT@480,"    'M' FOR MORE O
R 'E' TO END";
370 EN$=INKEY$
380 IF EN$="M" THEN 40 ELSE IF E
N$="E" THEN 430 ELSE 370
390 DATA HE,SHE,FRED,A BOY,A GIR
L,A DOG,A CAT,ADAM,MARY,SUE,SAL,
JIM,TOM,BOB,LOU
400 DATA RUNS,WALKS,SKIPS,HOPS,E
ATS
410 DATAAT THE STORE,AT NOON,AT
HOME,NEAR MY SCHOOL,IN YOUR CLAS
S
420 DATA QUIETLY,QUICKLY,SOMETIM
ES,OFTEN,NEVER
430 CLS:PRINT"          CHECK-UP T
IME":PRINT
440 PRINT"       YOU TRIED"NN"EXAM
PLES":PRINT
450 PRINT"        YOU DID"CC"CORRE
CTLY":PRINT
460 PRINT@448,"PRESS 'E' TO END
OR 'B' TO BEGIN";
470 EN$=INKEY$
480 IF EN$="B" THEN RUN ELSE IF
EN$="E" THEN END ELSE 470
```

A Question Of Intelligence And Anthropomorphic Charm

By Michael Plog, Ph.D.

Have you named your Color Computer? If you have not actually named your machine, have you assigned a sex to it — do you call your Color Computer a "he" or a "she"? Does your machine have a personality?

Throughout history, we have shown a tendency to assign human characteristics to things which are not human. This is called "anthropomorphism," and can easily be seen in the ways people have related to their pets and deities.

Anthropomorphism is not limited to things that are biological or mythical. We have always named objects that have no life, such as cars, boats, hurricanes, etc. In addition to naming such objects, we also assign personality traits to them. It is as if these inanimate objects have minds of their own. Indeed, that seems to be the key to anthropomorphism — a mind that operates independently of the human wishes.

The things that we personalize tend to be either big, complex or mysterious. They are things that are not easily mastered or understood by people; we have a difficult time with them. Children tend to personalize nonliving objects more than adults, which is consistent with their limited understanding of the world around them.

But there are many adults who personalize things they know very well, possibly from a sense of affection for the inanimate object. You have probably seen grown men speak affectionately of an automobile, and call the car by a pet name. In many cases, these people know the inner workings of the car very well, and have perhaps taken the entire thing apart and put it together again. Anthropomorphism seems to be based not on ignorance, but more on either respect, affection (possibly love?) or fear.

It is easy to personalize computers, especially ones like your small friend the Color Computer. True, it is not big — you can hold the entire thing in one hand under your arm while walking. The computer is, however, complex and mysterious. Sometimes it seems to have a mind of its own (especially when it wins a game).

Is your machine intelligent? Does it have a mind? Can we really assign motivations and mentality to silicon? The answers may not be simple. First of all, we do not know the meaning of intelligence. We have been prepared through movies such as *E.T.* and *Close Encounters of the Third Kind* to accept that things other than humans can have intelligence.

We have always considered other animals to have a sort of intelligence, but only recently realized that animals can use language. (Not an oral language, but the hand signs for the deaf or a keyboard connected to a computer.) Human intelligence may not be the same type of thing as ape or porpoise intelligence; we simply do not know.

In talking about computers, we often use terms which give the machine a type of intelligence. For example, "He got lost when there was no place to RETURN from" or "That subroutine is confusing her when you change the variable."

Computers are often personalized when there is blame to be shared. It is easy to say the computer fouled up a hotel reservation, or accidentally took $1,000 out of your bank account.

Children especially consider the computer to have intelligence, even something approaching a human mind and motivations. Computers can "cheat" or be "friendly." Computers are smart enough to beat you in a game, or tell you the right answer to a math question. For some children, computers are alive — because they know what to do next, and are smart enough to know right answers. As children grow older, their concept of "alive" becomes more mature and restricted to things biological. Yet, the question of life becomes more complicated, as does the question of intelligence.

Many people are fond of an old saying that computers will only do what their program tells them to. The first person to say this was the first "modern" programmer, Lady Ada Lovelace, in 1842 (working on the "analytical engine"). Her statement was: "The analytical engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform." A century and a half later, we are still using Lady Lovelace's thought to maintain the superiority of humans over machines.

That thought may be true, but in one sense it elicits a question. A computer without a program may be only plastic, metal and silicon. With a program, however, is there intelligence operating? Consider the question with a human being — without life, there is no intelligence; we are simply a collection of very complicated proteins and other chemicals. After death, a person cannot be said to be intelligent. With life added (or before taken away), we are an intelligent entity. Perhaps computers work the same way; programs for computers create intelligence which are not more stupid for being less biological. The problem remains that we do not have a good definition of intelligence.

There is a whole file of study around these questions. It is called "AI," or Artificial Intelligence. People working in the field of AI gained a lot of attention about two decades ago, with the perfection of chess programs and using broad pattern recognition instead of linear instructions representing rules. There was then a decline of public interest in AI.

Recently, more attention has been given to AI research. Much of the material is philosophical in nature, asking hard questions and discussing the essence of such things as humanity, intelligence, mind, etc.

Information from AI inquiry is important on a less esoteric level. What we think about computers determines how we react with them. And beyond that, the question of a mind in the machine quickly turns into the human mind being considered as a machine.

People are thinking and speaking of themselves in computer terms. Sometimes we have to "clear our buffers" before entering a conversation filled with emotion. Some people always "fall back" to the "default" position in times of crisis, or when a "decision branch" needs to be made. Have you ever "debugged" a personal relationship? Do you know anyone who "flip-flops" on issues? Of course, you know a few people who should be "re-programmed," hopefully not "deleted."

The language of computers has become incorporated into our general speech and thought patterns. Thus, our language carries with it an assumption that what takes place in a machine is the same type of thing that takes place in a human mind:

Words determine the way we think about ourselves and the world around us. Computer jargon has not only introduced new words into our vocabulary, it has also introduced new ways of reacting to our environment.

For many of us who are older, the problem may not be too great. Our minds have already become accustomed to our interaction with our environment. For young school children, the way they relate to machines may well determine how they relate to other people, and even themselves.

It is difficult, maybe even impossible, to predict the different ways these relationships might be expressed. Possibly the best of all outcomes will be a society with computers and people working together to solve problems. Does it really matter if computers have intelligence, as long as we use that intelligence to help us to be more understanding, more tolerant and less ready to harm our neighbors?

Learning about the extension of mind and the relationship of mind and machine is not just a job for kids in school; all of us need to keep learning, to keep questioning. Education may well be the process of questioning, not necessarily the answers we get. Let's keep it up. Have you named your Color Computer? Does your machine have a personality?

If you have thoughts about anything mentioned here, please share them with me. My address is 829 Evergreen, Chatham, IL 62629.

Expanding The Computer's Role In Student Testing

By Michael Plog, Ph.D.

In the spring of each year, students all over the country take achievement tests. Parents are called into school to see the results of these tests, and to understand how their children performed on instruments that might influence future school experiences and careers.

Tests are generally given for three major purposes. The first is for assessment — to get an idea of how well students know a subject matter, such as social studies, science, etc. Another major purpose is for screening, to determine what additional information is needed before making a decision about the student. A third purpose of testing is for diagnosis, to identify specific strengths or weaknesses of students and plan an educational experience accordingly.

There are three types of achievement tests, each measuring achievement in different ways. First, there are criterion-referenced tests. Each item (question) on a criterion-referenced test is related to at least one specific objective of a curriculum (for example, learning to add two-digit numbers).

There is a standard, or criterion, for determining successful performance on the test. If a student gets 80 percent of the items correct, that student has mastered the objective being measured and is ready to go on to a higher objective (possibly learning to multiply two-digit numbers).

Criterion-referenced tests are especially useful to classroom teachers for planning instruction for students. The information from the test concerns student mastery, not how well the student performed in relationship to other students.

The second type of achievement tests are classified as objective-referenced tests. They are similar to criterion-referenced tests in that each test item is associated with a specific objective of the curriculum. Objective-referenced tests, however, are not used with predetermined standards or criteria for successful performance. These tests are especially useful for determining student placement. Also, objeetive-referenced tests can be used to obtain an overall perspective of student growth for a classroom or a district.

The third type of achievement test is called norm-referenced. These tests compare performance of an individual student (or even a large group of students) with the performance of a "norming" group for the tests. Generally, the norming group should be students across the entire country. Items on the test may be grouped into subtests (i.e., mathematics, science, etc.), but are not related to specific objectives of a curriculum.

Norm-referenced tests are used to compare an individual student with the "typical" student taking the test. They can be used for screening purposes by identifying students in relationship to other students taking the test. Thus, those students who score either very high or very low on the test may be eligible for different types of programs.

So far we have used the term "referenced" while talking about tests. This term merely means what the test is related to — a specific criterion, an objective or a norming population.

Most tests given in schools are prepared by teachers. These are given frequently during the year and are normally used to help the teacher assign grades or determine what students need what kind of help.

Commercial tests are also used by schools for many purposes. These tests are purchased from test companies, and can be either criterion-, objective- or norm-referenced. Most commercial tests are "standarized," meaning the administration and scoring procedures are standard for all students taking the test, no matter where in the country that student is.

Commercial tests can be valuable for school personnel, but must be used with caution. These tests have been designed for specific purposes, and are not appropriate in some situations. All such tests come with manuals which explain the correct and incorrect purposes of the test.

It is fair to say that the widespread use of commercial (and especially standardized) tests is a result of computers. Without the number crunching ability of computers, most commercial achievement tests would never have been developed.

The role of the computer in testing is expanding. Part of this expansion is due to the microcomputer. We no longer need large mainframes to deal with achievement tests. In general, there are three roles the computer plays in testing: scoring, analysis and administration.

Most commercial tests given today use "mark sensitive" scoring answer sheets. Students mark their answers on a special sheet containing "bubbles." The sheets are then fed into machines that "read" which bubbles were marked by the students. Student responses go directly into the memory of the machine and then to permanent storage, such as a diskette or tape.

It is possible today for a school system to purchase a test-reading machine to connect with the Color Computer, or almost any microcomputer. Some districts even encourage their teachers to administer classroom tests using these answer sheets and score them on the microcomputer located in the school building. The whole set of hardware (computer, disk, reading machine and printer) can be put on a cart and transported to classrooms. Students have results of their tests before a class period is over; they can no longer count on parents forgetting they took an exam before the scores are ready.

Computers are a "natural" for analysis of test results. Norms for tests can easily be obtained in a short time for a classroom, building, entire district or even several districts giving the same test. Graphics output of individual student results, or groups of students, can be easily ordered and examined. When giving a criterion-referenced test, it is a simple matter to determine which students have mastered specific objectives at a glance.

Possibly the greatest expansion of the use of computers in testing is with the administration of tests. Instead of having a teacher stop all classroom work and give a test, students can now be assigned to go to a microcomputer, take a test and return to class. Questions appear on the screen and students type their choice of answers on the keyboard. A permanent record of each answer can be saved on diskette, and the student can be given the score for the test immediately after responding to the last question.

But, using computers for administration of tests can go beyond this simple example. Consider administration of a criterion-referenced test. The purpose of the test is to determine student mastery of a given body of content.. Instead of determining mastery by the answers to a few (generally less than five) test items, it is possible to use many test items to determine the developmental point of an individual student.

A computer program (and diskette) can contain over a thousand questions. The computer program begins with a few key items, then determines which items should be asked next, based on student responses. If the student gets three consecutive questions wrong, the computer goes to simpler questions. If the student gets three consecutive questions correct, the computer skips the next few questions and presents harder items.

After going through several groups of items, the computer can determine, with as much accuracy as the questions allow, the level of knowledge for each specific student. It becomes much easier for the teacher to then determine what lessons should come next for that special student.

This type of testing (and curriculum) depends on a lot of advance work to determine a sequence of knowledge in a subject matter. The authors of the computer program must know — in painful detail — the sequence of the educational program.

There is one other example of creative use of a microcomputer for administration of tests. Science teachers have stressed the importance of learning the process of science. Our test instruments, however, have stressed the product of learning. Based on student responses to most science tests, it is impossible to determine how a student arrived at that answer and what information a student used to make scientific inferences.

An innovative microcomputer program has been written to attempt to examine the process a student uses to arrive at answers for a science test. The program presents the student with a series of simulations. (We will use a plant growth simulation for illustration.)

The student is required to make observations, organize data, develop and test hypotheses, make predictions based on data, make inferences, and identify a scientific model that accounts for most of the data presented. In essence, the student must know the scientific method in order to progress through the simulation appropriately.

The screen shows graphics of a plant growing under conditions selected by the student. Also, the screen shows graphs and tables of information based on the selection of conditions by the student.

The microcomputer program records each step the student takes. Results of the test can provide information about steps students use to arrive at a conclusion. Statements about science education can be made that were never possible before. This knowledge would not be possible without the innovative use of the microcomputer.

This science simulation is being used as a pilot test this year in a statewide assessment program. We do not know the results of the assessment at the time of this writing; the pilot effort may not be worthwhile. The knowledge that is learned from the effort, however, cannot help but provide one more step forward in education. And, you can never take the second step until the first step is history.

I welcome your comments and thoughts about this most exciting activity of using computers in education. My address is 829 Evergreen, Chatham, IL 62629.

SIMULATION CONTEST WINNER

32K
ECB

SURFACE!

Program By Christopher Pfeifer

It is August 1958. The USS Skate has left its moorings in New London, Conn., glided quietly down the moonlit Thames River, passed Long Island Sound and then heads north. As the continental shelf steadily drops away, the blasts of the diving alarm siginal that it is time to go below.

When the Nautilus made headlines for the being the first submarine to pass under the North Pole, the USS Skate and her crew of 97 men and nine civilian specialists were only four days behind. But Captain Calvert had been handed a far more dangerous mission - the Skate must try to surface at the Pole - thus providing the usefulness of the Arctic Ocean as an operational area.

Surface is a program designed to stimulate the historic Arctic voyage of the nuclear submarine Skate as seen from its control room. It can be made to run from disk or cassette. When the program is run, an ML color test routine is called, making the screen either red or blue. Press the Reset button until it is red and then hit a key to start the voyage. The first thing Surface will do is PCLEAR8 to make room for the two high resolution screen dumps that are created by the DRAWINGS program listed below (it checks to see if you have a disk controller plugged in - if so, the screen dumps will load at address $E00 from disk as LOADM''CHARTS/ART'' - if not, then they will load at address $600 from cassette as CLOADM''CHARTS'').

After the screens are loaded, you will see a map of the Arctic Ocean. This map is on the navigation chart table and is continually updated. Look between Greenland and Spitsbergen for a tiny flashing dot. That dot represents the Skate's current position, after its first attempt at surfacing dot enters the small circle at the center of the chart, the Skate will be at the North Pole: The only place on earth where every direction is south.

The numbers around the map represent compass headings in degrees. Starting from the center of the map, straight up equals zero, straight down equals 180, left equals 270, and right equals 90. You may find this useful when changing the ship's bearing.

The chart is the first of four screens that comprise the activities of the control room. You can alternate between three of these screens by pressing the space bar. The fourth screen is a work area and is not displayed unless the periscope is raised. The second screen is the ship's ice scanner (if you see a text screen instead, press the space bar again).

The original ice scanner was a remarkable device. It was a grey metal box with a glass window, behind which a sensitive stylus traced a profile of the ice above on a slowly rolling paper tape. A straight line meant no ice (or very thin ice) above. Your scanner represents open water as a single row of dots.

The tiny submarine on the scanner indicates the position and depth of the Skate in relation to the ice. The scale on the right-hand side of the screen shows depth in 10-fathom increments (a fathom is six feet, so 10 fathoms equals 60 feet).

Press the space bar and move on to the third screen the text screen. Along the top are five boxes showing readings from the various instruments on board. Press the '0' key to see the menu. At the upper left-hand corner is the trim control. Below the word "TRIM" are the letters 'R' and 'P.' The number after the 'R' is the angle of the rudder, and it varies from -45 degrees (hard-a'port) to zero degrees (rudder-a'midships) to +45 degrees (hard-a'starboard). The number after the 'P' gives the angle of the diving planes ("30 degree down-bubble," etc.).

A submarine normally uses forward motion to control its depth, and the diving planes, acting like the elevators on an airplane, will pull the vessel up or push the vessel down depending on the angle at which they are set. If the submarine is not moving, however, the diving planes will have no effect on changing its depth.

The next box is labelled "BEARG," and the number given is the direction the ship's bow is pointing, in degrees. Thanks to the internal navigation set recently installed, you will not have to surface to get your bearings. You can change direction by pressing either '1' for right rudder, or '2' for left rudder. As with the diving planes, the ship must be moving for the rudder to be of any use.

The third box, "BLLST," tells you how full the ballast tanks are. The figure given is in percent of the total tank capacity. Therefore, a reading of zero percent means the ballast tanks are empty; 50 percent means the tanks are half full; 100 percent means the tanks are completely flooded. When the tanks are half full (50 percent) the submarine has neutral buoyancy (it won't rise or sink), but if the figure is less than 50 the submarine has negative buoyancy (it will rise toward the surface), and if the figure is greater than 50 the submarine has positive buoyancy (it will sink). The ship does not have to be moving for the ballast control to change its depth.

The fourth box isn't labelled, and gives three readings. The top figure is the depth of the submarine in fathoms. If it reads 'S,' then the ship is at the surface. A submarine measures its depth from the surface to its keel but, when travelling under the ice, it is more important to know the depth from the surface to its sail. The middle figure will explain why.

It is the reading from the ice scanner which tells you how deep the ice is, in fathoms, directly above the sail. Therefore, if the Skate's depth is 10 fathoms and the ice above is 10 fathoms, it means that the top of the submarine is touching the bottom of the ice.

The topside fathometer is a sonar device located on the sail, and it will beep whenever there is ice above it. The closer you are to the ice, the faster it will beep.

The bottom figure in the box is your latitude in hours and minutes, as given by the internal navigation set. Latitude is how far you are from the equator in degrees. If you sliced the world in half (in hemispheres) along the equator, the edge of that slice would be zero degrees latitude. Going from the center of the earth straight up to the Pole would be an angle of 90 degrees, the maximum latitude possible. Halfway from the equator to the Pole is 45 degrees. You will know when you are at the North Pole when your latitude reading is 90:00 (Ninety hours, zero minutes).

The last box, marked "SPEED," is fairly self-explanatory. It gives the Skate's velocity in knots. Full-speed ahead is 20 knots; half-speed is 10; all-stop is zero; full-a'stern (reverse) is 20 knots. Pressing '7' increases the speed and pressing '8' decreases it.

The menu below the boxes lists the commands which you will use to accomplish your mission. Pressing '9' raises the Skate's periscope. If the ship's depth is greater than 10 fathoms (60 feet), the periscope will not reach the surface and all you will see is water. Before raising the periscope, it is important to check the ice depth reading for ice above. If the ice depth reads zero, then the periscope is below open water and you may raise it safely.

When you are about to reach the Pole, slow down. As soon as you see "90:00" on the latitude indicator, come to a complete stop. Check the ice scanner for openings large enough to accommodate the submarine (five single dots or more in a row will give you enough open water).

If you see none, then before moving on again, turn the rudder over hard. The ship will now travel in a tight circle and soon return to (almost) the same spot. Upon your return, you will see that the ice cover over the Pole has changed because of drift. The crew of the Skate used a method of criss-crossing the Pole, waiting for drift to carry a decent-sized opening over the area, and it involved a lot of changes in direction. Some may prefer circling, but you may use whatever method you like. When you have succeeded at surfacing at exactly 90:00 (the depth indicator must read 'S' for "surfaced"), raise the periscope to send a message home.

Practice diving and surfacing while on your way to the Pole, so you won't make any mistakes when it counts. There is no SAVE feature to this Simulation — if you damage the Skate, your mission is scrubbed, and the Simulation is over.

You will also note that pressing the BREAK key doesn't stop the program. The program runs faster this way, and it would be a shame to accidentally touch the BREAK key and ruin your whole voyage.

There are several machine language routines in the program, such as the one that draws the ice on the ice scanner, and we must caution you to SAVE the progam first before running it or you may lose it completely!

It took the original Skate 13 days to journey from New London to the North Pole. This Simulation begins as the Skate encounters the ice pack. Traveling at 20 knots, at a safe depth of 60 fathoms, you should reach the Pole in 30 minutes. But bear this in mind: Getting to the Pole is easy — your orders are to SURFACE!

— Charles Springer

9	126	37	185
14	249	59	79
19	209	66	158
24	218	77	147
28	22	END	118
33	240		

Listing 1:

```
0 'CHARTS FOR U.S.S. SKATE BY
   CHRISTOPHER PFEIFER
   1348 N. DALE ST.
   ST. PAUL, MN  55117
1 GOTO3
2 GOTO5
3 PCLEAR8:GOTO2
4 GOTO2
5 IFPEEK(&HC000)=68 THEND=&H800
6 A=&H600+D:Z=A:CLS:PRINT:PRINT:
PRINT:PRINT"THIS PROGRAM WILL DR
AW THE TWO  HI-RES SCREENS NEEDE
D FOR THE    'SURFACE!' SIMULATIO
N. IT WILL  SAVE THE SCREENS AS
";:IFD=0THENPRINT"'CHARTS' ON YO
UR CASSETTE"ELSEPRINT"'CHARTS/AR
T'ON YOUR DISK"
7 FORR=1TO5000:NEXT:PMODE4,1:PCL
S:SCREEN1,1
8 B$="AA00A2XX03B2A200AA00A888A8
XX028000AA008A888A808A00AAXXA700
AA00XX0222A2XX022200AA00A282A2XX
028200AA008A888A888A00AAXX34002A
022A202AXX1C002A022A202AXX4D00AA
00A222A2822200AA0088088808880 0AA
00":GOSUB41
9 B$="XX0288BAXX028000AAXX100002
XX030002XX1F002A222A222AXX1C00XX
02202A222AXX1C000200XX0302XX2C00
AA002A202A022A00AA00A282A2XX0282
00AAXX0200XX0480008 0XX1000A222A2
22A2XX1F002AXX03222AXX1C002AXX03
222AXX1C00":GOSUB41
10 B$="A222A200A0XX2C00AA002AXX0
40800AA008A088A0A8800AAXX1800A2X
X0322A2XX1300XX050102XX020700050
9081FXX021E0200XX0A15XX0955XX0A1
51005XX0315XX0605XX0601XX0F00022
2A2XX0222XX2C00AA002AXX03222A00A
A00":GOSUB41
11 B$="XX038000800080XX1800A0XX0
```

```
320A000XX02010001XX0405XX0415545
1574F4E4D495145XX035515E07F3874F
484XX035515405552XX025150XX0451X
X0255XX054555XX0851XX02554015XX0
255XX0454XX1055XX0415XX040501A02
1XX0220A0":GOSUB41
12 B$="XX18000200XX0302XX0F00AA0
02A222AXX022000AAXX0600A8082808A
8XX1000XX0301050415XX0214XX02545
5XX03541543135C4F4E564D1DXX02FBX
X02F7XX026F6C0015XX021415145505F
0FF07505553XX02574F813EFCE115XX0
35554":GOSUB41
13 B$="XX10555401XX0C55XX0215XX0
355XX0345XX0255XX0251XX0255XX025
4555451451 5XX0655XX0315XX0205XX0
301XX0F00A220A202A2XX0F00AA00XX0
2222AXX022200AAXX0600A808A880A8X
X080001XX0203051413 47XX024F57XX0
24F3F7F":GOSUB41
14 B$="XX02FFXX02FEFDFBXX02F7EFX
X02DFBF3FDFE7F9FEXX08FF7F3F5F4B4
70FXX023F1FXX08FF07F83FXX025F390
5XX025554XX0D555005XX05555455434
F535554XX0255XX02535554XX0355535
550XX0254514555XX0215XX025545XX0
2555155":GOSUB41
15 B$="XX0254XX0A55XX0215XX0205X
X0201XX0700A222A202A2XX0F00AA002
82028202800AAXX0600A8XX0388A8000
10501XX020056571 3D7E7F3XX03FFFEF
DFBF7EFXX02DFBF7FXX05FFCF9753511
015E5XX02F5F139C1F1F9FCFDFEXX13F
F1FE0XX06FF":GOSUB41
16 B$="1F5FXX024F5F4F57XX034FXX0
253554015XX065515D5E5F5F8XX0AFFF
EFDF3CFXX023FXX035F174B574F5F47X
X0257XX024F5F380555XX02455551555
4XX085554XX021501050 1A0XX0320A0X
X0F00A800A888A8A08800A8XX0600XX0
2010515":GOSUB41
17 B$="XX065511C67FBFDEDDE3E7DB3
DXX02FEXX11FF7FXX063F5FXX024F174
B5390D1XX04D3C3XX02E7XX0AFF7F81F
EXX0FFFFE813F4757471F7F1F4FXX045
70FXX06FFFCF3EF9F7FXX14FF7FXX023
F47575352501551454155541 5XX09551
505":GOSUB41
18 B$="XX0201XX0900A808A880XX02A
8082808A8XX0600010515XX065550574
F4E5553479F3FXX08FF7FBFDFXX02EFF
7FBFDXX02FEXX15FF3FDFE7F9FEXX08F
FXX02FEXX03FDXX02FB0BF3XX04FBXX0
4F7XX05FBF30BFBXX03FDXX02FEXX09F
FFEF9E7DF3F":GOSUB41
19 B$="XX15FFXX02FEFDFBF7XX02EFD
38415XX0255XX0254XX0455155545515
554XX0955150501XX0500A8XX0388A88
088A8XX0208XX040015XX09555451479
E3DA19FBFXX14FF7FBFDFEFXX02F7FBF
DFEXX0BFFXX02FEFDXX02FBF72FCFXX0
2DF":GOSUB41
20 B$="XX02BFXX027FXX1EFFXX027FX
X02BFXX02DFEFCF37F7XX02FBFDFEXX0
CFFFEFDFBXX02F7EFDFBF7FXX09FF7F3
F7FF34313XX0353XX02575354XX02551
5455154XX095515XX0300A8XX0388XX0
2A8XX0388A80004545 5XX03531713XX0
25346414F2F":GOSUB41
21 B$="3777XX03FBXX03FDXX02FEXX1
AFFXX027FBFDFEFF4F3E7DFBF7FXX3AF
FXX027FBFDFE7F3F4EFDFBFXX027FXX1
AFFXX02FEXX03FD0B4B53XX0247XX020
F61065055XX0215XX05550501XX07000
73FXX09FFFCC33FXX0DFFXX037FXX03B
FXX03DF":GOSUB41
22 B$="XX02EFXX03F7F8XX03F90102X
X02FEXX02FFFEFDE2E8E5917FXX47FF7
F9FE7F9FEXX04FFXX03FEXX02FDXX03F
BXX03F7XX02EFXX03DFXX03BFXX037FX
X0CFFF134C1545515XX07551505XX040
01FXX0AFFF00FXX1DFFXX033F1F5F4FX
X025753":GOSUB41
23 B$="504195433FXX027FXX4CFF3F9
F637C7FXX22FFF17479097079752 5XX0
75515XX02000FXX0AFFF00FXX26FFFC0
3XX28FFFCFDF9FDFCXX28FF03FCXX23F
F7F3F5F0F5057535554XX0355XX02545
5XX020078XX0A7F0F70XX267F3FC0XX2
7FF7F1F":GOSUB41
24 B$="DFCFDF1F7FXX27FFC03FXX267
F700FXX037F3F5F4FXX02575355XX030
0FCXX0AFF0FF0XX26FF3FC0XX02F8859
55087FFFCF0FFF8XX02F9FCXX43FFFEF
9C73FXX26FFF00FXX0AFFFCXX0400F0F
EXX09FF3FC3FCXX10FFXX02FEFCXX03F
DF9C1D5":GOSUB41
25 B$="D4D1C7EEEDDDD9XX02B985XX0
25515XX025515059154XX025515E505X
X025515E1FDFEXX3CFFFEF9E79FXX047
FXX02BFXX03DFXX03EFXX02F7XX03FBX
X02FDXX03FEXX12FFFCC33FFFFCF1E5C
5F8XX04FEF0XX02000200XX0302C0F8X
X05FF":GOSUB41
26 B$="XX02FEXX02FD3DCBF0FAXX02F
7XX02EFECDDDEDFBFBE60XX026515XX1
8555455511545555154XX065515E5F1F
9F8XX03FFXX03FEXX03FDFEXX28FFFEF
9F7E7977BFDFEXX17FFXX037FXX02BFX
X03DFEFEEF6F5XX03F1F0F1C524E4C41
454XX0555":GOSUB41

27 B$="5040XX0400A222A202A2XX030
0A07CXX027FXX04FF994255154591FCX
X02FD7185XX15555455XX02515545551
5XX0B55XX021555XX024555505155 11E
6FEFF3F4750555154XX0355945414E4X
X02F5F8XX11FFXX02FEXX02FDFBF9F6E
7EFCFDF":GOSUB41
28 B$="BFXX027FXX08FFXX027FBFDFE
FXX02F7FBFDFEXX15FF3F50455554514
515XX0755XX02151440XX0200XX0522A
```

```
0XX0320A0XX0600C0F0FCXX02FF7C1D4
1XX04551555455 15554XX09555455XX0
251554555XX0215XX15555455514515X
X025551":GOSUB41
29 B$="097DFEXX027FXX02BFDFD1D82
D4D482DCDE8EDXX0267E3E4E00B4B86C
CEFE7E0EFDFDCDABCBFXX027FXX02FFC
1A5C5D97FBFCFF2XX02F8FDXX02FCXX1
1FF7FXX02BFDFEFF7XX02FBFDFEXX09F
FFEE511XX024515XX09555450XX0600X
X0202A2":GOSUB41
30 B$="22A2XX0F0080005054XX09555
414554151 4554XX0215XX1555545553X
X03555450441 5XX0555414EXX024F5FX
X033F7FXX03FF134A401253XX0217C7X
X02FFXX02FE7DB87F7377XX02FF3FC0F
177FFFCFBXX05FFFAF4FCF8F9FDFEXX0
27F1F43":GOSUB41
31 B$="931CF8XX03F9FDFEXX14FF7FX
X02BFDEEDF3E3DDA01454XX085554504
0XX0A00A0XX0320A0XX0F00XX02080AX
X0300405040541415XX0855155545555
1XX025554XX0555XX0454504F5FXX023
F7FFFFEF9E7DF3FXX03FF7FXX023F5F4
F1F":GOSUB41
32 B$="XX037FXX06FFFEXX02FF1C03X
X0357XX03D7970717475F3FFEE3CBF7F
F3F7F03FC1F3FXX04FFAFCFDF5F1FXX0
2CFEFCFXX03FFE7D3131555455154XX0
25397C6FEXX0BFFFEFDXX02FBF0E9DD3
DXX02FDF98115XX0355151444405040X
X23000A":GOSUB41
33 B$="XX03888AXX0800XX0240XX025
0XX0254XX0455XX0354534F5FXX023F1
F2977C7FBXX02FDXX02FEFDF3CFBF7FX
X12FFFEF8F97949518009FCXX02FFFEF
CXX02FFFEXX02FFBF5F2F07FBXX05FFC
F00E0C08FXX07FFF1FAFDFEFFF8XX02F
5E5D5XX0295":GOSUB41
34 B$="XX02E5F56494855355 4E5EBDF
DFBF4F1E9D9BDB105XX0295D515XX055
5XX0254XX0250XX0240XX0600XX0322X
X0220XX1F00XX0580XX1000XX024000F
0F8XX02FCFEXX06FFFCFBE79F7FXX02B
FXX02DFXX02EFXX02F7XX03FBFCFDFCF
0F4":GOSUB41
35 B$="XX02E5XX02D595XX075540155
55414E4XX0215055550 8FXX05FFFEF1C
5E5F0XX02FF1FDF00F6XX03FFFCFDF9F
5C515XX0255XX0254XX0355514101257
469EFDFDDB8B54595B59154XX0655XX0
354XX0250XX0340XX0E000222A2XX022
2XX3900":GOSUB41
36 B$="XX02080A080A80XX02C020XX0
4E0D0XX0294XX02D4E5XX03F9XX03F50
5XX065551101 3030FXX035F1C035E0E2
EECEEXX046F77XX05F7F67105484F2FX
X04EFXX02DF03D8XX02DFXX023F1F237
74F5F5E5DXX03595D5E5F5E3EFEF9E5D
4XX029454":GOSUB41
37 B$="XX0450400242XX0302XX1700A
0XX0320A0XX39000A08XX02888AXX130
0XX0640XX035010E0F8XX03FCF00CFC0
0XX045414C4F4F8F9F5E5XX0295D515X
X02555494C4F8FAF0E4XX02D4E404E0E
4D4E4E0XX02D090XX0250XX064000XX0
280XX0C00":GOSUB41
38 B$="2A022A202AXX5500XX0580XX1
F002A222A222AXX1C00XX0522XX1C002
AXX03222AXX79002AXX03222AXX1C00A
2XX0322A2XXBB00A0XX0320A0":GOSUB
41
39 GOTO54
40 'poke routine number one
41 FORR=1TOLEN(B$)STEP2
42 A$=MID$(B$,R,2)
43 IF A$<>"XX" THEND$="01":C$="&
H"+A$:GOTO48
44 R=R+2
45 D$="&H"+MID$(B$,R,2)
46 R=R+2
47 C$="&H"+MID$(B$,R,2)
48 FORF=1 TO VAL(D$)
49 POKE A,VAL(C$)
50 A=A+32:IF A>&H1DFF+D THENZ=Z+
1:A=Z
51 NEXTF
52 NEXTR
53 RETURN
54 'submarine
55 PMODE4,5:PCLS:SCREEN1,1
56 A=&H1C9C+D
57 F=0:Y=4:Z=&H1C
58 B$="03E000000320000003A00006F
F3E800FFFFEC00F8003C006FFFF":GOS
UB83
59 'v-scale
60 A=&H1F20+D
61 Y=3:Z=&H1D
62 B$="FFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFEFCFFFCFB7FFEFB7FFEFB7FFC
7CFFFFFFFFFFFFFFFFFFF000000FFFFF
FFFFFFFFFFFFFFFCFCFFFB7B7FFEFB7F
FDFB7FF87CFFFFFFFFFFFFFFFFFFF000
000FFFFFFFFFFFFFFFFFFFFCFCFFFB7B
7FFEFB7":GOSUB 83
63 B$="FFB7B7FFCFCFFFFFFFFFFFFFF
FFFFF000000FFFFFFFFFFFFFFFFFFFFE
7CFFFD7B7FF87B7FFF7B7FFF7CFFFFFF
FFFFFFFFFFFFFF000000FFFFFFFFFFFF
FFFFFFF87CFFF9FB7FFEFB7FFB7B7FFC
FCFFFFFFFFFFFFFFFFFFF000000FFFFF
FFFFFFF":GOSUB 83
64 B$="FFFFFFFFCFCFFFBFB7FF8FB7F
FB7B7FFCFCFFFFFFFFFFFFFFFFFFF000
000FFFFFFFFFFFFFFFFFFFF87CFFFF7B
7FFF7B7FFEFB7FFDFCFFFFFFFFFFFFFF
FFFFF000000FFFFFFFFFFFFFFFFFFFFC
FCFFFB7B7FFCFB7FFB7B7FFCFCFFFFFF
FFFFFFF":GOSUB 83
```

```
65 B$="FFFFFF000000FFFFFFFFFFFF
FFFFFFFCFCFFFB7B7FFD7B7FFF7B7FFC
FCFFFFFFFFFFFFFFFFFFFF000000FFFFF
FFFFFFFFFFFFFEFCFCFCFB7B7EFB7B7E
FB7B7C7CFCFFFFFFFFFFFFFFFFFFFF000
000FFFFFFFFFFFFFFFFFFEFEFCFCFCFB
7EFEFB7":GOSUB 83
66 B$="EFEFB7C7C7CFFFFFFFFFFFFF
FFFFF000000FFFFFFFFFFFFFFFFFFFEFC
FCFCFB7B7EFEFB7EFDFB7C787CFFFFFF
FFFFFFFFFFFFF000000FFFFFFFFFFFF
FFFFFEFCFCFCFB7B7EFEFB7EFB7B7C7C
FCFFFFFFFFFFFFFFFFFFFF000000FFFFF
FFFFFFF":GOSUB 83
67 B$="FFFFFFEFE7CFCFD7B7EF87B7E
FF7B7C7F7CFFFFFFFFFFFFFFFFFFFFFF
FFF":GOSUB83
68 'h-scale
69 A=&H1E00+D
70 Y=0:Z=0
71 B$="FFFFFEFFFFFEFFFFFEFFFFFE
FFFFEFFFFFEFFFFFEFFFFFEFFFFFFFF
FFFFFFFFFCFCFEFEFCFEFCFCFEFEFCFE
FCFCFEFCFCFEFE7CFEF87CFEFCFCFFFC
7CFCFB7FFB7B7EFCFB7EFB7B7EFCFB7E
FB7B7EFB7B7EFD7B7EF9FB7EFBFB7FFB
FB7B797FFEF":GOSUB83
72 B$="B7EFEFB7EFB7B7EFEFB7EFEFB
7EFEFB7EF87B7EFEFB7EF8FB7FFCFBFB
7A7FFDFB7EFEFB7EFB7B7EFEFB7EFDFB
7EFB7B7EFF7B7EFB7B7EFB7B7FFF7B7B
7B7FF87CFEFC7CFEFCFCFEFC7CFEF87C
FEFCFCFEFF7CFEFCFCFEFCFCFFF8FCFB
7B7FFFFFFEF":GOSUB83
73 B$="FFFEFFFFFEFFFFFEFFFFFEFF
FFFEFFFFFEFFFFFEFFFFFFFFFFFFFFFF
FFFFFEFFFFFEFFFFFEFFFFFEFFFFFEFF
FFFEFFFFFEFFFFFEFFFFFFFFFFFFFFF"
:GOSUB83
74 'uss nameplate
75 A=&H3480+D
76 Y=0:Z=0
77 B$="FFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFF5555555555555555555555555
55555555555555555555555555555555
55555555555555555555555555555555
55555555555555555555555555555555
5555555555":GOSUB83
78 B$="5577557D557D55557D775D7F7
F555557F7D7F557D7D5D77777F7D555
555555557755775577555577777D5D7
5555555D7775557777D77777567555
555557D5577557555755555757D675D7
5557D555D757555757567 7F7F7567557
D55557D5577":GOSUB83
79 B$="557D557D55557D7D675D7D557
D555D757D557D75677F7F7D7D557D555
55555775555755575555577D7F5D75555
5555D757555577757F7777757D5555555
5555577757775777555777775D75555
5555D77755577777777775775555555
555557D757D":GOSUB83
80 B$="757D75557D77775D7F5555557
F7D7F557D7D7777777F77555555555555
55555555555555555555555555555555
55555555555555555555555555555555
55555555555555555555555555555555
55555555555555555555555555500":G
OSUB83
81 CLS:SOUND50,4:IFD=&H800 THEN8
2ELSEPRINT@200,"READY CASSETTE";
:INPUTM:CSAVEM"CHARTS",&H600,&H3
5FF,0:END
82 PRINT@200,"READY DISK";:INPUT
M:SAVEM"CHARTS/ART",&HE00,&H3DFF
,0:END
83 'poke routine number two
84 FORR=1TOLEN(B$)STEP2
85 F=F+1:IF F=Y THEN A=A+Z:F=0
86 A$="&H"+MID$(B$,R,2)
87 POKEA,VAL(A$):A=A+1:NEXTR
88 RETURN
```

16	57	123	138
39	38	130	89
57	191	152	45
75	245	161	185
100	255	END	89

Listing 2:

```
0 'SURFACE!' * 1984
  BY CHRISTOPHER PFEIFER
  1348 N. DALE ST.
  ST. PAUL, MN 55117
1 '*****************************
2 'NOTE 1: SAVE THIS PROGRAM
   BEFORE RUNNING IT!
   NOTE 2: THE CASSETTE VERSION
   SHOULD HAVE "CHARTS" (MADE BY
   THE "DRAWINGS" PROGRAM) AFTER
   THIS ONE ON TAPE
3  *****************************
4 CLEAR200,&H6000
5 IFPEEK(&HC000)=&H44 THENDK=&H8
00
6 IFPEEK(&H72)=&H60 THEN12
7 GOSUB164
8 GOSUB183
9 EXEC&H6070
10 GOTO14
11 'pclear
12 PCLEAR8:GOTO10
13 GOTO10
14 IFPEEK(&HC000)=&H44 THENDK=&H
800
15 IFPEEK(&H145E+DK)=&HA0 THEN17
16 IFDK=&H800 THENLOADM"CHARTS/A
RT" ELSE CLOADM"CHARTS"
17 'get sub
18 DIMQQ(5)
```

```
19 PMODE4,1:GET(220,179)-(242,18
7),QQ,G
20 POKE&H6097,&H12:POKE&H73,&H97
 'revise reset
21 POKE&H167,&H39 'disable texts
creen after print
22 GOSUB173 'draw ice routine
23 GOSUB159 'break disable
24 DR=220:D=13:IT=4:SR=50:GB=RND
(300)+400
25 PX=138:PY=40.9:K=30 'start lo
cation
26 GOSUB151
27 FORR=&H6000 TO &H6038:POKER,R
ND(IT)+D-1:NEXT:FORR=&H6020 TO &
H6030:POKER,D:NEXT
28 PMODE4,5
29 EXEC&H60E0 'erase ice
30 KL=(DP*1.155)+17
31 IF KL=>179 THEN KL=179
32 IFDP<147THENPUT(52,(DP*1.155)
+9)-(74,KL),QQ,PSET 'put sub
33 EXEC&H6040:IFSP=0ORPS>0THENPS
=PS-1:GOTO54ELSEPS=5-(ABS(SP)/4)
34 H=RND(IT)-1
35 PX=PX+((FX/100)*SP)/500
36 PY=PY+((FY/100)*SP)/500
37 IFPX<128THENKX=(PX/3)+9ELSEKX
=((256-PX)/3)+9
38 IFPY<96THENKY=((PY*2)/3)-9ELS
EKY=(((192-PY)*2)/3)-9
39 K=((KX+KY)/2)-3.3:IFK<0THENK=
0ELSEIFK>50THENK=50
40 ZZ=PEEK(&H603F)
41 IFGG=1THEN44
42 IF ZZ<>&H38 THEND=PEEK(ZZ+&H6
001) ELSED=PEEK(&H6000)
43 GOTO45
44 IF ZZ<1 THEND=PEEK(ZZ+&H5FFF)
ELSED=PEEK(&H6038)
45 IFRND(110)>K THEN47
46 IFD<170THEND=D+H:GOTO48
47 D=D-H
48 IF D<13 THEN D=13
49 IFGG=0THENPOKE(ZZ+&H6000),D E
LSE POKE(ZZ+&H5FFF),D
50 IFGG=1THEN53
51 IFZZ=0THENPOKE&H603F,&H38 ELS
E POKE&H603F,ZZ-1
52 GOTO54
53 IF ZZ=&H38 THENPOKE&H603F,0 E
LSE POKE&H603F,ZZ+1
54 PMODE3,1:LD=PPOINT(PX,PY):DD=
PPOINT(PX+1,PY):DL=PPOINT(PX-1,P
Y):PSET(PX,PY,1):IFLD=6ANDDL=6AN
DDD=6THENPSET(PX,PY,LD):GB=4:GOT
O133 ELSE GOSUB178:PSET(PX,PY,LD
)
55 PMODE4,5:EXEC&H60B0
56 IFDP>5THENIFPPOINT(64,KL-8)=5
THENGB=3:GOTO133
57 IFDP>2THENIFPPOINT(56,KL-5)=5
ORPPOINT(60,KL-5)=5ORPPOINT(68,K
L-5)=5ORPPOINT(72,KL-5)=5THENGB=
3:GOTO133
58 'get key
59 A$=INKEY$:IFA$=" "THEN86ELSEI
FA$="0"THENGOSUB155
60 IFA$=""THEN62ELSESOUND1,1
61 ON VAL(A$) GOSUB93,95,97,99,1
01,103,105,108,111
62 DR=DR+((L*SP)/100): 'change d
irection
63 GOSUB76
64 GOSUB80
65 IFDR<0THENDR=DR+360ELSEIFDR>3
60THENDR=DR-360:PRINT@72,"    ";
66 DP=DP+(TR/10)+((SP*RS)/300)
67 IF DP>GB THENGB=1:GOTO133 'cr
ash
68 IFDP<0THENDP=0
69 'print outs
70 F$=STR$((K+40)*100):F=VAL(MID
$(F$,4,2)):F=INT(F*.6)
71 TE=PEEK(&H603F)+40:IF TE=>&H3
9 THEN TE=TE-&H39
72 IC=INT((PEEK(&H6000+TE))/1.16
-11)
73 C$=STR$(F/100)+"00":IFDP=0THE
NPRINT@51," S";ELSEPRINT@51,INT(
DP-4);
74 PRINT@66,L;:PRINT@71,INT(DR);
:PRINT@77,SR;:PRINT@83,IC;:PRINT
@92,SP;:PRINT@98,RS;:PRINT@115,L
EFT$(F$,3);":";MID$(C$,3,2);:GOS
UB178
75 GOTO28
76 IFDR<90THENFX=DR:FY=DR-90:RET
URN
77 IFDR<180THENFX=180-DR:FY=DR-9
0:RETURN
78 IFDR>270THENFX=DR-360:FY=270-
DR:RETURN
79 FX=180-DR:FY=270-DR:RETURN
80 TR=SR-50:RETURN
81 'poke routine
82 FORR=1TOLEN(B$)STEP2
83 C$="&H"+MID$(B$,R,2)
84 V=VAL(C$):POKEY,V:Y=Y+1:NEXTR
85 RETURN
86 SOUND1,1:CS=CS+1:IFCS>3THENCS
=1
87 ONCS GOSUB89,90,91
88 GOTO28
89 PMODE4,1:SCREEN1,1:RETURN
90 SCREEN0:RETURN
91 SCREEN1,1:EXEC&H60D0
92 RETURN
93 IFL<45THENL=L+5
94 RETURN
```

```
95 IFL>-45THENL=L-5
96 RETURN
97 IFRS<45THENRS=RS+5
98 RETURN
99 IFRS>-45THENRS=RS-5
100 RETURN
101 IFSR>0THENSR=SR-2
102 RETURN
103 IFSR<100THENSR=SR+2
104 RETURN
105 IFSP<20THENSP=SP+4
106 IFSP<0THENGG=1ELSEGG=0
107 RETURN
108 IFSP>-20THENSP=SP-4
109 IFSP<0THENGG=1ELSEGG=0
110 RETURN
111 'periscope
112 PRINT@160,"":FORR=1TO9:PRINT
:NEXT:PRINT@197,"PERISCOPE AYE,
CAPTAIN"
113 IFIC=0THEN114ELSEIFDP-IC<14T
HENGB=2:GOTO133
114 FORR=1TO5:SOUNDR*10,1:NEXT:P
MODE3,5:PCLS3:FORR=80TO94STEP2:C
IRCLE(128,96),R:NEXT
115 GOSUB131
116 IFDP>15THENPAINT(128,96),2,0
:GOSUB131:CIRCLE(128,96),80,1:CI
RCLE(128,96),95,1:GOTO124
117 CI=0:FORR=&H6000 TO &H6038:C
I=CI+PEEK(R):NEXT:CI=CI/57-13:LI
NE(50,96)-(220,96),PSET:PAINT(12
8,90),0,0:PAINT(128,100),2,0
118 GOSUB131
119 DW=96:FORR=60TO210STEP10:WD=
RND(CI*10):WD=97-WD
120 IFWD>130THENWD=130ELSEIFWD<0
THENWD=RND(90)
121 LINE(R-10,DW)-(R,WD),PRESET:
DW=WD:NEXT
122 GOSUB131
123 CIRCLE(128,96),95,1:CIRCLE(1
28,96),80,1:PAINT(128,20),1,1:PR
INT@160,"":FORR=1TO9:PRINT:NEXT:
PRINT@203,"WE ARE ";:PRINTUSING"
###.#";(50-K)*25:PRINT"    NAUTI
CAL MILES FROM POLE"
124 PMODE4,5:SCREEN1,1
125 A$=INKEY$:IFA$=""THEN125 ELS
E POKE&H60B0,&H10:POKE&H60B4,&H1
2:POKE&H60BC,&H12
127 SCREEN0:EXEC&H60B0
128 POKE&H60B0,&H12:POKE&H60B4,&
H10:POKE&H60BC,&H10
 129 IF DP=0 AND MID$(F$,2,1)="9"
 THENGOTO142
 130 FORR=1TO5000:NEXT:PRINT@160,
"":FORR=1TO9:PRINT:NEXT:RETURN
 131 PRINT"          UP . . .":R
ETURN
```

```
132 'goodbye
133 CLS:SCREEN0:SOUND30,30
134 PRINT@200,"MISSION SCRUBBED"
135 ON GB GOSUB137,138,139,140
136 A$=INKEY$:IFA$=""THEN136ELSE
RUN
137 PRINT"      EXESSIVE PRESSUR
E HAS":PRINT"         DAMAGED H
ULL":RETURN
138 PRINT"    PERISCOPE DAMAGED
BY ICE":RETURN
139 PRINT"       COLLISION WITH
ICE":PRINT"   REQUIRES EXTENSIVE
 REPAIRS":RETURN
140 PRINT"       SUB HAS RUN AGR
OUND":RETURN
141 'messages
142 PRINT@160,"":FORR=1TO9:PRINT
:NEXT
143 PRINT@160,"":B$=" ANY SHIP O
R ANY STATION X THIS  IS THE USS
 SKATE X WE HAVE A    MESSAGE TO
 SEND X OVER":GOSUB148:GOSUB147
144 PRINT:PRINT:B$=" THIS IS RAD
IO MANILA X HEAR     YOU LOUD AN
D CLEAR X WILL       RELAY YOUR 
MESSAGE X OVER":GOSUB148:GOSUB14
7
145 PRINT@160,"":FORR=1TO8:PRINT
:NEXT:PRINT@160,"":B$=" FROM USS
 SKATE TO NAVY          DEPARTME
NT X HAVE SURFACED AT   NORTH GE
OGRAPHIC POLE X OVER":GOSUB148:G
OSUB147
146 PRINT:PRINT:B$=" TO USS SKAT
E FROM CHIEF OF      NAVAL OPERA
TIONS X              CONGRATULAT
IONS ON A JOB WELL   DONE X OVER
 AND OUT":GOSUB148
147 A$=INKEY$:IFA$=""THEN147ELSE
RETURN
148 FORR=1TOLEN(B$)STEP2
149 C$=MID$(B$,R,2):SOUND255,1:P
RINTC$;:NEXT:RETURN
150 'draw control panel
151 A=175:CLS0::FORR=0TO31:PRINT
CHR$(A);:NEXT:PRINT" TRIM ";CHR$
(A);"BEARG";CHR$(A);"BLLST";CHR$
(A);"      ";CHR$(A);"SPEEDR:
   ";CHR$(A);"     ";CHR$(A);"
  ";CHR$(A);"          ";
152 PRINTCHR$(A);"     P:    ";C
HR$(A);"DEGRS";CHR$(A);"% CAP";C
HR$(A);"        ";CHR$(A);"KNOTS"
;:FORR=1TO32:PRINTCHR$(A);:NEXT
153 FORR=1TO10:PRINT:NEXT
154 GOTO157
155 PRINT@160,"  <SPC> = CHANGE
```

continued on page 55

ASSEMBLY FILE
BY KEVIN and MICHAEL MONK

(Michael Monk from Bayne and Trembath, and Kevin have teamed up to produce the following article.)

This month I shall attempt to bash into your weary brow an explanation of the 6809 instruction set. By now you should have some understanding of the architecture of the 6809 along with the format of a typical assembly language program both prior to and following the assembly into machine code.

Make a quick study of this program (don't worry too much about the details for now)

```
SC      EQU   $0400    SC=&H0400 START OF SCREEN
RESET   LDA   #65      ACCUMULATOR A=65
        LDB   #01                  B=01
        LDX   #0000    CLEAR REGISTER X
START   STA   SC,X     STORE A AT ADDRESS X + SC
INCX    ABX            ADD ACCUM B TO REGISTER X
COMP    CMPX  #512     DOES REGISTER X = 512 YET
FIN     BEQ   LOOP     IF YES THEN GOTO LOOP
        JMP   START    OTHERWISE GOTO START
LOOP    JMP   LOOP     LOOP UNTIL THE END OF TIME
        END
```

All we are concerned with at the moment is the second column. By now if you don't already have an editor assembler package of some sort (eg. TANDY's EDTASM+) then you really should get one or at least have a copy of William Barden's book TRS-80 COLOR COMPUTER ASSEMBLY LANGUAGE PROGRAMMING ($9.95 from all TANDY stores Cat No. 62-2077)

What we are looking for anyway is a list of the 6809 instruction set and in my copy of the EDTASM+ manual that begins on page 39 or in appendix I and II of William Barden's book. Also on page 35 of the EDTASM+ manual you will find details of its Pseudo Operations but we'll talk more about those later.

The instructions act as the 6809's commands. When an instruction is encountered it knows that it must do something and that something is dictated by that instruction. So now comes the day of reckoning. Its time for you to start studying in earnest. You must know at least the most frequently used instructions and what they do with the data they act on. But you're in luck. There is an easy way. Drag out your old RAINBOW magazines and find those <u>SHORT</u> assembly listings and try to understand how they work. Those listings in which the author has explained each piece of code in detail are best. With your list of the 6809 Instruction Set open in front of you see how the author has used that instruction in his program. You will find that many instructions are dependent on the results of the preceding operation eg. the BRanch instructions which come after a CMP instruction. Keep in mind too the architecture of the 6809 chip as discussed in April. Remember the D accumulator is a 16 bit combination of the A & B 8 bit accumulators and that the registers X,Y,U,S & PC are all 16 bit.

Very soon you will find that you have developed a very crude familiarity with the more common instructions and better an ability to determine what the more obscure instructions do. But I stress DON'T spend too much time trying to fathom out a complex instruction. All you will achieve a complete lack of progress. Better to pass over the problem for now. There will come a time in the future when you will know how the instruction works.

Its time we got back to our program so lets examine it step by step just to see what the instructions do. To keep things simple all we are going to do is fill the entire screen with the character A, which you should already know has the decimal character code 65. The text screen occupies addresses (memory locations) $0400 through to $5FF which is exactly 512 bytes. Remember the $ (dollar sign) preceding a number means that number is base 16 hexadecimal and that the # (shifted 3 on your keyboard) means the following number is a data value as opposed to an address.

Our first instruction is EQU which in actual fact happens not to be an instruction but rather a Pseudo Operation. When the program is assembled, the assembler program sometimes needs to know information external to the program and in this case it is the address of the start of the text screen. The EQU or equate Pseudo Op in this case makes the variable SC equal to $0400. Another common Pseudo Op is ORG which simply tells the assembler where in memory the assembled program is to begin (ORiGinate).

Now we come to our first instruction. Actually the next three instructions are all almost identical. They are all LoaD (LD) instructions LDA and LDB are 8 bit and LDX a 16 bit load instruction. LDA loads accumulator A with in this case the decimal data value of 65. This is the same as saying make the contents of accumulator A = 65 and similarly in the next line we make accumulator B = 1. Likewise we make the X register = to 0, ie. we clear the X register.

Its time we got back to our program so lets examine it step by step just to see what the instructions do. To Keep things simple all we are going to do is fill the

entire screen with the character A, which you should already know has the decimal character code 65. The text screen occupies addresses (memory locations) $0400 through to $5FF which is exactly 512 bytes. Remember the $ (dollar sign) preceding a number means that number is base 16 hexadecimal and that the # (shifted 3 on your keyboard) means the following number is a data value as opposed to an address.

Our first instruction is EQU which in actual fact happens not to be an instruction but rather a Pseudo Operation. When the program is assembled, the assembler program sometimes needs to know information external to the program and in this case it is the address of the start of the text screen. The EQU or equate Pseudo Op in this case makes the variable SC equal to $0400. Another common Pseudo Op is ORG which simply tells the assembler where in memory the assembled program is to begin (ORiGinate).

Now we come to our first instruction. Actually the next three instructions are all almost identical. They are all LoaD (LD) instructions LDA and LDB are 8 bit and LDX a 16 bit load instruction. LDA loads accumulator A with in this case the decimal data value of 65. This is the same as saying make the contents of accumulator A = 65 and similarly in the next line we make accumulator B = 1. Likewise we make the X register = to 0, ie. we clear the X register.

The next instruction STA is the opposite of the load instruction in that its purpose is to store the contents of the A accumulator to an address. Now we start to get tricky. We have already made the label SC EQUal to $400 and register X = 0. What we will now do is STA not at the memory address pointed to by X but rather at an offset past X of $400 (the value EQU to SC). See how the result of this instruction depends on the form of the data following it and more importantly the power we have to manipulate the data itself. Incidentally this is our first example of indexed addressing.

Now we come to an odd man out but useful instruction. ABX or add the contents of register B to the contents of register X and store the result in register X leaving B unchanged. In this case we have simply incremented the contents of X by a value of one now making the contents of X = to 1. Notice there is no ABY or AAX or AAY instruction thus the B and X registers have in this case a rather exclusive relationship.

Unlike ABX, CMP can be used on any old register and here we are going to compare the contents of register X with the data value of 512. Last month we spoke of the Condition Code Register (CC) and then promptly ignored its existence. Most instructions however do have some effect on this register depending very much on the results of the operation. If you look at your description of the CMP instruction you will see which bits in the CC register are effected by this instruction (In William Barden's book this is the right hand group of columns in appendix II under the headings H N Z V C). In our program the next instruction BEQ now looks at the zero bit ie. the Z bit - of the condition code register. If that bit is equal to zero then the program will branch to the instruction found at the label LOOP otherwise it will fall through to the next instruction. What the CMP instruction actually does is subtract 512 from the contents of register X. If the result is zero then the Z bit is set and when we test BEQ (Branch if Equal to Zero) and find the Z bit set then we can follow the branch instructions alternative path to loop.

One more thing about the Branch instruction. This causes a Relative Branch ie. When the Program Counter Register (remember him) is given its new value it is not given an absolute address but rather a relative offset to its current value. This becomes important when we study Position Independent Code (way down the track yet). Don't worry too much as all this is handled by the appropriate BRanch instruction.

Well I'm going to leave you there for now. You have enough information to be able to figure out the remainder of the program and perhaps even how it works. We will talk more about the program itself next month.

White's Language

Assembly Language, the gurus all say
Is better than BASIC, the only way
So when my copy of RAINBOW I found,
I became all excited & began to jump up & down.
For there written before my eyes
Was an article written by Kevin - just my size
Kevin declared me by the bible, the book.
I already had it so I gave it a look.
My body was wracked with sigh after sigh,
For I found I had been living a lie
You see I thought STD was cheaper after six,
And you'd be in a COMA if drinks you'd mix
That BRA was something a woman burnt,
And cows eat hay by the STACK - that's what I learnt.
I thought things would become CLRA after awhile,
But SORTING things out took me a mile!
That BITA was what a mozzie did best
And RELATIVE's were more of a pest.
If you ADDA the fact that I thought I was right
In assuming if it's all black it's MINITE
That INCA was a member of a lost race,
And SEX is a bit of a disgrace??
That MUL was thinking things over - to be unaffected by a
bluster & BLO,
And ADBA was a rock group of a few years ago
I've still SCAnty knowledge from studying this 'bible'
If I soon do not learn I will be arrested for LABEL,
er..LIBEL!
After this I have one thing to say,
Give me OS9 anyday!

Andrew White

FORMATTER

By James A. Sanford

Long, multiple statement BASIC lines are quick to type and efficient in terms of memory and run time, but they are *so* hard to read. This is especially true when you're trying to debug a long program after a long, bleary-eyed session at the keyboard.

Wouldn't it be nice if there were some way to take advantage of multiple statement lines and still have a convenient, readable listing? The listing formatter described here allows the best of both worlds.

How it Works

The concept involved is easy. Just check each character about to be printed. If it happens to be a colon (multiple statements coming), move down a line and indent some from the left margin before printing it, then print the rest of the line.

Making a program actually do this is not much harder. Listing 1 shows an assembly language listing which will make it happen.

The routine called *Begin* initializes the program. First, it gets the address stored at $0168 by BASIC: $0168 is an intercept called prior to executing BASIC's character output routine. The original contents are stored so that *Formatter* can be disabled.

Preserving the original information in this manner allows the same program to be used on any system, with or without disk, and even with some other patch already installed. This done, the program will place the address for *Formatter* at $0168 and return control to BASIC.

When a character is to be output, BASIC will jump to $0167; this will tell BASIC to jump to the beginning of the *Formatter*. When *Formatter* is called, the character to be printed is contained in 'A.'

The routine labeled *Start* first preserves all 6809 registers, then checks to see if the character to be printed is a colon. If not, all registers are restored, and the program now jumps to where it would have without *Formatter*.

It is necessary to preserve registers prior to checking for a colon. This is the case since BASIC cares very much what happens to the condition code register.

If the character to be output is a colon, the program branches to the routine labeled *Form*. This prints a carriage return and five spaces. Following this, all registers (including the colon contained in 'A') are restored and program control returns to BASIC, as if *Formatter* wasn't even there.

The actual printing is done by the routine called *Print*. This routine checks the device code to see where the character is destined. It then routes the character to the screen or printer as appropriate, and does nothing if the character is to go anywhere else.

Finally, the routine labeled *Kill* exists to restore BASIC's original pointer, disabling *Formatter*. This is necessary if you have some other patch you may need to use periodically.

Note that the routine is written in position-independent code, so it can be placed in any convenient memory locations.

Using the Formatter

With a program this short, entering it is easy. If you have an editor/assembler, use the source code in Listing 1, then assemble and save to tape or disk. To enable *Formatter*, type EXEC; to disable it, it will be necessary to EXEC the address where the *Kill* routine is stored.

If you don't have an editor/assembler, it is possible to POKE in the object code in Listing 1. Operating instructions are the same as if it had been assembled.

Finally, Listing 2 shows a BASIC program which will generate the machine code and give directions for use.

Once *Formatter* is enabled, any time a colon appears, it will be printed one line down and indented five spaces. This will take place whether the colon is in a program listing or some printed text.

One word of caution is in order. The formatter will work when editing programs, but must be used with care. If you backspace into the five blank spaces, you will be spacing over the text which preceded the colon in the buffer. This happens since there is no longer

Listing 1:

```
                   00100 *************************
                   00110 *        FORMATTER        *
                   00120 *  A ROUTINE TO FORMAT    *
                   00130 *    BASIC LISTINGS       *
                   00140 *       COPYRIGHT         *
                   00150 *     (C) 1983  BY        *
                   00160 *   JAMES A. SANFORD      *
                   00170 *  ALL RIGHTS RESERVED    *
                   00171 *************************
                   00172 *
                   00173 *
                   00174 *
                   00175 *
                   00180 * RELOCATABLE, PUT IN GRAPHICS
0E10               00190         ORG     $0E10
0E10 34   77       00200 START   PSHS    X,Y,U,A,B,CC    *PRESERVE ALL
0E12 81   3A       00210         CMPA    #':     *DOES A CONNTAN A COLON?
0E14 27   06       00220         BEQ     FORM    *IF SO, FORMAT
0E16 35   77       00230 RETURN  PULS    X,Y,U,A,B,CC    *OTHERWISE, RESTORE
0E18 6E   9D 0011  00240         JMP     [ADDR,PCR]      *AND RETURN
0E1C 86   0D       00250 FORM    LDA     #$0D    *MAKE A A CARRIAGE RETURN
0E1E BD   0E44     00260         JSR     PRINT   *PRINT IT
0E21 86   20       00270         LDA     #$20    *NOW MAKE A A SPACE
0E23 C6   05       00280         LDB     #5      *FOR 5 TIMES,
0E25 BD   0E44     00290 SPACE   JSR     PRINT   *PRINT IT
0E28 5A            00300         DECB            *COUNTER
0E29 26   FA       00310         BNE     SPACE   *DONE?
0E2B 20   E9       00320         BRA     RETURN  *IF SO, BACK TO BASIC
0E2D               00330 ADDR    RMB     2       *STORAGE OF RETURN ADDRESS
                   00340 *ROUTINE TO DISABLE THE FORMATTER
0E2F BE   0E2D     00350 KILL    LDX     ADDR    *RESTORE BASIC'S
0E32 BF   0168     00360         STX     $0168   *POINTER
```

a one-to-one correspondence between what is on the screen and what is in the edit buffer. If you then insert or delete something, you will actually be modifying the program line, not the spaces before the colon.

Because of this potential problem, I recommend that you use *Formatter* to determine where you need changes, then disable it prior to editing any lines. Unless you're extremely sure of what you are doing, this is the safest course.

(For those having questions, you may contact Mr. Sanford at the following address: 15 Whitlor Dr., Mt. Holly, NY 08060.)

```
0E35 39              00370              RTS                *RETURN TO BASIC
                     00380 *ROUTINE TO ENABLE THE FORMATTER
0E36 BE      0168    00390 BEGIN    LDX        $0168    *STASH BASIC'S
0E39 AF      8D FFF0 00400          STX        ADDR,PCR *POINTER .
0E3D 8E      0E10    00410          LDX        #START   *TELL BASIC
0E40 BF      0168    00420          STX        $0168    *ABOUT FORMATTER
0E43 39              00430          RTS                 *RETURN TO BASIC
                     00440 *ROUTINE TO PRINT A CHARACTER
0E44 7D      006F    00450 PRINT    TST        DEVNUM   *WHICH DEVICE7
0E47 1027 94BF       00460          LBEQ       SCREEN   *IF 0 THEN SCREEN PRINTT
0E4B 34      02      00470          PSHS       A        *PRESERVE A
0E4D B6      006F    00480          LDA        DEVNUM   *GET DEVICE
0E50 8B      02      00490          ADDA       #2
0E52 35      02      00500          PULS       A        *RESTORE A
0E54 1027 9467       00510          LBEQ       PRTR     *IF -2 THEN PRINTER
0E58 39              00520          RTS                 *RETURN
             A30A    00530 SCREEN   EQU        $A30A
             A2BF    00540 PRTR     EQU        $A2BF
             006F    00550 DEVNUM   EQU        $6F
             0E36    00560          END        BEGIN
00000 TOTAL ERRORS
```

Listing 2: *FORMAT*

```
10 'PROGRAM TO GENERATE FORMATTE
R MACHINE LANGUAGE PROGRAM.
15 'COPYRIGHT (C) 1983 BY JAMES
A. SANFORD
16 ' ALL RIGHTS RESERVED
20 FOR A = &H 0E10 TO &H0E58:REA
D D:POKE A,D:NEXTA
30 CLS:PRINT:PRINT"NOW CSAVEM OR
 SAVEM 'FORMAT',&H0E10,&H0E58,&H
0E36":PRINT
40 PRINT"TO ENABLE, TYPE <EXEC>
OR ":PRINT"<EXEC &H 0E36>.":PRIN
T:PRINT"TO DISABLE, TYPE":PRINT"
     <EXEC &H 0E2F>."
50 DATA 52, 119, 129, 58, 39, 6,
 53, 119, 110, 157
60 DATA 0, 17, 134, 13, 189, 14,
 68, 134, 32, 198
70 DATA 5, 189, 14, 68, 90, 38,
250, 32, 233, 203
80 DATA 74, 190, 14, 45, 191, 1,
 104, 57, 190, 1
90 DATA 104, 175, 141, 255, 240,
 142, 14, 16, 191, 1
100 DATA 104, 57, 125, 0, 111, 1
6, 39, 148, 191, 52
110 DATA 2, 182, 0, 111, 139, 2,
 53, 2, 16, 39
120 DATA 148, 103, 57
```

16K
Disk

Personalize Your Printouts With Script

By Benjamin W. Brunotte

Tandy's CGP-115 Color Graphic Printer has always been great for printing business letters, charts and graphs, and special reports.

If you are among those who feel something is missing when you receive a personal letter written in regular typewriter-style fonts, you will welcome a program that allows you to type correspondence in cursive script. That letter from home will have a bit of a personal touch.

If you sometimes tend to scribble your words in cursive, you will be pleased with the readability of Script's results.

Script was originally written for use with spooler files created by Tandy's Color Disk SCRIPSIT, but most other CoCo word processors will work. Use a 27-character line length, and save the file with an extension of /SPI. When you run the program, type in the file name without the /SPI.

Those personal letters will be a joy to write!

-- Ben Brunotte

Editor's Note: This program ends with an IE (input past end of file) Error in line 200. This does not affect normal operation of the program.

If you have any questions about this program, you may contact Mr. Brunotte at 3155 French Rd., Apt. 218, Beaumont, TX 77706.

		545	19
18	157	556	253
240	72	520	60
450	94	580	233
539	211	END	238

continued on page 30

CoCo LINK

I mentioned previously that whilst at the Personal Computer Show in Sydney, I saw Telecom's Viatel and was impressed.

At that stage, various prices were floated, and I found it difficult to accept some of what I heard. My caution was well founded!

There was a price of $400 quoted by one consultant at the show. It turns out this was for a modem only - not for the (would you believe) $3000 "Computerphone" he was showing me at the time!

I have reported at length on Stars in the Education section, so I am not about to repeat that information here. Briefly however, Stars is a database Bulletin Board which operates in a superior manor to our own CoCoLink. Never the less, Stars runs on 300/300 protocol, like CoCoLink.

Viatel on the otherhand, will require a modem with 1200/75 baud abilities and a Color Computer compatable "Pestel" program.

Currently on Viatel, you can obtain information on Qantas schedules, and the WA TAB Racing information. You can bank with the CTB, there is a computer magazine, you can download Commodore programs, and you can obtain information on courses you can study in a university in Western Australia, information on life insurance and also NRMA information.

Soon to be added will be articles and programs for CoCo - by us!

I can not say whether Viatel will be worth spending the extra to buy the correct modem and program.

Having a 1200/75 protocol, it is not especially interactive, and tends to have a "magazine" feel to it. The technology is old, and it has that aura.

Conversely, displays are colourful enough, and it has a big potential, given sufficent interest from others. Its just like Telecom to choose a second rate system like this with which to introduce the masses to computer communication!

I doubt that the present services will be enough to encourage CoCo users to get on line with Viatel!

At last we can down load programs from CoCoLink! I don't believe it is happening - but it is! There are not a lot of programs to down load at this stage - we'll wait till after CoCoConf and then will start to load the system with programs.

All being well, we should be selling CoCoOz through CoCoLink in August!

There are some continuing problems with CoCoLink which we are still attempting to define. Until we overcome, what we think will be the last of the major problems, we ask that you especially adhere to the convention of logging off, BEFORE hanging up!

Graeme Nichols of the North Shore Users Group has sent news of an OS9 Bulletin Board they are running. Presently, the system is on line from 19.30 to 21.30 Tuesdays and Thursdays. All OS9 users are welcome to use it to swap ideas and proceedures. Log on as a visitor and leave your name, address and phone number, and you will be rung back with a password.

The group do this as a no charge service, but if you intend to use their system often, have the good manners to offer something towards the costs. (My words, not theirs.)

COCOCONF

Registration for CoCoConf is now closed. Those of you who have registered should be hearing from us in the first week of June. If you do not, then PLEASE call us, because it is likely that there will be a change of venue. We don't want you going to the wrong place!

Unfortunately Lonnie Falk is ill and is unlikely to attend. The other bad news is that the Expo has been cancelled. We don't seem to be able to take a trick! We'll be making it up to you with a couple of little surprises on the weekend.

These changes aside, I am confident that we have a very worthwhile weekend ahead of us.

The games competition comes to an end this month, and winners will be announced on the Saturday night.

The prizes are

* a disk drive, donated by Software Spectrum,
* an amber screen monitor donated by Blaxland Computer Services.

and * a Tandy Speech Pack, donated by Bayne and Trembath.

The entries have been of a very high standard, and we are looking forward to presenting the prizes to several very deserving people!

For those of you who are not attending, we will report on proceedings next issue!

Tandy
ELECTRONICS

50% Off Compact Color Graphics Printer

149^{95}

- *With Serial and Parallel Interfaces*

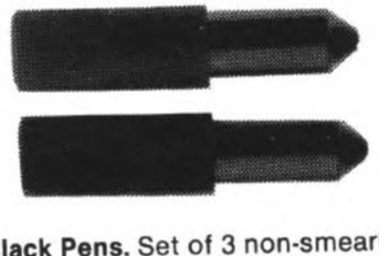
Reg **299.95**

Save $150

CGP-115. Color Graphics versatility — that won't eat into company profits! This little marvel responds to your computer commands instantly to produce anything from charts to computer-generated "doodles". Prints in red, blue, green and black from easily replaceable ink cartridges. Drawing and plotting are simplified by Built-in commands. In Text Mode CGP-115 will even take the place of a Daisy Wheel Printer by listing your programs and data on paper, at a speedy 12 characters per second with 40 or 80 characters per line. Prints on 11.4cm wide white paper. Software provides additional character size and rotation. 26-1192

Dust Cover 26-518...**6.95**

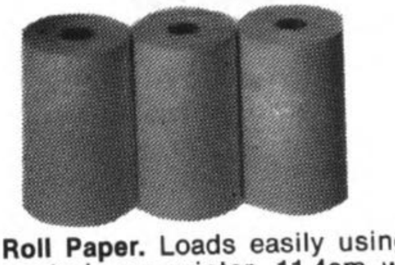
Replacement Pens



4^{95} EACH

Black Pens. Set of 3 non-smearing pens. 26-1480
Color Pens. Set of 3, with one of each color. Red, green and blue. Non-smearing. 26-1481

Replacement Paper Roll



12^{95}

Roll Paper. Loads easily using feed up and down controls on printer. 11.4cm wide, 45 metres long white paper. 26-1428

SALE!

75% Off Program Pak™ File Box

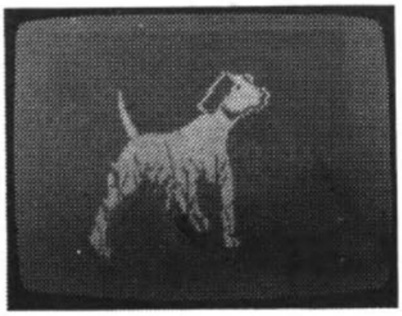
$60 Off!

Reg **79.95**

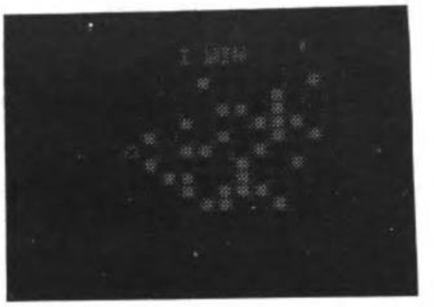
19.95

Keep your programs and cassettes "out of harms way" at a fraction of the normal cost — with this dustproof mini file cabinet. Holds 16 program paks and their manuals, or 16 cassette boxes. Tinted, slide down cover protects software from harmful rays to extend their life. Sloping display for easier selection. 26-1314

Color Games for a Sense of Adventure



Save upto $30



A. Art Gallery. Draw landscapes, cartoons and more with up to eight colors! Save with optional cassette recorder. 26-3061 Was 49.95 .... **Now 19.95**

B. Robot Battle. Program your robot in BASIC to fight with more daring tactics and win the battle. 26-3070 Was 49.95 .... **Now 19.95**

C. Gomoku and Renju. A classic oriental game of strategy. It's harder than you think to place five men in a row. 26-3069 Was 39.95 .... **Now 19.95**

D. Gin Champion. Test your skill at 16 different versions of gin by challenging the computer. 26-3083 Was 49.95 .... **Now 19.95**

Tandy ELECTRONICS

A DIVISION OF TANDY
AUSTRALIA LIMITED
INC. IN N.S.W.

Nearly 350 Stores Australia-Wide

WE SERVICE WHAT WE SELL!

**Available from
350 Stores Australiawide
including Tandy Computer Centres**

**Independent Tandy Dealers may not be participating in this ad or have every item advertised. Prices may also vary at individual Dealer Stores*

continued from page 26

The listing: *SCRIPT*

```
5 GOTO 100
10 PRINT#-2,"J1,0,2,1,2,2,2,4":R
ETURN
11 PRINT#-2,"J6,0,1,1":RETURN
12 PRINT#-2,"J1,1,2,1,1,0,2,-1,-
2,1,-1,0,-2,-1,-1,-1,-1,-2,0,-1,
1,-3,2,-1":RETURN
13 PRINT#-2,"R13,24":PRINT#-2,"J
2,-2,2,-1,1,1,0,1,-5,1":RETURN
14 PRINT#-2,"R9,23":PRINT#-2,"J-
3,-2,-2,-1,-2,1,0,1,3,1,6,0,7,1"
:PRINT#-2,"R-7,-1":PRINT#-2,"J-1
,-20,-1,-2,-2,-1,-1,0,-2,1":RETU
RN
15 PRINT#-2,"R2,0":PRINT#-2,"J3,
2,2,2,2,4,2,7,1,5,0,1,-1,3,-1,0,
-1,-3,0,-2,1,-4,1,-2":RETURN
16 PRINT#-2,"J-2,-11,-1,-1,-1,0,
-1,2,0,2,1,2,2,3,2,2,3,1,1,0":RE
TURN
17 PRINT#-2,"R7,24":PRINT#-2,"J-
1,-2,-1,-1,-1,0,-1,1,0,1,1,1,2,1
,1,0":RETURN
18 PRINT#-2,"R11,24":PRINT#-2,"J
-1,0,-3,-1,-2,-2,-1,-2,-1,-6,0,-
2,1,-6,1,-2,2,-2,3,-1,1,0,3,1,2,
2,1,2,1,6,0,2,-1,6,-1,2,-2,2,-3,
1,-1,0,-3,-2,0,-2,1,-1,2,0,5,1,4
,4":RETURN
19 PRINT#-2,"R3,21":PRINT#-2,"J1
,0,2,1,1,1,0,1,-1,-24,-1,2,2,18,
1,3,2,1,3,0,3,-1,1,-1,1,-3,0,-1,
-1,-2,-2,-2,-3,-1,-6,0":RETURN
20 IF PV=72 OR PV=73 OR PV=98 OR
 PV=111 OR PV=118 OR PV=119 THEN
 PRINT#-2,"R0,-6"
21 RETURN
22 IF PV<65 THEN RETURN
23 IF PV<91 THEN 25
24 IF PV<97 THEN RETURN
25 PRINT#-2,"R4,0":T=T+4:RETURN
100 CLS:CLEAR 1000
110 PRINT#-2,CHR$(13);
120 PRINT#-2,CHR$(18)
130 PRINT#-2,"S2"
140 PRINT#-2,"R5,0"
150 PRINT#-2,"I"
160 T=5:PV=0
162 CLS:INPUT"ENTER scripsit SPO
OL FILE NAME";NM$
164 IF LEN(NM$)>8 THEN 162
166 NM$=NM$+"/SPL"
170 OPEN"I",#1,NM$
200 LINEINPUT#1,LL$
201 L=LEN(LL$):IF L=0 THEN 450
202 FOR W=1 TO L:V$=MID$(LL$,W,1
)
210 V=ASC(V$)
211 IF V=0 THEN 1000
230 IF V=13 THEN 450
240 IF V<32 OR V>122 THEN NEXT W
:GOTO450
250 PRINT V$;
260 IF V<65 THEN 360
270 IF V<91 THEN 330
280 IF V<97 THEN 360
300 IF PV=72 OR PV=73 OR PV=98 O
R PV=111 OR PV=118 OR PV=119 THE
N GOSUB 11 ELSE GOSUB 10
310 Z=V-96:ON Z GOTO 565,566,567
,568,569,570,571,572,573,574,575
,576,577,578,579,580,581,582,583
,584,585,586,587,588,589,590
320 GOTO200
330 GOSUB 20
340 Z=V-64:ON Z GOTO 533,534,535
,536,537,538,539,540,541,542,543
,544,545,546,547,548,549,550,551
,552,553,554,555,556,557,558
350 GOTO 200
360 GOSUB 20:GOSUB 22
370 PRINT#-2,"R0,-4":PRINT#-2,"P
"+V$
380 PRINT#-2,"R0,4":GOTO 419
410 T=T+10:GOTO 430
411 T=T+11:GOTO 430
412 T=T+12:GOTO 430
413 T=T+13:GOTO 430
414 T=T+14:GOTO 430
415 T=T+15:GOTO 430
416 T=T+16:GOTO 430
417 T=T+17:GOTO 430
418 T=T+18:GOTO 430
419 T=T+19
430 IF T=>460 THEN 450
440 PV=V:NEXT W:GOTO450
450 PRINT CHR$(13);
460 PRINT#-2,"H"
470 PRINT#-2,"R0,-37"
480 PRINT#-2,"I"
490 T=5:PV=0:GOTO200
533 PRINT#-2,"R17,22":PRINT#-2,"
J-1,1,-2,1,-2,0,-2,-1,-3,-3,-3,-
5,-1,-5,0,-2,1,-3,1,-2,2,-2,3,-1
,4,0,2,1,1,2,1,5,0,9,-1,5,-1,-5,
0,-9,1,-7,1,-1,1,0":GOTO 419
534 PRINT#-2,"R3,21":PRINT#-2,"J
2,1,1,2,-1,-22,1,-2,0,24,7,0,2,-
1,2,-2,1,-2,0,-1,-1,-3,-1,-2,-3,
-2,-4,0,-3,1,3,1,4,0,2,-1,2,-2,1
,-3,0,-2,-1,-3,-1,-1,-3,-1,-7,0,
1,1,3,1,5,-1,3,-1,1,0":GOTO 419
535 GOSUB 13:PRINT#-2,"J-2,0,-3,
-1,-2,-1,-2,-3,-1,-5,0,-4,1,-5,2
,-3,4,-2,9,0":GOTO 419
536 PRINT#-2,"R7,24":PRINT#-2,"J
-2,-24,-1,0,-1,1,0,1,1,0,3,-1,4,
-1,2,0,2,1,2,4,1,5,0,4,-1,5,-2,3
,-3,2,-6,0,-2,-1,1,-1,6,-1,4,1,4
,2":PRINT#-2,"R0,-24":GOTO 419
537 GOSUB 13:PRINT#-2,"J-3,0,-3,
-1,-2,-1,-2,-3,0,-2,1,-2,2,-2,4,
-1,5,0,2,1,-2,1,-6,0,-3,-2,-2,-2
,-1,-3,0,-3,1,-2,2,-1,4,-1,9,0":

GOTO 419
538 GOSUB 14:PRINT#-2,"J-1,3,0,1
,1,3,1,1,2,1,9,2,-1,-5":PRINT#-2
,"R3,-7":GOTO 419
539 GOSUB 15:PRINT#-2,"J2,-1,1,0
,3,1,1,1,-1,-5,-1,-3,-1,-3,-2,-2
,-2,-1,-2,0,-5,1,-1,1,1,1,6,-2,6
,-1,3,0":GOTO 419
540 PRINT#-2,"R6,22":PRINT#-2,"J
-1,-1,-1,0,-1,1,0,1,1,1,1,0,1,-1
,-1,-23":PRINT#-2,"R12,24":PRINT
#-2,"J-1,-24,1,1,0,7,-1,2,-11,1,
-1,-1,0,-1,1,-1,12,-1,2,-1":GOTO
 419
541 PRINT#-2,"R18,0":PRINT#-2,"J
-2,1,-1,1,-2,4,-2,6,-1,7,0,1,1,3
,1,1,1,0,1,-1,1,-3,0,-1,-1,-8,-1
,-5,-1,-4,-2,-2,-2,0,-2,1,-2,3,-
1,4,3,-1,5,-1,8,0":GOTO 419
542 PRINT#-2,"R11,0":PRINT#-2,"J
-4,5,-2,6,0,3,1,7,1,2,1,1,1,0,1,
-1,1,-2,1,-6,0,-3,-2,-12":GOSUB
16:GOTO 414
543 GOSUB 17:PRINT#-2,"J1,-1,-3,
-23":PRINT#-2,"R13,24":PRINT#-2,
"J0,-1,-3,-5,-7,-7,-1,0,-1,1,1,1
,1,0,2,-2,3,-5,2,-5,3,-1,1,0":GO
TO 419
544 PRINT#-2,"R3,23":PRINT#-2,"J
3,-1,4,-1,7,0,1,1,0,1,-1,1,-1,0,
-1,-1,-1,-5,-1,-6,-1,-4,-1,-3,-2
,-3,-3,-2,-2,0,-1,1,0,1,1,1,2,0,
2,-2,3,-1,6,-1,2,0":PRINT#-2,"R0
,1":GOTO 419
545 PRINT#-2,"R5,21":PRINT#-2,"J
-1,0,-1,1,0,1,1,1,1,0,1,-1,-1,-2
3,-1,2,2,21,2,1,1,0,2,-2,1,-18,-
1,-2,-1,2,1,18,2,2,1,0,2,-1,1,-2
2,1,-1,1,0":GOTO 419
546 GOSUB 17:PRINT#-2,"J1,-1,-1,
-23,-1,2,2,17,1,3,2,2,3,0,1,-1,1
,-3,1,-19,1,-1,1,0":GOTO 419
547 GOSUB 18:PRINT#-2,"R0,-24":G
OTO 419
548 GOSUB 19:PRINT#-2,"R13,-13":
GOTO 419
549 GOSUB 18:PRINT#-2,"R-8,-18":
PRINT#-2,"J6,-7":PRINT#-2,"R2,1"
:GOTO 419
550 GOSUB 19:PRINT#-2,"J1,1,5,-1
,2,-4,2,-8,1,-1,1,0":GOTO 419
551 GOSUB 15:PRINT#-2,"J4,-4,2,-
3,-1,-2,-2,-2,-3,-1,-2,0,-5,1,-1
,1,1,1,6,-2,6,-1,3,0":GOTO 419
552 GOSUB 14:PRINT#-2,"J-1,2,-1,
4,4,-1":PRINT#-2,"R12,-6":GOTO 4
19
553 GOSUB 17:PRINT#-2,"J1,-1,-1,

-17,1,-3,1,-2,2,-1,3,0,2,1,1,2,1
,3,0,18,-1,-18,1,-6,1,0":GOTO 41
9
554 GOSUB 17:PRINT#-2,"J1,-1,1,-
3,2,-17,1,-3,1,0,1,3,2,17,1,3,1,
1,1,0":PRINT#-2,"R0,-24":GOTO 41
9
555 PRINT#-2,"R4,21":PRINT#-2,"J
-1,1,0,1,1,1,1,-1,-1,-20,2,-4,2,
0,2,4,2,16,-1,3,-1,-3,0,-16,2,-4
,2,0,2,4,1,20,1,1,1,0":PRINT#-2,
"R0,-24":GOTO 419
556 PRINT#-2,"R4,21":PRINT#-2,"J
-1,-1,-1,0,-1,1,0,2,1,1,1,0,3,-4
,2,-5,2,-6,2,-5,2,-4,1,0,1,1,0,2
,-1,1,-1,0,-1,-1":PRINT#-2,"R4,2
1":PRINT#-2,"J-1,0,-1,-1,-12,-22
,-1,-1,-1,0":PRINT#-2,"R16,0":GO
TO 419
557 GOSUB 17:PRINT#-2,"J1,-1,-1,
-17,1,-3,1,-2,1,-1,3,0,1,1,1,2,1
,3,2,15,0,3,-3,-24":GOSUB 16:GOT
O 419
558 GOSUB 17:PRINT#-2,"J2,-1,2,-
3,1,-2,1,-5,0,-2,-1,-4,-1,-3,-1,
-2,-3,-2,-1,0,-1,1,0,1,1,1,1,0,2
,-1,0,-2":GOSUB 16:PRINT#-2,"J2,
0":GOTO 419
565 GOSUB 12:PRINT#-2,"J2,0,2,1,
1,5,-1,2,0,-5,1,-2,1,-1,1,0":GOT
O 416
566 PRINT#-2,"J1,7,0,2,-1,4,-1,0
,-1,-4,0,-2,1,-7,1,-5,1,-2,2,0,1
,5,0,1,2,0":GOTO 413
567 GOSUB 12:PRINT#-2,"J6,0":GOT
O 415
568 GOSUB 12:PRINT#-2,"J2,0,2,1,
1,8,0,6,-1,5,-1,-4,1,-13,1,-2,1,
-1,1,0":GOTO 416
569 PRINT#-2,"J1,2,0,1,-1,1,-1,0
,-1,-1,0,-1,3,-8,1,-1,1,0":GOTO
410
570 PRINT#-2,"J2,6,0,2,-1,4,-1,-
1,-2,-24,1,-2,2,4,0,1,-1,2,-2,2,
5,-1,1,0":GOTO 411
571 GOSUB 12:PRINT#-2,"J2,0,2,1,
1,5,-1,2,-1,-15,-1,-1,-1,2,0,1,2
,4,3,1,1,0":GOTO 416
572 PRINT#-2,"J1,7,0,2,-1,4,-1,-
20,0,5,2,2,2,0,2,-2,0,-4,1,-1,1,
0":GOTO 414
573 PRINT#-2,"J0,2":PRINT#-2,"R0
,6":PRINT#-2,"J0,-1":PRINT#-2,"R
0,-5":PRINT#-2,"J0,-2,1,-4,1,-2,
1,-1,1,0":GOTO 411
```

```
574 PRINT#-2,"J0,2":PRINT#-2,"R0
,6":PRINT#-2,"J0,-1":PRINT#-2,"R
0,-5":PRINT#-2,"J0,-2,-1,-14,-1,
-1,-1,2,0,1,2,4,3,1,1,0":GOTO 41
0
575 PRINT#-2,"J1,6,0,3,-1,5,-1,0
,0,-21,0,8,1,1,3,0,1,-1,0,-3,-5,
0,3,0,2,-5,1,0":GOTO 413
576 PRINT#-2,"J1,7,0,2,-1,4,-1,0
,-1,-4,0,-2,1,-7,2,-6,1,-1,1,0":
GOTO 410
577 PRINT#-2,"J1,0,1,-1,-1,-6,0,
3,1,3,1,1,2,0,1,-1,0,-5,-1,-1,1,
6,1,1,2,0,1,-1,0,-5,1,-1,1,0":GO
TO 419
578 PRINT#-2,"J1,0,1,-1,-1,-6,0,
3,1,3,1,1,2,0,1,-1,0,-5,1,-1,1,0
":GOTO 415
579 PRINT#-2,"J2,1,1,0,-1,0,-2,-
1,-1,-2,0,-2,1,-2,2,-1,1,0,2,1,1
,2,0,2,-1,2,-2,1,-1,0,-1,-1,0,-1
,1,-1,5,1,1,0":GOTO 415
580 PRINT#-2,"J0,2,-1,-16,-1,-1,
-1,2,0,1,3,12,2,1,1,0,2,-1,1,-2,
0,-2,-1,-2,-2,-1,-4,0,1,1,7,-1,1
,0":GOTO 415
581 GOSUB 12:PRINT#-2,"J2,0,2,1,
0,7,-1,-14,1,-2,2,4,0,1,-1,2,-2,
2,4,-1,1,0":GOTO 417
582 PRINT#-2,"J0,1,1,-1,4,0,-1,-
3,0,-1,1,-2,1,-1,1,0":GOTO 414
583 PRINT#-2,"J1,3,1,-1,2,-4,0,-
1,-1,-3,-2,-1,-1,0,-3,2,3,-1,5,-
1,1,0":GOTO 413
584 PRINT#-2,"J0,8,-1,-8,2,-6,1,
-1,1,0":PRINT#-2,"R-8,11":PRINT#
-2,"J8,1":PRINT#-2,"R0,-12":GOTO
 410
585 PRINT#-2,"J0,2,-1,-6,1,-2,2,
-1,2,0,2,1,1,8,-1,-6,1,-2,1,-1,1
,0":GOTO 416
586 PRINT#-2,"J0,2,-1,-5,1,-3,1,
-1,1,0,1,1,1,3,1,5,0,-2,1,-1,1,0
":GOTO 414
587 PRINT#-2,"J0,1,-1,-7,1,-1,2,
0,1,1,1,5,-1,-2,0,-3,1,-1,2,0,1,
1,1,7,1,-2,1,0":GOTO 417
588 PRINT#-2,"J1,1,2,-1,1,-2,1,-
2,1,-2,1,-1,2,0":PRINT#-2,"R-2,8
":PRINT#-2,"J-6,-8":PRINT#-2,"R8
,0":GOTO 416
589 PRINT#-2,"J0,2,-1,-6,1,-2,2,
-1,1,0,2,1,1,2,0,6,-1,-17,-1,-1,
-1,2,0,1,2,5,2,1,1,0":GOTO 415
590 PRINT#-2,"J1,1,2,1,1,0,2,-1,
1,-2,0,-2,-1,-2,-1,-1,-2,-1,-1,0
,-1,1,1,1,1,0,2,-2,-1,-8,-1,-1,-
1,2,0,2,2,4,4,1,1,0":GOTO 416
1000 CLOSE#1:UNLOAD:END
```

CORRECTIONS

"Cooking With CoCo" (March 1985, Page 43): Colin Stearman passes along a note from a reader who says that Radio Shack's disk versions of *EDTASM*, *Scripsit* and *Spectaculator* (all of which use the Color TRSDOS operating system) use bytes 17 and 18 of the directory entry, which conflicts with the file creation date area used by his program. Dates are not put into the file when these programs create files and they should pose no real problem. Colin also cautions that files created by these programs should not be redated with the BASIC program on Page 148. This problem may also occur with future Radio Shack programs that use Color TRSDOS (which is supplied on each program disk in each package).

"Cooking With CoCo" (April 1985, Page 48): Two numbers were reversed in the paragraph beginning with "Mount the SN7404 . . . ." The corrected sentences read ". . . connect pins 1, 3, 5, 7, 9 and 11 together and also to pin 1 of the 40-pin socket. Connect pin 14 to pin 20 of the 40-pin socket." Also, in later models of the CoCo (NC board and CoCo 2) the circuit board is "piggybacked" over the MC6821 PIA chip; the second "PIA" is actually an MC6822 IIA chip in these machines (or in older computers with Radio Shack's upgraded keyboard installed).

hint

To those out there who have typed in "Road Race" (Nov. 1984) and have had problems with line 660, here is the answer to that problem:

It should read thus:

```
660 DRAW"BM106,182;"+N$(SA)+"BM114,182;"+N$(SB)+
 "BM122,182;"+N$(SC)+"BM160,182;"+N$(HA)+
 "BM168,182;"+N$(HB)+"BM176,182;"+N$(HC)+
 "BM184,182;"+N$(HD)
```

Alex Hartman.

One-Liner Contest Winner . . .

This program, adapted from an algorithm used on a Texas Instruments 57 programmable calculator, computes the interest per period for normal annuities.

The listing:

```
1 PRINT:INPUT"PRESENT VALUE";PV:
INPUT"PAYMENT            ";PMT:INPUT"#
 PERIODS         ";N:F=PMT/PV:I=1.05:
FORT=1TO30:I=(1-I^-N)*F+1:NEXT:P
RINT USING "INTEREST IS     ###.##
 PERCENT";(I-1)*100:RUN
```

Bob vanderMark
Groningen, The Netherlands

BITS AND BYTES OF BASIC

An Introduction To Spreadsheets

By Richard A. White

After writing about BASIC for over two years, it is time for a break. For the next several months we will concern ourselves with what is, in fact, a language, and perhaps a higher level language than BASIC. This is the spreadsheet program.

Spreadsheets are fairly new in computing history. *VisiCalc*, the first, was invented by Daniel Bricklin and Robert Frankston. The incentive for its invention was Bricklin's frustration with the accountant's pencil and paper that he had to work with in business school. Bricklin viewed *VisiCalc* as a program to facilitate accounting by easing the data entry and letting the computer do the calculations.

As with all good programs, users found more and better uses. One of the first out of the box was using the program to make "what if" business projections. A good spreadsheet can be considered a simulation kit with many uses, some of which are still to be discovered.

A spreadsheet can be used as a simple text editor since the cursor can be easily moved around the sheet to any desired cell. A spreadsheet can also be a database manager with a sort function to re-order records. Some spreadsheets even have a search function to locate a specified character sequence that may appear one or more times in the spreadsheet contents.

The spreadsheets we will talk about are organized into a rectangular array of cells. Cells are addressed by a one- or two-letter column identifier and a row number. The cell in the third row of the first column is "A3."

The user has total control of what goes into each cell. A cell may contain a number, a string or label or a formula that draws on data in other cells to generate the contents of the cell. I said generate rather than calculate since some spreadsheets can use formulas to select strings to appear in the cell.

Finally, formatting commands can be used to control how the data is shown in a cell or groups of cells. For example, numbers may be formatted as integers, in dollar format (e.g., $10.00) or in a number of other ways. Strings may be left-justified, right-justified or in more powerful spreadsheets, centered in a cell. And, when you go to print the spreadsheet, it's literally "what you see is what you get."

If you are thinking these types of capabilities are neat, many, many others have, too. And they will pay up to $500 for spreadsheet programs. *VisiCalc* initially sold for $200 and not only sold programs, but sold $2000 computers to run the program. The Apple II was a principal beneficiary, but buyers are fickle creatures with little loyalty. The IBM PC arrived and not long after a *VisiCalc* appeared for it. But, Mitch Kapor and his programmers were brewing trouble in the form of *Lotus 1-2-3*, which has coined about $100,000,000 in a year and a half. *VisiCalc* was caught with its enhancements down and is just a shadow on the market.

All this relates to bigger computers than the CoCo. We have had spreadsheet development as well, even if it does not draw "Silicon Valley" limelight. THE RAINBOW was just a toddler when Radio Shack brought us the *Spectaculator* ROM and the excitement among us old-timers was indeed great. While hardly a *VisiCalc*, it did many jobs well and was only $49.95. It also departed from the *VisiCalc* command methods which did not seem important at the time. *Disk Spectaculator* is similar to the ROM Pak version with a few enhancements.

The first RAINBOWfest was in Chicago in 1983 and there *Elite*Calc* was introduced with much acclaim. My

Spectacular ROM Pak went onto the shelf and has been there since. *Elite*Calc* won me over with Copy and Move, Lookup tables, If . . . Then . . . Else, better formatting and better printer control. So my spreadsheet investment was up to $120.

It was not to stop there. *DynaCalc* plucked another $100 from my pocket. But, I have three spreadsheets rather than one *VisiCalc*, and with *DynaCalc* I have some neat graphing abilities that are missing in *VisiCalc* and speed in areas where speed counts.

This is one reason I have drawn the line and not bought *VIP Calc*. *VIP*'s screen handling and calculating are so slow, while *DynaCalc* is speedy. On the other hand, *DynaCalc* will print a spreadsheet and that's all in that department, while *VIP Calc* has all sorts of "specialties" but lacks the graphing *DynaCalc* has, along with the ability to do a high resolution screen dump of the graph to many printers.

Which brings us to the 59, 69, 99 or 149 dollar question. If you want to buy a spreadsheet to use as you follow this series, what should it be? If you have a disk drive and 64K, you could go for *DynaCalc* running under CoCo DOS, not OS-9. But, the people at VIP Technologies, formerly Softlaw, would like to sell you *VIP Calc* along with their five other utilities, *Disk-Zap*, *Speller*, *Database*, *Writer* and *Terminal*, for only $149 in an "integrated package." Since the *Writer* and *Terminal* rank among the best in their respective areas, it's a tough package to turn down. The *VIP Integrated Library* is available from local dealers who handle CoCo software or by express order from any Radio Shack store.

If you have a tape-based system, the choice is much tougher. *VIP Calc* comes with tape and disk, and has the high resolution text screen. *Elite*Calc*, purchased either on tape or disk, uses the standard CoCo text screen, but is fast, easy to use and well-debugged. As one owner of the four CoCo spreadsheets observed, if the best features of each were combined into one program, near perfection would be achieved. I expect the serious spreadsheet user will end up with a number of programs to meet a variety of needs.

Whatever you buy will give you a major increase in the usefulness of your CoCo. Granted, you can do anything with a BASIC or PASCAL program that you can do in a spreadsheet, although working with a spreadsheet is much quicker and easier.

We are all used to dealing with forms, though we may not like to. A form serves to organize information in a specific way and may include instructions for processing that information for inclusion in other places on the same form.

Income tax forms are probably the best and most widely known (and maybe most disliked) example. Forget the pain of your money going to support programs you may not believe in, the time spent finding all those pieces of paper with the data needed for the income tax forms only to find that things are still confusing and troublesome. The calculations are fairly simple, consisting only of addition, subtraction and multiplication, but they must be right, which means checking and rechecking. Then there are the choices.

Here are some choice calculations from Schedule G (Form 1040) "Income Averaging."

15. Multiply the amount on line 14 by 25% (.25).
16. Write in the amount from line 7 above.
17. Add lines 15 and 16.
18. Write in the amount from line 11 above.
19. Add lines 17 and 18.
20. Tax on amount on line 19 (from Tax Rate Schedule X, Y, or Z).
21. Tax on amount on line 17 (from Tax Rate Schedule X, Y, or Z).
22. Tax on amount on line 16 (from Tax Rate Schedule X, Y, or Z).

Is this sounding more like computer work than people work? Would it be a lot of work to program in BASIC or some other high level language? Not only would you need to write code to do the calculations, but you would need to write PRINT routines to display the results so the user could transfer the results to the tax form to keep the IRS happy. And what about editing the data after it is input? Now there is a real can of worms.

While a spreadsheet does not remove the need to understand the task and write formulas to implement it, presentation of results and ability to edit data are integral to the spreadsheet such that the programmer need give these little thought. And the actual programming is easier.

Cells replace variables so the abstraction of some two character name is replaced by some X-Y location that can be labeled with real words and reviewed by scrolling through the spreadsheet under arrow key control. Further, a formula is written into a cell and the results of its labors are shown in that cell.

As one example of spreadsheet programming speed, I developed a Schedule G (Form 1040) spreadsheet in *Elite*Calc* with about six hours of work. To make the sheet fully general, it needs to calculate tax values based on four tax schedules (Schedule Y has two versions) for values from five different lines in the form.

The total Schedule G file uses over 10,000 bytes. About a third of this is *Elite*Calc* overhead, while the remainder is labels, data and formula.

Now I will grant you that having already written the tax calculations into Form 1040 and 1040A spreadsheets, I knew just what I needed to do. Still, to program and debug this in BASIC would have taken over twice as long. If you are reading this in March 1985 and feel the hot breath of April 15th down your neck, you can buy a package of eight income tax spreadsheets, including those mentioned above from Elite Software Users Group, Box 683, West Fargo, ND 58078, phone (701)281-0549. These are for disk *Elite*Calc* and cost $19.95 by check or money order.

While big, powerful spreadsheet programs can be fun and impressive, let's first talk about a very simple example. Enter four numbers to be added. We will put them in a column using cells A1 to A4.

```
  :     A     ::     B     ::     C     :
1      42.00
2       7.00
3      27.80
4     100.09
5   ---------
```

To enter the numbers, the user moved the cursor to each cell in turn and typed the number that is in that cell. The spreadsheet added zeros after the decimal as needed to provide the normal .XX default presentation. Individual cells or groups of cells can be formatted to change this. The row of dashes in A5 is a string of dash characters. The simplest way to add the numbers is to put the formula A1+A2+A3+A4 into A6. As long as the spreadsheet is in the automatic calculation mode, the answer immediately appears.

```
   :     A     ::    B    ::    C    :
1       42.00
2        7.00
3       27.80
4      100.09
5    ---------
6      176.89
```

Now, you could move the cursor over any of the cells A1 to A4, type a different number from that in the cell and immediately see the corrected total. A spreadsheet will support all arithmetic operations that are supported by BASIC.

Those numbers look sort of arbitrary. We can add labels to add meaning.

```
 :         A               ::     B     ::     C     :
1 PAID TO:                    CHECK #      AMOUNT
2 WATER WORKS                  2145.00        42.00
3 NEWSPAPER                    2146.00         7.00
4 TELEPHONE                    2147.00        27.80
5 RADIO SHACK                  2148.00       100.09
6                                          ---------
7 TOTAL                                      176.89
```

First, I inserted two new columns to the left of old Column A. This moved the data and total into Column C. It did not need to be retyped. Next, Column A was widened to 18 characters. Within some limits, column widths can be adjusted as you choose. Next, a new row was added at the top and the labels which are strings of characters were typed into Row 1 cells and cells A2-A5. The check numbers in Column B were entered as numbers and appear in the default format having two decimal places justified to the right. In the default case, labels justify to the left.

In some ways, our spreadsheet just does not look as neat as it could. The decimal point and zeros on the check numbers aren't right. Check numbers should be integers and it would be nice for them to line up under the Check # heading, so let's retype them as labels. Also the Amount heading would be nicer if it were right-justified over the column of numbers. That is easily done by formatting cell A3 to right-justify the text.

Here is the final result.

```
 :         A               ::     B     ::     C     :
1 PAID TO:                   CHECK #          AMOUNT
2 WATER WORKS                2145              42.00
3 NEWSPAPER                  2146               7.00
4 TELEPHONE                  2147              27.80
5 RADIO SHACK                2148             100.09
6                                          ---------
7 TOTAL                                       176.89
```

To current spreadsheet users, this is a trivial example. Many non-spreadsheet owners have only the foggiest notions of what a spreadsheet is. Perhaps even this simple example will demonstrate how powerful the simplest operations can be.

Now I have talked grandly about typing in numbers and labels, inserting columns and rows, putting a formula into a cell and reformatting cells as if it were as easy as buttering bread; it is, almost. However, there are a couple of different approaches used by different spreadsheets that bear discussion, since this may influence which spreadsheet you buy.

First, there is the *VisiCalc* method that is also used by *VIP Calc*, *DynaCalc* and *Lotus 1-2-3* on IBM PC-type machines. Here, the spreadsheet is normally in the "entry" mode and is looking for a number or a label. If the first character typed is a number, it expects the rest of the entry will be a value. If the first character is a letter or some other non-numeric character, with a few exceptions, it assumes the remainder of the entry will be a label. How, then, did we enter a check number as a label? Simply tell the program a label is coming by typing a single quote, SHIFT-7 and then whatever characters we want in the label. In this case, it was a string of numbers.

To enter a formula or expression, one of several operators must appear as the first character. We really need to remember two, '+' and '@.' In the example above, +A2+A3+A4+A5 enters the formula that totals the amounts. However, this is a messy and memory-wasting way to tell the program to add if 20 or 30 cells in a column or row are involved. All spreadsheets provide a selection of arithmetic functions. A basic one is SUM(X . . . Y) that will add all values in cells 'X' to 'Y.'

In a *VisiCalc*-type sheet, the @ symbol as the first character clues the program that an expression using a function is coming next. Typing @SUM(A2 . . . A5) does the same job as +A2+A3+A4+A5.

We must be able to escape the entry mode to do such basic things as save or print our spreadsheet, as well as do all the other neat things like changing cell formats, adjusting column widths and many more. Striking the slash key will call up the main program menu.

In a memory-eating, 80-column display program such as *Lotus 1-2-3*, the menu is displayed as a series of single words at the top of the screen. The words are generally selected so each starts with a different letter and the user can select the desired function or submenu by typing a single letter. Submenus work the same way. In *DynaCalc*, only the single letters are displayed. In both cases, typing a question mark will get one or more Help screens that elaborate on the choices.

Up to this point, we see a number of programs working in a standard manner. Now, that uniform approach starts to break down. *DynaCalc* maintains close similarity to *VisiCalc* in that the examples in the book *VisiCalc Applications* (Radio Shack Cat. No. 62-1051) work with very few changes. You may never need an "Accounts Receivable" spreadsheet, but that example is worth trying in *DynaCalc* or *VIP Calc* just to learn a few ways to manipulate data and expressions.

Lotus 1-2-3 has so many added features that there are submenus up to four deep. Here the problem is learning the new things more than unlearning what you might know from *VisiCalc*, *DynaCalc* or *VIP Calc*.

*Elite*Calc* exemplifies the alternate approach where the program is normally in the command mode and ready to accept single letter commands. The program must be told that a label is coming with a double quotation (") that applies to a single cell entry, or a 'T' that puts the program into Text Lock so it accepts only text until the BREAK key is pressed. Any number signals that a value is coming, while an equal sign indicates "here comes an expression or formula."

Seemingly, it should be a bit easier to learn *DynaCalc* or *VIP Calc* at home and then pick up *Lotus 1-2-3* at the office than to start with *Elite*Calc* and then learn *Lotus*. Actually, the basic concepts in spreadsheets are what is important and the command implementation differences are fairly minor.

As with any programming language, the documentation provided cannot begin to describe or even imply all the ways each of the functions can be used. In some respects, the bigger the book, the more obscure and confusing it becomes. The Help screens in *Lotus* are much better at getting to the heart of the user's basic needs than is the documentation.

In subsequent months, we will continue to introduce concepts and uses to help you understand and use a spreadsheet. However, hang onto the documentation for your spreadsheet and become familiar with it so you can quickly refer to the exact syntax or usage that your sheet requires.

If you are considering which spreadsheet or sheets to buy, remember that *Lotus 1-2-3* and *VisiCalc* are not available for the CoCo. The others mentioned are available and are well-described in advertisements in THE RAINBOW. These ads may be referenced through the "Advertiser's Index" in the back of the magazine by the manufacturer's names and not by the program names. *DynaCalc* is published by Computer Systems Center, *Elite*Calc* by Elite Software and *VIP Calc* by VIP Technologies, formerly Softlaw Corp. The versions of *Spectaculator* may be seen and tried at any Radio Shack Computer Center and at many Radio Shack stores.

GRAPHICS

16K
ECB

Different Sights and Sounds With Graphics Bazaar

By Paul Vernon Miller

Here's a program in which you have to do absolutely nothing except type *RUN* and press ENTER! *Graphics Bazaar* is a BASIC program requiring a 16K Extended CoCo. It consists of three graphics screens, each running for five cycles and changing according to the value in the *FOR . . . NEXT* loop in Line 5.

The basic premise running throughout the whole program is the switching of the on screen design between different *PMODEs* and *SCREENs* while playing a series of sounds (lines 13, 24 through 26, and 31 through 33). Each of the three screens uses the 'pattern *POKE*' at one time or another, which creates some interesting effects when using this switching routine. The program uses a BREAK key disable (Line 3) and a speedup *POKE* (Line 2) to help the sound and graphics flow more smoothly. Most of the sounds in the program are contained in string statements (lines 34 through 45). These sounds were contrived through a great deal of experimentation with the *PLAY* command. Some of these sound strings use the effective volume controls, V+; and V-;, to increase or decrease the volume.

Screen one can start from either side of the screen, depending on what happens in the random statement in Line 6. Screen one (basically Line 9) alternates between *PMODEs 1, 3,* and *4,* creating a series of mirages and bright flashes. It also employs a 'Box Subroutine', which does nothing more than add a 'B' to both *LINE* statements and uses the 'pattern *POKE*' to paint around these boxes, creating bright colors in *PMODE4*.

The idea behind the circular line patterns in screen two and three was borrowed from the ECB manual. A little knowledge of trigonometry is required to understand the 'hows' and 'whys' of this technique (lines 18 and 29 through 30), but even if you don't understand the reason behind it, you can still experiment to gain the desired results.

The program will continually renew the global loop in Line 5 and run forever, so just push the Reset button to exit the program. I hope you enjoy *Graphics Bazaar*. Happy computing.

9	251
18	28
24	28
36	72
40	183
END	255

The listing:

```
0 ' GRAPHICS BAZAAR
1 ' TITLE SCREEN
2 POKE65495,0:CLS:GOSUB35:DIMR(1
6):K=178:PRINT@101,"GRAPHICS":PR
INT@169,"BAZAAR":PRINT@239,"BY":
PRINT@306,"PAUL":PRINT@373,"MILL
ER":PLAY"T255L255V104":FORX=1TO3
0:PLAY"V+;ADBEADBE":NEXTX:FORX=1
TO30:PLAY"V-;GFGFEFEF":NEXT
3 PRINT:FORX=1TO15:PLAY"V+;O4ABC
DEFGFEDCBAV+;CDEABFCD":PRINT:NEX
TX:POKE248,50:POKE249,98:POKE250
,28:POKE251,175:POKE252,126:POKE
253,173:POKE254,165:POKE410,126:
POKE411,0:POKE412,248
4 ' SCREEN 1
5 FORQ=1TO5:X=RND(-TIMER):PLAY"V
4":PMODE4,1:PCLS:SCREEN1,1:IFQ=2
```

```
ORQ=5THENM=3ELSEIFQ=4THENM=1ELSE
M=4
6 R=RND(Ø):IFR>.5THENX=4:Y=196:Z
=4ELSEX=196:Y=4:Z=-4
7 PLAYA$:IFQ=3ORQ=5THEN11
8 FORA=X TO Y STEPZ:IFA=1ØØTHENP
LAYA$
9 PMODEM,1:SCREEN1,Ø:LINE(A-1,19
6-A)-(A+56,A-4),PSET:PMODE4,1:SC
REEN1,1:LINE(A-1,A-4)-(A+56,196-
A),PSET:NEXTA:PLAYA$:IFQ=1ORQ=4T
HENV=RND(255):W=RND(255):POKEK,V
:PAINT(15Ø,15Ø),,1:POKEK,W:PAINT
(15Ø,5),,1:GOTO13ELSE13
1Ø ' BOX SUBROUTINE
11 FORA=X TO Y STEPZ:IFA=1ØØTHEN
PLAYA$
12 PMODEM,1:SCREEN1,Ø:LINE(A-1,1
96-A)-(A+56,A-4),PSET,B:PMODE4,1
:SCREEN1,1:LINE(A-1,A-4)-(A+56,1
96-A),PSET,B:NEXTA:PLAYA$:FORX=1
TO2ØØ:V=RND(255):POKEK,V:PAINT(V
,X),,1:NEXTX
13 PMODE4,1:PLAYI$:PLAYB$:PLAYC$
:PLAYD$:PLAYE$:PMODE3,1:SCREEN1,
Ø:PLAYF$:PLAYG$:PLAYH$:PLAYI$:PL
AYJ$:PLAY"V31":FORX=1TO1Ø:PMODE3
,1:SCREEN1,1:PLAYM$:PMODE4,1:SCR
EEN1,1:PLAYK$:NEXTX
14 ' SCREEN 2
15 IFQ=5THENPMODE4,1:PCLS5:SCREE
N1,1:COLOR8ELSEPMODE4,1:PCLS:SCR
EEN1,1
16 LINE(Ø,96)-(255,96),PSET:LINE
(128,Ø)-(128,192),PSET:Y=Ø:FORX=
128TO255STEP4:LINE(128,Y)-(X,96)
,PSET:Y=Y+3:NEXTX
17 X=Ø:FORY=96TO192STEP3:LINE(X,
96)-(128,Y),PSET:X=X+4:NEXTY:Y=9
6:FORX=1TO128STEP4:LINE(X,96)-(1
28,Y),PSET:Y=Y-3:NEXTX:Y=192:FOR
X=128TO255STEP4:LINE(X,96)-(128,
Y),PSET:Y=Y-3:NEXTX:PLAY"V4O3"
18 FORT=3ØTO-25STEP-5:A=(2*3.141
5)*T/6Ø:LINE(5Ø,5Ø)-(25*SIN(A)+5
Ø,25*COS(A)+51),PSET:LINE(2Ø5,5Ø
)-(25*SIN(A)+2Ø5,25*COS(A)+51),P
SET:LINE(5Ø,142)-(25*SIN(A)+5Ø,2
5*COS(A)+142),PSET:LINE(2Ø5,142)
-(25*SIN(A)+2Ø5,25*COS(A)+142),P
SET:PLAY"V+;V+;AFAF":NEXTT
19 ' PAINT ROUTINE
2Ø IFQ=2THEN23ELSEIFQ=3ORQ=5THEN
24
21 CIRCLE(5Ø,5Ø),26:CIRCLE(2Ø5,5
Ø),26:CIRCLE(5Ø,142),26:CIRCLE(2
Ø5,142),26:FORX=1TO16:R(X)=RND(2
55):NEXTX:POKEK,R(1):PAINT(55,35
),,1:POKEK,R(2):PAINT(6Ø,65),,1:
POKEK,R(3):PAINT(31,55),,1:POKEK
,R(4):PAINT(21Ø,4Ø),,1:POKEK,R(5
)
22 PAINT(215,65),,1:POKEK,R(6):P
AINT(186,55),,1:POKEK,R(7):PAINT
(55,125),,1:POKEK,R(8):PAINT(6Ø,
157),,1:POKEK,R(9):PAINT(31,147)
,,1:POKEK,R(1Ø):PAINT(215,157),,
1:POKEK,R(11):PAINT(185,147),,1:
POKEK,R(12):PAINT(2Ø6,132),,1
23 POKEK,R(13):PAINT(1,12Ø),,1:P
OKEK,R(14):PAINT(1,1),,1:POKEK,R
(15):PAINT(255,1),,1:POKEK,R(16)
:PAINT(255,12Ø),,1:IFQ=4THENFORX
=1TO255STEP3:V=RND(255):POKEK,V:
PAINT(X,1),,1:V=RND(255):POKEK,V
:PAINT(X,19Ø),,1:PAINT(X,8Ø),,1:
PAINT(X,11Ø),,1:NEXTX
24 PLAY"V31":FORX=1TO11:R=RND(Ø)
:IFR>.5THENR=1ELSER=Ø
25 S=RND(Ø):IFS<.5THENS=1ELSES=Ø
26 PMODE 3,1:SCREEN 1,R:PLAYK$:P
MODE4,1:SCREEN1,S:PLAYL$:NEXTX
27 ' SCREEN 3
28 IFQ=4THENPMODE4,1:PCLS5:SCREE
N1,1:COLOR8ELSEPMODE4,1:PCLS:SCR
EEN1,1
29 FORT=3ØTO-27STEP-3:A=(2*3.141
5)*T/6Ø:LINE(128,96)-(75*SIN(A)+
128,75*COS(A)+96),PSET:LINE(25,2
5)-(75*SIN(A)+128,75*COS(A)+96),
PSET:LINE(23Ø,167)-(75*SIN(A)+12
8,75*COS(A)+96),PSET:LINE(23Ø,25
)-(75*SIN(A)+128,75*COS(A)+96),P
SET
3Ø LINE(25,167)-(75*SIN(A)+128,7
5*COS(A)+96),PSET:NEXTT:IFQ<3ORQ
=4THENV=RND(255):POKEK,V:PAINT(1
,1),,1ELSEPOKEK,25Ø:PAINT(1,1),,
1:FORX=1TO3ØØ:V=RND(255):Y=RND(2
55):Z=RND(191):POKEK,V:PAINT(Y,Z
),,1:NEXTX
31 PLAY"V11":FORX=1TO2Ø:R=RND(Ø)
:IFR>.5THENR=1ELSER=Ø
32 S=RND(Ø):IFS<.5THENS=1ELSES=Ø
33 PMODE3,1:SCREEN1,R:PLAYN$:PMO
DE4,1:SCREEN1,S:PLAYO$:NEXTX:NEX
TQ:GOTO5
34 ' SOUND STRINGS
35 A$="V+;V+;V+;V+;V+;V+;V+;V+;V
+;O1ABFFABCDEFGFEGEFEABGABFAAO2A
BCDEFGABADEFABCDEEBFGAO3ABDEFGAB
DEFGABDEFGABAEFGO4ABDEEFGABFECDE
ABGABEEBAAA"
36 B$="O5GFEDCBABCDEFGFEDCBAO4BC
DEFGFEDCBABCDEFGO3GFEDCBABCDEFGF
EDCBAO2ABCDEFGFEDBCAO1BCDEFGFEDC
BABCDEFGFEDCBAO2BCDEFGFEDCBABCDE
FGO3FDECBABCDEFGO4EDCBABCDEFGO5F
EDCBABCDEFGFEDCBABCDEFFFGGG"
37 C$="O4V1ØAAAAAAFFFFFFFAAAAAAF
```

```
FFFFFV15O2AAAAAAAFFFFFFFV2ØAAAAA
AAFFFFFFGABACAADAAFAAV31O1AAAAAB
AAACAAADAAAFAAAFFFFFAAAAACAAACAA
FAAFFFFFACAAAAFAAFFFFFV15FFFFFAA
AAAAFFFFFGGGGGGAAAAAV6FFFFFAAAAA
GGGGGFFFFFAAAAAV3FFFFAAAAGGG"
38 D$="V3O1GGGGGFFAAAAV6AAAAAFFF
GABCDABDDDEEEAABBAAFFV1ØAABBBCAA
ACDDDFFFFGGGGGGGGV15AAAAFFFFFAAA
AAGGGGGDDDFFFBBBBV31ABCDEFGABCDE
FGABCDEFGO2AAAAFFFBBBBBGGGGAAAAA
BBBBBFFFFFFFAAAAAAAGGGGGGO3ABCDE
FGAAAABBBBBFFFFAAAAAGGGGGAAAAAGG
GGGBBBBBBAAAAAAAFFFAABB"
39 E$="O3GGGGGAAAABBFFFFGGGAAAFF
FFBBBBBGGGGGGGAAAAABBBDDDGGGGGAA
AAFFFFFBBBGGGGO4AAAAAAAABBCCCCCC
CDDDDDDEEEEEEFFFFFGGGGABCDFFFFFF
AAAAGGGGGGGAAAADDAFADDDDDGGGGGGG
GG"
4Ø F$="AAAAAAADDDDDADBAAAAGGGGGG
GAAAAAAAFFFFFFBBBBBBGGGGGGO3AAAA
AFFFFFFBBBGGGGDDDDGGGGEEEEEAAAAA
AGGGGGGFFFFFFFBBBBBBAAAAACCCCCCF
FFFFFFBBBBGGGGGGO4FFFFFFFGGGGGGA
AAAFFFBBBGGGGGGGGGAAAAGGAAAGGAABF
BBBO2V15AAAAAAFFFAAAGGGGBBBFFFAA
AABBBGGGGFFFFFFAAGGAAB"
41 G$="BBFFFAAAFFFGGAABBBGGGFFFA
AGGGAAADDDDV31O1AABBGGGFFAABBFFF
CCCGGGAAABBGGGGBBBAAFFFAAAGGGBBB
AAAAGGGCCFFFAAAFGFGFGFGFGFABABAB
ABACDDCDCDCDCDDCDAFFFFFAAAV2ØAAA
FFFFGGGGGAAABBBBFFFGGGABACDDABCD
FDABDFDABDFEFFFV15ABCDGGGAAABBFF
FAABBBGGGAAAFFFFBBBGC"
42 H$="V1ØABCDGGAGAGBBBBBV5BBBCG
BCBGV3ABABABAB"
43 I$="O1V31AAAABABABABV15BBBBAB
ABABV31O2AAAAAAABABABABABABV15BBB
BABABABABABV31O3BBBBBABABABABABA
BAV15AAAAAAAABABABABABABABV31O4AA
AAAABABABABABV15BBBBBABABABABABA
BAV31FFFFFFFFFAFAFAFAFAFAFAFAFAF
AFAFAV15GGGGGGABABABABAGAGAGAGV1
ØFFFFFABABABBABAFAFAFAFAFA"
44 J$="AFAFAV5DDDDDDADADADAV3BBB
BBABABABABV2ABABABABABABV1GGGGGG
GGGGGGGGGAAGAGAGAGAGAGAGAGAGAGAG
AGV31O3ABCDEFGABGGAAGAGAGAGAV15O
2BABABABCDCDCABCDCBCDV1ØO1ABABAB
CGCGCGCGV5ABABABV2GAGAGAGAV31O4A
BCDEFGABCDEFGABCDEFGABCDEFG"
45 K$="V-;O4GFEDCBABCDEFGFEDCBAB
CDEFGABCDEFGABCDEFG":L$="V-;O5FE
DCBABCDEFGABCDEFGABCDEFGABCDEFAB
CDEF":M$="V-;V-;O3ABABCFCFABABCF
CFO2ABCDEFGFEGFE":N$="V+;O3AAFAA
FAAFAAFGBBGBB":O$="O4AABCDEABFCD
CDEABFCD":RETURN:PCLEAR8
```

Color BASIC ROM Switcher

By Clay Howe

Early models of the CoCo, the 'C' and 'D' boards, were originally marketed with Version 1.0 of the Color BASIC ROM. When the 'E' board came on the scene, it was followed by the Version 1.1. This was an improvement for all concerned, as it allowed printer operations that were not possible with the 1.0 version, as well as modifying the way RAM was to be handled.

Then came the next board, the one with no real official designation, but referred to as the "NC," 'F,' or "285" board. It did not bring with it another Color BASIC ROM version, but within a matter of months Version 1.2 was introduced in the new CoCo 2 model. This time, a problem did appear for those who were using software that was written for the previous models – not all software, but quite a large segment of the more popular machine language word processing, database and utility programs. It seems that the keyboard

(Clay Howe is a self-taught electronic and computer hobbyist. He has developed several hardware modifications for both the CoCo and radio control systems. When he can get away from his CoCo, and weather permits, he enjoys flying RC sailplanes.)

was being polled in a different way for one thing, plus other changes that created other problems. We're not concerned with the cause of the problems, but rather with a cure.

As an 'F' board owner, which had the 1.1 Color BASIC, I had a fairly large group of utilities, on disk and cassette, that would not work correctly with the CoCo 2 that was purchased shortly after they were introduced. Fortunately, Tony DiStefano had an article in the December 1983 issue of RAINBOW that showed how to overcome the problems by simply "piggybacking" a 1.2 ROM with a 1.1 Version ROM, with a couple of the pins wired through a switch, so either could be selected.

This construction article is a modification, and I feel an improvement, to Tony's switcher. The reason I say improvement is that the ROMs do not require soldering to each other. In fact, neither are soldered to anything, and they may be simply plugged into sockets to provide the switchable setup. There are seven illustrations that accompany this text, and they will be referred to by number as we go through the construction process. It will be best to read the complete article, and study the illustrations, before beginning actual assembly of the switching unit.

To the best of my knowledge, the ROM Switcher will work with any board version of the CoCo. In the case of the newest "Korean" version, it will only be possible to use the switcher if the Color BASIC ROM is de-soldered from the board, and a socket put in its place.

Construction

The assembly steps that follow assume that you have some knowledge of hardware construction practices, although it should be no problem for a novice to build the ROM Switcher. If you have doubts concerning the assembly, have a "hacker" friend construct it with you — hands on experience is the best teacher, and will help prepare you for your next hardware project. The soldering should be done with a low heat iron, one with a 25 watt to 35 watt heating element is best. Don't try the project with a soldering "gun" — most of these are rated at 80 to 100 watts, and would be too hot and clumsy to work with.

Figure 1 shows a 24-pin dual inline header. You will notice that the only modifications required to the header are the removal of the top portions of the connectors at positions 20 and 24. The best way I found to remove the tops of the connectors is to insert the header into a spare 24-pin, wire-wrap IC socket, then use small nippers, like Radio Shack Cat. No. 64-1833, to snip off the tops of the two connectors. Be sure to leave a portion that looks like those in the illustration.

Figure 2 is a top view of the header. To make it possible to connect the switch to the computer's five volt bus, and the Chip Enable signal, two small wires must be soldered to the portions of the connectors left at the 20- and 24-pin positions. The wires may be fashioned from resistor lead cut-offs, bent as per the illustration, then soldered in place. Keep the header in the spare IC socket while doing this and future soldering. After the wires are soldered in place, put a small glob of fast-curing epoxy on the ends as per the figure. *Do not let the ends of the two wires touch!*

Next, you should prepare four pieces of printed circuit board as shown in Figure 3. These may be cut from any board that has the pattern of parallel holes joined by a copper trace. There are several boards that are useable in the Radio Shack selection of PC boards. (Note: The four little strips of PC board will be referred to as "2-strips" in following construction steps. This is a little easier than saying "take one of the pieces of printed circuit board that has 12 sets of parallel circuit traces.")

Referring to figures 4, 5, and 6, you will see how the 2-strips are soldered to the 24-pin IC sockets. Be sure that the IC sockets are fully inserted into the strips before soldering, and that they are straight, not sloping away from the socket. Solder the four 2-strips to the IC sockets using only enough solder to make a good connection. After soldering, trim the remaining portion of the socket legs *of one socket only*, as close as possible to the underside of the 2-strip. Trim the legs on the other socket at positions 20 and 24 *only*, as in Figure 4A.

Again referring to Figure 4A, solder the socket/2-strip with the 20- and 24-position legs removed, to the top of the header. 22 pins from the socket/2-strip should be soldered, and when finished it should look like figure 4B on the side where the two legs were removed. The legs should fit right into the depressions in the header pins for soldering. The header should still be in the extra socket to keep the pins straight while the soldering is being done. The header may be fairly soft plastic, and the extra socket will act as a heat sink.

For the next step, you will need 22 pieces of small diameter wire about ¾ inch long. A good type wire to use is the cut-offs from ¼-watt resistors. If you're a hardware hacker, you probably have some from a previous project. If not, Radio Shack has spools of small bus wire. The 24 gauge (Cat. No. 278-1341) will work fine. These wires will be referred to as "standoffs" in following construction steps.

With the socket/header assembly still in the heat-sink (spare socket), solder the standoffs at all positions except 20 and 24. The standoffs are soldered into the outer holes in the 2-strip that is attached to the socket/header. Trim all standoffs as close as possible to the underside of the 2-strip. Figure 5 and the lower part of Figure 6 show where the standoffs are to be placed.

Figure 6 shows ROM 'A' and ROM 'B' plugged into the upper and lower sockets. *Do not* have ROMs plugged into the sockets during construction.

Referring to figures 5 and 6, place a spacer of some sort between the upper and lower sockets to provide a spacing of no less than .4 inch. A small piece of soft wood like balsa may be easily carved to serve this purpose. Slip the standoffs that are protruding up from the socket/header assembly into the respective holes in the upper 2-strips. Again, there should be nothing at positions 20 and 24. When all are properly aligned, solder the standoffs to the bottom side of the top 2-strips, then trim the standoff wires flush with the top side of the 2-strips.

The switcher assembly should look like Figure 5, except that it will also have the header soldered to the bottom socket legs. Hold the switcher assembly up and look at the end. It will look like Figure 6.

The next step is very important. Take a magnifying glass and look the switcher over very carefully. Make sure all solder joints are well-done and none of the standoffs are touching. The first prototype of the switcher used insulated wire for the standoffs, but others, in use for several months with the bare standoff wire, have created no problems. It may be a good idea to use a flux remover at this time to clean the complete assembly. The spray-can flux remover from Radio Shack (Cat. No. 64-2324) will do a nice job. Stand the switcher on end on some paper towels, and wash it down with the flux remover.

The switch wiring is next. Figure 7 shows where the wires are to be attached. Note that one end pair will go to the upper 2-strip holes 20 and 24, the other end pair goes to the lower 2-strip holes 20 and 24, and the center pair goes to the two wires soldered and epoxied to the header. Be sure that all three wires on one side of the three pair of switch connections go to position 20 and the others to 24. *Do not accidentally cross the wires.*

The length of the switch wires depends on which model CoCo you have, and where you want to mount the switch. Allow enough wire to do the installation, but don't have unneeded wire looped around inside the CoCo. On the CoCo 2, a good place to mount the switch is the cavity under the ROM pack port. It will be easy to get to for switching, but out of the way to prevent accidents. Almost any small insulated wires may be used from the switch to the switcher. A good selection would be colored flat multi-conductor, with a different color to each connection, and all wires kept as a strip with about 1 inch of each end separated for soldering.

Once the switcher is assembled, and you begin the switch installation, be very careful while working inside the CoCo. When the switch is mounted, the Color BASIC ROMs may be inserted into the two sockets. It makes no real difference which one goes where, although I keep the most used ROM in the bottom socket. The top ROM may then be easily removed for use in one of my other CoCos if desired. The ROMs may be plugged into the sockets before or after the switcher is plugged into the board. It is easier to insert them with the switcher out, as it may be positioned away from the board clutter. *Be certain the ROMs are inserted with the notch at the correct end.* If they are plugged in backwards, it will damage the ROMs, and possibly the CoCo.

If the switch should get bumped from one position to the other, it should not hurt anything physically, but the CPU and SAM chips will "get confused," and whatever is in RAM may be either lost, or unuseable.

Although you may wish to turn the CoCo off, then flip the switch, then turn the CoCo back on again to go from one ROM version to another, the method I have used for almost a year, with no unusual results is:

1) Hold the Reset button in
2) Flip the switch
3) Release the Reset button
4) Type POKE113,0:EXEC40999 ENTER

The Color BASIC ROM will be switched from one to the other. This may be verified by typing: EXEC 41175 ENTER before and after switching to get the Color BASIC sign-on message on the screen.

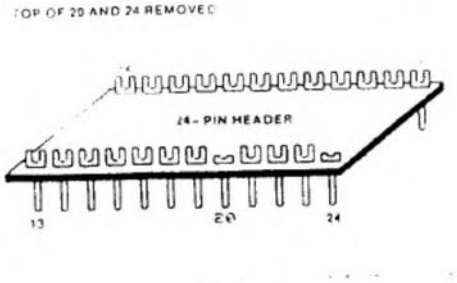


Figure 1

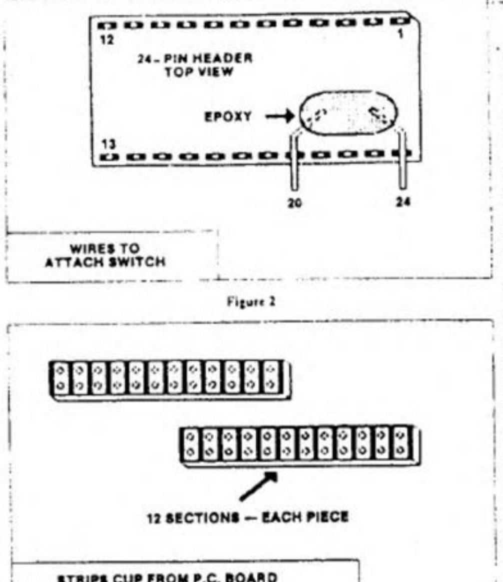


Figure 2

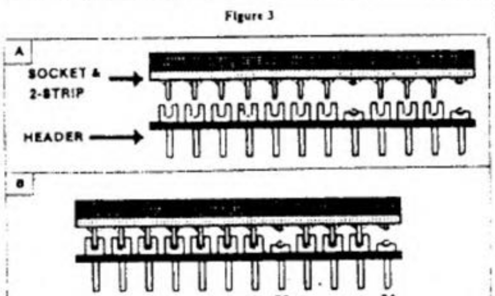


Figure 3

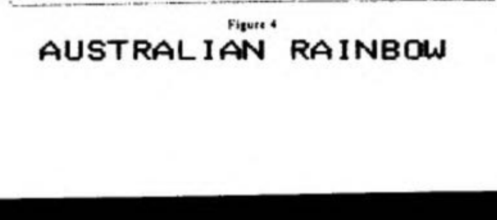


Figure 4

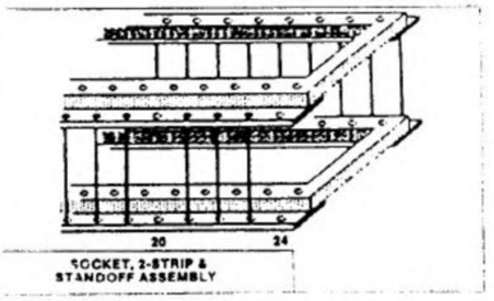


Figure 5

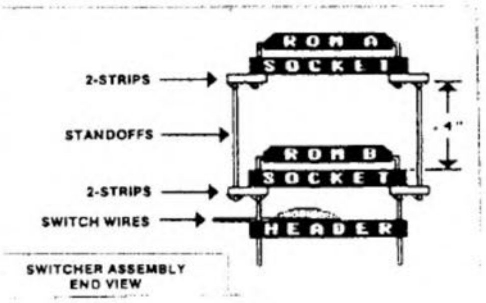


Figure 6

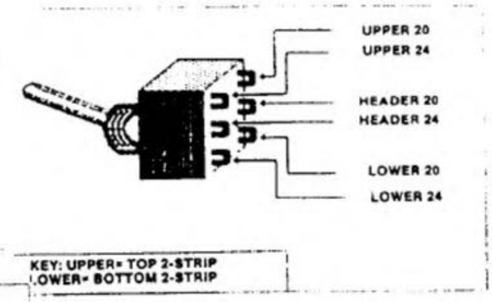


Figure 7

Maybe the name of this unit should have been ROM Switcher, rather than Color BASIC ROM Switcher, as its use is not restricted to use with the Color BASIC ROMs. It may also be used to switch Extended BASIC ROMs 1.0 and 1.2, or a modified version of Color BASIC, or Extended BASIC, that has been programmed into a 24-pin EPROM such as the Motorola 68766. To go to extremes, if the disk controller was used with a short flat ribbon cable adapter, the ROM Switcher could be used to switch 1.0 and 1.1 versions of Disk BASIC ROMs. The cover would need to be left off the controller to do this, so the ribbon cable adapter would position the controller outside the ROM Pak port.

Parts List

1	24-Pin Header Plug		(JE)
2	24-Pin Low Profile Socket		(JE)
1	DPDT Micro-Mini Switch	#275-626	(RS)
Misc. — wire, solder, epoxy, PC board.			

The 24-pin wire wrap socket, which is used as a heat sink during construction, is also available from (JE) Jameco Electronics.

The address of Jameco Electronics is: 1355 Shoreway Road, Belmont, CA 94020, phone (415) 592-8097.

TURN OF THE SCREW

Adding A Numeric Keypad To Your CoCo

By Tony DiStefano

Just the other day, I walked into my local electronics store and saw they had recently opened a warehouse bargain section in the rear of the store. I immediately went in and started to browse. This place is a gold mine of old parts and nifty gadgets. Some items I found were individual keyboard switches. They were surplus from who knows where, were of good quality and very inexpensive.

If you recall, a while back I did an article on adding function keys to your keyboard. I explained that in the eight by seven matrix that makes up the CoCo keyboard, there are four free areas and how to add switches. Ever since then, I get requests to write an article on how to add a numeric keypad to the Color Computer.

I looked into it and found that it would be quite easy to wire one up. Very few components would be needed and it would not cost too much. About the only thing that was keeping me from doing such an article was the actual keypad switches — there were none to be had around here. I could have used regular switches; after all, that is all that makes up the keyboard part of a CoCo, but it would not look like a nice keypad. Therefore, I put the numeric keypad article on the back burner.

Back at the electronics store, I picked up about 20 keyboard switches along with an assortment of keycaps. In no time at all, I had myself a nice numeric keypad. It was then that I decided I should submit "Adding a Numeric Keypad" to THE RAINBOW.

It is still up to you to find your own keyboard switches and keycaps. You will also have to build your own keypad case since the size and shape of your case will depend on what kind of keyboard switches you get and how many you decide to add (I'll explain later). In other words, all of the cosmetic side of this project will be left up to you. I will supply the schematic, parts list and method of putting together a numeric keypad.

For the benefit of those of you who do not know how the CoCo keyboard works, a little background information may help you with this project.

The keyboard has 53 keys. A PIA (Peripheral Interface Adapter) is used to scan these switches (keys). The eight keyboard columns are attached to the 'B' side of the PIA. These eight lines are programmed to be outputs.

The seven keyboard rows are attached to the 'A' side of the PIA. These seven PIA lines are programmed to be inputs. To read the keyboard, only one column is enabled by writing a zero in the bit that corresponds to that column and by writing ones in all the other bits. If a key has been pressed in that column, one of the input lines will be a zero and the key location will correspond to the bit that is low. By scanning each column in the keyboard, all of the keys may be checked.

My idea is (if I run some wires in parallel to that of the keyboard lines) to take those wires and run them to a connector, and finally to some keyboard-like switches. Since the switches would be in parallel, this would give you the choice of pressing, for example, the number '1' on the main keyboard or on the numeric keypad You could enter all your numeric data from the keypad.

But also, I included a few more keys that would be handy: the plus key(+), the minus key (-), the multiplication key (*) and the division key (/). Then there is the decimal point (.) and the ENTER key (CR).

The schematic in Figure 1 shows how to wire the above keys to the main keyboard connector. I chose those keys because it suited my needs. There is no reason why you could not change them to fit your needs, or for that matter, you can add a complete second keyboard. All you have to do is get the right wiring.

Figure 2 shows the complete wiring diagram of the CoCo keyboard. All versions of the CoCo or CoCo 2 keyboards are the same, even though the keyboards look different. That is one of the few things that did not change in the ever-changing CoCo.

Now, the next thing I didn't like was that if you wanted to enter a multiplication sign or a plus sign, you had to press the SHIFT key. I had two choices: 1) include the SHIFT key and press it every time you wanted these functions, or 2) make a small electronic circuit to automatically press the SHIFT key when you hit these keys. I elected to do the latter of the two.

The switch that is normally used for the keyboard is an SPST (Single Pole Single Throw) momentary on. To automatically hit the SHIFT key and the key you want shifted would require a DPST (Double Pole Single Throw) momentary on. That way, both circuits would connect and we would get the shifted function, if any, of that key. That is a good way to do things, but I could not find a DPST switch in the shape of a keyboard key. So, I decided to make an equivalent transistorized circuit.

Look at the transistor part of the circuit diagram in Figure 1. Each transistor acts like a switch. The 10K bias resistor makes sure the transistor stays off when not being accessed. That is the equivalent of the switch being off or open (no key pressed).

The emitter of the transistor is connected to the output side (Port B) of the PIA. If you recall, all the 'B' lines are programmed to be outputs and are all high or five volts. Only one line at a time goes low, so when the line that has the emitter of the transistor connected to it goes low, the transistor's emitter is effectively connected to ground.

The 1K base resistor is used to limit the base current, but enough to turn the transistor on. The switch in this circuit is connected to five volts. When the switch is on (key pressed), current flows through the resistor, therefore, turning the transistor on. That makes the collector of the transistor ground potential. In turn, the ground potential on the collector grounds one of the corresponding input pins on the 'A' Port of the PIA. To the computer, this translates into a pressed key.

Now, take the plus key for example. The Port 'B' output that connects to this key is PB3 (keyboard connector #12). The input pin is PA5 (keyboard connector #7). I placed my transistor circuit on these two points as described above. Now, when I press the switch connected to the transistor, I get the semicolen (;), the unshifted plus.

I then made another transistor circuit for the SHIFT key and connected the base resistor to the same switch as the plus key. Now when I press the switch connected to the two transistors, I get the shifted plus in one key press. Nice, but this would require two transistors for every shifted key I needed. Use a simple diode to isolate the two transistors and now you only need one diode per shifted key. (I'm sure that someone will write me saying, "I found a way to do it with fewer parts," but this one works, so I'll use it.)

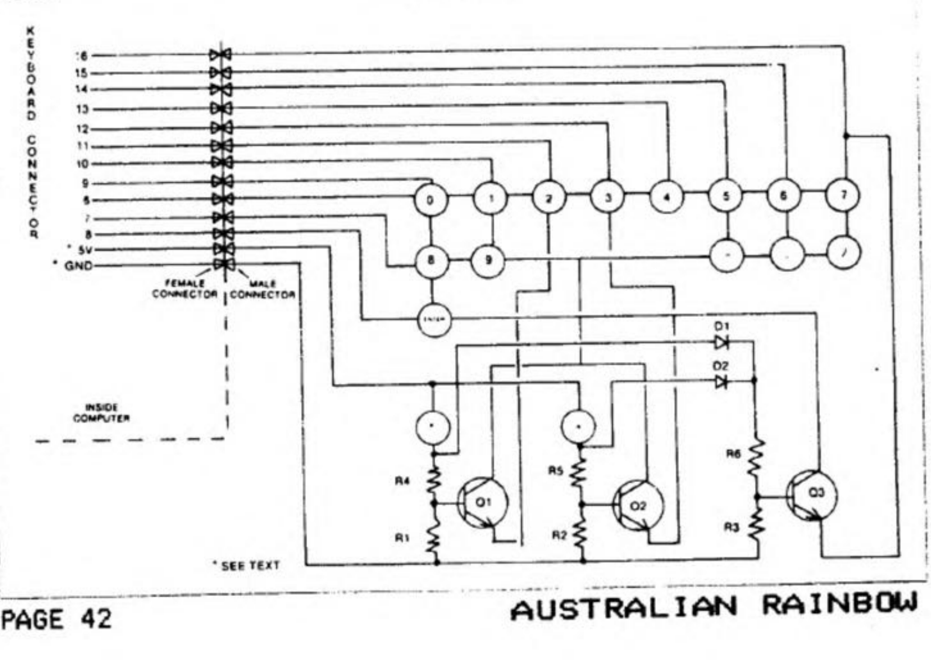
As a point of interest, this circuit can make an easy pause key. When you want to stop a listing, you press the SHIFT @ key. Well, this would make a one-key pause button.

You can really get carried away and make all of the shifted keys "one-key only." For example, "SHIFT backspace" means backspace the complete line. You can now have one key to "delete line." Another good one is the question mark (?). It is used as a short form for the PRINT statement.

The construction of this project requires a bit of doing and cutting. I'll leave that part up to you. As you can see in the photo, I used a proto-board and glued the keys onto it. You can see the transistors and resistors at the bottom. The important parts, like the theory of operation, schematic diagrams and keyboard layout, are here. There should be enough information here to get you going.

Figure 1

Keypad Schematic



Since there are many board revisions to the CoCo, there might be a problem as to where to find the right connections to the keyboard. The best way to cover all versions is to connect directly to the keyboard connector itself. It is a 16-pin connector and all of them are wired the same way, even though the connector might be different.

I suggest you solder your wires to the connector. If the connector is too close to the board and you cannot reach its soldered pins, you could always remove the board and solder to the pins from the bottom side.

Remember, when soldering from the bottom, the pin numbers are backwards. The pin numbers go from 1 to 16; it is marked which side is 1 (left) and which side is 16 (right).

Find a good spot to mount the 15-pin connector — on the side of the computer directly under the keyboard is not bad. On the left or right depends if you are left- or right-handed.

Mount the female connector to the computer. Using a short length of ribbon wire, solder all the pins needed from the keyboard connector to the 15-pin connector.

There are two more connections that go to the 15-pin connector that do not go to the keyboard connector: the ground wire and the five volt wire. There is always a question of where is the best place to connect the five volts and ground. I always look for a 1. uf decoupling capacitor. They usually connect to the right points. If you are not sure, use pin #8 on the 74LS138 for ground and pin #16 on the same chip for the five volts. That is all the wiring you have to do on the computer side; the rest is all in the keypad adapter.

Solder another short length (the length is up to you) of ribbon wire to the male 15-pin connector. Make sure all the wires match the pinout of the female side. The rest of the wiring is done on the proto-board with the keyboard switches and other parts.

The parts list matches the needs of the schematic in Figure 1. If you are adding more keys, you will have to add more parts. The connector I used has 15 pins. There are a few free ones, but if you decide to do a complete remote keyboard (or somewhere in between) you will need to move up to the 25-pin connector since the 15-pin connector is not enough.

One last thing, if you write me and expect an answer, please include a SASE or, for faster responses, you can call me on Monday nights *only* (please). My number is (514) 474-4910.

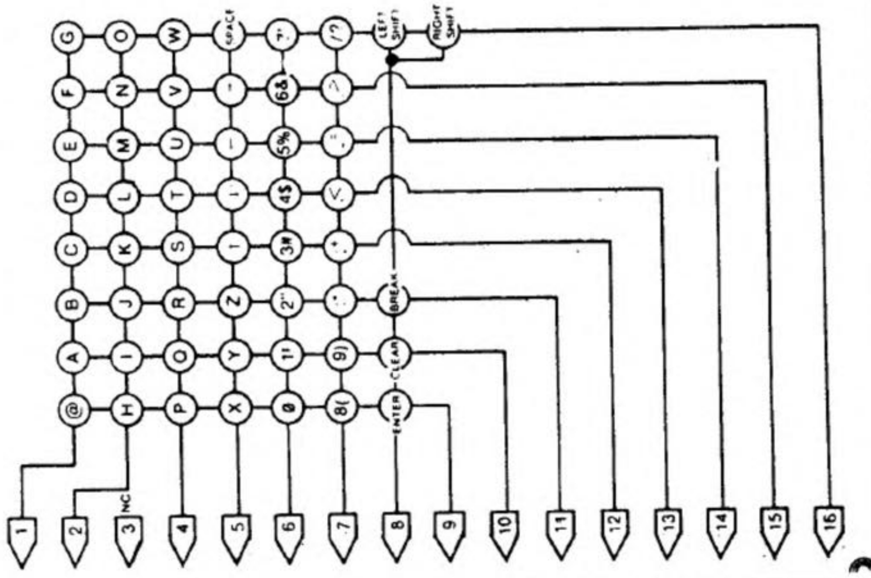
Parts List

ID	Description
Q1, 2, 3	2N3904 transistor
R1, 2, 3	10K 1/4W resistor
R4, 5, 6	1K 1/4W resistor
D1, 2	1N914 diode
C1	15 pin sub-D male
C2	15 pin sub-D female
Miscellaneous	16 (or more) conductor-ribbon wire 12 key-switches 12 key-caps proto-board
Hardware	plastic or metal case screws and mounting lugs, etc

Reference

TRS-80 Color Computer Technical Reference Manual

Figure 2 **Keyboard Wiring Diagram**



TAKING BASIC TRAINING

16K ECB

Make The Most Of Versatile EDIT

By Joseph Kolar

Aside from acutally creating a program, the most important feature of CoCo is the EDIT statement that is so vital to debugging a program. It is a feature that often confuses newcomers. At times, instead of making desired changes, imprudent use of EDIT causes new problems to pop up.

EDIT is a versatile tool that CoCo offers as a good friend to beginning programmers. At this point, you might refer to your manual for an overview of the subject at hand: EDIT. We will offer hints and methods of attack on editing a typical homemade program. Key in Listing 1, as presented, mistakes included.

When you are creating a program, try to compose your program lines in an orderly sequence. Reserve Line 0 for a REM program name line. You may want to start your program at Line 10 and proceed in steps of 10. You may want to begin at Line 100, incrementing each line "+10," reserving lines 1-99 for a title or instructions. This way, your listing will be linear. The listing will show in ascending lines the resultant program as it evolves on the screen.

In other words, a title may be shown first on the screen, followed by instructions or text and then a graphic. In the listing the title will be composed first using low numbers, then the instructions will be composed with larger line numbers. The graphic, being created last, will have the highest numbers (see Listing 1). As each line of text is created, it is assigned the next highest line number.

This is important! Look at the first display panel of Listing 1. "Proceed" is spelled incorrectly. To correct this glaring error, we must find the proper line. Since it is a straightforward program, where the line numbers become greater as you proceed through the displayed program, you can assume it is a low number near the beginning of the listing.

LIST-60 and ENTER. Conduct a search for the line to be edited. LIST-60 tells CoCo to list all the program lines from zero through 60. A quick search will pick up Line 40, which contains the error.

EDIT40 and ENTER, or if you prefer to give it your undivided attention, CLEAR EDIT40 and ENTER to get you into the edit mode.

Our object is to change the "DE" in "PROCEDE" to "ED." You could bang away at the space bar until the cursor is positioned over the offending 'D.' That is a lot of work!

You could take a guess as to the number of characters and spaces in front of it, say, 70. While in the edit mode, press '7' '0' and the space bar CoCo will advance the cursor on top of the 'O' in "HOW." You have just made an educated guess. You are close! Space until the cursor is over the 'D.' Now, press '2' because we want to tell CoCo how many characters to change, beginning with the one the cursor is covering. Press 'C' and CoCo will be ready to change them. Key in ED. With that done and no other corrections to be made, press ENTER to exit the edit mode.

You can imagine that in a very long line of text, zeroing in on the error is a matter of guessing the target location. Remember, to make any changes, you must be in the edit mode. After completing each change, you must get out of the edit mode by pressing ENTER, then check your work.

If an error is near the beginning of a pr gram line, it is quicker to get on target by advancing with the space bar.

If the error is near the end of a program line, it is simpler to press 'X' and the cursor will jump to the first free space after the end of the line. Use the left-arrow key and backspace until you are over the target; make the correction. Refer to the reference line and retype the balance of the line.

If the target is in a very short program line, it is quicker to re-key the line correctly.

Suppose you want to substitute the word "PRODUCTIVE" for "EFFICIENT" in Line 40. Hit CLEAR then type EDIT4Ø and enter. The quickest way, since it is near the end of the line, is to press 'X' to get to the end. Use the left-arrow key until the cursor is over the first 'E' of EFFICIENT, type PRODUCTIVE MANNER and press ENTER to exit the edit mode.

Look at the second line of the first panel. There's no period, so CLEAR EDIT3Ø and enter it as the very end. Press 'X,' the period and ENTER. If you forgot to get out of the edit mode and keyed in RUN and ENTERed, you got out of edit OK, but not until you added an unwanted RUN that will show up on the screen.

If the error is somewhere in the middle of a long program line, it is better to take an educated guess concerning the target location.

How do you take an educated guess? If you read the second line of the first panel, you will note that grammatically, it's wrong. "FEATURE" should be plural and "IS" should be "ARE." So, CLEAR EDIT3Ø and ENTER. Assuming that each line has about 30 characters spaces, "FEATURE" is near the beginning of the third line. The second line has about 30 characters/spaces, so the third line should begin at about the 60th character. The target letter is somewhere past the middle of the third line, about 20 units from the left margin So, add 20 to 60 and your educated guess is 80!

Press '8' '0' and space. CoCo placed the cursor over 'U' of "FEATURE." Closing in fast! Space over to get to the space after 'E.' If you overshot the target, use the left-arrow key to return to the target location.

To insert a letter, press SHIFT 'I' together to get into the insert option, then press 'S.' Get out of insert mode and also the exit mode with ENTER.

It is safer to make one correction at a time, RUN and check out the result.

Come to think of it, the second "USED" is superfluous, so hit CLEAR EDIT3Ø and ENTER. "USED" is on the third line, which begins at about the 60th character/space. The target is about 20 spaces over. Once again, the educated guess is 80. Press '8' '0' and space. Not quite! Space over until the cursor is on top of 'U.' We want to delete five spaces, ("USED" and the following space). Press '5' (the number of spaces to be zapped) and 'D' to tell CoCo to delete the five spaces. Get out of the edit mode and check it out.

Now, we will delete "IS" and insert "ARE" in the same operation: CLEAR EDIT3Ø and ENTER. We recall that our educated guess was 80. Press '8' '0' and space, then space over so the cursor is on top of 'I.' We will leave the space after "IS" alone because we need a space after "ARE." To tell old faithful CoCo to delete two spaces, press '2' 'D.' They are gone! To insert "ARE," we must get into the insert mode. Press SHIFT and 'I' together and key in ARE. Get out of the edit mode with ENTER.

Due to deleting "USED IS" and inserting "ARE," about four spaces must be removed after "VARIOUS," so CLEAR EDIT3Ø and ENTER. Our target is on the fourth line. The fourth line begins at about the 90th character/ space. It is about eight more spaces from the left margin. The educated guess is 98. Press '9' '8' and space over to the space after 'S.' First, we delete three spaces to be safe. Press '3' 'D' to delete them. Space over a few spaces to reveal what comes next. Sure enough, one more space to yank out! Use the left-arrow key to get over to either of the two spaces. Press 'D' and get out of edit.

Checking it out, we discover a new error. The last word must be moved left one space to align at the left margin. Press CLEAR EDIT3Ø and ENTER. The space between "EDITING" and "FEATURES" must be removed. Our educated guess will be 90 (beginning of the fourth line), and about 15 more to get to the middle. Press '1' '0' '5' and space over to the space to be extracted. Press 'D' and ENTER to get out of you know what!

The sentence is OK but is poorly worded. Let's look at the second display panel.

"EDITTING" in the top paragraph has one 'T' too many. Note that even if you edit the 'T' out, a comma will be on the next line. That is a no-no! Now, hit CLEAR EDIT6Ø and ENTER. It is on the fifth line, so it must be located about 120 characters/spaces from the beginning. Add about 15 more spaces because it is in the middle of the line. Press '1' '3' '5' and space: our educated guess. Space over to any one of the 'T's, and press 'D' and ENTER to get out of edit.

Now run to see the comma. We must insert a '-' after "EDIT" and add two blank spaces to fill the end of the line. "ING" must be placed on the next line.

Again, press CLEAR EDIT6Ø and ENTER. We know 135 is a good, educated guess. Press '1' '3' '5' and space over the 'I.' We tell CoCo to get into the insert mode: SHIFT 'I' together and press '-' and the space bar twice. ENTER and RUN to get out of the edit mode and check.

Now, we must put a space between "IN" and "THE" on the last line, so CLEAR EDIT6Ø and ENTER. Being lazy, press 'X' to get to the end of the line. Press the left-arrow key until you are on top of the target, 'T' of "THE." This killed the end of the line. Key in a space and the rest of the line. Get out of the edit mode and double-check your work.

Nuts! The second paragraph is not only grammatically incorrect, but "KOLARS" requires an apostrophe. Now, let's CLEAR EDIT7Ø and ENTER. Knowing that inserting "TO" after "LIKE" will knock out the alignment, and also that an apostrophe must be placed in "KOLARS," the best thing is to target "LIKE." Since "LIKE" is at the end of the second line, it must be at about location 60. Press '6' '0' and space, then left-arrow one space SHIFT 'H' together to chop off the rest of the sentence. Using the EDIT 7Ø line above it on the screen as a guide, retype the rest of the sentence beginning with the missing "TO" and make sure to insert the apostrophe, then ENTER and check.

When you see problems arising, chicken out and retype the part of the line that will cause misalignment. We could separate the last line with a blank row. Find the proper line number and get in the edit mode. Now, CLEAR EDIT8Ø and ENTER. PRINT will be put at the beginning to create the blank row. SHIFT 'I' together, key in PRINT: or ?: and get out of edit. The text is fine. On to the graphics panel.

It could be raised about 10 spaces on the vertical axis for better centering. LIST it (you know it must be at the end of the listing). Sure enough, it is in lines 120 and 130. First, let us tack Line 130 to the end of Line 120 to keep the graphics on one program line, then EDIT12Ø and ENTER. Press 'X' to the end. Add the contents of Line 130 after the colon.

Note: CLEAR was not used so that the listed line would be on the screen for easy copying. Get out of edit and RUN. If the graphics look unchanged, then you have copied the contents of Line 130 properly. If a distortion appears, recheck your typing. LIST to make sure you are deleting the right line, then DEL130 and ENTER.

Now to raise the graphics higher on the display (10 units on the vertical axis), we subtract 10 from the vertical component (y) of the four (BM x, y) that create the design.

CLEAR EDIT120, ENTER and space over to the '8' of 86 and press 'C' '7'; space over to '9' of 96 — Press 'C' '8.' Space over to the '1' of 106, press 'D' to remove the '1.' Make sure you are on top of '0' then Press 'C' '9.' Space over to the '9' of 92 and press 'C' '8' and get out of edit. If you want to fool around, move it left five units on the horizontal axis.

We will recreate a typing error that occurs in the normal course of events, so CLEAR EDIT200 and ENTER. You desire to press 'X' to get to the end of the line. Without realizing it, you press 'C' instead. The cursor did not move. Strange! You know CoCo moves to the end of the line automatically on 'X.' What you did was delete the first letter and CoCo was waiting for you to make a change. Therefore, if you press 'X' in the edit mode and nothing appears to happen, look at the reference line above and press the same key to replace the letter. If you don't replace the letter, it will be deleted and ruin the line.

Sometimes, when you are in the edit mode, you may forget to tell CoCo what you want to do. If the first letter you press is 'Q,' you will be jerked out of the edit mode and "OK" will be displayed. If the first letter pressed is 'A,' the line will be redisplayed. If it is 'E,' you will exit the edit mode. You know where 'X' will bring you. These errors are harmless, but startling; CoCo is keeping us on the alert. There are other EDIT features not covered that you may care to check out.

Try not to use RENUM while you are developing a program. If you RENUM in midstream, you inflict a lot of unnecessary woes. You get accustomed to a line being at 'X,' and all of a sudden it is at 'Y.' When you change a large number of lines, the confusion gets greater. Wait until you are finished debugging. As long as there is room to enter new lines between existing program lines, you have the luxury of postponing RENUM.

The next five tutorials will be a series on the various aspects of DRAW.

Listing 1:

```
0 'LISTING1
5 CLEAR500
10 '(C) 1984, J KOLAR
20 CLS:PRINT:PRINT"  THIS PROGRA
M IS TO BE USED AS A GUIDE TO VA
RIOUS HINTS CON-   CERNING THE E
DIT CAPABILITY OF  COCO.
30 PRINT:PRINT"  ASIDE FROM ACTU
ALLY PROGRAM-  MING, THE MOST FR
EQUENTLY USED  FEATURE USED IS T
HE VARIOUS     EDITING FEATURES
40 PRINT:PRINT"  IT IS HOPED THA
T YOU WILL GET A FEW NEW INSIGHT
S AS TO HOW TO PROCEDE IN AN EFF
ICIENT MANNER.
50 PRINT:INPUT"           PRESS <E
NTER>";AO
60 CLS:PRINT:PRINT"  A FEW MISTA
KES WERE PLACED     DELIBERATELY
TO ILLUSTRATE SOME SUGGESTIONS.
 IF YOU ARE AN EX- PERT, AND KNO
W ALL ABOUT EDITTING, SKIP THIS
TUTORIAL AND KEY INTHE GRAPHIC P
ROGRAM.
70 PRINT:PRINT"  THESE GRAPHIC P
ROGRAMS ARE FORTHOSE WHO LIKE PR
ACTICE TYPING  LISTINGS OR ARE I
NTRIGUED WITH  KOLARS CREATIONS.

80 PRINT"  COCO LOVES BEGINNERS!
"
90 PRINT:INPUT"          PRESS <EN
TER>";OA
100 PMODE4,1:PCLS:SCREEN1,1
110 A$="BL4U2E2R4F2D4G2L4H2U2BR2
UER2FD2GL2HUBD4"
120 DRAW"S12BM128,86"+A$:DRAW"BM
135,96"+A$:DRAW"BM124,106"+A$
130 DRAW"BM114,92"+A$
200 GOTO200
```

Listing 2:

```
0 'BOUNCY
5 '(C) 1984, J KOLAR
30 PMODE4,1:PCLS
40 A=126:B=90:R=80:S=15
50 DIM S(13),T(13)
60 DRAW"BM8,4NU4NL4NR4ND4NE4NF4N
G4NH4"
61 DRAW"BM28,8BL4U2E2R4F2D4G2L4H
2U2BR2UER2FD2GL2HU"
70 GET(0,0)-(12,16),S,G
71 GET(20,0)-(32,16),T,G
80 PCLS:SCREEN1,1
100 DRAW"A0S8BM90,74U6R3D3NL3RD4
NL4BR3NR4U6R4D6BR3NU6R4NU6BR3U5N
UF4NU5DBR3NR4U6R4BR3D2F2ND2E2U2"

110 DRAW"BM98,100U6R3D3NL3RD4NL4
BR3U6R4D4NL4D2BR3NU6R4BR3NU6R4BR
5U2H2U2BR4D2G2"
180 FOR Z=6 TO -30STEP-3
185 FOR S=200 TO 100 STEP-20
195 K=C+LOG(R)+Z
200 X=INT(A+R*COS(K)):Y=INT(B+R*
SIN(C))
210 PUT(X+12,Y+60)-(X,Y+80),S,PS
ET
212 SOUND50,1
213 X=INT(A+R*COS(K)):Y=INT(B+R*
SIN(C))
215 PUT(X,Y+60)-(X+12,Y+86),T,PS
ET
220 NEXT S,Z
230 PLAY"O3V25L8CV30EV25CO2BABV2
0O3CL16EEFFGGV15EEFFGGL8GFEL4CO2
V20BO3L2C"
240 GOTO180
```

TURN OF THE SCREW

4K

How To Hook Up The Radio Shack Voice Synthesizer

By Tony DiStefano

A little while back, I did a project using the Votrax SC-02 voice synthesizer chip to make CoCo talk. It was an interesting project and I got a lot of correspondence about it. However, not all of it was good. People found that the chip was hard to find, and when they found it, it was very expensive. Ever since that time, I have been getting letters inquiring about how to hook up Radio Shack's own speech synthesizer to the CoCo.

I just came back from the Irvine RAINBOWfest in California, and believe it or not, more than one person asked me about this synthesizer. I know I am slow at times, but I think I finally got the message. So, this month we are presenting "How to Hook up the Radio Shack Voice Synthesizer to Your Color Computer."

But, first things first! I got a good piece of information while I was at the Irvine RAINBOWfest. I was talking to a gentleman about the "ins and outs" of the CoCo and we came upon the subject of repairing. If you have the 'F' board (also known as the 285 board) this is for you.

On some of these boards, there is a problem (an intermittent one at that). The symptoms are as follows: the computer works fine for a while, then all of a sudden random characters start to appear in columns one and nine of the screen. Just about this time, the computer freezes up and all work that is there gets lost. According to the gentleman I spoke with, the problem stems from the SAM; it is some sort of heat problem.

He says that Radio Shack is aware of this and is offering some help. Go to your Radio Shack dealer and order the "Final Fix" for the old CoCo.

Let's get back to the synthesizer. When I went to buy the chip, I saw there were two different sets: the older 276-1783 chip set and the newer 276-1784. Only the 1784 is listed in the new catalog, so I decided to go with it. This is just one chip while the 1783 is two. The package of this chip says that it comes 1) complete with specifications, applications data and programming information, and 2) requires additional components and skill in project assembly.

That's fine, but they don't tell you how to hook it up to your Color Computer.

Usually, in this next section I describe the functions of my project. This time, the project I am doing comes with a 20-page manual. I must say it is not a bad manual; the only thing left out was the circuit to connect it to the CoCo. But once the circuit is up and running, all you need to start writing programs that talk is in the book. It has all of the "allophone" set (as they call it) and even has a dictionary of words. It also includes a set of rules for using these allophones.

From the diagram and the description of this chip, I think you can add more chips to it so it can speak more sounds, possibly whole sentences and phrases. There is, however, no reference to part numbers or where and when these chips will be available. There will probably included a BASIC program listing you can use to try out your project.

This chip, as is, is quite easy to implement to the CoCo. It is basically divided into two parts. The first part is to get the data to the chip. The second is to poll the chip until it is not busy. Then you can give it the next piece of data.

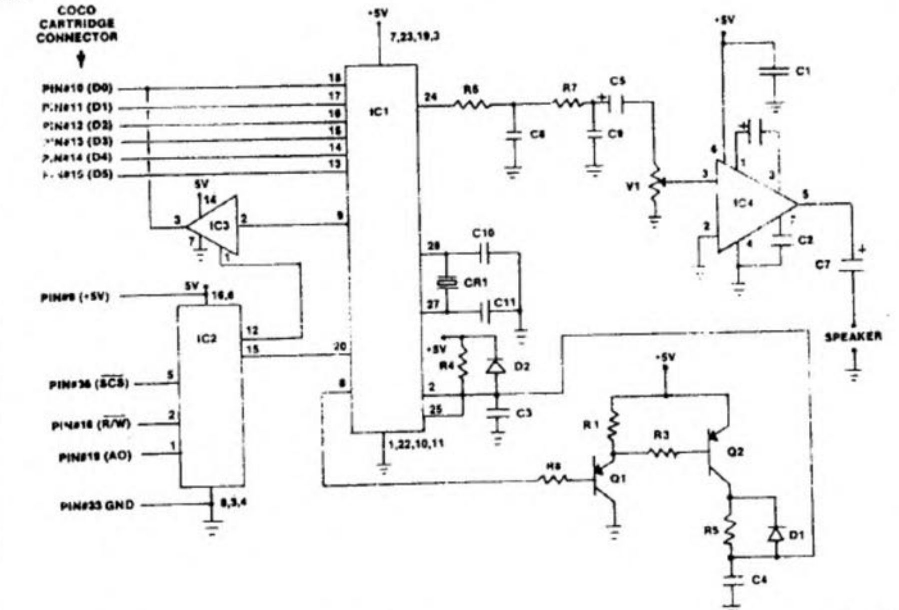
It is just like a parallel printer; in fact if the CoCo had a parallel printer port, I would have used it without any other extra parts. But that is not the case, so I used the old cartridge method which means it is connected to the CoCo via the cartridge connector, so if you have a disk drive, you will need a multipack or some kind of a switch.

It will not work with just a Y-cable because I used the *SCS output of the computer and it is used by the disk controller (see my previous article on the Multi-Pak Interface). With a few more chips, you *can* make it work with just a Y-cable. I will be doing an article soon on the technique of memory mapping and how to memory map something anywhere in memory.

To get data to the chip is easy. All you have to do is strobe the *ALD pin with data valid on the data lines and the data is entered. You can use the *SCS pin and you would not need any other parts, but there is another location to monitor. That is the pin that says when the chip is no longer busy with the last command you gave it, which is the *LRQ pin.

This is where the two TTL chips I added come in. The first chip is a 74LS138; this is a decoder chip. It is capable of decoding a three-bit binary

Figure 1



number into its eight different outputs. It also has three other select lines. Examine the 74LS138 in Figure 1 and notice that all I used is three lines. That is all that's necessary for this project.

We need two locations, one to write the data to the chip, and the other to read the busy pin of the chip. The *SCS pin of the CoCo selects the 74LS138 chip. A0 selects which location and the *R/W lines select a read or a write. Since we are using the *SCS pin on the CoCo, location $FF40 (65344 in decimal) is the base address. We are using only A0 so the two locations are $FF40 and $FF41.

In this case, $FF40 (65344) is the write location which is used to transfer data to the chip. Location number $FF41 (65345 in decimal) is used to monitor if the chip is busy. Reading (or PEEKing in BASIC) this location reveals whether it is busy or not.

The *LRQ line is connected to the input of a tri-state buffer. This is the 74LS125 chip. Only one of the four gates is connected. The output of this gate is connected to D0 on the CoCo bus. When you read the location, all other bits in the byte are irrelevant. If bit 0 is a logic 1, the buffer is full and the chip is busy. When this bit is logical 0, the chip is free and waiting for another command.

The rest of the circuit is the same as the recommended circuit by Radio Shack. There is one thing that confuses me about the Radio Shack circuit and I don't have the solution. It is the reset circuit: the two transistors, diodes, capacitor and resistors that connect to the reset and SBT reset. This circuit, as is, does nothing. I think it has something to do with the little arrow and the "NOTE" sign. What does that note refer to? Where is that note? What does that do?

I constructed the whole circuit, along with my circuit, and it worked fine. I monitored the SBY pin on the synthesizer chip and found it did nothing. It was always a logical 1. I disconnected the pin from the rest of the circuit; it still made no difference, so I cut out all the components except for the 100K resistor.

If you feel you must leave this circuit in, fine. Better yet, if you have an answer as to why it is there, please write me; I would like to know.

If you have a Multi-Pak from Radio Shack, a simple poke will give you access to the chip by changing the soft switch inside the Multi-Pak Interface. Remember that the Multi-Pak can change access to the *CTS pin and the *SCS pin. The *CTS pin controls 16K of software and the *SCS controls 32 bytes of I/O. The control byte is $FF7F (65407 in decimal).

To change the selector, you must poke a number into this byte. The most common configuration is to have the controller in slot #4 and the voice in slot #1. In that case, the value you must poke in the control location is a value of $30 (48 in decimal). Refer to your MPI manual for more details.

Table 1 lists all the parts necessary to build this voice synthesizer, including the parts in that reset circuit. At the end of this article, there is a list of mail-order stores that you can get parts from. There is no guarantee that any or all of these stores will have these parts. Except for the TTL chips and the protoboard, Radio Shack will have all of these parts.

There is one more thing to note. The diagram requires a 3.12 MHz crystal. The manual says you can order this crystal from Radio Shack, but you will have to wait. I didn't want to wait, so I bought Radio Shack's 3.579545 MHz crystal instead, which they had in stock. It works just as well, except the voice will be about 14 percent faster.

Electronics Parts Mail-Order Houses

JDR Microdevices
1224 S. Bascom Ave.
San Jose, CA 95128
(800) 538-5000 or (800) 662-6279 (CA)

Jameco Electronics
1355 Shoreway Road
Belmont, CA 94002
(415) 592-8097

Dokay Computer Products, Inc
2100 De La Cruz Blvd.
Santa Clara, CA 95050
(800) 538-8800

Table 1

Number	Description
IC1	SPO256-AL2 (Radio Shack #276-1784)
IC2	74LS138
IC3	74LS125
IC4	LM386
C1,2,3,4	.1uf 15V
C5	1uf 15V
C6	10uf 15V
C7	100uf 15V
C8,9	.022uf 15V
C10,11	22pf 15V
R1	1K ¼W
R2	10K ¼W
R3	200K ¼W
R4,5	100K ¼W
R6,7	33K ¼W
R8	10 ¼W
V1	10K POT
CR1	3.12 MHz (see text)
D1,2	1N914
Q1,2	MPS 2907 or 2N2907
Misc.	Proto-board, speaker, solder, wire, case

The listing: *TURNSCRW*

```
1 ' THIS IS FOR THE RADIO SHACK
2 ' VOICE SYNTHESIZER IC
3 ' FROM RAINBOW'S
4 ' TURN OF THE SCREW
5 ' BY TONY DISTEFANO
6 '
10 FOR I= 1 TO 55
20 READ A
30 POKE&HFF40,A
40 IF (PEEK(&HFF41) AND 1) = 1 T
HEN 40
50 NEXT I
60 END
100 DATA 27,7,45,15,53,4
110 DATA 24,06,04
120 DATA 26,26,16,4
130 DATA 20,04
140 DATA 13,23,23,2,42,12,44,4
150 DATA 42,15,16,9,49,22,13,51,
4
160 DATA 4
170 DATA 63,24,06,4
180 DATA 13,53,11,19,4
190 DATA 33,12,55,0,13,7,0,40,26
,26,56,53,1
```

PART VIII

By Colin J. Stearman

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.

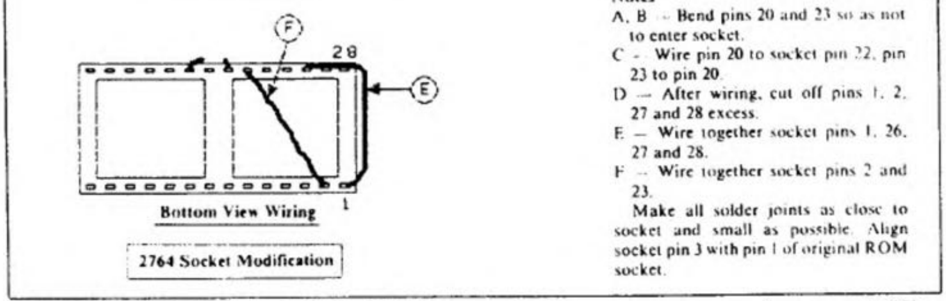
When the EPROM programmer is started up, load a 2764 EPROM into the ZIF socket and check that it is erased. Then transfer the memory contents from $4000 through $5FFF into the EPROM starting at EPROM address 0. This completes the programming. You can check the EPROM by powering down and moving the EPROM 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

Figure 1



Preparing The IC

Side View Wiring

Bottom View Wiring

2764 Socket Modification

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.

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; 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 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 whir, looking for *AUTOEXEC.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 *DECB 1.0* and another runnning *DECB 1.1*, some of the BASIC tokens will be different. This is due to the DOS command in *DECB 1.1*. 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.

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 *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.

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.

CORRECTIONS

"Rescue On Alpha II" (February, Page 13): Steven Mitchell has informed us that there have been some reports of syntax errors in DATA statement lines. According to him, these were usually caused by a missing or extra comma in a *previous* line. One change suggested is to change Line 346 to read:

```
346 IF X=16 AND E(5)=L THEN560
```

to fix a problem he has noted with the box of poison pellets in the game.

WEFAX (April 1985,Page 34): While testing this program here at THE RAINBOW, we found out that it did not work on the Star Micronics Gemini printers. Glenn Little has sent us some changes to the program that allow it to work on the Gemini-10X and Gemini-15X printers. (Note that the modified program will not work with the Epson printers.)

1) Edit Line 380 to remove one space between the first quote and the first '*'. Change the word EPSON in this line to read GEMINI. This makes the menu item read ''*PRINT TO GEMINI PRINTER*''.

2) Edit Line 903 to change the first 04 to 03 and then change the checksum on the end of this line to 1153. This makes the black cursor bar the correct length for the new message.

3) Edit Line 940 to change the first 17 to 0F and then change the checksum on the end to 2789. This is the actual printing fix; it changes the linefeed step to match the Gemini specifications, which don't quite match the Epson "standard."

WEFAX users who have the Epson MX printers should know that this program was designed for use with Graftrax Plus, which all MX-80 and MX-100 printers made after March 1982 have. We do not know if it will work with MX-80 and MX-100 printers that had Graftrax-80 installed later (or with the MX-70, which had Graftrax-80 as a standard feature). The earlier MX-80 and MX-100 printers that did not have Graftrax installed cannot do bit-image graphics printing and will not work with *WEFAX* until Graftrax Plus ROMs are installed.

OS-9 UTILITY

Hierarchal Directory

By Donald L. McGarry

One of the most convenient features of Microware's OS-9 operating system is its hierarchal directory structure. With a disk full of programs in Extended BASIC, a directory listing flies by on the screen. Pausing the display with SHIFT-@ is a test of reflexes as well as patience.

OS-9, on the other hand, allows many separate directories so each one contains only related files. I often have several levels of directories on one disk.

This is a fine method of file storage and I wonder how I ever lived without it, but it also has its problems. I often forget which file is which directory on a disk. Worse, I sometimes forget which disk holds the particular file I'm looking for.

The problem is compounded for those of us who keep several backup copies of important files on different disks. If one of these files is modified and the others are not immediately updated, the newest version is often difficult to identify.

Finally, "dir," as supplied by Radio Shack for the Color Computer, displays a directory which wraps around two screen lines to display one line of information. Redirecting output to the printer just produces a hard copy of the narrow display.

What I needed to end most of the disk confusion and to overcome the limited directory listings was a program that would display or print a complete directory for a disk. *HDirect* does just that!

In addition to displaying a complete directory, the programs display the date

The listing:

```
PROCEDURE HDirect

(* Prints an indented, hierarchal directory of /d0 or /d1
(* to the printer or the screen
(* Returns with working directory set to
(* 'root directory' of /d0 or /d1
(* whichever was used for the directory
(* D. McGarry

(* Calls: Direct and FixDate

PRINT CHR$(2) \REM Clear the screen

DIM Response:STRING[1]
DIM OutPath,DPath,Char:BYTE
DIM Level,Lines:INTEGER
DIM Date(5):BYTE
DIM SDate:STRING[14]

PRINT " Directory of which drive (0*/1) ";
GET #0,Response
PRINT  \ PRINT
IF Response=CHR$(13) THEN
Response="0"
ENDIF
IF Response<"0" OR Response>"1" THEN
END
ENDIF
CHD "/d"+Response \REM Point to the correct drive

PRINT " Print to Screen or Printer (S*/P) ";
GET #0,Response
PRINT
IF Response=CHR$(13) OR Response="s" THEN
Response="S"
ELSE IF Response="p" THEN
Response="P"
ENDIF
ENDIF
IF Response<>"P" AND Response<>"S" THEN
END
ENDIF
IF Response="P" THEN
CLOSE #1 \REM Close standard output path
OPEN #OutPath,"/P":WRITE \REM and substitute printer
ENDIF
PRINT  \ PRINT
REM Get disk name
OPEN #DPath,"@":READ
SEEK #DPath,31
LOOP
GET #DPath,Char
EXITIF Char>127 THEN
PRINT CHR$(Char-128); "      ";
ENDEXIT
PRINT CHR$(Char);
ENDLOOP
PRINT "Created  ";
REM Get disk creation date
SEEK #DPath,26
GET #DPath,Date
RUN FixDate(Date,SDate)
PRINT SDate;
```

the file was last modified and the size of the file. Other pieces of information such as file attributes, creation date, owner's ID and link count could be added easily (I left them out because I have no use for them). Details of the file descriptor sector which contains this information is given on pages 35 and 36 of the OS-9 Technical Information Manual supplied with OS-9.

The output of the programs is an indented, hierarchal directory starting with the root directory. Each level is indented from the previous level so that a disk's structure is easy to see. The programs are relatively straightforward, but they do make use of some interesting techniques. I will describe them briefly.

Names of variables are capitalized for a reason. If a variable is entered for the first time with a capital letter in its name, BASIC09 will change all future references to that variable to the original capitalization. Thus, if a variable name is misspelled in the program, the listing will show the name in lowercase. This acts as an instant spelling checker during program development.

Most of *HDirect* is self-explanatory, but a trick is used to read the disk name. The entire disk can be read as a single random access file if the filename used in the OPEN statement is '@.' Using this technique allows *HDirect* to get the disk name and creation date.

The same method is used in *Direct* to get the file descriptor information of each entry. If a printed listing is chosen, the standard output path is closed and the printer is substituted. Be careful closing and opening any of the standard paths a mistake can create a mess.

HDirect prints a heading, sets two variables and calls *Direct*, which does all of the real work. *Direct* reads each file in a directory. It prints the name and checks the descriptor to see if the file is a directory. If it is, the variable level is incremented and *Direct* calls itself. Each call to *Direct* ends when it has read all of the files in a directory.

Since all disks begin with a root directory, the sequence of calls to *Direct* begins and ends at the root of the disk being read. Changes to *Direct* will allow the addition of the file information mentioned above.

The other program, *FixDate*, just converts a date from the format used by OS-9 to a string. I prefer month day year format to OS-9's year month day format, so I changed the order also. You could keep the original date order by removing the lines in *FixDate* which

```
REM Change the next line for a narrower screen
PRINT TAB(57); "Last Modified      Size"
CLOSE #DPath
PRINT  \ PRINT

Level=0
Lines=4
RUN Direct(Level,Lines)
PRINT
REM Restore correct standard output path
CLOSE #1
OPEN #OutPath,"/Term":WRITE

END
PROCEDURE Direct

(* Reads, formats, and prints file descriptor information
(* from the default drive and changes directories
(* D. McGarry

(* Calls: Direct

TYPE Descriptor=Attr,Owner(2),MDate(5),LCount,FSize(4):BYTE

PARAM Level,Lines:INTEGER

DIM Name:STRING[29]
DIM Temp:STRING[24]
DIM I:INTEGER
DIM DPath,Path,LSN(3):BYTE
DIM FileInfo:Descriptor
DIM Sector,TSize:REAL

OPEN #Path,".":READ+DIR
REM Skip "." and ".."
GET #Path,Name
GET #Path,LSN
GET #Path,Name
GET #Path,LSN

WHILE NOT(EOF(#Path)) DO
GET #Path,Name
GET #Path,LSN
Sector=65536.*LSN(1)+256*LSN(2)+LSN(3)

REM Change Name into BASIC09 format
FOR I=1 TO 29
EXITIF MID$(Name,I,1)>CHR$(127) THEN
Name=LEFT$(Name,I-1)+CHR$(ASC(MID$(Name,I,1))-128)
ENDEXIT
NEXT I

OPEN #DPath,"@":READ
SEEK #DPath,Sector*256
GET #DPath,FileInfo
CLOSE #DPath

IF ASC(Name)<>0 THEN
TSize=256*256*256*FileInfo.FSize(1)+256*256*FileInfo.FSize(2)+256

*FileInfo.FSize(3)+FileInfo.FSize(4)
 RUN FixDate(FileInfo.MDate,Temp)
 Temp=LEFT$(Temp,14)+RIGHT$("            "+STR$(TSize),11)
 PRINT TAB(4*Level+1); Name;
 PRINT TAB(56); Temp \REM Change for narrower screen

 Lines=Lines+1
 IF Lines=60 THEN
 FOR I=1 TO 6
 PRINT
 NEXT I
 Lines=0
```

change Date(1), Date(2) and Date(3).

The programs were designed for an 80-character line. If your screen display is not that wide, *HDirect* and *Direct* should probably be changed to fit your screen. Change the lines containing "PRINT TAB(57)" and "PRINT TAB(56)". Keep a full-width version of the programs for printing.

When you enter the programs, enter *HDirect* first. This way when the programs are packed, they will work correctly. Don't forget to save the programs before you pack them.

If you have more than two disk drives, you can change the drive number response in *HDirect* to the number of drives you have.

Using *HDirect* is simple. The program prompts for the information it needs. If you pack the programs and want to print directories of several disks, you should load BASIC09 and *HDirect* so they don't have to be reloaded after each disk.

Please remember that *HDirect* will leave you logged onto the root directory of the disk that it accessed when it is finished. If your working directory was somewhere else, you will have to log back on using "chd."

(Anyone having questions regarding this program may contact Mr. McGarry at 2 Heather Dr., Northport, NY 11768, phone (516) 754-3069.)

```
ENDIF

IF FileInfo.Attr>127 THEN CHD Name
RUN Direct(Level+1,Lines)
CHD ".."
ENDIF

ENDIF

ENDWHILE

CLOSE #Path

END
PROCEDURE FixDate

(* Changes a five byte date into a 14 character string
(* with the date in the format: MM/DD/YY HH:MM
(* D. McGarry

PARAM Date(5):BYTE
PARAM SDate:STRING[14]

DIM Temp:BYTE
DIM I:INTEGER

SDate=""

REM remove the next 4 lines to keep date in YY/MM/DD format
Temp=Date(1)
Date(1)=Date(2)
Date(2)=Date(3)
Date(3)=Temp

FOR I=1 TO 5
SDate=SDate+RIGHT$("0"+STR$(Date(I)),2)+MID$("// : ",I,1)
NEXT I
END
```

OS9

The OS9 User Group has produced it's latest newsletter, and its a gem!

For anyone with OS9, or anyone contemplating the purchase of OS9, this group is to be highly recommended; not only do you get access to experts in OS9, but there is a growing range of software that is available only through the group, most of it pretty good stuff too!

THE big news is that Dr Patrick Tuohey of Montmorency in Victoria has found a patch to fix OS9's clock. Now we can get some sense out of that feature of our own Bulletin Board! And of course, it enables a range of other software.

The group has access to a Public Domain OS9 communications program. Currently, this is available only as a listing, but hopefully some kind typist will make a disk version available soon!

OS9 groups are springing up all over the place! The latest is in Cooma, of all places! Fred Bisseling is the contact there, and his phone number occurs on the back of this magazine.

Commercial software for OS9 is also becoming more widely available. Elsewhere in this magazine is a review of "Trivia", a program for OS9 we have seen in Australia already. There are also several new packages due from Tandy soon too.

But the software is not going to materialise if people don't BUY some occasionally. No merchant, let alone Tandy, is going to bother getting stuff in if it, 1. just gets copied, or 2. no one buys it!

NOW! is the time to subscribe to Australian RAINBOW

Copies of back issues can be obtained, subject to the availability of stocks, by using this order form and marking clearly which issues you require to be sent to you.
Each issue costs $4.50 including postage and packing. Please enclose your cheque/money order made payable to: Australian Rainbow Magazine, PO Box 1742, Southport, 4215.

RATES

AUSTRALIAN CoCo/MiCo/softgold		AUSTRALIAN RAINBOW
□ $3 45	Latest per copy	$4.50 □
□ $19	6 months	$27.75 □
□ $31	12 months	$39.95 □

BACK ISSUES

MiCo —
First Issue Oct '83
□
$3/copy

CoCo/MiCo/softgold
First Issue Aug '84
□
$3.45/copy

Feb '85-Apr '85	□	$3.95
Aug '84-Jan '85	□	$3.25
to July '84	□	$3.00

DISKS & TAPES

Rainbow on Tape (programme listings) $12 for month of □
Please note that RAINBOW on TAPE is issued irregularly

or Annually $144 □ or Debit my Credit Card Monthly □

Tape Monthlies

CoCoOz (Aust CoCo on Tape)		MiCoOz (MiCo on Tape)
□	Latest $8	□
□	6 months $42	□
□	12 months $75	□
First Issue Mar '83		Dec '83

or Debit my Credit Card Monthly

Blank tapes 12 for $18 or $1.70 ea □

Cassette Cases 10 for $5 □

Disks — $3.50 ea □ 10 for $29.99 □

BOOKS

Byte
Elementary $5.95 □

Help
Medium $9.95 □

Facts
Advanced $11.95 □

MiCo Help
Medium $9.95 □

MiCo Exposed
SORRY
OUT OF STOCK □

BULLETIN BOARD

Let your computer talk to ours!
CoCoLink —
Annual Sub $29 □

□□□□□□□□□□□□□□

□ VISA BANKCARD □
MASTERCARD

______________ Signature

□ CASH
□ CHEQUE
□ MONEY ORDER

Authorised:
Amount $ ______________

BLOCK CAPITALS PLEASE

If you already subscribe to either Australian Rainbow or Australian CoCo please place Subscription No
□□□□□□□

Complete the section below with one letter, figure or space per square

FIRST NAME SECOND NAME
□□□□□□□□□□□□□□□□□□□□

Address
□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□ PC □□□□□

Telephone Number STD Code □□□□□ Local Number □□□□□□□□□□

Subscription □ Renewal □

Of Back Issues, Tapes and Things.

With the exception of June 82 and Nov 82, we have copies of all back issues available, and in fact need to reduce our stocks of many of them. The early copies of Rainbow are an excellent source of information for the new CoCo owner. The later copies reflect the growing knowledge of the average user of the time. There are games, utilities, hints, and programs for educational, business and club use. There are also many tutorials and articles of interest.

We also have considerable numbers of GoCo Magazine. If you don't have a full set then give me a call.

For those who want to complete their Rainbow collection, we are offering a one for three deal. Buy any three pre August 1984 Rainbows, and we'll give you one more of your choice free!

ANNOUNCING The BEST of CoCoOz!!

To assist teachers and others who are involved with children in learning situations, we have compiled a 14 program tape (or disk) which reflects some of the better Educational programming.

Programs include Quizes on Flags and Rivers, the classic "Fractut", a fractions tutor, and "Taxman", a program which teaches factors. Many of our best writers are represented and we fully recommend this tape to Educators with CoCo's who can't decide what to do with them!

'The Best Of CoCoOZ' is available for $10.00 (tape), or $21.95 (disk), post paid.

Recently I was accused of being frustrated. My accusor was closer to the truth than I care to admit. Further cause for my frustration is very much linked with Tandy.

Four years ago Tandy took a gaint leap forward and launched our beloved CoCo in its humble 4K guise. Other computers since then have come and gone, and I agree with Graham when he says that the CoCo has at least 5 years of life yet. But that is no thanks to Tandy!

It almost seems accidential that the CoCo should be as popular as it is today! The Commodore people would LOVE to have a computer as capable as ours! (I believe they could sell a lot more too!)

Commodore are about to release a computer with a screen described as being "T.V. Standard". I understand this to mean that the computer has extremely high resolution. I also understand that this is not ho-hum - it IS a big step forward.

What is at issue here, is not that Commodore have the new screen. Sooner or later, if it is good, someone will make one for CoCo - like they did with CoCoMAX, MUSICA, GRAPHICOM, 128K upgrades, disk drives, monitors, 64K upgrades etc!

BUT Tandy have not, throughout the developmeant of all the above, lifted a finger to improve CoCo. CoCo is still the same computer it was 5 years ago - plus some memory (forced on Tandy) and a pretty keyboard/case, (and minus a great board).

How about it Tandy? Why not show some intestinal fortitude and put some cash back into CoCo? There's lots you can do - add some some colour, redefine the screen, give CoCo an optional 80 column text screen, build in a modem, a disk drive and a screen, add a new dimension to CoCo's abilities!

The clever things we see today for CoCo have been developed by external parties - and there's nothing wrong with that! - but Tandy what are you doing?

I can't imagine what must be going through the minds of the Tandy dealers in Victoria. I know of at least one there who recently spent alot of money to set up a new shop, specificially to market computers in his local area. Along comes Tandy with some state wide deal with a third party to sell computers anywhere this third party likes - even against Tandy's own dealers!

Could it be that Tandy are dissatisfied with the current computer sales performance?

One thing is certain - if you are paid to sell computers for Tandy - I'd be chasing business very hard at present!

I wanted to go to the launch of the Tandy 1000 in Brisbane in May, but I wasn't allowed to go. Just because I got cross with Tandy for launching a mediocure computer, Graham said I couldn't be there.

Well he doesn't silence this little black duck so easily! If we ever see one in the office, you can be assured that I'll qive you my unbiased opinion of it!

continued from page 21

```
SCREEN":PRINT"     <1> = RIGHT RU
DDER (+)":PRINT"     <2> = LEFT R
UDDER (-)":PRINT"     <3> = DIVIN
G PLANE DOWN (+)":PRINT"     <4>
= DIVING PLANE UP (-)"
156 PRINT"     <5> = BLOW MAIN BA
LLAST":PRINT"     <6> = FILL MAIN
 BALLAST":PRINT"     <7> = INCREA
SE SPEED (+)":PRINT"     <8> = DE
CREASE SPEED (-)":PRINT"     <9>
= PERISCOPE (UP/DOWN)"
157 FORR=1TO31:PRINTCHR$(A);:NEX
T:POKE&H5FF,&HAF
158 RETURN
159 'break disable
160 B$="32621CAF7EADA5":Y=&HF8:G
OSUB82
161 POKE&H19A,&H39:POKE&H19B,0:P
OKE&H19C,&HF8:POKE&H19A,&H7E
162 RETURN
163 'color test
164 B$="121A5086F8B7FF22B7FFC3B7
FFC5B7FFC7B7FFC9":IFDK=0 THEN165
 ELSEB$=B$+"B7FFCB"
165 B$=B$+"8E0"+HEX$(&H600+DK)
166 B$=B$+"CCAAAAED818C"
167 B$=B$+HEX$(&H1E00+DK)
168 B$=B$+"25F9BDADFB128EAB83108
EAA96CE02E010CE7F331A847EAE7512"
169 Y=&H6070:GOSUB82
170 POKE&H71,&H55:POKE&H72,&H60:
POKE&H73,&H70
171 RETURN
172 'draw ice
173 Y=&H603E
174 B$="600086FF97B586E0BE603EC6
0DD7C6E680D7C097BE97C48004271380
603822093412BD94A1351220E68E6000
20F239"
175 GOSUB82
176 RETURN
177 'sonar
178 IFTIMER>E THENTIMER=0 ELSE R
ETURN
179 E=(DP-4-IC)*5:IFE<-20THENGB=
3:GOTO133
180 IF IC=0 THENRETURNELSESOUND1
50,1
181 RETURN
182 'screen routines
183 B$="128E6200108E"
184 B$=B$+HEX$(&H1E00+DK)
185 B$=B$+"ECA1ED81108C"
186 B$=B$+HEX$(&H3600+DK)
187 B$=B$+"25F686FFC60E8E0152A78
05A270820F98662447E960F39"
188 Y=&H60B0:GOSUB82
189 'erase ice
190 B$="125F8E"
191 B$=B$+HEX$(&H1F00+DK)
192 B$=B$+"861DE7804A27068C"
193 B$=B$+HEX$(&H347C+DK)
194 B$=B$+"25F639300320EF12"
195 Y=&H60E0:GOSUB82
196 RETURN
```

user group CONTACTS

(Stop between numbers = b.h. else a.h.; but, hyphen between = both.)

ADELAIDE	JOHN HAINES	08 278 3560
ADELAIDE NTH	STVN EISENBERG	08 250 6214
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	TOM STUART	067 72 8162
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
BANKSTOWN	KEN HAYWARD	02 759 2227
BLACKTOWN	KEITH GALLAGHER	02-627-4627
BLACKWATER	ANNIE MEIJER	079.82.6931
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BOWEN	TONY EVANS	077 86 2220
BRASSALL	BOB UNSWORTH	07 201 8659
BRIGHTON	GLENN DAVIES	08 296 7477
BRISBANE EAST	ROB THOMPSON	07 848 5512
BRISBANE SH	PATRICK SIMONIS	07 209 3177
BRISBANE SW	GRAHAM BUTCHER	07 376 3400
BRISBANE WEST	BRIAN DOUGAN	07 30 2072
BUNDABERG	JIM McPHERSON	071 72 8329
CAMBERWELL	TONY BALDWIN	03 728 3676
CAMPBELLTOWN	LEO GINLEY	02 605 4572
CANBERRA	SHAUN WILSON	062 51 2339
CAULFIELD	JEFF SHEEN	03 528 3724
CHATSWOOD	BILL O'DONNELL	02 411 3336
CHURCHILL	GEOFF SPOWART	051 22 1389
COLYTON TEENS	DWAYNE MANSON	02 623 5805
COOMA	ROSS PRATT	0648 23 065
DANDENONG	DAVID HORROCKS	03 793 5157
DARWIN	BRENTON PRIOR	089.81.7766
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 2095
EMERALD	LEIGH EAMES	059 68 3392
FORSTER	GARY BAILEY	065 54 5029
FRANKSTON	BOB HAYTER	03.783.9748
GIPPSLAND STH	PAT KERMODE	056 74 4583
GLADSTONE	ALBERT VAN GORKUM	079 72 2353
GOLD COAST	SHERYL BENTICK	075-39-2003
GOSFORD	PETER SEIFERT	043 32 7874
GOULBURN VALLEY	TONY HILLIS	058 59 2251
GRAFTON	DAVID HULME	066.42.0627
GREENACRES	BETTY LITTLE	08 261 4083
HASTINGS	MICHEAL MONK	059.79.1513
HERVEY BAY	LESLEY HORWOOD	071 22 4989
HILLS DIST	DENNIS CONROY	02 671 4065
HOBART	BOB DELBOURGO	002 25 3896
IPSWICH	MILTON ROWE	07 281 4059
JUNEE	PAUL MALONEY	069 24 1860
KALGOOLIE	TERRY BURNETT	090-21-5212
KENMORE	GRAHAM BUTCHER	07 376 3400
LEETON	CHRIS NAGLE	069 53 2969
LITHGOW	STUART RAYNER	063 51 4214
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACKAY	LEN MALONEY	079511333x782
MACLEOD	ROBIN ZIUKELIS	03 450211x465
MacQUARIEFIELDS	KIETH ROACH	02 618 2858
MAFFRA	MAX HUCKERBY	051 45 4315
MAITLAND	LYN DAWSON	049 49 8144
MARYBOROUGH	NORM WINN	071 21 6638
MELBOURNE	JEFF SHEEN	03 528 3724
MELTON	MARIO GERADA	03 743 1323
MILDURA	SCOTT HEWISON	050 23 6016
MOE	STEPHEN SEMPLE	051 27 6841
MORPHETTVALE	KEN RICHARDS	08 384 4503
MOREE	ALF BATE	067 52 2465
MORWELL	GEORGE FRANCIS	051 34 5175
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280
MUDGEE	BRIAN STONE	063-72-1958
MURGON	PETER ANGEL	071 68 1628
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NEWCASTLE	LYN DAWSON	049 49 8144
NOARLUNGA	ROBBIE DALZELL	08 386 1647
NOWRA	ROY LOPEZ	044 48 7031
PARKES	DAVID SMALL	068 62 2682
PENRITH	ALEX SCHOFIELD	047-31-5303
PERTH	IAN MACLEOD	09 448 2136
PORT MacQUARIE	RON LALOR	065 83 8223
PORT NOARLUNGA	ROB DALZELL	08 386 1647
PORT PIRIE	KEVIN GOWAN	086 32 1368
RINGWOOD	ANDREW RAWLINGS	03 726 6521
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
ROSEVILLE	KEN UZZELL	02 467 1619
SALE	BRYAN McHUGH	051 44 4792
SANDGATE	MARK MIGHELL	07 269 5090
SEACOMBE HTS	GLENN DAVIS	08 296 7477
SMYTHESDALE	TONY PATTERSON	053 42 8815
SPRINGWOOD	DAVID SEAMONS	047 51 2107
STURT	MARY DAVIS	08 296 7477
SUNBURY	JACK-SMIT	03.744.1355
SUTHERLAND	IAN ANNABEL	02 528 3391
SWAN HILL	BARRIE GERRAND	050.32.2838
SYDNEY EAST	BOB JONES	02-331-4621
SYDNEY TEENS	ROD HOSKINSON	02 48 5948
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
TONGALLA	TONY HILLIS	058 59 2251
TOOWOOMBA	MIKE CARTER	076 35 6911
' BEGIN NTH	DAVID PROUT	076.32.7533
' BEGIN STH	LEW GERSEKOWSKI	076 35 8264
' ADVANCED	GRAHAM BURGESS	076 30 4259
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
TRARALGON	MORRIS GRADY	051 66 1331
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
WAGGA WAGGA	BRUCE KING	069 25 3091
WESTLEIGH	ATHALIE SMART	02 848 8830
WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
WOLLONGONG	BRIAN McCAULEY	042 71 4265
WONTHAGGI	PAT KERMODE	056 74 4583

SPECIAL INTEREST GROUPS

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696
BRISBANE OS9	JACK FRICKER	07 262 8869
CARLISLE MiCo	STUART HALL	08 361 1922
MONARO OS9	FRED BISSELING	0648 23263
BLAXLAND OS9	BOB THOMSON	047 30 2468
BLAXLAND 128K	BOB THOMSON	047 30 2468
ROCKHAMPTON MiCo	TIM SHANK	079 28 1846
SYDNEY MiCo	RAJA VIJAY	02 519 4106
SYDNEY MS DOS	ROGER	047-39-3903
BRISBANE MS DOS	BRIAN DOUGAN	07 30 2072

AUSTRALIAN CoCo

THIS MONTH

- * INVADER 2000 by Stuart Sanders
- * LABEL PRINTER by Graham Nichols
- * FLASHWORD by Tom Lehane
- * MACHINE LANGUAGE DATA GENERATOR by Greg Watson
- * WIZARD SOCIETY by Darrell Berry
- * MUSIC RECORD by Darrell Berry
- * PYTHAGARUS by Les Thurbon
- * LOG BOOK by Les Thurbon
- * CROSS REFERENCE INDEX by Robbie Dalzell
- * TANGLE by Craig Stewart

PLUS Letters, OS8, Reviews and articles of interest to YOU!!

CoCoOz tapes are produced monthly and contain all the programs from CoCo magazine, already typed up and ready for you to RUN!

AUSTRALIAN RAINBOW MAGAZINE
Registered by Australia Post
Publication No. QBG 4009

P.O. Box 1742,
Southport. QLD. 4215.