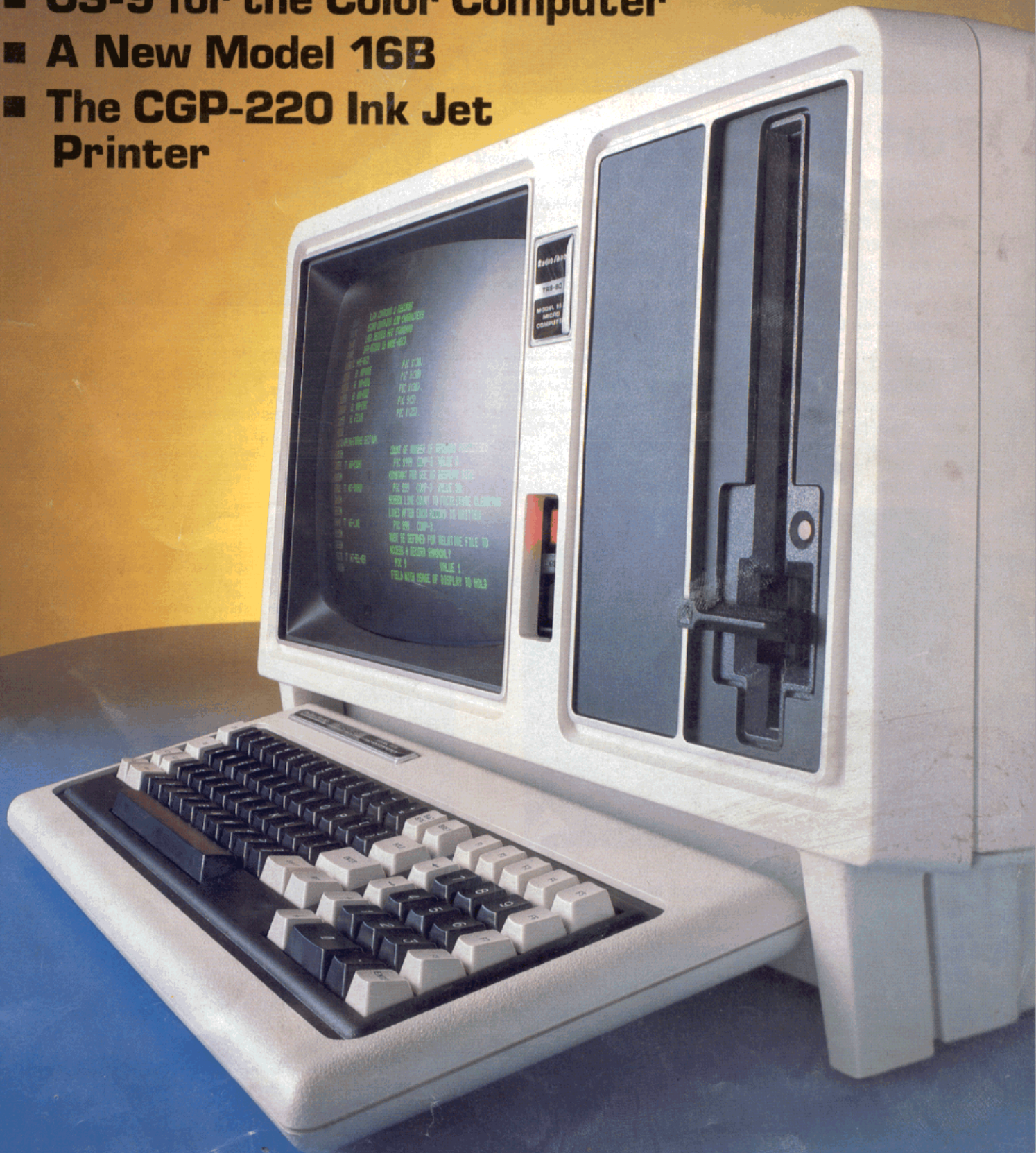


Microcomputer News

- **OS-9 for the Color Computer**
- **A New Model 16B**
- **The CGP-220 Ink Jet Printer**



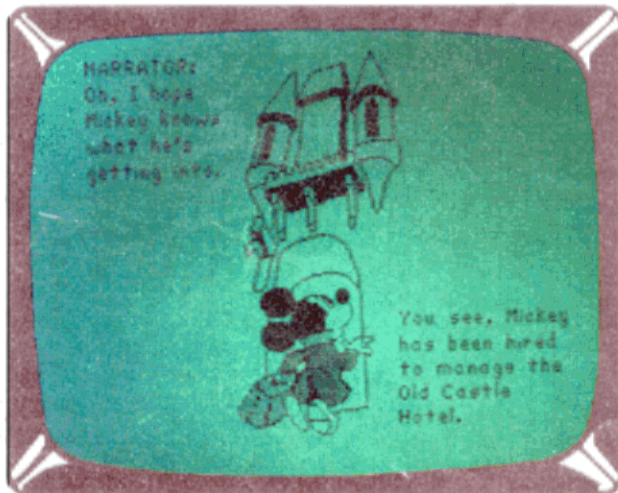


Fort Worth Scene

Several computer products including a new computer, new printer, new peripherals, and new software are described in this issue.

Just when you think that you've seen everything, something else appears that makes you realize you haven't seen anything yet. The CGP-220 Ink Jet Printer is really a terrific value in a printer. It has color. It's self cleaning. The ink comes in cartridges so there's no mess. It uses plain or roll paper. Enough said. This is not intended to be the second article on the CGP-220.

Radio Shack will be offering OS-9 for the Color Computer for those who want the ability to write their own operating system, multi-processing or multi-tasking on the Color Computer. You can read about OS-9 this month, too. There are several new hardware items for the Color Computer: the Multi-Pak Interface, the Mouse, and the new Deluxe Joystick.



AND FOR THE CHILDREN

Educational packages designed to run on the Color Computer are coming our way from two of the most recognizable names in the fields of education and entertainment for children: Walt Disney Productions and Children's Computer Workshop (CCW).

Many favorite Disney characters are found talking and teaching basic education skills in the eight educational packages from the Walt Disney Educational Media Company. In addition to helping a child learn, these programs are designed to amuse and entertain with their high resolution graphics, recorded narration, and music.

CCW is a part of the Children's Television Workshop, the same folks who produced "Sesame Street," "The Electric Company," and "3-2-1 Contact." The seven programs in this series are designed to develop basic skills in 3 to 6 year olds and to teach cooperative strategy to children ages 7 through 10 in an interactive, entertaining environment.

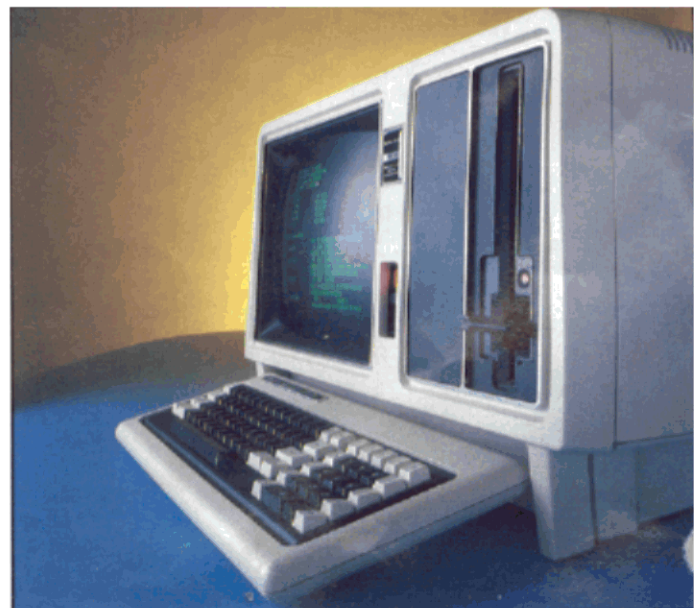


FOR THE PC-2

For the PC-2 there's the fifth article in the popular series on PC-2 Assembly Language and an article on the PC-2 RS-232 communications device. I would recommend the latter to anyone who has ever wondered about communication terms like parity, stop and start bits, framing errors, etc. This article offers fundamental information on RS-232 terminology.

MCN'S NEW SIZE

Something else new, in case you didn't notice right off, is that *TRS-80 Microcomputer News* has grown again. We leaped to fifty-six information packed pages this month. By doing this we hope to provide you with additional useful material throughout the year. We are striving to see that there will be something in every issue for all our readers.



This Model 16B has an internal 15-meg hard disk.

TRS-80[®] Microcomputer News

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation, One Tandy Center, Fort Worth, Texas U.S.A. 76102. Copyright 1983 by Tandy Corporation, One Tandy Center, Fort Worth, Texas U.S.A. 76102. All rights reserved.

Reproduction or use, without express written permission from Tandy Corporation, of any portion of the Microcomputer News is prohibited. Permission is specifically granted to individuals to use or reproduce material for their personal, non-commercial use. Reprint permission for all material (other than Ivan Sygoda's Profile article), with notice of source, is also specifically granted to non-profit clubs, organizations, educational institutions, and newsletters.

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single six month subscription is available free to purchasers of new full size TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, Canada and APO or FPO addresses. Certain smaller TRS-80 Microcomputers will not include this free subscription. Subscriptions to other addresses are not available.

The subscription rate for renewals and other interested persons with U.S., APO or FPO addresses is twelve dollars (\$12.00) per year, check or money order. Single copies of the Microcomputer News may be purchased from Radio Shack Computer Centers or Computer Departments for \$1.50 suggested retail each.

The subscription rate for renewals and other interested persons with Canadian addresses is Fifteen dollars (\$15.00) per year, check or money order in U.S. funds. All correspondence related to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76113-2910.

Retail Prices in this newsletter may vary at individual stores and dealers. The company cannot be liable for pictorial and typographical inaccuracies.

Back issues of Microcomputer News prior to January, 1981 are available through your local Radio Shack store as stock number 26-2115 (Suggested Retail Price \$4.95 for the set). Back issues of 1981 copies are available as stock number 26-2240 (Suggested Retail Price \$9.95 for the set).

The TRS-80 Newsletter welcomes the receipt of computer programs, or other material which you would like to make available to users of TRS-80 Microcomputer systems. In order for us to reprint your submission, you must specifically request that your material be considered for reprinting in the newsletter and provide no notice that you retain copyrights or other exclusive rights in the material. This assures that our readers may be permitted to recopy and use your material without creating any legal hassles.

Material for publication should be submitted on magnetic media (tape, disk, or CompuServe). If you submit material on tape or disk, and it is accepted for publication, we will send you two cassettes or diskettes for each one you sent us. Cassettes will come from our box of mixed blank cassettes. If you submit material on CompuServe, and we think we may use the material, we will extend your Microcomputer News subscription by six months for each article accepted. If you are submitting material over CompuServe, please include your name and address or your subscription number so we can find you. If the material is very short, send it to us in E-Mail. If you have more than a few lines, you need to place the material in the ACCESS area of CompuServe and then let us know it is there by leaving a message on E-Mail.

Material may be submitted by mail to P.O. Box 2910, Fort Worth, Texas 76113-2910, or through CompuServe. The Microcomputer News' CompuServe user ID number is 70007,535.

Programs published in the Microcomputer News are provided as is, for your information. While we make reasonable efforts to ensure that the programs we publish here work as specified, Radio Shack can not assume any liability for the accuracy either of the programs themselves or of the results provided by the programs.

Further, while Microcomputer News is a product of Radio Shack, the programs and much of the information published here are not Radio Shack products, and as such can not be supported by our Computer Customer Service group. If you have questions about a program in the Microcomputer News, your first option is to write directly to the author of the program. When possible, we are now including author's addresses to facilitate communications. If the address is not published, or if you are not happy with the response you get, please write us here at Microcomputer News. We will try (given the limited size of our staff) to find an answer to your question and, in many cases, will publish the answer in an up-coming issue of Microcomputer News.

Trademark Credits

| | |
|--------------------------|-----------------------|
| CompuServe [™] | CompuServe, Inc. |
| CP/M [®] | Digital Research |
| Dow Jones [™] | |
| NEWS/RETRIEVAL | |
| Service [®] | Dow Jones & Co., Inc. |
| LDOS [™] | Logical Systems, Inc. |
| VisiCalc [®] | VisiCorp, Inc. |
| XENIX [™] | Microsoft |
| Program Pak [™] | Tandy Corporation |
| SCRIPSIT [™] | Tandy Corporation |
| TRSDOS [™] | Tandy Corporation |
| TRS-80 [®] | Tandy Corporation |

Contents:

Color Computer

| | | |
|--|------------------------|----|
| Color Computer Draw Statement | by Mike Kim | 10 |
| Color Computer Multi-Pak Interface | | 19 |
| Color Mouse and Deluxe Joystick | by Linda Miller | 20 |
| IF/THEN/ELSE Statements | by Ray B. Blessum | 42 |
| OS-9—A New Color Computer Operating System | by Bruce Elliott | 16 |
| Programs | | |
| Calendar | by Tim McDuffie | 32 |
| Drawing with the Color Computer | by David Andrew Palmer | 34 |
| Spiral | by Leo Gilbride | 23 |

Computer Clubs

Computer Customer Service

| | | |
|-----------------|--|----|
| TRS-Xenix Power | | 24 |
|-----------------|--|----|

Data Bases

| | | |
|--|--|----|
| Profile | | 28 |
| Profile and the Model 100 | | |
| AgriStar | | 38 |
| Write A Codefile! | | |
| by Kimberly Bilstad Ness and Mary Turner | | |

Education

| | | |
|--|------------------------|----|
| Educational Computing from Walt Disney, Inc. [™] | by Linda Miller | 4 |
| Learning Can Be Child's Play: Children's Computer Workshop | | |
| Develops Educational Software for Radio Shack | | 7 |
| Pod Concept in Classroom Networking | by Warren Hornsby, Jr. | 14 |

Fort Worth Scene

| | | |
|--|--|---|
| | | 2 |
|--|--|---|

General Interest

| | | |
|---|--------------------|----|
| Fast Data Lines for Models I/III/4 and II/12/16 | by B.M. Tennyson | 12 |
| On-Line Computer Telephone Directory | | 31 |
| Poem—Frustration | by J. De Augustine | 13 |
| Word Processing and Programming | by Davy L. Barron | 18 |

Magazines

| | | |
|--|--|----|
| | | 40 |
|--|--|----|

Model I/III/4

| | | |
|---|---------------------------|----|
| Install the Model III Business Graphics Package | | |
| On Your 5 Meg Hard Disk | by Annette Zamberlin-Main | 39 |
| Programs | | |
| Bar, Line and Scatter Graphs for the Model III | by A.F. Bell | 51 |

Model II/12/16

| | | |
|--|---------------------------|----|
| A New Model 16B with 15-Megabyte Hard Disk | by Annette Zamberlin-Main | 11 |
|--|---------------------------|----|

Model 100

| | | |
|-------------------------|--|----|
| Debug for the Model 100 | | 21 |
|-------------------------|--|----|

Notes on Previous Issues

| | | |
|-------------------------------------|--|----|
| | | 43 |
| November 1981 | | |
| Reading Profile II Files from BASIC | | |
| February 1982 | | |
| 3-D Color Graphics | | |
| March 1982 | | |
| Relocating Machine Language Prog. | | |
| April 1982 | | |
| Perpetual Calendar | | |
| May 1982 | | |
| Ultra Precision Multiplication | | |
| Verifying Programs and Data Files | | |
| June 1982 | | |
| Concentric Circles | | |
| Graphs for the PC-2 | | |
| July/August 1982 | | |
| Accounts Payable | | |
| September 1982 | | |
| G.E. ASCII Sequential Files | | |
| October 1982 | | |
| Coded Message | | |
| December 1982 | | |
| Christmas Eve | | |
| January 1983 | | |
| Non-Reset of Random No. Generator | | |
| Day of the Week/Monthly Calendar | | |
| February 1983 | | |
| Variable Swapping | | |
| Labels and Renumbering on PC-2 | | |
| Periods to Commas in Data Strms. | | |
| Merge BASIC Prog. on CoCo | | |
| March 1983 | | |
| Renumbering on Models I and III | | |
| High Res Screen Dump CGP-115 | | |
| Additional PMODE3 Color Set | | |
| Music Program Pak (26-3151) | | |
| Plotter Cass. Interface Manual | | |
| Engineering Math II (26-3526) | | |
| April 1983 | | |
| Expressive, Expeditious, X-Pad | | |
| Sieve of Eratosthenes | | |
| May 1983 | | |
| Disk Editor Disassembler | | |
| PC-2 INPUT Statements | | |
| Disk Directory | | |
| June 1983 | | |
| USA Flag for CoCo and MC-10 | | |
| Document Listing for Model II | | |
| Bargraph | | |
| Grid for CGP-115 | | |
| Variable Swapping Routine | | |
| A Tribute to Columbia | | |
| July 1983 | | |
| Record Chess Play | | |
| Invasion | | |

Peripherals

| | | |
|-------------------------|-----------------|----|
| CGP-220 Ink Jet Printer | by Linda Miller | 53 |
|-------------------------|-----------------|----|

Pocket Computer

| | | |
|--|------------------|----|
| PC-2 Assembly Language—Part 5 | by Bruce Elliott | 35 |
| PC-2 RS-232 Communications Device—Part 1 | by Peter Levy | 41 |

Prices shown in TRS-80 MICROCOMPUTER NEWS are in U.S. Funds.

Educational Computing from Walt Disney Productions

By Linda Miller

With thirty years of experience in the educational field and fifty years in the entertainment industry to its credit, Walt Disney Productions now lends its expertise to the field of educational computing. The Walt Disney Educational Media Company has developed eight interactive, educational packages to run on the cassette based, 16K TRS-80 Extended Color BASIC Computer.

These packages cover language arts, math, government, and problem solving. Each package has:

- Excellent high-resolution screen graphics
- Recorded narration—Real voices are interspersed with the program activities.

- Music to create drama and add interest to the presentation

The teachers in the programs include such old friends as Donald Duck, Mickey Mouse, Goofy and many other beloved Disney characters. Even the intrepid crew of the Palomino space explorer offer instruction along the way.

Each package has a story line which makes the learning interesting and fun and creates added incentives for completing the learning activities.

All eight programs utilize the Radio Shack Talk/Tutor system format of instruction. This means that an audio-visual education program is presented to the child, and at various points in the program, the child is asked a question. When the child responds with correct keyboard input, positive reinforcement such as "excellent" or "very good" is displayed. If the response is incorrect, the child is asked to try again.

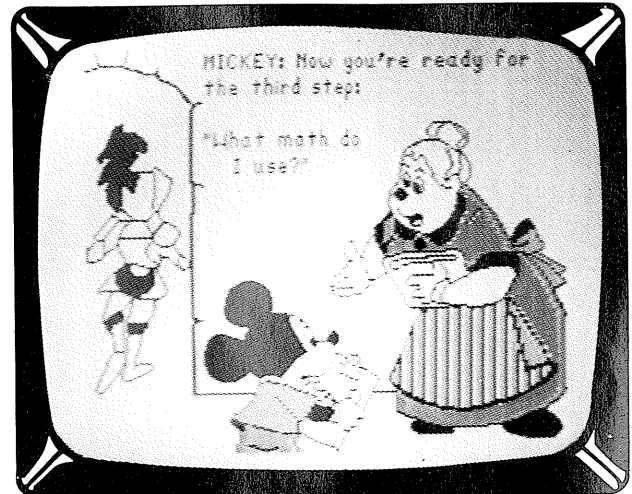
Each software package is accompanied by a player's guide which contains information for parents, directions for loading and using the program, player's instructions including a summary of the story and several follow-up activities. The follow-up activities provide opportunities for the parent and child to review and reinforce the concepts taught in the program.

PROGRAM DESCRIPTIONS

To acquaint you with each package, a brief description follows of each one.

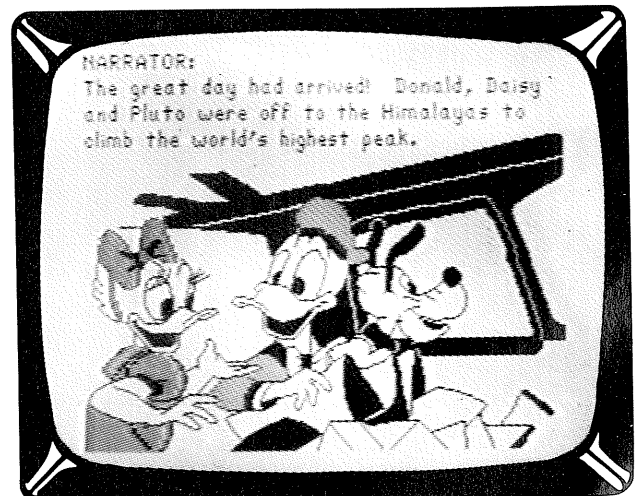
Telling Time with Donald (26-2530) for ages 5-8. This program has the dual purpose of teaching how to tell time to the half hour and introducing telling time on a digital clock. While fishing, Donald Duck and his cousin, Gladstone Gander, discover a magic lamp. When the lamp is rubbed, a friendly genie appears and grants Donald and Gladstone three wishes. Each wish must be made exactly on the half hour.

The terms digital clock, minutes, one-half, and half past are introduced to the student's vocabulary in this program.



From Math Adventures with Mickey, Part 1

Problem Solving with Scrooge McDuck (26-2531) for ages 9-13. Part 1 of this package is Estimating with Scrooge McDuck. When Scrooge takes Huey, Dewey, and Louie to lunch on their birthday, he teaches them a valuable lesson in estimating cost that eventually enables the three boys to recover money previously stolen from Scrooge's bank. The objectives of Part 1 are to: show the value of estimating in everyday money handling, demonstrate that understanding "why" in math is important even when using a calculator, and illustrate that a math answer must be checked for both accuracy and reasonableness.



From Math Adventures with Mickey, Part 2

Part 2, Graphics with Goofy, teaches how graphs help us easily record and present information, the three basic types of graphs (bar, line and pictographs), and the speed with which we can understand information via graphs.

Daisy Duck is acting as Goofy's campaign manager in his bid for the office of mayor. She keeps tabs on voter trends and polling through graphs which she uses to demonstrate to Goofy how well he is faring against his opponent.

Mickey's World of Writing (26-2532) for ages 8-11. In Part 1 Jiminy Cricket narrates this fanciful tale of the Knights of the Alphabet. When the citizens of Wordland complain that the Knight of I is the only "capital" knight, King Mickey announces new laws. These laws state the basic rules of capitalization. Each new law is tried out by the knights with great excitement. The program concludes as all the knights are given the Order of the Capital to be used according to the laws. The Knight of I is still special because the letter I is always a capital when it makes a word by itself. Concepts taught in this program are:

The titles Mr., Mrs., Ms. and Miss begin with capital letters.

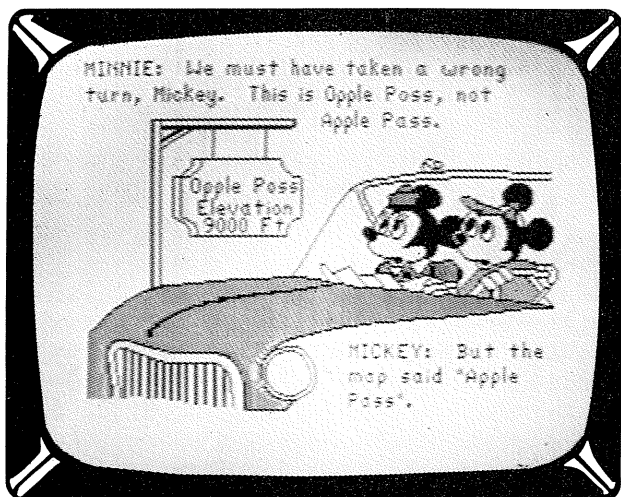
The first word in every sentence begins with a capital letter.

The names of months and the days of the week begin with capital letters.

The pronoun I is always capitalized.

In Part 2, Writing Sentences with Ludwig Von Drake, Louie is writing his entry for the Best Sentence Contest held by the local baseball team, the Duckburg Drakes. All contestants are required to write a sentence explaining why they like the Drakes. Louie has included every reason he could imagine in one sentence. When Professor Ludwig Von Drake comes along, he points out that Louie has not really written a good sentence. Ludwig takes Louie to the baseball stadium where he teaches him how to recognize and write a good sentence. Louie revises his sentence and wins the contest.

The objectives of Part 2 are to: teach the difference between a good sentence and an incomplete or run-on sentence, identify what makes a sentence, point out that a sentence is usually a complete thought, and explain that a sentence must make sense.



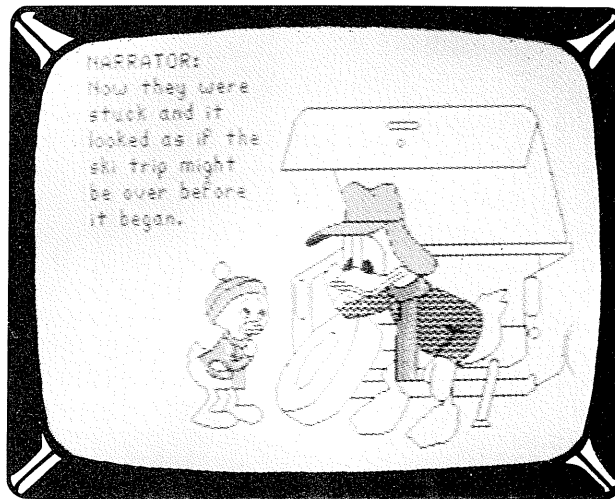
From Mickey's Alpine Adventure, Part 1

Goofy Covers Government (26-2533) for ages 10-14.

As a reporter on his first assignment, Goofy proceeds to learn about the government of the United States. With a tour guide and a bright young tourist, Goofy takes a tour of the Capitol building where he learns about the jobs of congresspersons and senators.

Next Goofy learns the qualifications and responsibilities of the office of President of the United States.

- The purposes of this program are to:
- Understand the importance of Congress
- Distinguish between senators and representatives
- Explain election procedures and requirements
- Define the duties and powers of Congress
- Point out that there are limits on congressional power



From Mickey's Alpine Adventure, Part 2

Mickey's Alpine Adventure (26-2534) for ages 7-10. In Part 1 Mickey and Minnie visit the town of Apple Pass where they find that the town artist has difficulty with short vowel sounds and has misspelled all of the signs in the town. Mickey and Minnie set about to help the artist fix the signs, and in the process, they teach him some key words to help him remember short vowels.

The objectives in Part 1 are to: learn the five short vowels and the specific sound of each, identify and recognize specific short vowel sounds, and practice reading and spelling short vowel sound words.

In Part 2 Donald Duck, Huey, Dewey, and Louie are on their way to a ski lodge when they discover that due to mistakes in spelling, they have forgotten some important items. When they visit a nearby town to purchase the needed supplies, they meet some unusual shopkeepers and learn about spelling with vowel digraphs. As the story progresses, the student will learn to identify long vowel sounds, recognize vowel digraphs, identify the long vowel sound patterns in the digraphs ai, oa, and ea, and practice spelling words which contain the vowel digraphs ai, oa, and ea.

Math Adventures with Mickey (26-2535) for ages 9-13. The two major objectives of this package are to demonstrate an effective method for problem solving and to demonstrate the correct use of decimals.

Part 1 is Problem Solving with Mickey Mouse. Mickey is hired to manage a scary old castle where he discovers that the staff cannot plan intelligently because they don't know

how to solve problems involving math. Mickey and Minnie decide to teach the staff how to solve their math problems by demonstrating a five-step problem-solving method, establishing the need for planning in our daily lives, and encouraging a confident attitude about problem solving.

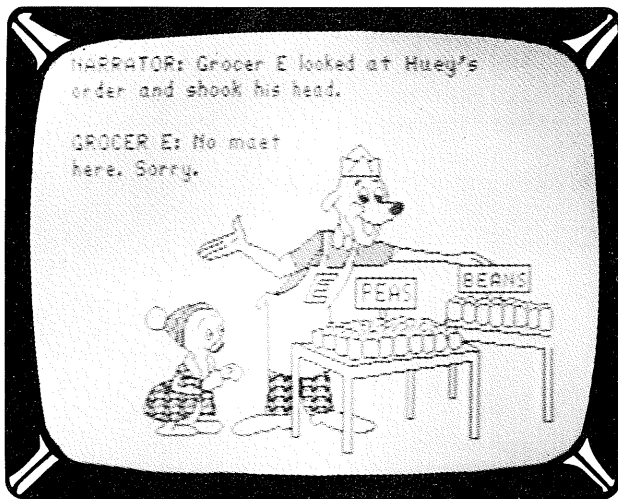
In Part 2, Decimals with Donald and Daisy, these two friends find themselves on an adventure in the Himalayas preparing to climb Decimal point. Donald has forgotten to buy the necessary supplies, but a traveling department store cart opportunely appears carrying all the missing items. Donald makes many calculation errors which necessitates his friends helping him determine the correct prices of his purchases. When an avalanche separates Donald from his friends he finds his newly learned math skills crucial to his survival.

Donald's misadventure is used to teach the concept that math is a necessary skill to be an intelligent consumer. The activities are directed at curing common difficulty with decimals, establishing the need in math for attention to detail, and encouraging students to check answers to see if they are reasonable.

Space Probe: Reading (26-2536) for ages 10-15. This outer-space adventure helps develop important reading comprehension skills.

Part 1, Understanding Cause and Effect, is designed to develop the ability to identify cause and effect situations and to demonstrate the relationship of sentence parts. When the explorer craft Palomino sustains damage in a meteor shower, the crew charts a course for a space station to get repairs.

The space station has a shortage in the parts required to repair the Palomino. The Palomino crew sets out to discover the cause for all the shortages.



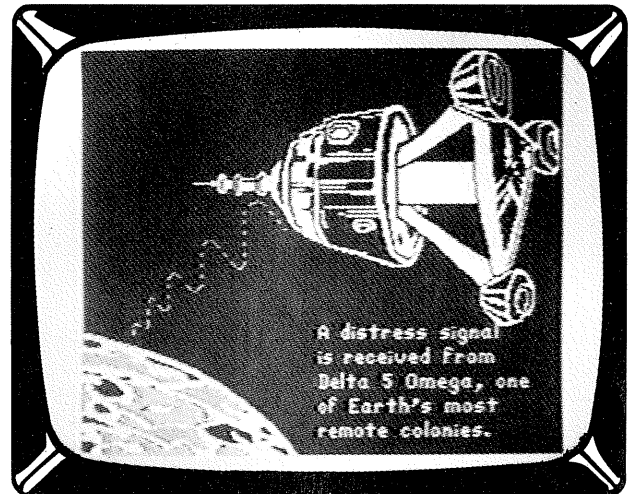
From Mickey's Alpine Adventure, Part 2

In Part 2, Drawing Conclusions, the crew of the Palomino discovers an unexplored solar system. The third planet in this solar system has an atmosphere similar to that of earth. The crew begins exploring the planet only to become captives of R-1000, a master computer, who informs the Palomino crew that he had been directed to run the city and protect all living things. The only effective way that R-1000 has found to protect the living creatures is to lock them up. Only the quick thinking of the Palomino's robot Vincent saves the crew from

captivity. When the Palomino leaves the planet, the crew agrees that they have learned much about drawing faulty conclusions.

The objectives of Part 2 are to: demonstrate the use of facts to draw conclusions, help students develop the ability to make judgements, provide practice drawing conclusions, and apply skills in reading a schedule.

Space Probe: Math (26-2537) for ages 7-14. In Part 1, Problem Solving: Multiplication and Division, the Palomino responds to a distress call from a remote Earth colony. The previously populous colony's only inhabitants are a handful of unconscious survivors lying in capsules in suspended animation.



From Space Probe: Math

The crew of the Palomino has to use multiplication and division skills to determine the number of survivors and the amount of antidote necessary to combat the mind diffusion disease that afflicts them.

Part 1 is designed to demonstrate the concepts of multiplication and division, how to write a number sentence for a multiplication or a division problem, when to use multiplication and when to use division in solving math problems, how to set up a multiplication or a division problem, and how to take a systematic approach to problem solving.

In Part 2, Problem Solving: Area and Perimeter, the Palomino explorer ship is exploring still another unknown planet. The inhabitants are about to lose their crops to the threatening eruption of a nearby volcano unless something can be done to protect their fields.

The terms area, base, height, and perimeter are defined while using both a rectangle and a square. The formulas for determining area and perimeter are also taught.

FINALLY . . .

All of these programs are interactive so the student is drawn into the problem solving process. Student responses to the problems or situations become an integral part of the presentation. The graphics are outstanding and the voices and music in the story make the programs as entertaining as they are educational. Each of the eight packages are sold at the suggested retail price of \$34.95.

Disney software and characters © 1983 Walt Disney Productions.

Learning Can Be Child's Play

Children's Computer Workshop Develops Educational Software for Radio Shack

For thirteen years Children's Television Workshop has demonstrated successfully the philosophy that one of the greatest vehicles for learning is fun. The computer games developed by Children's Computer Workshop, a wholly owned subsidiary of Children's Television Workshop, continue this tradition of combining fun with education. The first two clusters of Children's Computer Workshop games for use with the Radio Shack TRS-80 Color Computer, using Extended Color BASIC, are now available through all Radio Shack stores.

The first cluster of four activities is designed to present "Basic Pre-School Skills" to children ages 3 through 6. The second cluster is a series of three "Cooperation and Strategy" games for children ages 7 and older that reward collaboration rather than competition. A third cluster for ages 10 and older will be distributed later this year.

Children's Computer Workshop's computer games have extended the goals, pioneered by Children's Television Workshop in television with "Sesame Street," "The Electric Company" and "3-2-1 Contact."* The aim is to create software that is wholesome and engaging, encouraging children to play constructively and learn actively. Children's Computer Workshop games allow children to experiment and explore, to think and solve problems, and to practice skills while they are having fun. The games are designed to be a family experience where children and parents play together and learn from each other.

Both Children's Television Workshop and Children's Computer Workshop were founded with the same goal in mind: to use technology that was beguiling to the eye, mind and imagination of a child and use it as a tool for engaging young minds.

When Children's Television Workshop was established, that new technology was, of course, television. "Sesame Street," the first program produced by Children's Television Workshop, brought Ernie, Bert, Big Bird and all the wonderful Muppets to popularity on TV, and has won more awards than any children's show in history. "Sesame Street" was seen in over ten million American households last year. After "Sesame Street," Children's Television Workshop created "The Electric Company," designed to build reading skills in 6 to 10 year olds, and "3-2-1 Contact" which aims to stimulate the interest of 8 to 12 year olds in science and technology. No company has done more research and testing of children's response to the television screen or gained greater insight into learning capabilities and entertainment preferences of children.

The computer has brought an exciting new dimension to electronic learning—that of interaction. A great deal of planning and research have gone into making Children's Computer Workshop games responsive to children of different ages and learning styles. Children's Computer Workshop products are different from many other software products in purpose—learning while playing - and their creation process is entirely different as well.

Children's Computer Workshop games come from teamwork, harmonizing the diverse contributions of a skilled, creative staff, educational specialists, and research and testing with children themselves. Whenever possible, their level of difficulty, pacing and direction of the games are under the control of the player. So, as the child responds to the computer, the computer responds to the child.



Photo 1. Grover's Number Rover (Ages 3-6).

ACTIVITIES FOR PRE-SCHOOLERS (AGES 3-6)

The "Basic Skills Series" of games, for children ages 3 through 6, focuses on four important pre-school skill areas: working with numbers, letters, matching shapes and colors, and classifying objects.

The four games in the "Basic Skills Series" are:

Grover's Number Rover™ **, which lets children play with numbers in an engaging and fun environment. Each of the six activities is designed to provide increasing levels of challenge, while allowing the child to explore number operations and number facts. Children play with the basic operations that form the foundation of later mathematical skills.

Cookie Monster's Letter Crunch[™] ** lets kids match letters and words in order to feed the Cookie Monster—and you know he's always hungry! The fun of moving Cookie Monster from letter to letter and watching him eat will encourage children to practice letter recognition and letter sequencing skills.

Ernie's Magic Shapes[™] ** helps children to sort out the things they see, in a process called visual discrimination. By looking closely at Ernie's figures, children can match shapes, structure parts into meaningful wholes, recognize embedded figures and notice similarities and differences.



Photo 2. Ernie's Magic Shapes (Ages 3-6)

Big Bird's Special Delivery[™] ** provides a playful setting in which children can practice an important skill—classification. In order to help Big Bird deliver packages to the right stores, a child must consider and compare objects according to the attributes of form and function.

DESIGNING SOFTWARE FOR PRE-SCHOOLERS

Children's Computer Workshop executive producer, Dan Oehlsen, explains how Children's Computer Workshop approached the design of software for pre-schoolers. "Working with numbers, letters, matching shapes and colors, and classifying objects are the kinds of activities that parents and teachers immediately recognize and the things they want their pre-school children to learn about. We decided to concentrate on these areas because they match specific goal areas for the 13th season of 'Sesame Street.' They draw on some of the same skills that are part of the television show, but the computer allows for a level of interaction that is impossible with television."

Oehlsen also knew that in addition to working in a new medium, computer programs for pre-school children are a very new idea. "There are just not very many products out there for children in this age range, and I think many people are skeptical about whether computers are appropriate for children this young. So, we decided to stay with traditional and accepted educational goals. We wanted to do some things with the computer that could not be done with other media. We were not in any way trying to suggest these programs should replace coloring books, or paper and pencils, or blocks, or paints or teachers. Rather, our games should provide kids in this age group with another opportunity to play with and to learn as they play with a certain set of skills." The Children's Computer Workshop production team used the ability of the computer to give children immediate feedback on what they are doing and the flexibility to move

from one level to another that a computer allows to give kids a new experience with these traditional skill areas. "We were not trying to teach numbers or letters," says Oehlsen. "We were trying to give pre-schoolers an environment in which they could play with concepts they may have learned from their teacher, or from their parents or doing other things. We took traditional educational goals and some of the unique things that a computer can do and put them together in a way that works for pre-schoolers."

One big question that Children's Computer Workshop faced in developing these products was what was going to happen when they took a task that they knew a pre-school child could do, and put a computer between the child and the task. They knew, for instance, that pre-schoolers can count and stack blocks. That is a straightforward task. Depending upon the age of the child and the level of ability, it is something that is relatively easy for a pre-schooler to do. But, what happens if a child has to count something on a computer screen and then has to push a button on the keyboard in order to have that count register? Does that change the task in a way that makes it more confusing to the child?

"The first question that we had to ask was how we could use the computer, the keyboard and the screen to expand the child's ability and not create a roadblock to learning," Oehlsen explained. "We spent a lot of time working with prototypes of the programs, going out and testing in pre-schools. We wanted to see if three-year olds could find an arrow key on the keyboard if we showed them an arrow on the screen. If three objects came out on the screen could the children count them and then find the appropriate numeral on the keyboard? Did that sequence of events make sense to them? Could they make the association between the number of objects and the symbols on the keyboard? By breaking the tasks down into fine components and continually testing with kids from the very beginning of the software development process, I think what we have done is put together four programs that work very well for children, as well as work very well with children. We have learned that kids are very comfortable with and really enjoy our programs, that the computer facilitates their interaction with the concepts and that they learn the mechanics of operating the computer very, very quickly. Little kids in this age range have no fear of computers."

EXTENSIVE RESEARCH AND TESTING

What Children's Computer Workshop has found while testing these programs is that on the very basic level, the level for the three-year-olds, simply being able to push a button on the keyboard and have something predictable happen on the screen fascinates them. In *Grover's Number Rover*, for instance, when three-year-olds play, they can push the arrow key and a "Twiddle Bug" comes racing across the screen. They push another arrow key and it goes off the screen. Oehlsen reports that three year olds find their simple task engaging and have been observed to play this way on their own, exploring the computer environment, for up to a half hour at a time. He feels that the control over the elements of the activity that the computer allows is a very powerful concept for young children.

"If the computer is programmed to be responsive to children, it becomes an incredibly intriguing device for learning. Kids will sit and explore and play with it as they try new

things. The computer does not replace other techniques for helping children learn; it does open another door for them and, most importantly, the kids have a lot of fun."

CO-OPERATIVE COMPUTER GAMES (AGES 7 AND OLDER)

The second cluster of Children's Computer Workshop games are designed for children ages 7 and older. Entitled the "Cooperation and Strategy Series," these three games encourage players to work toward a common goal. Children play in an environment where they may share information, divide responsibilities, and build on one another's strengths. The three games in the "Cooperation and Strategy" series are:

Peanut Butter Panic™ *** is a two player game in which success depends on cooperation. As the players jump for stars to make peanut butter sandwiches, they soon find that they must work together to catch the most valuable stars, and they must share the sandwiches they make to maintain their jumping energy. Teamwork develops naturally and is well-rewarded. Players must plan their jumps carefully to move to higher levels of the game as they work together and share resources.

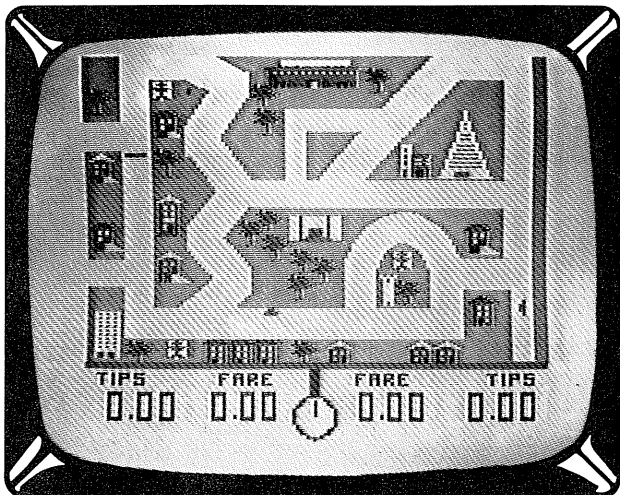


Photo 3. Taxi (Ages 7 and Up).

Taxi™ *** allows kids to get "behind the wheel" of a cab. By maneuvering around street grids based on city maps, players try to deliver as many passengers as possible before their time is up. When played cooperatively, the game encourages communication and division of labor. As players develop mutually effective strategies, their scores will increase.

Star Trap™ *** is a dynamic maze game where cooperation really pays off. The challenge is to trap a shooting star by blocking the paths with x's and using the special maze gates. One player can make a trap, but two players working as a team can play more efficiently. By talking and planning together, players will trap the star more quickly and move to higher levels in the maze.

In creating the "Cooperation and Strategy" game series, Children's Computer Workshop established an educational goal, but education in a very broad sense of the term. They set out to create a set of games that kids could play cooperatively and still have a good time. Dan Oehlsen explains,

"What we tried to achieve in the 'Cooperation and Strategy Series' was a little different than what we did in the pre-school cluster. We did not establish an educational goal that was tied to a content area or a curriculum area like spelling or math. We chose a broader area, but one we felt was just as important.



Photo 4. Taxi (Ages 7 and Up).

"Our first objective was to create a group of games that would be a lot of fun to play. Games that would give kids the same sort of interactive excitement that they get playing other video-games. We wanted these games to have that type of strong appeal." Oehlsen feels that what they have accomplished in this series of games is very similar to what "Sesame Street" does on the screen. "First of all, it is a really fun show to watch. We know that if kids will not watch the program, the content does not really matter. They will have no opportunity to experience it. So, our first objective was to develop a set of three games that children would like to play. After that, we wanted to design games that kids would enjoy playing together, cooperatively."

In each of the three games in this series the way players succeed is to cooperate with their partners. They are all played by two players simultaneously, and in every case, the players reach a point where, in order to do better in the game, they have to work together. Children's Computer Workshop research has shown that soon after beginning play the kids discover that working against one another is a very limiting strategy.



COOPERATION AND COMPETITION

The Children's Computer Workshop folks do not suggest there is anything wrong with kids playing competitive games. "What we set out to do with these games was to provide some alternatives," says Oehlsen. "Competitiveness is certainly a positive thing in some circumstances, but in other situations, cooperation and collaboration with another person may be a good strategy also. There are lots of competitive games out there. We wanted to demonstrate that it is possible to play and have fun in a cooperative environment. Some games that we make in the future may involve competition. But, we really think there should be more of a 'balance'."

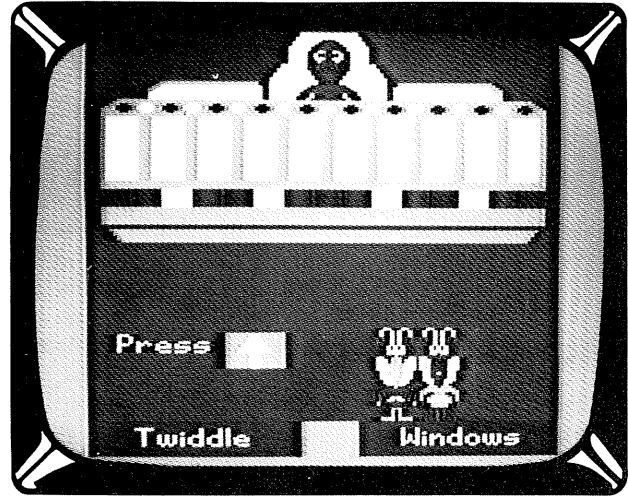


Another important issue concerning computer games is that of isolation of the individual. "Sometimes kids want to play by themselves. Sometimes they want to have something they can do on their own, away from other people. That can be a good thing. Again, we think these should be alternatives for children. We think this is a medium that should not necessarily isolate people. It can be a medium that facilitates communication between people. We set out to create a group of games that would use the computer medium to get children to talk to one another and work together. In fact, that is what happens in these games. When kids play them, they are usually very quiet in the beginning and then, a couple of minutes into the game, an amazing thing starts happening—they start talking to each other. Because they have to communicate to go any further, to improve their score, the cooperation and collaboration happen naturally, as a part of game play. Once they get to that point, there is a lot of interchange between the kids throughout the rest of the game. We think that is a positive thing and good for the kids. The same kids might want to play a game that allows them to play by themselves another time, but we feel that this alternative is a unique and important one for children to have."

Children's Computer Workshop researchers have gotten extremely favorable feedback on these games from parents. They enjoy playing them with their children and are pleased that they have at least one cooperative strategy game in the library of games they have for their kids.

The people at Children's Computer Workshop believe that children can learn through play. They view children as their partners in the development of their products. Their software is thoroughly tested for age appropriateness, appeal, accessibility, and comprehension of learning goals.

Their games are designed and redesigned to best serve the needs of kids, and resulting products are unlike anything else on the market.



*Sesame Street, Electric Company, and 3-2-1 Contact are trademarks of Children's Television Workshop.

**Trademarks of Muppets Inc.

***Trademark of Children's Computer Workshop Inc.
CCW and Children's Computer Workshop are trademarks of Children's Computer Workshop Inc.

Color Computer Draw Statement

Mike Kim
CIS 70615,1040

Here is a hint for using the DRAW statement with the Color Computer. When using the DRAW statement, you can use variables rather than absolute numerals with the motion commands. All you have to do is take the direction (U,D,L,R,E,F,G, or H), put an equal sign and then the variable (i.e. U = A where A is the variable). As far as I can tell, you can use the variables B, M, U, D, R, L, etc.

Editor's Note: The last variable in U, D, R, L, etc. must have an absolute value or an FC error will occur.

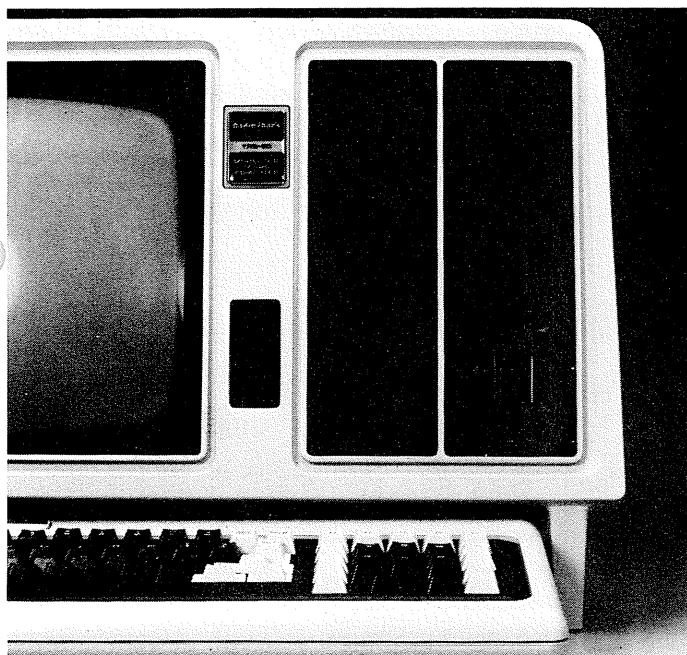


The New Model 16B with 15-Megabyte Hard Disk Built-In

by Annette Zamberlin-Main

One expression of the state-of-the-art in 68000 microprocessor technology is Radio Shack's Model 16B. It has proven to be well-suited for the office of today, providing simultaneous job-handling without the expense involved in purchasing and utilizing multiple microcomputers.

The 16B offers high operating speeds and large memory capacities that numerous business users require in their particular environments. It can handle complex computational jobs and highly sophisticated programs.



Now to all of this, Radio Shack introduces an innovation of revolutionary proportions, a 256K 1-Floppy Disk Model 16B with a built-in 15-Megabyte Hard Disk (Cat. No. 26-6006) at a cost of only \$6999.00.

That's right, built into the physical 16B microcomputer cabinet is a 15-megabyte hard disk drive.

Just imagine the power and capacity of this machine! You can access up to 15 million characters on the hard disk. And there is the ability to add another external hard disk drive for a total of 27 million characters of on-line hard disk storage.

Compare the old price (RSC-9, page 9) of a 128K Model 16 with 1-floppy disk drive and an external 12 Meg hard disk drive for \$8494.00. Let's see, \$8494.00 less \$6999.00 is \$1495.00. That is a significant price drop especially when considering all the features of the 16B.

MODEL 16B FEATURES

In addition to the 16 bit 68000 there is a second processor, the Z-80A. In the 16B the Z-80A 8-bit processor handles input and output plus a variety of other housekeeping chores.

PORT CAPABILITY

This computer has one standard parallel printer port and two RS-232C serial communications ports built-in which allow interfacing the 16B to modems, digitizers, serial printers, and a variety of other peripherals. The 16B also has high-resolution graphics as an option.

THE KEYBOARD

The 16B has a detachable typewriter-style eighty-two character keyboard with a numeric keypad. The eight special function keys are programmable so often repeated tasks can be performed with the stroke of a single key. Other special keys include HOLD, ESCape, BREAK, CTRL, CAPS, and REPEAT plus up, down, right, and left arrow keys.



WHAT A MEMORY!

The 16B has 256K RAM bytes of memory, expandable to 768K. That is a lot of memory! There is 64K Input/Output memory and 2K video memory.

VIDEO DISPLAY

The 16B has a high-resolution 12" green display monitor which allows you to display either 80 or 40 characters per line with 24 lines per screen. It displays upper and lower case characters with true descenders on the video. It also will display an additional 32 "business graphics" characters. There is automatic scrolling plus an available partial screen scroll protection.

Fast Data Lines for Models I/III/4 and II/12/16

Barney M. Tennyson, CPA
146 South Battery
Charleston, S.C. 29401

Using the programs NEWLINE and TEXTLINE, data lines can be entered free of syntax and continuity problems and with less typing on the operator's part. Specifically developed to run on either the Models I/III/4 or the Models II/12/16, these two programs allow you to make corrections to data during the entry process.

NEWLINE

In NEWLINE numerical information can be entered with ease from the keypad since all separating commas are added by the program. Each data item is entered individually and stored in the same data line until you type in #. A # indicates completion of data entry in the current line and causes the next line number to be displayed. Each line number is incremented by 10. Entering END stops data from being entered and causes the data lines to be written to a disk file named TEMPXFER. If you discover that you made a mistake on the previous item of data, just type XX instead of the next data item and you can type that item over again.

If NEWLINE is to be used without the character limitation routine suggested below, it would still be wise to insert a caution line following line 10160 to alert you that the line is nearly full. For example:

```
10165 IF LEN(T$(N)) => 240 THEN PRINT "WARNING....."
```

(The character limitation number—240 above—depends on the nature of the information being entered.)

You may want to use the following suggestions which can be successfully added to the primary routine.

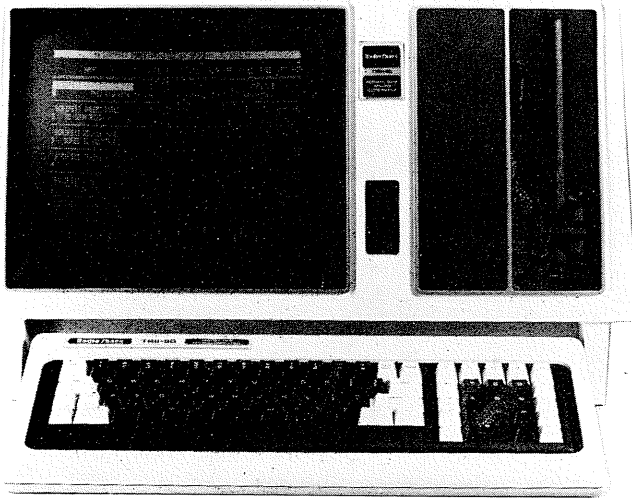
1. Add a live keyboard input subroutine to limit the length or nature of individual items and to control the length of the line.
2. Customize the input process at lines 10060 through 10120 to provide descriptive prompts for the information required in a particular application.
3. When lines are to be entered consecutively without any interval, eliminate the prompt for the starting line number and have the program set the starting line. For example, using an assigned data block with a single item on each line:

```
1000 '-----START OF DATA BLOCK-----  
1999 DATA END
```

Read and discard the data lines already in place with a counter variable, and set SN equal to 1000 plus the counter variable.

4. Display operator instructions for continuation of the main program just before line 10190.

```
100 ' *** TEMPORARY LINES 100-116 ***  
110 CLEAR 5000  
112 DEFINT A-Z  
113 DIM T$(100)  
114 CLS  
115 PRINT"ADD DATA LINES TO THE RESIDENT PROGRAM:"  
116 PRINT  
10000 ' ** SUBROUTINE TO ADD DATA LINES **
```



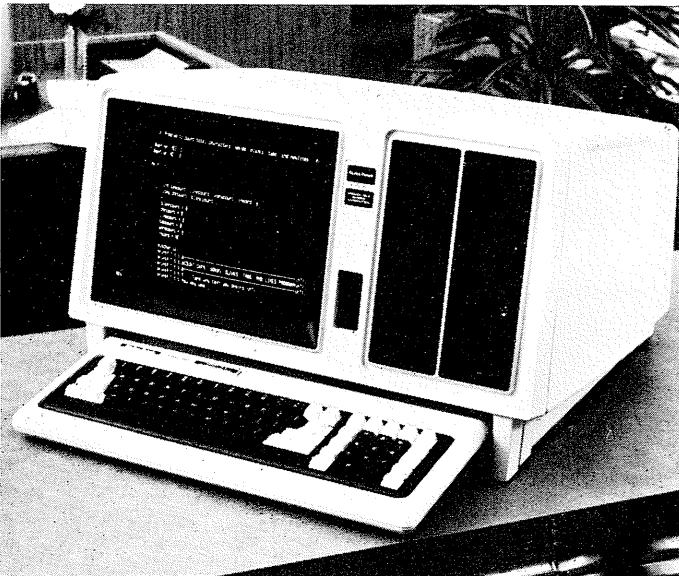
SOFTWARE COMPATIBILITY AND EXPANSION SLOTS

The Model 16B can run existing Model II/12 software in the single user mode. It will also run new software which has been designed to take advantage of the Model 16B's advanced features.

Add to this two more expansion slots (4 in the 16B, 2 in the 16) in the user accessible card cage. The hard disk uses one of the expansion slots.

OPERATING SYSTEMS

The Model 16B includes two operating systems: TRSDOS for single user environments and TRS-Xenix Core System Version 1.3 which is a powerful multi-user operating system.



WHAT A VALUE!

The 16B is a tremendous value in a computer. It is economical in that it provides simultaneous job-handling without the expense of multiple computers. The Model 16B is for the individual who wants the most advanced microcomputer technology at his fingertips.

```

10010 OPEN"O",1,"TEMPXFER"
10030 INPUT"STARTING WITH LINE NO. ";SN
10040 FOR N=1 TO 100
10050 PRINT
10060 PRINT"Entering Line Number "SN
10070 SN$=STR$(SN)
10080 L=LEN(SN$)-1
10090 SN$=RIGHT$(SN$,L)
10100 T$(N)=SN$+" DATA "
10110 PRINT
10120 INPUT"Enter Data Item ";D$
10130 IF D$="# " THEN GOSUB 10200
: NEXT N
10140 IF D$="END" OR D$="end" THEN GOSUB 10200
: GOTO10180
10150 IF D$="XX" OR D$="xx" THEN GOSUB 10280
: GOTO 10120
10160 T$(N)=T$(N)+D$+" , "
10170 GOTO 10120
10180 CLOSE
10190 MERGE"TEMPXFER"
10200 ' *** TO WRITE LINE TO FILE ***
10205 Z$=SN$+" DATA "
10210 CLS
: IF T$(N)=Z$ THEN SN=SN-10
: N=N-1
: RETURN
10220 L=LEN(T$(N))-2
10230 T$(N)=LEFT$(T$(N),L)
10240 PRINTT$(N)
10250 PRINT#1,T$(N)
10260 SN=SN+10
10270 RETURN
10280 ' *** ITEM CORRECTION ROUTINE ***
10285 Z$=SN$+" DATA "
10290 IF T$(N)=Z$ THEN N=N-1
: SN=SN-10
: FL=1
10300 L=LEN(T$(N))
10310 FOR X=(L-2) TO 1 STEP -1
10320 IF MID$(T$(N),X,1)="," THEN 10500
10330 NEXT X
10340 IF FL=1 THEN PRINTT$(N)
: GOSUB 10400
: RETURN
10350 T$(N)=SN$+" DATA "
10360 PRINTT$(N)
10370 GOSUB 10400
10380 RETURN
10390 ' *** PRINT LINE FOR CORRECTION ROUTINE ***
10400 PRINT"CORRECTION OF PRECEDING ENTRY"
10410 FL=0
10420 RETURN
10500 ' ** PREPARE LINE FOR CORRECTION **
10510 T$(N)= LEFT$(T$(N),X)+" "
10520 PRINTT$(N)
10530 GOSUB 10400
10540 RETURN

```

TEXTLINE

With TEXTLINE each data line will contain a single item of data. It is particularly useful for form letters and standard documents which are regularly used. The use of double quotes after the word DATA in the program permits the use of commas and other prohibited characters within the data string. As in NEWLINE, XX lets you correct the previous line and END stops DATA line entry and causes the DATA lines to be written to the disk file TEMPXFER.

```

100 ' *** TEMPORARY LINES 100-116 ***
110 CLEAR 5000
112 DEFINT A-Z
113 DIM T$(100)
114 CLS
115 PRINT"ADD DATA TEXT TO THE RESIDENT PROGRAM:"

```

```

116 PRINT
10000 ' ** SUBROUTINE TO ADD DATA LINES **
10010 OPEN"O",1,"TEMPXFER"
10030 INPUT"STARTING WITH LINE NO. ";SN
10040 FOR N=1 TO 100
10050 PRINT
10060 PRINT"Entering Line Number "SN
10070 SN$=STR$(SN)
10080 L=LEN(SN$)-1
10090 SN$=RIGHT$(SN$,L)
10100 T$(N)=SN$+" DATA "+CHR$(34)
10110 PRINT
10120 LINEINPUT"Enter Data Item ";D$
10130 IF RIGHT$(D$,1)=CHR$(13) THEN GOSUB 10200
: NEXT N
10140 IF D$="END" OR D$="end" THEN GOSUB 10200
: GOTO10180
10150 IF D$="XX" OR D$="xx" THEN SN=SN-10
: N=N-1
: PRINTT$(N)
: GOSUB 10390
: NEXT N
10160 T$(N)=T$(N)+D$+" , "
10170 GOSUB 10200
: NEXT N
10180 CLOSE
10190 MERGE"TEMPXFER"
10200 ' *** TO WRITE LINE TO FILE ***
10205 Z$=SN$+" DATA "+CHR$(34)
10210 CLS
: IF T$(N)=Z$ THEN SN=SN-10
: N=N-1
: RETURN
10220 L=LEN(T$(N))-2
10230 T$(N)=LEFT$(T$(N),L)
10240 PRINTT$(N)
10250 PRINT#1,T$(N)
10260 SN=SN+10
10270 RETURN
10390 ' *** PRINT LINE FOR CORRECTION ROUTINE ***
10400 PRINT"CORRECTION OF PRECEDING ENTRY"
10410 FL=0
10420 RETURN
10500 ' ** PREPARE LINE FOR CORRECTION **
10510 T$(N)= LEFT$(T$(N),X)+" "
10520 PRINTT$(N)
10530 GOSUB 10400
10540 RETURN

```

Frustration

J. De Augustine
2275 Grove Way #19
Castro Valley, CA 94546

Since I saw a poem submitted by a reader in your January issue, I thought perhaps you might like to share a few of my thoughts with your readers.

I bought a home computer,
Thinking even I could learn
This mysterious new language
That appeared at every turn.

In a week I was addicted:
At the keyboard day and night,
Using strings and variables,
But nothing came out right.

Futility has made me doubt
My ability to learn.
I may have ample GO-SUBS
But I'm short a few RETURNS!

The Pod Concept in Classroom Networking

By Warren Hornsby, Jr.

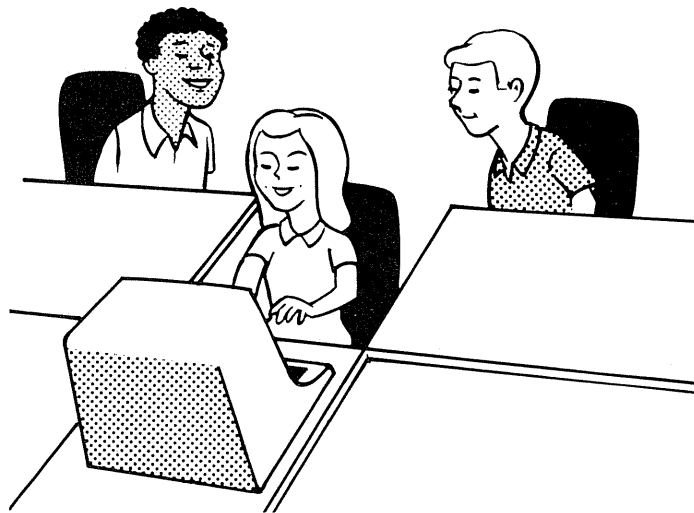
Editor's note: Warren Hornsby is a teacher and systems programmer who independently produces and sells special-features operating system software for use with Radio Shack Network 2 systems. For more information, write to Hornsby at "ClassWare", 8192 N. Dilcrest Ct., Florence, KY 41042. While the article below describes a Network 2 configuration with Model III student stations, the same arrangement could be used with other Radio Shack Network 2 or Network 3 configurations.

This article describes a special arrangement of tables and computers that allows one Radio Shack TRS-80 Network 2 system in a classroom or resource center to serve 30 students at once. Hardware required is:

- Ten Radio Shack TRS-80 Model III or Model 4 computers with at least 16K memory each
- One Radio Shack TRS-80 Model III or Model 4 disk system with at least 32K memory
- One Radio Shack Network 2 Controller, with all the necessary connecting cables that come with the controller.

THE SET-UP

A special system of grouping students around the network's computers (called the "pod" concept) is used to reach the goal of teaching the maximum number of students that can effectively be taught with the minimum amount of computer hardware.



Pictured above is a typical "pod" of three students. Notice that each member of the pod has a direct line of sight to the computer screen. While not physically separated from

the teacher or from the rest of the class, the "pod" is an entity of these three students working and learning together.

At the end of whatever time period the teacher dictates, the pod should rotate (see illustration of THE POD CONCEPT, below). Normally the rotation is clockwise; often on a daily basis. What this means to each student is that every third day the computer is his or hers. The other two days, the student is not just an observer, but a vital member of the "pod." (During testing or short exercises, of course, the pod may rotate more often.)

THE BENEFITS

The pod system has been in use in several schools, where the following benefits have been noted:

1. While a one-to-one ratio of student-to-teacher is often voiced as the ideal, this is not necessarily true in the student-to-computer relationship, especially in younger (primary and junior high) grades. It has been observed that in both programming and Computer Assisted Instruction classrooms, that students can sometimes get lost or confused with some simple procedures for operating the computer. With two other "pod" members, this rarely occurs.

2. The "three against one" (students vs. computers) atmosphere of the pod is very reassuring to new computer students and goes a long way towards relaxing them in their relationship with the computer. (The computer is, after all, almost human . . .?)

3. If a real problem should occur in the lesson, a pod is much more likely to alert the instructor than a lone individual might be.

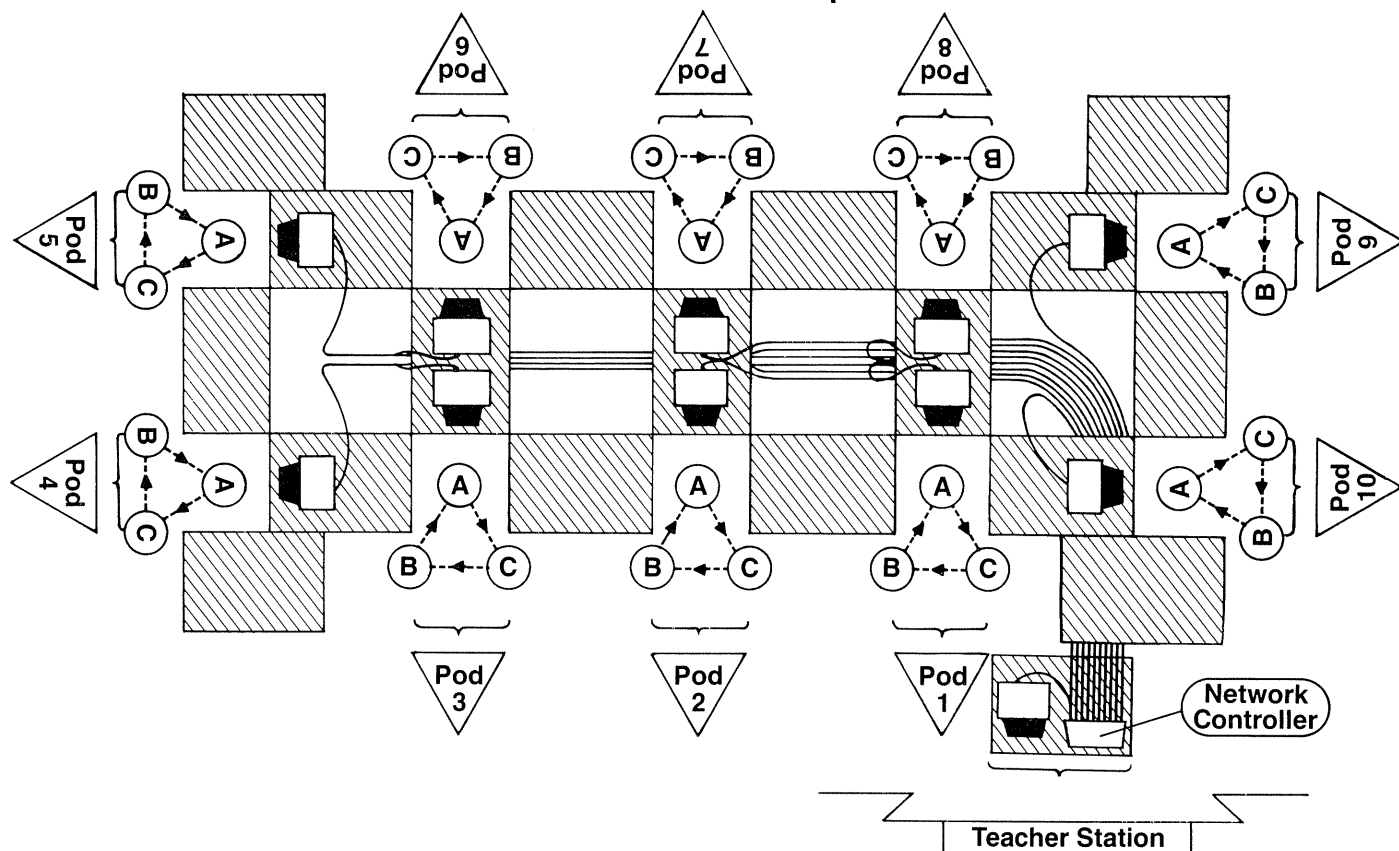
4. Nearly three times the learning takes place as the student not only does his or her lesson or test, but also observes as other pod members do their lessons.

5. "Brainstorming" quickly becomes the norm when programming and debugging get underway.

6. The usual practice of "lining the outside walls" with computers in the typical computer lab is no longer necessary to avoid tripping hazards from wires on the floor. The floor plan of the pod system illustrates the complete lack of wires on the floor where foot traffic could be expected; this works especially well if Radio Shack power strips are used to supply power to all computers. If no floor plugs are available, then one power cable will have to exit from the pod arrangement at some point close to a wall outlet. The cable should be covered and shielded from foot traffic.

7. Finally, for less than what is usually spent on textbooks, and for about half of what is spent on setting up the average typing lab, 180 students (30 x 6 class periods) can be given

The "Pod" Concept



extremely comprehensive and powerful exposure to programming, CAI, remediation, tutorials, and anything else associated with having computers available on a truly "workable" basis.

CONCLUSION

All in all, the "pod" system of arranging the computer

classroom has many good features, not the least being the maximum coverage of the student population with the maximum computer exposure at the minimum cost. At higher grade levels and/or in advanced programming classes, of course, the closer to a one-to-one ratio of computer-to-student, the better.

Computer Clubs

The Color Byte Club
 c/o Randy Blunt
 869 Innes Ave.
 San Francisco, CA 94124

The Color Computer Club
 of San Francisco
 c/o Thai Howard
 1771 Page St.
 San Francisco, CA 94117

Color Computer Users Group
 c/o A. Arnold Weiss
 Apt. 1626 Kennedy House
 1901 J. F. Kennedy Blvd.
 Philadelphia, PA 19103

Halifax-Dartmouth Color Computer
 Users Group
 P.O. Box 572
 Dartmouth, N.S. B2Y 3Y9
 Canada

Kansas City TRS-80 Users Group
 c/o Mary Youngblood
 300 N. W. 83rd Street
 Kansas City, MO 64118

Portland Area TRS-80 Users Group
 P. O. Box 02500
 Portland, OR 97202

San Antonio TRS-80 Users Group
 14310 Pembridge
 San Antonio, TX 78247

Shippensburg Color Computer Club
 c/o Dept. of Mathematics &
 Computer Science
 Shippensburg University
 Shippensburg, PA 17257
 1-717-532-1406

South Brevard Color Computer Club
 c/o Dossey E. Evans, III
 2727 N. Wickham Road, 10-203
 Melbourne, FL 32935
 1-305-254-0575

Ventura County Color Computer Club
 c/o Carol Simpson
 524 Kitty Street
 Newbury Park, CA 91320
 1-805-499-3055

OS-9—A New Color Computer Operating System

By Bruce Elliott

The following information is a composite of quotes from the OS-9 documentation, paraphrases from that documentation and other relevant materials.

WHAT IS OS-9?

OS-9 is a versatile operating system for the 64K TRS-80 Color Computer. It is based on the UNIX operating system developed by Bell Laboratories Inc. Unix is widely used on larger computers, including the Radio Shack Model 16 which runs the UNIX based Xenix™.

OS-9 opens many new doors by expanding the Color Computer's capabilities. OS-9 offers sophisticated features that are normally available only in much larger computers. Among these features are:

Multi-Level Filing System—Like most operating systems, OS-9 lets you store information on disk in a "file" and index these files with a directory. OS-9, however, goes one step further by letting you create a hierarchy of directories and files.

Multi-user/Multi-tasking Operation—Multi-user means that more than one person can use the system at the same time. The number of users is limited by the number of terminals. The TRS-80 Color Computer can have one terminal; this means that two people can use OS-9 at the same time, one person on the Color Computer and one on the terminal.

Multi-tasking means that two or more tasks (programs) can run at the same time. For example, with OS-9 you could print reports and enter information at the same time.

Device-Independent Input/Output System—OS-9 uses a very efficient method for inputting and outputting information. OS-9 expects all input to come from the "standard input device" and all output to go to the "standard output device." On the Color Computer, OS-9 expects all input to come from the keyboard/console and all output to go to the video display. You can easily "redirect" the standard I/O devices to other devices such as printers or disks. This means that an OS-9 program needs only one output routine and one input routine. From there you can redirect the routines to other devices. This saves time for the programmer and space on the disk because programs can be shorter.

WHAT IS THE ADVANTAGE OF OS-9?

The first thing to do in looking at OS-9 is to answer the question "What is an operating system?"

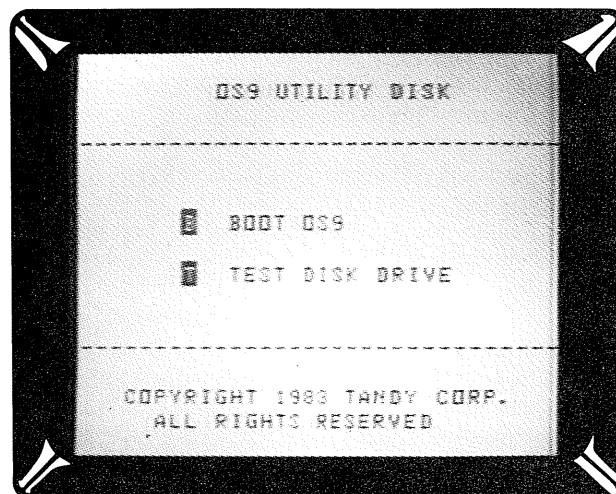
An operating system acts as a manager for your computer. It sends information to the disk drives, printer and video. The operating system manages the storage space on your disks and in your computer's memory. The operating system also responds to your commands.

OK, an operating system manages the computer. What does that have to do with the Color Computer?

If you presently have a Color Computer disk system, you may not be fully aware of the computer's operating system. Radio Shack deliberately made the operating system under Color Disk BASIC "invisible" to you in normal operations. If you want to format a disk you simply ask BASIC to initialize the diskette using DSKINI. BASIC then goes out to the proper drive, initializes the diskette and returns control to you in BASIC. There is no need (or way) for you to move out of BASIC and into the operating system.

Under "most" operating systems, including those of the Models 4/12/16, BASIC is simply one of many programs which is available to you for use under an operating system. To format a diskette you use a disk format utility, which is NOT part of BASIC. In Color Disk BASIC, the operating system is "hidden" in BASIC so that you do not have to deal with it.

Under OS-9, you bring up the operating system by "booting" it from the operating system diskette. OS-9 loads into your Color Computer and takes control. BASIC is gone, as is the hidden nature of the operating system. Under OS-9 you will have to learn how to function within the operating system, if you are going to successfully use the power of the system.



If your main thing is running program paks, or if you buy your programs off the shelf, then you probably don't have a lot of need for OS-9. Color Spectaculator (ROM or Disk) will still function and will still do the same functions that they have always done. Color Scripsit will continue to function and does not need OS-9. If you write your own programs, or if you want to learn a lot more about your Color Computer, then OS-9 may have some advantages for you.

At the time I am writing this article, the only OS-9 product is the bundled package of the OS-9 operating system and the Editor/Assembler/Debugger. By the time you are reading this, BASIC09, a compiled BASIC for use under OS-9, should also be available. The first products that you can expect for OS-9 are languages. First BASIC09, a little later we expect to have some other popular languages available. It will take some time for all of us to get comfortable with OS-9 on the Color Computer and for applications programs to begin to appear.

Does this mean you should not buy OS-9? NO! It only means that OS-9 is an advanced operating system which will unleash the power of the Color Computer. If you are still uncomfortable with Color BASIC, or if you have no intentions of writing your own programs, then your money might be better spent on a different product.

WHAT DO I NEED TO RUN OS-9

The hardware required to run OS-9 includes a 64K TRS-80 Color Computer with at least one floppy disk drive. The OS-9 standard system disk includes modules to support the following TRS-80 Color Computer hardware:

- 64K RAM
- Keyboard
- Alphanumeric Video Display
- Color Graphics Display
- Disk Drives (1 or 2)
- Joysticks (1 or 2)
- Serial Printer
- RS-232C Communications Port

Because OS-9 makes extensive use of the Color Computer's disk drives, the OS-9 Boot Disk includes a test for your disk drives. This test checks the speed of the drives. A disk drive's speed should be about 300 RPM (rotations per minute). The test will tell you if your drives are too fast, too slow, or if they are working within a proper speed range. If your disk drives should need adjustment, this work can be done at a Radio Shack repair facility.

Beyond the hardware requirements, you should be very familiar with Extended Color BASIC and Color Computer Disk BASIC. OS-9 is an advanced operating system designed for experienced computer users. Radio Shack strongly recommends that you feel comfortable with Color Disk BASIC before you move up to OS-9. The OS-9 documentation assumes that you have a working knowledge of the Color Computer disk operating system.

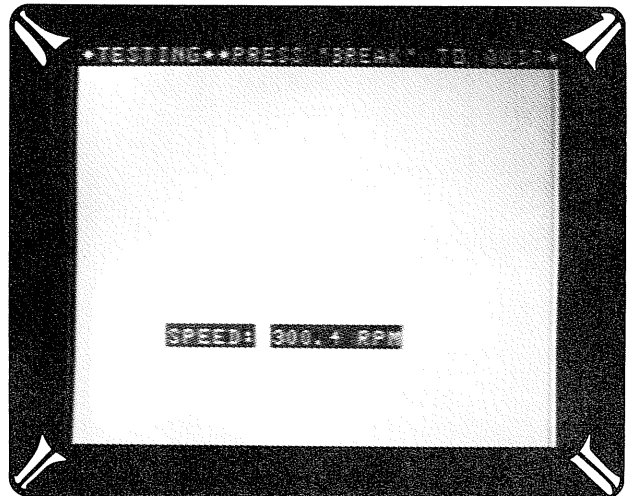
TELL ME MORE ABOUT OS-9

I have known for some time now that OS-9 was coming as a new operating system for the Color Computer. When I finally got a copy (xeroxed manuals and labelless diskettes), the first thing that struck me was a similarity to Xenix™. This similarity was the large stack of manuals for each. I received six separate manuals to support the initial release of OS-9:

- Getting Started With OS-9
- OS-9 Operations Manual
- OS-9 Technical Information
- OS-9 Macro Text Editor User's Manual
- OS-9 Assembler User's Manual
- OS-9 Interactive Debugger User's Manual

I am not quite sure how many pages that represents, but it certainly represents an incredible amount of information about both OS-9 and the Color Computer.

The first thing to get used to about OS-9 is its file system. Both Color Disk BASIC and OS-9 store information in disk "files" just as you might store a memo or other information in a file folder. Like most disk files, these files can contain a wide variety of information.



Under Color Disk BASIC, each diskette has a directory which tells you what information is stored on that particular diskette. OS-9 also uses directories, but in a slightly altered way. As in Color Disk BASIC, each directory points to (shows) information which is on a diskette. The difference with OS-9 is that one directory may contain or be contained in another directory, and each diskette may have many such directories.

The OS-9 system starts at the "system device directory." This directory contains a directory entry for each device in the Color Computer system, such as the keyboard/display, disk drives, printer, and the optional terminal. The system device directory looks like this:

System Device Directory



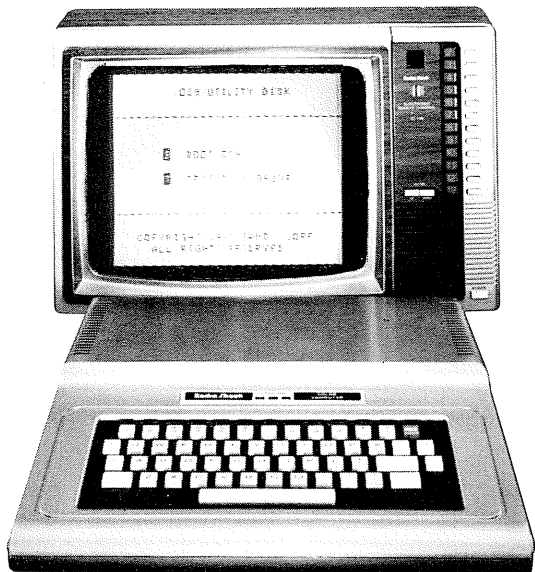
The P is the printer, D0 is the first disk drive, TERM is the keyboard/display, D1 is the second drive, and T1 is a terminal connected to the RS-232C serial port. (Even if you don't have the actual device, your system device directory still has the entry.)

The disk drives, referred to as /D0 and /D1, are the only devices that can form their own "tree" structure by storing other directories and files. Each drive has a "root" (or main) directory which is the beginning of its particular directory tree.

Generally, the disk in drive 0 is the OS-9 system disk. Its root directory contains two files (startup and OS-9Boot) and three sub-directories (SYS, DEFS and CMDS). The sub-directories contain another level of files. As an example, CMDS contains the files copy, list, dir and del, among many

others. (Note that OS-9 uses the convention of lowercase entries for files and UPPERCASE entries for sub-directories.)

To help you find your way inside the OS-9 structure, each device or file has its own unique "pathname." A pathname is the description of the path to follow to move from the system device directory to the desired file. The pathname for the file dir in the previous paragraph is: /D0/CMD5/dir. This name tells us that dir is located on the diskette currently logged into drive 0, and that it resides in the sub-directory CMD5. A pathname can be as long as needed to uniquely describe the path to a particular device, directory or file.



OS-9 COMMANDS

- attr - Change file attributes.
- backup - Make a disk backup.
- binex - Convert binary to s-record.
- build - Lets you create or build simple files.
- chd - Change working data directories.
- chx - Change working execution directory.
- cmp - File comparison utility.
- cobbler - Make a bootstrap file.
- copy - lets you duplicate a file.
- date - displays the current system date and time.
- dcheck - Check disk file structure.
- del - lets you delete a file.
- deldir - lets you delete a directory and all files in its system.
- dir - list the contents of the current directory.
- display - Display converted characters.
- dsave - Generate a procedure file to copy files.
- dump - Formatted file dump.
- echo - Echo text to output path.
- exbin - Convert s-record to binary.
- format - Initialize disk media.
- free - tells you the amount of free space remaining on a diskette.
- ident - Print OS-9 module identification.
- kill - Abort a process.
- link - Link module into memory.
- list - displays the contents of a file.
- load - Load module(s) into memory.
- login - Timesharing system log-in.
- makdir - allows you to create your own directories.

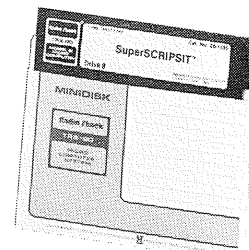
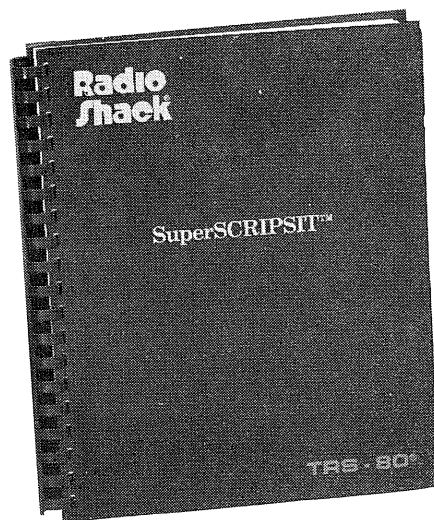
- mdir - Display module directory.
- merge - Copy and combine files.
- mfree - displays the amount of available memory in your OS-9 system.
- OS9Gen - Build and link a bootstrap file.
- printerr - Print full-text error messages.
- procs - Display processes.
- pwd - prints the name of the current working directory.
- pxd - Print current execution directory.
- rename - lets you change the name of a file.
- save - Save memory module(s) on a file.
- setime - lets you set the system time and date.
- setpr - Set process priority.
- sleep - Suspend process for a period of time.
- shell - OS-9 command interpreter.
- tee - Copy standard input to multiple output paths.
- tmode - Change terminal operating mode.
- tsmon - Timesharing monitor.
- unlink - Unlink memory module.
- verify - Verify or update module header and CRC.
- xmode - Examine or change device initialization mode.

Word Processing and Programming

Davy L. Barron
311 Montgomery St.
Troy, Al 36081

Most word processors save their text in ASCII format or have a utility program to convert their text to ASCII and back for proofreading. To use this tool to edit programs, all that needs to be done is to follow these simple directions.

1. Save your program in ASCII format. In most cases it would be saved like this: SAVE "program", A.
2. Load and run the word processor.
3. Load program or text into the word processor.
4. You are now ready to edit your program.
5. Save the program out in ASCII format (Model I/III Scripsit) or convert the file to ASCII format (Model III/4 SuperSCRIPSIT or Model II/12 Scripsit).



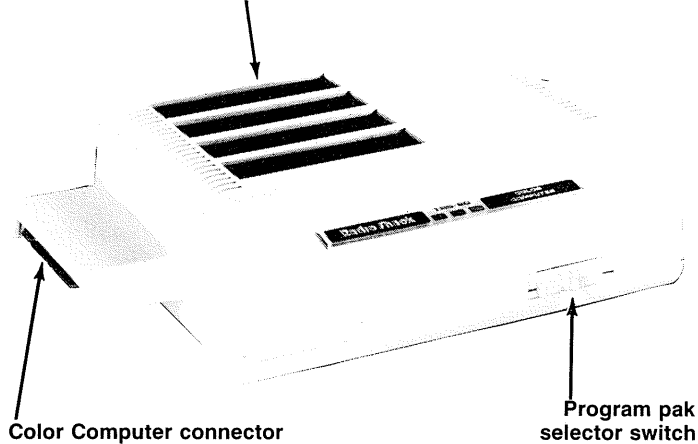
The Color Computer Multi-Pak Interface

By Linda Miller

With the Multi-Pak Interface (Cat. no. 26-3024, suggested retail price \$179.95) it is possible to connect up to four Color Computer Program Paks or Interface Controllers such as the Disk Controller to the Color Computer at the same time. By selecting one of the four available slots you can change from one program pak to another.

Frequently used program paks no longer have to be removed and reinserted each time you want to use them. They can be plugged into the Multi-Pak interface and used over and over without having to change the setup or continually change the cartridges.

Four program pak slots



HOOKING UP THE MULTI-PAK INTERFACE

When everything is turned off, the Multi-Pak Interface Computer Connector is aligned with the cartridge slot. The Multi-Pak Interface's connector is then gently slipped into the Color Computer's recessed cartridge receptacle. Next the computer and the interface are plugged into a power source. After the program paks have been inserted in the appropriate slots, the computer and the interface are turned on.

While it doesn't matter which of the four slots the program paks are plugged into, it is recommended that the Color Computer Disk Interface Pak be plugged into slot four.

HARDWARE OR SOFTWARE SLOT SELECTION

Slot selection can be accomplished via the switch on the interface (hardware) or by POKEing a memory location (software).

Switch positions one through four select the corresponding slots. When the switch is set to slot four, then the program pak in slot four is selected. Moving the switch to another position selects a different program pak. Sometimes after changing to a different slot, you may have to press the reset button.

By POKEing the appropriate value at address 65407, it is possible to software select any of the four interface slots.

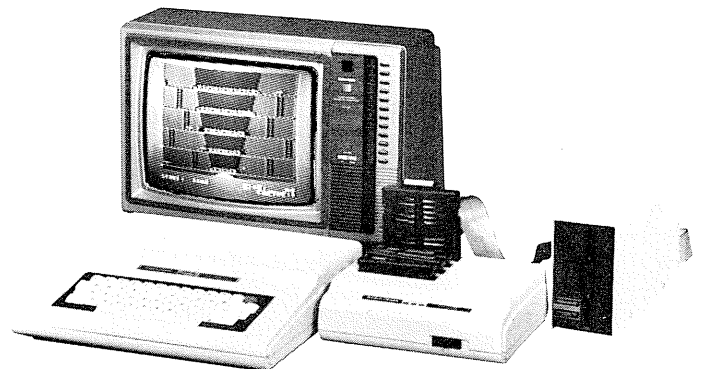
| POKE Address | with the Value of | for slot # |
|--------------|-------------------|------------|
| 65407 | 0 | 1 |
| 65407 | 17 | 2 |
| 65407 | 34 | 3 |
| 65407 | 51 | 4 |

The BASIC statement POKE 65407,17 selects slot two.

WHAT DOES IT ALL MEAN?

What this means is that I can have Dungeons of Daggorath, Double Back, Canyon Climber and the Disk Controller all plugged in at the same time. If I decide to play a different game or decide to use my disk for more serious applications like Scripsit, I just reposition the switch on the front of the Multi-Pak Interface to the appropriate slot, press the reset button, and I'm ready to go.

Convenience is the watchword here. The Multi-Pak Interface makes frequently used programs only the press of a button away.



The Color Mouse and the Deluxe Joystick

By Linda Miller

The TRS-80 Color Computer Color Mouse (cat. no. 26-3025, suggested retail price \$49.95) can be used in place of or in conjunction with the Color Computer joysticks (26-3008). With programs that require exact positioning of the cursor such as the three listed below, the Mouse is far superior to conventional X-Y controllers.

Galactic Attack (26-3066)
Polaris (26-3065)
Wildcatting (26-3067)
Reactoids (26-3092)

GETTING READY

The mouse connector is plugged into the joystick connector in the back of the Color Computer. The Color Mouse should be placed on a flat plane (such as a table top, desktop, or floor) with a sheet of paper between the flat surface and the Color Mouse. The sheet of paper is just a precaution to prevent marring of the flat surface. When I placed the Mouse on a stack of printer paper, it seemed a little more responsive. Maybe that was because the surface of the paper was not as rigid as my desk and that allowed more contact with the rotating ball on the bottom of the Mouse.

The Mouse is positioned so that the fire button is at the top and the power cord comes out the side opposite to the operator. In this position the lettering on the Mouse is easily readable.



CURSOR CONTROL

Controlling the cursor with the Color Mouse is a matter of logic. Cursor movement is relative to the movement of the

Mouse. Horizontally and vertically, the Mouse divides the screen into 64 steps (0 - 63). The center point of the screen is $X=32, Y=32$. The maximum movement of the Mouse is approximately 4.5" both horizontally and vertically. The range of movement and direction that the cursor can be moved is software dependent.

Figure 1 indicates the maximum movement capabilities of the Mouse and the directions of movement.

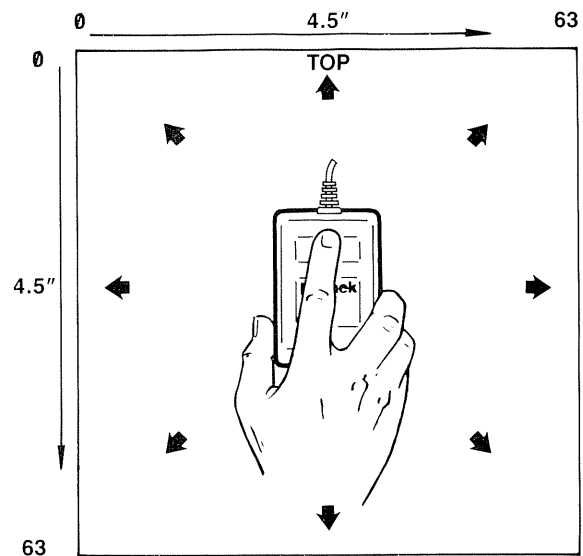


Figure 1.

SOFTWARE AND THE MOUSE

Software is a critical factor in the way that the Mouse operates. The available movement area and the direction of movement of the mouse are determined by the Color Computer program being used. With some programs the entire screen may be used while with others, the movement may be restricted to certain areas of the screen. Diagonal movement is permissible as can be seen in Figure 1.

The Mouse is programmed like the joysticks, and pages 84-88 of Getting Started with Color BASIC provide the necessary information for writing software to use it.

THE MOUSE IS FUN TO USE

The Mouse feels quite different from a joystick and lends itself better to some software applications than others. The Color Mouse can make drawing color graphics as well as many other applications great fun.

THE DELUXE JOYSTICK

The TRS-80 Color Computer Deluxe Joystick is a high-tech, high-performance joystick designed to increase screen positioning power and fun.

TO BEGIN WITH

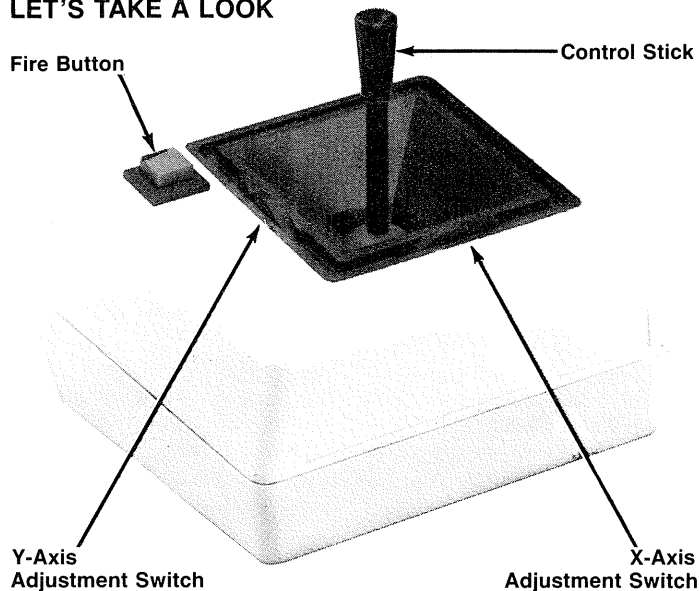
The Deluxe Joystick (cat. no. 26-3012, suggested retail price \$39.95) is easily installed. Its five pin connector plugs directly into either of the JOYSTK connectors at the rear of the Color Computer. Two Deluxe Joysticks can be connected to the computer.

Separate X and Y axis controls give you precise screen positioning.

Selectable "free-floating" or "spring center return" control stick adjustments let you tailor the stick action to the software being used.

The convenient "fire" button lets you execute game or program functions, such as starting a new game, quickly and efficiently.


LET'S TAKE A LOOK



The control stick lever lets you move in the direction of your choice. The fire button fires missiles or initiates program functions. Move the X-Axis adjustment switch to the left or right to adjust your horizontal position as close as possible to the screen's center. The Y-Axis adjustment switch moves up or down to adjust your vertical position as close as possible to the screen's center.

The control stick has both spring center return and free-floating modes of operation. The spring center return mode means that the control stick will automatically return to center position when you release the lever. When the lever is released, the screen position will be at or near dead center. The control stick can be free-floating for the X, Y or X and Y axes. Free-floating means that the control stick will remain in its present position when you release the lever.

In addition there are two levers on the control stick housing which allow fine adjustments to the electrical center of the Joystick.

With spring center return and free floating modes plus the ability to fine adjust the electrical center of the Joystick, it is easy to customize it to your own playing specifications. 

DEBUG for the Model 100

This BASIC debugging program has been run in an 8K Model 100. It ran successfully after deleting some user programs and executing a CLEAR 50 to free additional memory space.

The commands are:

V - Will prompt you for the address that you want to see.

Enter a Hex value and press **(ENTER)**.

Q - Returns to BASIC (Quits Debug).

N - Displays the Next memory block

P - Displays the Previous memory block.

S - Performs a Screen print. The contents of the display are sent to the line printer.

H - Produces a hardcopy of memory. When prompted, enter a hex value for the address at which to start the dump and press **(ENTER)**. Then enter a Hex value for address at which to stop the dump at the next prompt and press **(ENTER)**.

(←) (→) (↑) (↓) - The arrow keys are used to position the cursor.

0-9, A-F - These characters modify memory.

```
90 FOR X=0 TO 3
  : READ Z(X)
  : NEXT X
91 DATA 251,253,252,254
100 CLS
110 GOSUB 1000
120 GOSUB 3000
130 GOSUB 900
  : I$=INKEY$
  : IF I$="" THEN 130
140 IF I$="V" THEN GOSUB 1000
  : GOSUB 3000
  : GOTO 130
145 IF I$="Q" THEN CLS
  : END
150 IF I$="N" THEN GOSUB 5000
  : GOTO 130
160 IF I$="P" THEN GOSUB 6000
  : GOTO 130
170 IF I$="S" THEN PRINT@ CP, " ";
  : LCOPY
  : GOTO 130
175 IF I$="H" THEN GOSUB 13000
  : GOTO 130
180 IF I$=CHR$(28) THEN GOSUB 7000
  : GOTO 130
190 IF I$=CHR$(29) THEN GOSUB 8000
  : GOTO 130
200 IF I$=CHR$(31) THEN GOSUB 9000
  : GOTO 130
210 IF I$=CHR$(30) THEN GOSUB 10000
  : GOTO 130
220 IF I$<"0" THEN 130
230 IF I$>"F" THEN 130
240 IF I$>"9" AND I$<"A" THEN 130
250 GOSUB 11000
899 GOTO 130
900 CC=CC+1
  : IF CC>3 THEN CC=0
910 PRINT@CP,CHR$(Z(CC));
920 RETURN
999 END
1000 PRINT@CP, " ";
  : PRINT@280, " ";
  : LINE INPUT "ADDRESS: "; A$
```



```

1005 IF A$ > "FFD0" THEN A$="FFD0"
1010 P$=A$
      : GOSUB 2000
1020 A=P
      : RETURN
2000 IF LEN(P$)<4 THEN P$="0"+P$
      : GOTO 2000
2005 P=0
2010 FOR P1=0 TO 3
2020 P2$=MID$(P$,4-P1,1)
2030 P3=ASC(P2$)-48
2040 IF P3>9 THEN P3=P3-7
2050 P=P+P3*16^P1
2060 NEXT P1
2070 RETURN
3000 CLS
3005 CP=5
3010 FOR P1=A TO A+47 STEP 8
3020 P=P1
      : GOSUB 4000
3030 PRINT P$;" ";
3040 FOR P2=P1 TO P1+7
3050 P=PEEK(P2)
      : GOSUB 4000
3060 P3$=RIGHT$(P$,2)
3070 PRINT P3$;" ";
3080 NEXT P2
3083 PRINT" ";
3090 FOR P2=P1 TO P1+7
3100 P4=PEEK(P2)
3110 IF P4<32 OR P4>126 THEN P4=ASC(".")
3120 PRINT CHR$(P4);
3130 NEXT P2
3140 PRINT
3150 NEXT P1
3160 RETURN
4000 P$=""
4010 RESTORE 4100
4020 FOR P0=1 TO 4
4030 READ P9
4035 P8=INT(P/P9)
4040 P=P-P8*P9
4050 P8=P8+48
4060 IF P8>57 THEN P8=P8+7
4070 P$=P$+CHR$(P8)
4080 NEXT P0
4090 RETURN
4100 DATA 4096,256,16,1
5000 A=A+48
      : IF A>65488 THEN A=65488
5010 GOSUB 3000
      : RETURN
6000 A=A-48
      : IF A<0 THEN A=0
6010 GOTO 5010
7000 PRINT@CP," ";
7010 CP=CP+3
7020 P1=CP
7030 P1=P1-40
      : IF P1<0 THEN P1=P1+40 ELSE 7030
7040 IF P1<27 THEN RETURN
7045 CP=CP+16
      : IF CP>240 THEN CP=CP-40
      : S=CP
      : A=A+8 ELSE RETURN
7050 IF A>65488 THEN A=0
      : S=5
7060 GOSUB 3000
      : CP=S
      : RETURN
8000 PRINT@CP," ";
8010 CP=CP-3
8020 P1=CP
8030 P1=P1-40
      : IF P1<0 THEN P1=P1+40 ELSE 8030
8040 IF P1>4 THEN RETURN

```

```

8045 CP=CP-16
      : IF CP<0 THEN CP=CP+40
      : S=CP
      : A=A-8 ELSE RETURN
8050 IF A<0 THEN A=65488
      : S=226
8060 GOSUB 3000
      : CP=S
      : RETURN
9000 PRINT@CP," ";
9010 CP=CP+40
      : IF CP>239 THEN CP=CP-40 ELSE RETURN
9020 A=A+8
      : IF A>65488 THEN A=0
      : CP=CP-200
9030 S=CP
      : GOSUB 3000
      : CP=S
      : RETURN
10000 PRINT@CP," ";
10010 CP=CP-40
      : IF CP<5 THEN CP=CP+40 ELSE RETURN
10020 A=A-8
      : IF A<0 THEN A=65488
      : CP=CP+200
10030 GOTO 9030
11000 PRINT@CP," ";
11010 CP=CP+2
11020 PRINT@CP-1,I$
11030 P$="00"+I$
11040 GOSUB 9000
      : I$=INKEY$
      : IF I$="" THEN 11040
11050 IF I$=CHR$(27) THEN CP=CP-2
      : GOSUB 12000
      : P=PEEK(P)
      : GOSUB 4000
      : P$=RIGHT$(P$,2)
      : PRINT@CP+1,P$;
      : RETURN
11060 IF I$<"0" OR I$>"F" THEN 11040
11070 IF I$>"9" AND I$<"A" THEN 11040
11075 PRINT@CP,I$;
      : CP=CP-2
11080 P$=P$+I$
11090 GOSUB 2000
      : S=P
      : GOSUB 12000
      : POKE P,S
      : S=PEEK(P)
11100 Q=P-A
      : R=31+40*INT(Q/8)
      : Q=Q-8*INT(Q/8)
      : R=R+Q
11110 IF S<32 OR S>126 THEN S=ASC(".")
11120 PRINT@R,CHR$(S)
      : RETURN
12000 PA=8*INT(CP/40)+A
12010 PB=CP
12020 PB=PB-40
      : IF PB<0 THEN PB=PB+40 ELSE 12020
12030 PB=PB-5
12040 PB=INT(PB/3)
12050 P=PA+PB
12060 RETURN
13000 PRINT@CP," ";:PRINT@240,"";
13010 LINE INPUT"SIART: ";P$
13012 P$=P$
      : GOSUB 2000
      : PS=P
13018 PRINT@240,STRING$(40,32);
13020 PRINT@240,"";
13030 LINE INPUT"END: ";PE$
13032 P$=PE$
      : GOSUB 2000
      : PE=P

```

```

13034 LPRINT CHR$(12);
13036 PP=0
13040 FOR PL=PS TO PE STEP 16
13050 P=PL
      : GOSUB 4000
13060 LPRINT P$;" ";
13070 FOR PI=PL TO PL+15
13080 P=PEEK(PI)
      : GOSUB 4000
13090 LPRINT RIGHT$(P$,2);" ";
13100 NEXT PI
13110 LPRINT " ";
13120 FOR PI=PL TO PL+15
13130 PX=PEEK(PI)
13140 IF PX<32 OR PX>126 THEN PX=ASC(".")
13150 LPRINT CHR$(PX);
13160 NEXT PI
13170 LPRINT
13175 PP=PP+1
      : IF PP>59 THEN LPRINT CHR$(12);
      : PP=0
13180 NEXT PL
13183 PRINT@240,STRING$(40,32);
13187 LPRINT CHR$(12);
13190 RETURN

```

Spiral

Leo Gilbride, Age 12
614 Alameda Padre Serra
Santa Barbara, CA 93103

I am a twelve year old boy, and I received my 32K Color Computer as a combined birthday and junior high school graduation present. "Spiral" is a program that, given proper coordinates, will produce a colorful spiral. It will run on a 16K Extended BASIC Color Computer.

After you enter and run the program, the computer will then ask whether the spiral should grow or shrink. Once this decision is made, you must enter the first spiral circle size and position, which is determined by the horizontal and vertical center points and its radius. The limits of these coordinates are included in the program to ensure that the spiral does not completely exit the screen. After all the coordinates are entered, the spiral will appear on the screen for 15 seconds. You can continue making many different spirals with a "yes" response to the final question.

The following coordinates will give you a full spiral on the screen:

"Grow" H=64 V=46 R=10
"Shrink" H=150 V=100 R=85

If you have the Screen Print program and a printer, you can print out a spiral by loading the spiral program after loading the Screen Print Program. With this program, you will be able to print many variable spirals.

```

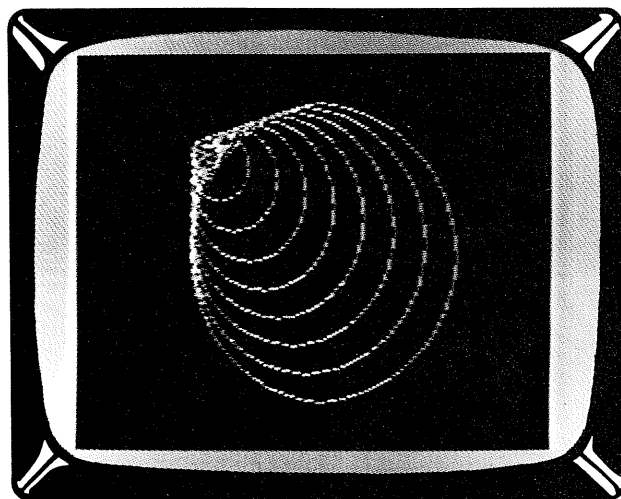
10 '*** SPIRAL ***
15 'BY LEO GILBRIDE
20 CLS
      : PRINT "          *** SPIRAL ***"
23 INPUT "DO YOU WANT THE SPIRAL TO GROW OR
SHRINK";W$
25 IF W$ = "GROW" GOTO 140 ELSE 30
30 IF W$ = "SHRINK" GOTO 33 ELSE 20
33 CLS
      : PRINT " LARGEST CIRCLE COORDINATES"
      : PRINT " "

```

```

35 SOUND 150,1
      : INPUT "HORIZONTAL CENTERPOINT (72 TO
180)";A
37 IF A < 72 OR A > 180 GOTO 35
40 SOUND 150,1
      : INPUT "VERTICAL CENTERPOINT (54 TO
120)";B
42 IF B < 54 OR B > 120 GOTO 40
45 SOUND 150,1
      : INPUT "CIRCLE RADIUS (72 TO 96)";C
47 IF C < 72 OR C > 96 GOTO 45
50 PMODE 4,1
55 PCLS
60 SCREEN 1,1
65 FOR I = 1 TO 10
70 CIRCLE (A,B),C
75 A = A - 8
80 B = B - 6
85 C = C - 8
90 NEXT I
100 FOR X = 1 TO 6000
      : NEXT X
110 PRINT " "
      : INPUT "WOULD YOU LIKE TO TRY IT AGAIN (Y OR
N)";F$
120 IF F$ = "Y" OR F$ = "YES" GOTO 20 ELSE END
140 CLS
      : PRINT " SMALLEST CIRCLE COORDINATES"
      : PRINT " "
145 SOUND 150,1
      : INPUT "HORIZONTAL CENTERPOINT (0 TO
175)";H
150 IF H > 175 GOTO 145
155 SOUND 150,1
      : INPUT "VERTICAL CENTERPOINT (0 TO
100)";V
160 IF V > 100 GOTO 155
165 SOUND 150,1
      : INPUT "CIRCLE RADIUS (0 TO 35)";D
167 IF D > 35 GOTO 165
170 PMODE 4,1
175 PCLS
180 SCREEN 1,1
182 FOR I = 1 TO 10
185 CIRCLE (H,V),D
190 H = H + 8
195 V = V + 6
200 D = D + 8
205 NEXT I
210 FOR X = 1 TO 6000
      : NEXT X
215 PRINT " "
      : INPUT "WOULD YOU LIKE TO TRY AGAIN (Y OR
N)";G$
220 IF G$ = "Y" OR G$ = "YES" GOTO 20 ELSE END

```



TRS-Xenix Power

The subject of TRS-Xenix gives us the chance to share with you some tips that we in Computer Services have discovered to be very helpful to many of our customers. Because Xenix is so flexible, you have a great deal of control over the system. The intent of this article is to help users gain some insight into the power of the system and become more comfortable with its use.

First, I would like to explain what TRS-Xenix is and what is meant by multi-processing/multi-user. TRS-Xenix is derived from the powerful UNIX operating system developed by Bell Laboratories. UNIX has been extensively field-tested for the past decade and has demonstrated outstanding performance under heavy workloads. The TRS-Xenix "core" is a derivative of this powerful UNIX system. The core comes with only those modules needed to run the current application software such as the Model 16 COBOL Accounting Software and Multiplan, but it can be enhanced with the purchase of the Development System which adds about 250 additional modules to the system. The development system with the powerful C language is for advanced programmers developing multi-user software but also contains many other powerful features such as: an on-line UNIX reference manual, system maintenance, system status information, electronic mail, spelling dictionary with over 20,000 words, text editors, a sort utility, typesetting, and many more. A COBOL Development System and a BASIC interpreter are also available as separate additions to the core.

Multi-processing and multi-user sometimes mean different things to different people. Here we define multi-processing as the ability to run one or more operations at a time, while multi-user is the ability for one or more persons to run the same or different programs at the same time. This means that one person could be in Accounts Receivable, another in General Ledger and still another in Order Entry. We could also have one person in General Ledger and two running Multiplan.

MAKING BACKUPS

This is perhaps the most important thing I could bring up in this article. Without backups you are asking for serious trouble. TRS-Xenix offers a number of backup procedures which are easy to understand and run. Before we take a few minutes to discuss them, let's talk about "VERIFY." TRS-Xenix has the option to verify what is being written to a disk. Whenever you backup, you should always turn verify on to insure that the backup is a good copy of your data. To enable verify simply type the command:

```
verify ^ b ^ o
```

where: ^ = space
b = -f (for floppy) or -h (for hard drive)
o = y (for yes) or n (for no)

After turning on verify you have three options of how you want to make your saves. The first is the "save/restore" utility

which is executed from the "tsh shell." This is perhaps the most preferred way to make backups of the entire system. The "tsh save" is as easy to use as the TRSDOS-II save was or perhaps even easier. Here is the syntax of the "tsh save" command, but first, remember you must be in the "tsh shell" before the command will execute.

```
save ^ :d ^ t ^ name
```

where: ^ = space
d = drive-number
t = -ss (single sided disk) or -ds (double)
name = the directory name you wish to save.

- Note:
1. One of the above (-ss, -ds) must be specified.
 2. You can't save devices.
 3. A / in place of the name will save the entire system.

Now that you know how to save under the "tsh save" utility, I bet you want to know how to put it back. Let me assure you that it's just as easy as the save. Here's the syntax:

```
restore ^ :d ^ name
```

where: ^ = space
d = drive-number
name = the directory name you wish to restore.

- Note:
1. If you use a -d in the place of name it will give you a list (directory) of what is on the floppy disk

With the most preferred option of save out of the way, let's discuss the least preferred option "sysadmin."

Sysadmin is menu driven in order to simplify its use. The sysadmin program cannot recognize the difference between files of the same name in different directories because it does not know about pathnames. When you restore a file which was saved under sysadmin, it assigns a number rather than the original file name. This number is unique to a single TRS-Xenix file. The file you restore is placed in your current working directory, and so to avoid possible confusion, rename it to its original name and move it back to its correct position in the directory hierarchy (using the mv command). This moving of files is what makes the sysadmin our least preferred option of backing up the system.

The last save option I wish to discuss is only used to save the three disk accounting packages and is also a menu driven program. All that is needed to invoke this save is to type "save". Under this menu you have the option of saving your data or programs. It is the recommended way to make daily data saves of your accounting packages.

USER SHUTDOWN

Now that we have covered Save commands, I would like to move on to something that everyone has been asking for. What is it, you ask. Well, it's how to shut the system down without knowing the root password.

Using the four steps outlined below, the root user sets up the usr/shutdown file so other users will be able to shut the

system down without knowing the root password. This process requires four steps. It will be very helpful if you know something about the text editor supplied with the core system. However, I will give you step by step instructions to follow for your convenience.

Duplicate the following key strokes exactly as they appear below in the Strokes to Duplicate column.

Step 1.

Login as root.

Strokes to Duplicate Explanation of Key Strokes

| | |
|------------------|--------------------------|
| ed /etc/shutdown | Loads file into buffer. |
| 14p | Displays line number 14. |
| d | Deletes line 14. |
| d | Deletes the new line 14. |
| i | Enters insert mode. |
| else : okay | Input new line. |
| . | Exits insert mode. |
| 14p | Displays current line. |
| w /etc/shutdown | Writes out new file. |
| q | Exits ed. |

Step 2.

| | |
|--------------------|--------------------------|
| mkuser | To create a new user. |
| n | No for instruction. |
| shutdown | User login name. |
| password | User password. |
| password | Retype for check. |
| to shutdown system | Comment. |
| n | For no change. |
| n | No for add another user. |

Step 3.

| | |
|----------------|-------------------------------|
| ed /etc/passwd | Loads file into buffer. |
| . | Displays line (Note 3.1) |
| s/211/0/ | Replace user id with roots id |
| s/50/0/ | Replace group id with root id |
| . | Displays line (Note 3.2) |
| w /etc/passwd | Writes out new file. |
| q | Exits ed. |

Step 4.

| | |
|------------------|--------------------------|
| cd /usr/shutdown | Change directory. |
| ed .profile | Loads file into buffer. |
| a | Append to end-of-file. |
| /etc/shutdown 0 | Sets shutdown (Note 4.1) |
| . | Exits append mode. |
| w | Writes out new file. |
| q | Exits ed. |

Notes

3.1 Except for the numbers 211 and 50, the line you display should read as follows:
 shutdown:j9InvqjAuhNJs:211:50:to shutdown
 system:/usr/shutdown:/bin/sh
 (211 is the user ID and 50 is the group ID. Your numbers will probably be different from those shown).

3.2 The line you display should read as follows:
 shutdown:j9InvqjAuhNJs:0:0:to shutdown
 system:/usr/shutdown:/bin/sh

4.1 You may set shutdown from 0-15 minutes (currently set at 0 minutes).

All that is needed to shutdown the system now is to login as shutdown.

RESTRICTING USERS-1

What's next on the list? Well, lots of customers have been asking questions on how they can have one person run general ledger and nothing else. All that is needed is to modify that user's entry in the "passwd" file. Let's take a look at Rick's entry in the /etc/passwd file and change it so he can only run general ledger. The following is the way it looks before we change it:

```
rick::212:50:rick's file:/usr/rick:/bin/sh
```

Now with "ed" (or another text editor) we can change this entry to read:

```
rick::212:50:rick's file:/usr/rick:rungl
```

As you can see we only replaced /bin/sh with rungl.

Now Rick can only run general ledger, and when he's through, he will be logged out.

The above procedure is good if you want the user to run only one package and nothing else. However, what if you want him to be able to run more than just one program? This brings us to our next topic.

RESTRICTING USERS-2

It is generally a good policy for the owner of the system to restrict access to the files and directories of the system. However, if some of the files and directories need to be shared by several individuals, the individuals can be identified to the computer as a group. The group affiliation is a facility that allows groups of users to share files while still restricting access to unaffiliated users.

With any file on the TRS-Xenix operating system, there are three levels of protection: the owner, the group, and all others. Using "chmod" (which we will discuss later), you can restrict or allow access to any file on the system. Chmod controls read, write, and execute ability for any user that logs into the system.

Before we get into chmod we must first understand the layers of protection that exist: the owner, the group, and all others. The first and last of these are the easiest to understand. The owner is the person who created the file, and all others are just that, all those other than the owner. The one layer that is not easily understood is the group. Every file and every user has a group id that is assigned by TRS-Xenix. By changing these group id's, we can allow users to run some programs but not all.

All users are given a group id of 50 when they are created with mkuser. All files, excluding system files, are also given a group id of 50. With ed we can edit these files and then with some special commands, we can change the group id for any file or user we wish. The first thing to do is to create the new group. To do this we must edit the file /etc/group. The syntax for adding the group is as follows:

```
Group name::group id#:user's login names
```

Where: *Group name* = the name you wish to assign the group.

group id# = the number that the group is to be identified by.

user's login names = all the users that you wish to be in that group.

Here's an example of a new group I've added:

```
Group name::group id#:user's login names
```

The keystrokes for adding this new group were as follows:

```
ed /etc/group
a
payroll::60:mike,sally,robert
.
w /etc/group
q
```

Now that we have a new group created, we must change group id's for the users selected and the programs to be protected. There are two ways of changing the group id's for a user: by changing their id when they login, or by a "newgrp" command that allows you to change back and forth between groups. To have the person login with the different group id, we must edit the file /etc/passwd. You should already know how to do this from our examples of shutting down the system.

To use the newgrp command the syntax is:

```
newgrp groupname
```

Now that we have changed the group id's for the user, we must change the id for the program. For this we use the command chgrp. The syntax is as follows

```
chgrp group# filename
```

For example: chgrp 60/usr/bin/runpr

This has changed the group id's for both users and the programs that we have decided to limit access to. Now we can change the permissions using the chmod command, so only the users with the same group id as the program can run them. This command is explained in your runtime manual, but I will also cover it here to help explain it further.

The syntax for chmod is:

```
chmod [ugoa] [= - +] [rwxugo] file.
```

To explain further, the [ugoa] is saying for whom (the u stands for all users, the g for groups, the o for owner, and a for all for these), the [= - +] is for leaving the same(=), taking away(-), or adding(+), and the [rwxugo] is the permissions to be affected (of these listed, only rwx are of interest to us). To give an example, let's change permissions for the program used above and change it to where only the owner and users with the same group id can run it.

```
chmod og+rwx /usr/bin/runpr
```

To disallow anyone else from running it.

```
chmod uo-rwx /usr/bin/runpr
```

MAKING MULTI-COMPANIES

Now for something new. How many times have you wished for the ability to place more than one company's accounting information on the same hard disk? With TRS-Xenix we have this capability! Just follow these few simple steps.

Step 1.

Install all of the accounting programs and the data for one of the companies. Use the "INSTALL" command. This will place all the data in a directory called "Dta.01"

Step 2.

For our example we will use Accounts Receivable and will be copying from Dta.01 to Dta.02, the 2 being for our company #2.

Now copy this information from one directory to another. For this we will use the copy command "cp." The syntax for this is:

```
cp -ro /Dta.01 /Dta.02
```

Step 3.

Check on the permission levels of the copied files to assure they are set correctly. This is done by the following steps:

```
cd /Dta.02/Ar
```

```
1 (to list the directory)
```

The permission levels should look like the following example.

```
-rwxrwxrwx 1 root 186 Jun 22 10:49 file.dta
```

If the permission levels don't match this example, then we need to use the "chmod" command to set them correctly. The key strokes would be:

```
chmod 777 /Dta.02/Ar/* (* = wildcard)
```

We use the wildcard here to set the permission levels of all the files in our new Dta.02 directory with one command.

Step 4.

Using the editor ed we will next modify the runar file to give us access to our newly created company data files. Here is the procedure to follow:

```
ed /usr/bin/runar (loads file into the buffer)
```

```
6s/1/2/ (changes 1 to 2 on Dta.0x)
```

```
w /usr/bin/runar2 (writes to a new file runar2)
```

```
q (exits ed)
```

```
chmod 777 /usr/bin/runar2 (sets permissions)
```

Now when you want to run company #2's accounts receivable, you just simply type in "runar2". This will take us to company 2's data files.

Step 5.

Repeat steps 3-4 for each accounting package.

Step 6.

Repeat the entire process for each additional company.

SAVING MULTI-COMPANIES

Now that we've installed our additional companies, we need a way to save them. To do this we have two options. We can either have it so that when we save all our data is saved, or we can have it so that only one particular company's data is saved. For our example we will edit the save command so that we can save company 2's data. If you want the save command to save all the data, place an "*" where we have a "2" in our example. Now for the syntax to accomplish this task.

```
ed /usr/bin/save (loads file into the buffer)
```

```
69s/01/02 (changes Dta.01 to Dta.02)
```

```
82s/01/02
```

```
114s/01/02
```

```
w /usr/bin/save2 (writes to a file save2)
```

```
q (exits ed)
```

After completing these steps, when you need to save the data for company 1, just use "save" and for company 2 you would use "save2".

MULTIPLE HARD-DISKS

For those of you that wish to include additional hard drives in your system, just follow these few simple steps.

Step 1.

Do a normal system shutdown using either "shutdown" or "haltsys." Then re-boot the system.

Step 2.

Format the secondary hard drive using this "diskutil" program as you did when you formatted the primary disk. Just follow these few basic instructions:

TRS-Xenix Boot

```
: diskutil ENTER
Copy or format (c or f)?
Enter "f" to format.
Hard or floppy disk (h or f)?
Enter "h" for hard disk.
Hard drive number (0..3)?
Enter the appropriate drive number (0-3).
```

Step 3.

Make a file structure on the drive.
For 8 meg use: /etc/mkfs /dev/rhd1 16966 1 17.
For 12 meg use: /etc/mkfs /dev/rhd1 23018 1 17.

Edit the /etc/rc file for automatic mounting of the secondary hard drive using the following syntax.

```
ed /etc/rc          (loads file into the buffer)
a                  (to append to the file)
/etc/mount /dev/hd1 /mnt1 (mounts the
                    secondary)
.                  (exits the append mode)
w                  (writes the file )
q                  (exits ed)
```

Now the disk will be referred to as directory "/mnt1."

In order to take data files to the secondary hard drive, use the following steps.

```
copy -ro /Dta.0x /mnt1/Dta.0x
check the permissions
edit the runxx files to say "cd /mnt1/Dta.0x/Xx"
```

With the above information you should be able to accomplish just about anything you need with the all-powerful TRS-Xenix operating system.

In response to some problems that have been reported to us in Computer Customer Services, we are including the following solutions.

PRINTER DIFFICULTIES

If, when you attempt to print information you receive a printer busy or a printer not ready message, you need to try these steps.

Step 1.

```
chmod 666 /dev/clp
This will set the permission levels correctly.
```

Step 2.

```
cd /usr/spool/lpd
l
rm lock
cd /
ps -alx
```

At this point look for "lpd" under the column titled CMD. After finding this, find the corresponding number under the column titled PID. Then we will kill this number as follows:

```
kill -9 PID#
```

Step 3.

```
Initialize the line printer by entering the following:
/usr/lib/lpd
```

Step 4.

If the above steps don't solve the problem, then shut the system down, boot your machine from a TRSDOS floppy disk and check the printer with DIR [PRT]. If it does not seem to be a hardware problem with the printer, then the last alternative is to reinstall the Xenix core.

DT-1 "@" SIGN

If you have a copy of Xenix core version 01.02.00 and you receive "@" symbols on your DT-1 screen, this is caused by an invalid termcap type. To correct this, type the following:

```
ed /.profile
g/TERM = trs16/.d
w
q
```

Ideally the .profile should have the following command in it:

```
if test -z "$TERM"
then
TERM = trs16 ;export TERM
fi
```

The Xenix 01.02.00 system will check for a DT-1 or TRS-16. If neither of these are being used, TERM will be set to TRS-16 by the above command.

READING A DIRECTORY OF FLOPPY DISK

To read a directory of a floppy disk that you have used the "tar" command to move files to, do the following.

```
tar tf /dev/fd0
```

CONCLUSION

With TRS-Xenix, not only do you have control over the applications being run, but you basically have control of the entire system. This gives one unlimited possibilities in the area of computer operations. With the ability to add two data terminals, the system increases its productivity to make it a very valuable tool in the business environment. With the addition of the TRS-Xenix development system, you are given a variety of more advanced commands and utilities to use, as well as the "C" language compiler to enhance your system. This addition will give added flexibility to further customize this already powerful operating system to meet your specific needs.



Computer Customer Service Address and Phone Numbers

8AM to 5PM Central Time
Computer Customer Services
400 Atrium, One Tandy Center
Fort Worth, Texas 76102

| | |
|---------------------------------------|----------------|
| Model I/III/4 Business Group | (817) 390-3939 |
| Model II/12/16 Business Group | (817) 390-3935 |
| Languages and Compilers | (817) 390-3946 |
| Color/Model 100/Pocket Computer Group | (817) 390-3944 |
| Hardware and Communications Group | (817) 390-2140 |
| Educational Software | (817) 390-3302 |
| Games, Books, and New Products | (817) 390-2133 |
| Newsletter Subscription Problems | (817) 870-0407 |

Profile and the Model 100

The Small Computer Company

P.O. Box 2910

Fort Worth, TX 76113-2910

By Ivan Sygoda, Director, Pentacle

Profile III Plus section copyright 1983, Ivan Sygoda.

All rights reserved.

If you're like me, you don't do all your business in the office or all your leisure time activities at home. As a manager of dance companies, I often travel to conventions of presenters, and as a philatelist, I occasionally attend stamp shows or make the rounds of the dealers. Thanks to my new Radio Shack Model 100 portable computer, I don't have to leave my Profile data bases at home when I go on the road. It's much easier to lug around a Model 100 than a Model III or 12.

THE OLD STAMPING GROUND

As I've been promising, here is a method of transferring your data to and from the Model 100. I'll use the STAMPCAT stamp collection data base I discussed in the June 1983 Profile article and then accessed from BASIC in the July issue. (For newcomers who can't get to a Radio Shack to leaf through recent issues, STAMPCAT is a one-segment Profile data base in which I store information about each item in my stamp collection. The fields are listed below.)

| Field | Heading | Length | Buffer Variable |
|-------|------------|--------|-----------------|
| 1 | Cat# | 6 | CT\$ |
| 2 | Suffix | 1 | SF\$ |
| 3 | Item ID | 2 | ID\$ |
| 4 | #Stamps | 2 | NS\$ |
| 5 | Year | 4 | YR\$ |
| 6 | Denom | 5 | DN\$ |
| 7 | Series | 15 | SR\$ |
| 8 | Color/Var | 12 | VR\$ |
| 9 | Dealer | 3 | DL\$ |
| 10 | Purch.date | 8 | PD\$ |
| 11 | Price | 7 | PR\$ |
| 12 | Mint/Used | 1 | MU\$ |
| 13 | Cond:Recto | 10 | CR\$ |
| 14 | Cond:Verso | 10 | CV\$ |
| 15 | Scott val | 7 | SV\$ |
| 16 | Mkt val | 7 | MV\$ |
| 17 | Source | 3 | SC\$ |
| 18 | +/- % | 5 | CH\$ |
| 19 | Last upd | 8 | LU\$ |
| 20 | Marker | 1 | MK\$ |
| 21 | Face val | 6 | FV\$ |

Figure 1.

Five relatively simple steps are involved, and I'll go through them one by one. Then I'll show you a neat way to make a menu to call these operations, using Profile's menu

creation program. First, I'll discuss a short BASIC program that plucks data from the Profile data base and puts it in a separate data file to be transferred to the Model 100. Then, I'll show the transfer process itself, from the desktop computer to the 100. The next step involves displaying and editing the data on the Model 100. The fourth step is transferring the updated data back into the desktop computer. And, finally, I'll discuss another short BASIC program that tucks the updated data back into the Profile data base.

ASCII A QUESTION, GET AN ANSWER

The whole process is quite simple thanks to the ASCII (American Standard Code for Information Interchange) method of storing alphanumeric information. As I've pointed out before, Profile stores data as strings of ASCII characters arrayed one after the other, character by character, field by field. This principle is the key to successfully manipulating files.

Figure 1 lists the fields in STAMPCAT/KEY—the only segment in this data base—along with the buffer-variable names I assigned them in the accompanying BASIC programs. Listing 1 is the Model III BASIC program STAMP100/BAS, which takes the data from my Profile data segment and puts it into another data file STAMP100/DAT. I used a Profile user index (STAMPCAT/IX1) so that my Model 100 file will be in catalogue number order. (The Scott catalogue numbering system is copyrighted by the Scott Publishing Company.) Here are the details.

```

10 'STAMP100/BAS- OPEN BASIC FOR 3V FILES
20 CLS
   : CLEAR 1000
   : DEFINT A-Z
30 OPEN "R",1, "STAMPCAT/IX1",10
40 FIELD 1,7 AS KY$,2 AS PH$,1 AS XX$
50 OPEN "R",2, "STAMPCAT/KEY",127
60 FIELD 2,6 AS CT$,1 AS SF$,2 AS ID$,2 AS NS$,4
   AS YR$,5 AS DN$,15 AS SR$,12 AS VR$,3 AS DL$,8
   AS PD$,7 AS PR$,1 AS MU$,10 AS CR$,10 AS CV$,7
   AS SV$,7 AS MV$,3 AS SC$,5 AS CH$,8 AS LU$,1
   AS MK$,6 AS FV$
70 OPEN "R",3, "STAMP100/DAT",240
80 FIELD 3,120 AS ZY$,120 AS ZZ$
90 FOR LR=4 TO LOF(1)
100 GET 1,LR
110 PR=CVI(PH$)
120 GET 2,PR
130 LSET ZY$= "Scott no.=" +CT$ +SF$ +" Year=" +
   YR$ +" Denom=" +DN$ +" Ser=" +SR$ +" Clr/vr="
   +VR$ +" ID=" +ID$ +NS$ +MU$ +" Cond:R=" +CR$
   +" v=" +CV$ +" "
```


| | | | | |
|----------------------|-----------|--------------|--------------------|----------------------|
| Scott no.=524 | Year=1918 | Denom= 5.00 | Ser=Wash.-Franklin | Clr/vr=FP, P.11 unw |
| ID=S 1* Cond:R=vf | | V=nh | Prch price= 60.00 | Dlr=SAM Dte=74/04/12 |
| Curr m'rkt= 450.00 | Src=SAN | Upd=83/03/27 | Curr Scott= 725.00 | Mkr= Rec.no.00009 |
| Scott no.=530 | Year=1918 | Denom= .03 | Ser=Wash.-Franklin | Clr/vr=OF, P.11 unw |
| ID=S 1* Cond:R=xf | | V=1h | Prch price= 5.00 | Dlr=ZEN Dte=78/11/16 |
| Curr m'rkt= | Src= | Upd=83/03/27 | Curr Scott= | Mkr= Rec.no.00010 |
| Scott no.=538 | Year=1919 | Denom= .01 | Ser=Wash.-Franklin | Clr/vr=RP, P.11x10 |
| ID=S 1* Cond:R=vf+ | | V=nh | Prch price= 9.00 | Dlr=ZEN Dte=78/11/16 |
| Curr m'rkt= 20.00 | Src=SAN | Upd=83/03/27 | Curr Scott= | Mkr= Rec.no.00008 |
| Scott no.=541 | Year=1919 | Denom= .03 | Ser=Wash.-Franklin | Clr/vr=P.11x10 TyII |
| ID=S 1* Cond:R=vf-xf | | V=nh | Prch price= 115.00 | Dlr=ZEN Dte=78/11/16 |
| Curr m'rkt= 100.00 | Src=SAN | Upd=83/03/27 | Curr Scott= 37.50 | Mkr= Rec.no.00007 |
| Scott no.=ZZZZZ | Year= | Denom= | Ser= | Clr/vr= |
| ID= | Cond:R= | V= | Prch price= | Dlr= Dte= |
| Curr m'rkt= | Src= | Upd=83/07/03 | Curr Scott= | Mkr= Rec.no.00015 |

Figure 3.

One more suggestion before I leave step 1 and the description of the BASIC program. The way to add new records to the data base while using the Model 100 is simply to anticipate your needs before creating STAMP100/DAT. I create a few "dummy" records while still in Profile, arbitrarily assigning them catalog numbers like "ZZZZZZ" so that they're included in the index file, but at the end. These empty records are waiting for me, when I need them on the road.

LET'S COMMUNICATE

Having created a data file with the desired information in it, the next step is to feed the file to the Model 100. There are a number of different ways to do this. You can connect the Model 100 directly to a Model III, 4, II, or 12 using an RS232 cable with a null modem adapter, which is how I did it. Or you can communicate over telephone lines using modems. In either case, you need a suitable communications package for the Model III such as Videotex Plus (26-1588), LCOMM under LDOS 5.1.3 (26-2213). The Models 4, II and 12 already come with a terminal utility that does the same thing. Finally, the new Model 4 has a Model 100 tape utility, which means you can save STAMP100/DAT on cassette for subsequent loading into the Model 100. Whichever way you do it, make sure the communications parameters for both computers match.

AWAY WE GO

I loaded STAMP100/DAT into my Model 100 as STAMPS.DO. Any six-character filename works, followed by the .DO (document) extension. When I want to go into my file, I position the cursor over STAMPS.DO at the main menu and press **ENTER**; the first record appears on the screen. I can scroll through them all by pressing the **F1** function key ("Find") and then entering the desired match string.

For example, entering "no.=335" calls up the "page" with all the information about Scott no. 335. All the Model 100's built-in text editing features are available. The one thing that takes a bit of getting used to is that the text editing program is always in "insert mode" rather than in "overwrite mode." So you have to be careful to leave fields the same length you found them.

There's another way to do the same thing if you're going to simply reference your data as opposed to updating it. The Model 100's ADDRSS and SCHEDL programs both have

built-in search functions. You can make either one available to your data by renaming either ADRS.DO or NOTE.DO to something like HOLD.DO and then renaming STAMPS.DO to either ADRS.DO or NOTE.DO. This is done from Model 100 BASIC: NAME "STAMPS.DO" AS "NOTE.DO". The text editing features are still available. But be sure to rename things back the way they were when you're done.

I also recommend that you back up your Model 100 file on cassette from time to time. My Model 100 is very reliable, but sometimes I'm not. One night, while preparing this article, I pressed download instead of upload and zapped the whole data file. Think, as they say.

GOING THE OTHER WAY

All voyages come to an end, and sooner or later, you'll want to feed your updated data back into Profile on your desktop computer. The first step is to upload STAMPS.DO from the Model 100 to the Model III (or whatever) as STAMP100/DAT. The new information simply overwrites the original data file.

Listing 2 shows STAMP100/BAS, the BASIC program I use to put the updated file back into my Profile data base. It involves the same principles as before.

```

10 'STAMP100/BAS- OPEN BASIC FOR 3V FILES
20 CLS
   : CLEAR 1000
   : DEFINT A-Z
30 OPEN "R",1, "STAMP100/DAT",240
40 FIELD 1,10 AS QA$,6 AS FA$,1 AS FB$,6 AS QB$,4
   AS FC$,7 AS QC$,5 AS FD$,5 AS QD$,15 AS FE$,8
   AS QE$,12 AS FF$,4 AS QF$,2 AS FG$,2 AS FH$,1
   AS FI$,8 AS QG$,10 AS FJ$,3 AS QH$,10 AS FK$
45 FIELD 1,119 AS FZ$,12 AS QI$,7 AS FL$,5 AS
   QJ$,3 AS FM$,5 AS QK$,8 AS FW$,12 AS QL$,7 AS
   FO$,5 AS QM$,3 AS FP$,5 AS QN$,8 AS FQ$,12 AS
   QO$,7 AS FR$,5 AS QP$,2 AS FS$,10 AS QQ$,5 AS
   FT$
50 OPEN "R",2, "STAMP100/KEY",127
60 FIELD 2,6 AS CT$,1 AS SF$,2 AS ID$,2 AS NS$,4
   AS YR$,5 AS DN$,15 AS SR$,12 AS VR$,3 AS
   DL$,8 AS PD$,7 AS PR$,1 AS MU$,10 AS CR$,10 AS
   CV$,7 AS SV$,7 AS MV$,3 AS SC$,5 AS CH$,8 AS
   LU$,1 AS MK$,6 AS FV$
70 FOR LR=1 TO LOF(1)
80 GET 1,LR
90 PR=VAL(FT$)
100 GET 2,PR

```

```

110 LSET CT$=FA$
    : LSET SF$=FB$
    : LSET ID$=FG$
    : LSET NS$=FH$
    : LSET YR$=FC$
    : LSET DN$=FD$
    : LSET SR$=FE$
    : LSET VR$=FF$
    : LSET DL$=FM$
    : LSET PD$=FW$
    : LSET PR$=FL$
    : LSET MU$=FI$
    : LSET CR$=FJ$
    : LSET CV$=FK$
    : LSET SV$=FR$
    : LSET MV$=FO$
120 LSET SC$=FP$
    : LSET LU$=FQ$
    : LSET MK$=FS$
130 PRINT "Processing Rec.#"; LR; "(Phys.Rec.#";
    PR; "Cat.#"; FA$+FB$;)"
140 PUT 2
150 NEXT LR
160 PRINT "DONE."
170 END

```

In lines 40-45, I FIELD the transfer file so that I can unmix the headings and fields. Hoping to be efficient and accurate, I followed a procedure that got me into trouble. When choosing field variable names, I decided to name the headings QA\$. . . QQ\$ and the actual fields FA\$. . . FT\$. That way, it would be easy to ignore the headings when it came time to LSET the updated fields into STAMPCAT/KEY, and I'd be sure not to skip any fields, since I know the alphabet rather well. Because one line of BASIC isn't long enough to do all the necessary FIELDing, I divided the task between two lines. FZ\$ in line 45 is a "dummy" field that gets me to where line 40 leaves off. Clever, eh?

Line 90 is a consequence of line 135 in listing 1. CVI(FT\$) wouldn't work because of the way I manipulated the record number. VAL(FT\$) works just fine.

Lines 110-120 LSET each field into STAMPCAT/KEY, field by field. This procedure takes care of the fact that there were one or two fields I didn't display on the Model 100, such as percentage change in the stamp's value. Such uninvolved fields remain unscathed.

Finally, line 130 tells me what's going on while it's going on.

FREE MENU

The Profile program you bought from Radio Shack has a menu creation program used to construct user menus for your Profile data bases. The Profile III Plus version is called EFCM/CMD, and it's on your Creation disk. The Model II/12 versions call it MAKEMENU/EFC. Did you realize you can use it to make any menu you want, whether Profile is involved or not?

Call the relevant program from TRSDOS, and then make careful notes of your answers to the prompts. To call STAMP100/BAS and STAMP111/BAS, Model III users have to construct DO files to record all the keystrokes involved and then call the DO file from the user menu. Model II/12 users can enter keystrokes directly into the menu format. Try it!

MORE TO COME

We've barely scratched the surface. The Model 100 comes with a powerful BASIC, which we didn't use at all this

month, and which should make it possible to manipulate your Profile data with even greater finesse—for instance, by re-creating some of Profile's math capabilities. If any of you have made headway in this area, let us know so that we all can share your expertise.

Also, I've been promising articles on interfacing Profile with Scipsit and Visicalc. One of these days I'll deliver. A round-up of the most interesting problems and questions posed by readers is also in the works. Keep those cards and letters coming.

PROFILE Editor's Note: This is Mr. Sygoda's eleventh article in a series of 'how-to' Profile articles. Other articles in the series will be published over the next few issues in this column. We hope that you enjoy this feature, and we look forward to your comments and questions on Profile.

Pentacle is a New York City-based non-profit service organization specializing in administrative services for performing art groups.

The On-Line Computer Telephone Directory

The On-Line Computer Telephone Directory
Post Office Box 1005
Kansas City, MO 64111

The "On-Line Computer Telephone Directory" (OLCTD) is a quarterly newsletter for telecomputing. They keep you up-to-date with the latest trends in microcomputer telecommunications. Plus, you get the latest information on:

- User operation procedures for many of the popular microcomputer bulletin board systems.
- Simple, straightforward explanations of the technical aspects of computer telecommunications.
- Software/hardware news and reviews.

AND a directory of over 400 free-access microcomputer bulletin board systems on-line in North America.

As the seasons change, so changes the field of microcomputer telecommunications. New products, developments and bulletin boards appear. Old, outdated information becomes useless. That's why the people at OLCTD go to great lengths to make sure that the directory section in each issue is thoroughly checked, revised and updated. According to OLCTD, nobody has a more accurate list of bulletin board systems.

For subscription information, write to:

Subscription Dept.
OLCTD
Post Office Box 10005
Kansas City, MO 64111-9990





Calendar

Tim McDuffie
P.O. Box 835
Huntington, TX 75949

Now you can have immediate calendar-oriented results at the touch-of-a-button on your Color Computer. This program offers three functions that are accurate, all the way from January 1, 1753 to December 31, 2099.

Function #1 tells you the day of the week for which a given month, day, and year falls on. The date that you entered will remain on the screen, along with the day of the week, in case you forgot what you entered.

Function #2 will produce a calendar along with its month, and year given it. An "L" appears to the right of all leap years.

During this mode you may press the  to view the next month or the  to view the previous month. You may also view the displayed month in the next year by pressing the  or the displayed month in the previous year by pressing the .

Function #3 will tell you, in days, how long it has been from one date to the next, given a "from" date and a "to" date.

At the end of functions 1 and 3 you will be prompted with the following:

```
ENTER [R]—RETURN TO MENU
      [A]—ANOTHER RUN
```

In function #2, however, the above prompts would be difficult to achieve, therefore you are prompted with "[R] OR [A]" in the upper right-hand of the screen. Both prompts ask for the same thing.

EXPLANATION OF ERROR MESSAGES

| | |
|---------------------|---|
| **1 2 OR 3 ONLY** | —Bad menu selection |
| BAD MONTH | —Three-character month abbreviation was misspelled |
| **BAD DAY** | —Too many or too few days given for specified month |
| *BAD YEAR | —Year given is > 1753 or < 2099 |
| **NOT A LEAP YEAR** | —Feb. 29 was given along with a year other than a leap year |
| **TO < = FROM** | —Total days displayed would be 0 or negative |

When entering the DATA statements, be sure they are in the order exactly as shown here and that no spaces appear before or after the commas, otherwise the program will not function properly.

```
100 'CALENDAR' BY TIM MC DUFFIE
120 'NEEDS 16K WITH STANDARD
125 'OR EXTENDED COLOR BASIC.
130 DIM DP(42), ML(12), MD$(12)
140 GOSUB 1560
150 H$="COLOR COMPUTER CALENDAR"
160 GOSUB 610
170 PRINT "DO YOU WANT TO:"
180 PRINT
190 PRINT "[1] - FIND WEEK-DAY FOR A GIVEN"
200 PRINT "      MONTH, DAY & YEAR"
205 PRINT
210 PRINT "[2] - FIND CALENDAR FOR A GIVEN"
```

```
220 PRINT "      MONTH & YEAR"
225 PRINT
230 PRINT "[3] - FIND NUMBER OF DAYS"
240 PRINT "      BETWEEN 2 DATES"
245 PRINT
250 INPUT CH$
255 CH=VAL(CH$)
260 IF CH<1 OR CH>3 THEN EX=0
      : GOSUB 1550
      : GOTO160
270 ON CH GOSUB 290, 380, 460
280 GOTO 150
290 H$="DAY-OF-WEEK FINDER"
300 GOSUB 610
310 GOSUB 650
320 GOSUB 718
330 IF DV=1 THEN GOSUB 1550
      : GOTO 300
340 GOSUB 800
345 GOSUB 930
350 GOSUB 1160
360 GOSUB 1400
370 IF RA$="R" THEN RETURN ELSE 300
380 GOSUB 610
390 GOSUB 650
400 GOSUB 718
410 IF DV=1 THEN GOSUB 1550
      : GOTO 380
420 GOSUB 800
425 GOSUB 930
430 GOSUB 1200
440 GOSUB 1400
442 IF RA$="R" THEN RETURN
444 IF RA$="A" THEN 380
446 IF RA$=CHR$(8) THEN GOSUB 700
      : GOTO 420
448 IF RA$=CHR$(9) THEN GOSUB 693
      : GOTO 420
450 IF RA$=CHR$(94) THEN GOSUB 708
      : GOTO 420
452 IF RA$=CHR$(10) THEN GOSUB 712
      : GOTO 420
460 H$="NUMBER OF DAYS"
470 GOSUB 610
490 FOR XD=1 TO 2
500 PRINT FT$(XD)
510 GOSUB 650
520 GOSUB 718
530 IF DV=1 THEN GOSUB 1550
      : GOTO 470
540 GOSUB 800
550 FT(XD)=TD
555 IF XD=1 THEN GOSUB 1540
560 NEXT
570 IF FT(1)>=FT(2) THEN EX=5
      : GOSUB 1550
      : GOTO 470
580 GOSUB 1350
590 GOSUB 1400
600 IF RA$="R" THEN RETURN ELSE 470
610 H=INT(32-LEN(H$))/2
620 CLS
630 PRINT @ H, H$
635 PRINT
640 RETURN
650 INPUT "ENTER MONTH (1'ST 3 CHAR.)"; MI$
655 IF LEFT$(MI$, 1)="R" THEN 150
670 IF CH=2 THEN DI=1
      : GOTO 690
680 INPUT "ENTER DAY-OF-MONTH"; DI
690 INPUT "ENTER YEAR (1753-2099)"; YI
692 RETURN
693 MP=MP+1
694 IF MP>12 THEN MP=1
      : GOSUB 708
698 RETURN
700 MP=MP-1
```

```

702 IF MP<1 THEN MP=12
   : GOSUB 712
706 RETURN
708 YI=YI+1
709 IF YI>2099 THEN YI=2099
   : GOSUB 1554
710 GOSUB 1010
711 RETURN
712 YI=YI-1
714 IF YI<1753 THEN YI=1753
   : GOSUB 1554
716 GOSUB 1010
717 RETURN
718 MF=1
719 DV=0
720 IF YI<1753 OR YI>2099 THEN DV=1
   : EX=3
   : RETURN
730 MC$=LEFT$(MI$, 3)
740 FOR X=1 TO 12
750 IF MC$=LEFT$(MD$(X), 3) THEN MP=X
   : MF=0
760 NEXT
770 IF MF=1 THEN DV=1
   : EX=1
   : RETURN
775 GOSUB 1010
780 IF DI<1 OR DI>ML(MP) THEN DV=1
   : IF MP=2 AND DI=29 THEN EX=4 ELSE EX=2
790 RETURN
800 TD=0
820 IF YI=1753 THEN 870
830 PY=YI-1753
840 LY=INT(PY/4)
850 RY=PY-LY
860 TD=(LY*366)+(RY*365)
870 IF MP=1 THEN 910
880 FOR X=1 TO (MP-1)
890 TD=TD+ML(X)
900 NEXT
910 TD=TD+DI
920 RETURN
930 DD=TD/7
940 DD=DD-INT(DD)
950 DD=INT(DD*100)
960 GOSUB 1030
970 IF CD<=18000229 THEN GOSUB 1070
   : RETURN
980 IF CD<=19000229 THEN GOSUB 1100
   : RETURN
990 GOSUB 1130
1000 RETURN
1010 IF (YI/4)-INT(YI/4)=0 THEN ML(2)=29 ELSE
   ML(2)=28
1020 RETURN
1030 IF DI<10 THEN CD$="0"+MID$(STR$(DI), 2) ELSE
   CD$=MID$(STR$(DI), 2)
1040 IF MP<10 THEN CM$="0"+MID$(STR$(MP), 2) ELSE
   CM$=MID$(STR$(MP), 2)
1050 CD=VAL(MID$(STR$(YI), 2)+CM$+CD$)
1060 RETURN
1070 FOR X=1 TO 7
1080 IF DD=D1(X) THEN DW=X
   : RETURN
1090 NEXT
1100 FOR X=1 TO 7
1110 IF DD=D2(X) THEN DW=X
   : RETURN
1120 NEXT
1130 FOR X=1 TO 7
1140 IF DD=D3(X) THEN DW=X
   : RETURN
1150 NEXT
1160 PRINT
1170 PRINT MD$(MP) DI ", " YI "IS " DD$(DW)
1180 GOSUB 1540
1190 RETURN

```

```

1200 CLS0
1210 PRINT @ 0, MD$(MP) ", " YI;
1215 IF ML(2)=29 THEN PRINT "L";
1220 FOR X=1 TO 7
1230 PRINT @ CH(X), LEFT$(DD$(X), 3);
1240 NEXT
1290 FOR X=1 TO ML(MP)
1300 DS$=MID$(STR$(X), 2)
1310 PRINT @ DP(X+(DW-1)), DS$;
1320 NEXT
1330 GOSUB 1540
1340 RETURN
1350 ND=FT(2)-FT(1)
1355 GOSUB 1392
1360 PRINT
1370 PRINT "TOTAL DAYS = " ND$
1380 GOSUB 1540
1390 RETURN
1392 ND$=MID$(STR$(ND), 2)
   : LD=LEN(ND$)
   : IF LD<4 THEN RETURN
1394 DR$=RIGHT$(ND$, 3)
   : DL$=LEFT$(ND$, LD-3)
1396 ND$=DL$+", "+DR$
   : RETURN
1400 IF CH=2 THEN PRINT @ 21, "[R] OR [A]"; ELSE
   PRINT
   : PRINT "ENTER [R] - RETURN TO MENU"
   : PRINT "      [A] - ANOTHER RUN"
1410 GOSUB 1500
   : RETURN
1500 RA$=INKEY$
1502 IF RA$="R" OR RA$="A" THEN RETURN
1504 IF CH=2 THEN IF RA$=CHR$(8) OR RA$=CHR$(9) OR
   RA$=CHR$(10) OR RA$=CHR$(94) THEN RETURN
1530 GOTO 1500
1540 SOUND 199, 2
   : RETURN
1550 PRINT @ 480, EM$(EX);
1552 SOUND 220, 12
   : RETURN
1554 SOUND 220, 2
   : SOUND 199, 2
   : RETURN
1560 FOR X=1 TO 42
   : READ DP(X)
   : NEXT
1570 FOR X=1 TO 7
   : READ D1(X)
   : NEXT
1580 FOR X=1 TO 7
   : READ D2(X)
   : NEXT
1590 FOR X=1 TO 7
   : READ D3(X)
   : NEXT
1600 FOR X=1 TO 7
   : READ DD$(X)
   : NEXT
1610 FOR X=1 TO 12
   : READ ML(X)
   : NEXT
1620 FOR X=1 TO 12
   : READ MD$(X)
   : NEXT
1630 FOR X=1 TO 2
   : READ FT$(X)
   : NEXT
1640 FOR X=1 TO 7
   : READ CH(X)
   : NEXT
1650 FOR X=0 TO 5
   : READ EM$(X)
   : NEXT
1660 RETURN
1670 DATA 128, 132, 136, 140

```

```

1672 DATA 144, 148, 152, 192
1674 DATA 196, 200, 204, 208
1676 DATA 212, 216, 256, 260
1678 DATA 264, 268, 272, 276
1680 DATA 280, 320, 324, 328
1682 DATA 332, 336, 340, 344
1684 DATA 384, 388, 392, 396
1686 DATA 400, 404, 408, 448
1688 DATA 452, 456, 460, 464
1690 DATA 468, 472
1700 DATA 00, 14, 28, 42, 57, 71, 85
1710 DATA 14, 28, 42, 57, 71, 85, 00
1720 DATA 28, 42, 57, 71, 85, 00, 14
1730 DATA SUNDAY, MONDAY, TUESDAY
1740 DATA WEDNESDAY, THURSDAY
1750 DATA FRIDAY, SATURDAY
1760 DATA 31, 00, 31, 30, 31, 30
1770 DATA 31, 31, 30, 31, 30, 31
1780 DATA JANUARY, FEBRUARY, MARCH
1790 DATA APRIL, MAY, JUNE, JULY
1800 DATA AUGUST, SEPTEMBER
1810 DATA OCTOBER, NOVEMBER
1820 DATA DECEMBER
1830 DATA 'FROM-DATE', 'TO-DATE'
1840 DATA 64, 68, 72, 76, 80, 84, 88
1850 DATA **1 2 OR 3 ONLY**
1860 DATA **BAD MONTH**
1870 DATA **BAD DAY**
1880 DATA **BAD YEAR**
1890 DATA **NOT A LEAP YEAR**
1900 DATA **TO < = FROM**

```

```

180 IF A$ = "*" THEN GOSUB 600
190 IF A$ = " = " THEN GOSUB 650
200 IF A$ = "^" THEN V = V-1
210 IF A$ = CHR$(10) THEN V = V+1
220 IF A$ = CHR$(8) THEN H = H-1
230 IF A$ = CHR$(9) THEN H = H+1
240 IF A$ = "C" THEN CLS(0)
250 IF H > 63 THEN H = 63
260 IF H < 0 THEN H = 0
270 IF V < 0 THEN V = 0
280 IF V > 31 THEN V = 31
290 IF A$ = "/" THEN C = C+1
300 IF C = 9 THEN C = 0
310 IF A$ = "L" THEN GOSUB 520
320 IF A$ = "S" THEN S = S+1
330 IF S = 2 THEN S = 0
340 IF S = 1 THEN C = RND(8)
350 IF A$ = "1" THEN H = 0
: V = 0
360 IF A$ = "2" THEN H = 63
: V = 0
370 IF A$ = "3" THEN H = 0
: V = 31
380 IF A$ = "4" THEN H = 63
: V = 31
390 IF A$ = ">" THEN GOSUB 720
400 IF A$ = "<" THEN GOSUB 780
410 IF A$ = "!" THEN RUN
420 IF H = 63 OR H = 0 OR V = 31 OR V = 0 THEN SOUND
255,1
430 IF A$ = "B" THEN CLS(C)
440 IF A$ = "J" THEN JJ = JJ+1
450 IF JJ = 2 THEN JJ = 0
460 IF JJ = 1 THEN SET(JOYSTK(0), JOYSTK(1)/2, C)
470 SET(H, V, C)
480 IF A$ = "E" THEN E = E+1
490 IF E = 2 THEN E = 0
500 IF E = 1 THEN RESET(H, V)
510 GOTO 170
520 FOR X = 0 TO 63
530 SET(X, 0, C)
540 SET(X, 31, C)
550 NEXT X
560 FOR X = 0 TO 31
570 SET(0, X, C)
: SET(63, X, C)
580 NEXT X
590 RETURN
600 OPEN"O", #-1, "DATA"
610 FOR X = 1024 TO 1535
620 PRINT#-1, PEEK(X)
630 NEXT X
640 CLOSE#-1
: RETURN
650 OPEN"I", #-1, "DATA"
660 FOR X = 1024 TO 1535
670 INPUT#-1, A
680 POKE X, A
690 NEXT X
700 CLOSE#-1
710 IF A$ = "<" THEN GOSUB 780
720 FOR X = 1024 TO 1535
730 P = PEEK(X)
740 IF P > 239 THEN RETURN
750 POKE X, P+16
760 NEXT X
770 RETURN
780 FOR X = 1024 TO 1535
790 P = PEEK(X)
800 IF P < 16 THEN RETURN
810 POKE X, P-16
820 NEXT X
830 RETURN

```

Drawing with the Color Computer

David Andrew Palmer
7 Frontenac Cresc.
Deep River, Ontario
Canada

This is a drawing program for Color Computers with a minimum of 4K RAM.

```

10 *****DRAW***** BY DAVID ANDREW
: PALMER 1982
100 CLS
110 PRINT"$-----PRINTS CHARACTERS
= -----READS FROM CASS.
*-----PRINTS TO CASS.
!-----DISPLAYS LIST"
120 INPUT"PRESS<ENTER>";PO
130 PRINT"$-----RANDOM COLORS
/-----COLOR CHANGE
C-----CLEARS SCREEN
B-----REVERSES COLORS
E-----ERASE CURSOR
L-----SCREEN BORDERS"
140 PRINT"1-----CURSOR IN UP LEFT
2-----CURSOR IN UP RIGHT
3-----CURSOR IN DN LEFT
4-----CURSOR IN TP RIGHT
J-----TURNS ON JOYSTICK
<-----REPAINTS(DOWN
ONE)>-----REPAINTS(UP ONE)
150 INPUT"PRESS <ENTER> TO START";PO
160 CLS(0)
: C = 5
: H = 31
: V = 15
170 A$ = INKEY$

```

PC-2 Assembly Language—Part 5

By Bruce Elliott

This is the fifth in a series of articles which describe the MPU (microprocessor unit) used in the Radio Shack PC-2 pocket computer. It is our intention to include specific information about the 8-bit CMOS microprocessor, the machine code used by the microprocessor, as well as information about the PC-2 memory map, and certain ROM calls which are available. Please realize that much of what we are talking about refers to the overall capabilities of the MPU, and does not imply that all of these things can be done with a PC-2.

The information provided in these articles is the only information which is available. We will try to clarify any ambiguities which occur in the articles, but cannot reply to questions outside the scope of these articles. Further, published copies of TRS-80 Microcomputer News are the only source of this information, and we will not be maintaining back issues. Parts One, Two, Three and Four of this series were published in the March, April, May, and September 1983 issues, respectively.

The first three articles described the MPU used in the PC-2, including information on the MPU's structure and its machine language. We also gave you details on the PC-2 memory map and the locations of ROM routines which are available. In the fourth article we presented two lists to make finding a particular machine language instruction easier. We also provided some information on how you might begin to use the information we have published. In this fifth article we want to present information on how to create your own machine language routines, and begin describing how to use the PC-2 ROM calls which are available.

CREATING YOUR OWN PROGRAMS

Last month we looked at an existing machine language program and described a procedure (disassembly) for determining how the program did what it was supposed to do. This month I want to define a program and then describe the procedure for creating a workable program that fits the definition. To make things simple, the program we are going to design will do only one thing—display on the LCD the key you press on the keyboard. I know that this program may sound silly. After all, doesn't the PC-2 automatically display the key you press? The answer is no, it doesn't. Try using the INKEY\$ command. With INKEY\$, if you want the character displayed you must display it yourself.

What we are really doing is designing a program which will accept characters from the PC-2 keyboard and display them on the LCD. This program should show you how to do three important things in assembly language: first, how to get information from the keyboard into the computer; second, how to take information that is in the computer and display it on the LCD; and third, how to use the PC-2's ROM subroutines.

In Part 1 of this series (March, 1983, pg. 26) we published a PC-2 memory map. It is in this section of PC-2 memory that we find ROM subroutines.

WHY DO ROM SUBROUTINES EXIST?

In general, any computer consists of similar basic parts. To function, a computer must have a processing unit, input and output functions, working memory to store temporary results, and some sort of control mechanism or program.

In the PC-2, the processing unit is the MPU which we have been describing in this series. The input function is handled primarily by the keyboard, and the output function is handled primarily by the LCD. The working memory is RAM (Random Access Memory), and the control mechanism is in the form of programs stored in ROM (Read Only Memory).

In order to make the PC-2 behave so that you can use it, the manufacturer wrote an operating system to control the various functions of the computer. Part of this operating system is instructions which control the keyboard, the LCD, and BASIC. This is where ROM subroutines come from. To function properly, the PC-2 has to have a routine which looks at the keyboard and stores any key which may be pressed. Likewise, there has to be a routine somewhere which takes a character and displays it on the LCD. The PC-2 memory map tells us where some of these routines are located, and we will use this information to create our machine language program.

IS THIS INFORMATION AVAILABLE ON OTHER COMPUTERS?

Radio Shack has received permission from the original manufacturer of the PC-2 to disclose the information which we are presenting in this series of articles. The information is fixed, and we do not expect it to change.

If you happen to own a different TRS-80 you may have tried to get similar information for that computer and you were told "I am sorry, but we cannot provide you with that information." Why? Well, there are two major reasons. The first and largest reason is that most computers are evolving products. As a computer evolves, the contents of its operating systems also change. If we give you information about where a particular routine is located in the first version of a program or operating system, you are going to expect that information to be true in the second version of that program or operating system also. With few exceptions, every change of a machine language program such as an operating system means a relocation of ALL of the contents of that program.

Because the contents of programs are subject to change with each revision, what Radio Shack typically does is to publish certain "published entry points." These published entry points won't normally change, even if the rest of the

program does change. Other than the published entry points, Radio Shack, in general, will not provide you with other information about the contents of the program. Using only published entry points protects your software from becoming obsolete as soon as Radio Shack issues a new version of the program.

The second major reason for not providing the information is that Radio Shack often does not have permission from the copyright holder to release the information. As an example, Microsoft BASIC on any of our machines is owned by Microsoft. Since Microsoft owns the code, they have the right to tell us what we can and cannot publish.

BACK TO THE PC-2

The stated function of our machine language program is to accept keyboard entries and display the pressed key on the LCD.

A quick glance at the memory map for System Program ROM shows two keyboard scan routines and two routines which output single characters to the LCD.

E243H Keyboard Scan—Wait for Character

E42CH Keyboard Scan—No Wait

ED4DH Output one character to LCD and increment cursor position by one

ED57H Output one character to LCD

(Remember that the H after the address, as in E243H, indicates that the number is in Hexadecimal notation and not decimal.)

E243H

My information on the E243H Keyboard scan routine tells me that the PC-2 will wait for a key to be pressed. Once a key has been pressed, the key's code will be placed in the MPU Accumulator. If a key is not pressed within about seven minutes, the PC-2 will be turned off automatically. Once power-down has occurred, pressing the **(ON)** key will return the computer to the keyboard scan routine.

E42CH

The information on the E42CH routine states that if a key has been pressed, the key code will be in the accumulator. If a key has not been pressed the accumulator will contain 00H.

ED4DH

To output a character using ED4DH, the ASCII code of the character to be displayed is placed in the accumulator and the routine is executed. The character will be placed at the current cursor position, and then the cursor position will be updated.

The current cursor position is stored in memory location 7875H. According to our information, if the old cursor position (before the call to ED4DH) was less than 96H the new cursor position (stored in 7875H) will be the old position plus 6H. If the old cursor position was 96H or greater, the new position will be 00H.

ED57H

To display a character using the ROM routine at ED57H, place the ASCII value of the character to be displayed into the accumulator and execute the ED57H routine. The character will be displayed at the current cursor location and the cursor position will not be updated.

LET'S WRITE THE PROGRAM

I try to program conservatively when I use machine language. What I mean by this is that I try to disturb as few

things as I can. So, the first part of my program will "save the MPU registers." What I mean by this is that I will save a copy of the various registers so I can restore the MPU when I am finished with my program. This is done by using the appropriate push (PSH) instructions to "push" the register values onto the stack.

```
FD C8      PSH A
FD 88      PSH X
FD 98      PSH Y
FD A8      PSH U
```

Now that I have saved a copy of the registers, I want to set the PC-2's cursor position to the left side of the LCD. This would make the cursor position (stored in 7578H) zero (0).

```
B5 00      LDI A, 00H
4A 75      LDI XL, 75H
48 78      LDI XH, 78H
0E         STA (X)
```

Notice that I used three Load Immediate (LDI) instructions. The first LDI puts the cursor position (00H) into the MPU's Accumulator (A register.) The next two LDIs load the X register with the address which stores cursor position (7578H). The fourth instruction (STA) tells the MPU to put the value currently in the A register into the memory location which is currently in the X register.

Now that the cursor is where I want it, it is time to get a keystroke from the keyboard. Since the only thing I want to do is to get a keystroke, I choose to use the routine which waits for a key to be pressed before returning. A ROM routine is executed by using the Subroutine Jump (SJP) command.

```
BE E2 43   SJP E243H
```

We learned earlier that once a key is pressed, the PC-2 stores the ASCII value of the key in the A register. Both display routines I am considering require the ASCII value of the character I want displayed to be in the A register. Since the keyboard scan routine already put the ASCII value in the A register, all I need to do is use a subroutine jump to the proper display routine.

```
BE ED 4D   SJP ED4DH
```

I chose to display each character in cursor position 0, so I used the display routine at ED4DH.

The purpose of this program was to get a character from the keyboard and to display it on the LCD. My program has done that, so I restore the registers by POPping their values (in reverse order) off the stack.

```
FD 2A      POP U
FD 1A      POP Y
FD 0A      POP X
FD 8A      POP A
```

There is one final task which any machine language program which is called from BASIC (as this one will be) must perform and that is to return control of the PC-2 to BASIC. This is accomplished by executing a return command.

```
9A         RTN
```

Here is the completed machine language program along with various comments so I can remember what is happening.

```
FD C8      PSH A      'Save Registers
FD 88      PSH X
FD 98      PSH Y
```

```

FD A8      PSH U
B5 00      LDI A, 00H 'Cursor Position
4A 75      LDI XL, 75H 'Cursor Storage
48 78      LDI XH, 78H 'Location
0E         STA (X) 'Store Cursor
BE E2 43   SJP E243H 'Read Keyboard
BE ED 4D   SJP ED4DH 'Display Character
FD 2A      POP U 'Restore Registers
FD 1A      POP Y
FD 0A      POP X
FD 8A      POP A
9A         RTN 'Return to BASIC

```

TURN IT INTO A BASIC PROGRAM

Now that I have the machine code for my program, I need a way to get the program into the PC-2 and executed. A very straight forward way to do this in the PC-2 is to put the machine language program into a BASIC program shell like the following:

```

10 WAIT 0
20 DATA &FD, &C8, &FD, &88
30 DATA &FD, &98, &FD, &A8
40 DATA &B5, &00, &4A, &75
50 DATA &48, &78, &0E
60 DATA &BE, &E2, &43
70 DATA &BE, &ED, &4D
80 DATA &FD, &2A, &FD, &1A
90 DATA &FD, &0A, &FD, &8A
100 DATA &9A
110 M=16999
120 FOR I=1 TO 30
130 READ A
140 POKE M+I, A
150 NEXT I
160 M=M+1
170 PRINT "      READY"
180 CALL M
190 GOTO 180

```

Line 10 simply sets the PC-2 PRINT command delay time to 0.

Lines 20-100 contain DATA statements into which I have placed the hexadecimal values for my machine language

program. Notice the use of a leading '&' to indicate that the values are in Hex.

Line 110 contains the address (minus one) where I will begin storing the machine language program in memory.

Lines 120-150 POKE the machine language routine into PC-2 RAM memory. Line 160 updates the memory pointer from line 110 so that it contains the actual starting address of my routine (17000 decimal).

Line 170 tells me that the machine language program has been put into memory and will begin executing with the next instruction.

Line 180 tells BASIC to turn control of the PC-2 over to the machine language program which begins at location M (my memory pointer). The PC-2 will set the cursor position to zero, wait for a key to be pressed on the keyboard, display the proper character and return to BASIC.

Line 190 tells BASIC to go back to line 180 and execute the machine language program again.

THAT IS ALL THERE IS TO IT!

If you have followed this series of articles all the way through, you now have enough information about the PC-2 and how it operates to begin writing your own programs in machine language.

Next month we plan on giving you some additional information about the various ROM subroutines which are available to you in the PC-2.

A CLOSING GIFT

Operation codes (op-codes, mnemonics) are short names which programmers give to machine language commands to make them more readable, and more memorable. We have given you several lists with op-codes and have provided some detail on what the commands do. At least one person has asked "How am I supposed to pronounce those funny looking things?"

Below is a listing of the various PC-2 op-codes and a recommended "name" or pronunciation for each.

| Op-Code | Suggested Name | Op-Code | Suggested Name | Op-Code | Suggested Name |
|---------|------------------------------------|---------|-------------------------------|---------|-------------------------|
| ADC | Add with Carry | PSH | Push | HLT | Halt |
| ADI | Add Immediate | POP | Pop | OFF | OFF |
| DCA | Decimal Add | ATT | Accumulator to T Register | JMP | Jump |
| ADR | Add Register | TTA | T Register to Accumulator | BCH | Branch |
| SBC | Subtract with Carry | TIN | Transfer and Increment | BCS | Branch Carry Set |
| SBI | Subtract Immediate | CIN | Compare and Increment | BCR | Branch Carry Reset |
| DCS | Decimal Subtract | ROL | Rotate Left | BHS | Branch Half Carry Set |
| AND | AND Accumulator | ROR | Rotate Right | BHR | Branch Half Carry Reset |
| ANI | AND Immediate | SHL | Shift Left | BZS | Branch Zero Set |
| ORA | OR Accumulator | SHR | Shift Right | BZR | Branch Zero Reset |
| ORI | OR Immediate | DRL | Decimal Rotate Left | BVS | Branch Overflow Set |
| EOR | Exclusive OR Accumulator | DRR | Decimal Rotate Right | BVR | Branch Overflow Reset |
| EAI | Exclusive OR Accumulator Immediate | AEX | Accumulator Nibble Exchange | LOP | Loop on Positive |
| INC | Increment | SEC | Set Carry | SJP | Subroutine Jump |
| DEC | Decrement | REC | Reset Carry | VEJ | Vector Jump |
| CPA | Compare Accumulator | CDV | Clear Divider | VMJ | Vector Unconditional |
| CPI | Compare Immediate | ATP | Accumulator to Port | VCS | Vector Carry Set |
| BIT | Bit | ITA | Port Input to Accumulator | VCR | Vector Carry Reset |
| BII | Bit Immediate | SPU | Set PU | VHS | Vector Half Carry Set |
| LDA | Load Accumulator | RPU | Reset PU | VHR | Vector Half Carry Reset |
| LDE | Load and Decrement | RDP | Resets display flip-flop | VZS | Vector Zero Set |
| LIN | Load and Increment | SDP | Sets display flip-flop | VZR | Vector Zero Reset |
| LDI | Load Immediate | SPV | Set PV | VVS | Vector Overflow Set |
| LDX | Load X | RPV | Reset PV | VVR | Vector Overflow Reset |
| STA | Store Accumulator | SIE | Set Interrupt Enable | RTN | Return from Subroutine |
| SDE | Store and Decrement | RIE | Reset Interrupt Enable | RTI | Return from Interrupt |
| SIN | Store and Increment | AM0 | Accumulator to Timer, Bit 9=0 | ME0 | Memory Enable 0 |
| STX | Store X | AM1 | Accumulator to Timer, Bit 9=1 | ME1 | Memory Enable 1 |
| | | NOP | No Operation | | |

Write A Codefile!

by Kimberly Bilstad Ness and Mary Turner

A USDA aide called AGRI-STAR headquarters to ask how he could access reports most efficiently. He wanted to print all current news stories early each morning for the Secretary of Agriculture to review. We suggested he build an on-line codefile.

A researcher from a large agricultural firm needs to access growing degree day weather reports from each climatic division in seven states each day. She uses AGRI-STAR codefiles.

David Prentiss follows the corn and soybean markets, checks long term and severe weather reports and reads ag news stories on AGRI-STAR early each evening from his farm operation near Delta, Ohio. He can get all the market and analysis reports he needs in less than six minutes. By using a codefile, he can access those reports even more quickly in sequence by entering one three-keystroke codefile code.

PICK REPORTS FIRST, BUILD A CODEFILE FAST

AGRI-STAR provides electronic access to thousands of volatile pieces of information; each piece is identified by title and by a report code. Reports and their codes are listed for quick reference in print form in the AgriScan Index of Reports and Keywords. Or, codes can be located on-line by entering words that identify the kind of information desired. For example, to get a list of all available cash corn market reports on AGRI-STAR, type CASH.CORN at any ★.

If the same reports are accessed regularly, storing those report codes in a codefile will save on-line connect-time and eliminate the need to locate, recall or enter a long series of codes each time the reports are accessed.

We are learning that many of our AGRI-STAR clients are becoming sophisticated codefile builders. Building codefiles is fast and easy; it's possible to build up to 10 different codefiles, each of which can store up to 10 report codes.

STREAMLINE TO SAVE TIME

Using codefiles can help streamline information gathering on AGRI-STAR according to one dedicated user, Debra Streeter, a Top Farmer market analyst. Says Streeter, "Codefiles are well worth the small investment in time it takes to learn how to use them." She has created her own codefiles for specific days of the week. For example, she has grouped reports that are updated weekly on Friday in one codefile and weekly Wednesday reports in another.

Streeter also creates a special codefile shortly before a major government report is due to be released and includes the trade expectations, the report itself and trade reaction reports.

A STREETER SAMPLE

Several days before the U.S. Commerce Department is scheduled to release its housing start figures, Streeter

builds a codefile that includes CNS434, PRE-HOUSING START SURVEY; CNS415, HOUSING STARTS; CNS842, HOUSING STARTS REACTION and CNS820, HOUSING REACTION—COMMERCE. The day the reports are released, she has at her fingertips the information she needs to help her determine her own economic analysis.

Streeter offers some tips to codefile users. She suggests that if reports are to be printed, fill a codefile with 10 reports and use the RUN command to scroll reports without interruption, saving connect-time. If reports are not to be printed, merely viewed on-line, she suggests building codefiles with only five or fewer reports in each.

A HOG OPERATION SAMPLE

Just as analyst Deborah Streeter does, many farmers make their livings analyzing market trends and deciding when to buy and sell commodities. And using codefiles can help make that job easier as well.

For example, a hog farm operator marketing from a central Illinois location might build a codefile to track daily cash hog markets as reported in the USDA's Federal-State Market Reports, private sources reporting directly from the major local livestock trading centers, and favorite hog market analysis and recommendation reports.

That farm operator would likely access the sample codefile below at about noon each day. The codefile contains the following reports:

DAR2 DOANE'S LIVESTOCK MARKET WATCH. This report is updated twice a day on AGRI-STAR to provide cattle and hog market reviews, analysis and advice.

CNS358 IOWA-MINN OP/MIDSESSION HOGS

CNS367 SIOUX CITY OP/MIDSESSION HOGS

CHS368 ST PAUL OP/MIDSESSION HOGS

CNS364 PEORIA OP/MIDSESSION HOGS

CNS373 WEST FARGO CASH HOGS. These are the USDA reports updated late each morning.

CNS838 ILLINOIS HOGS-PRIVATE SOURCES

CNS857 JOLIET HOGS-PRIVATE SOURCES

CNS830 PEORIA HOG OPEN-PRIVATE SOURCES.

These private source views on the local markets are updated just before 9:00 a.m. CDT each day.

TO BUILD A FILE

To create a codefile on AGRI-STAR, simply follow these steps:

Step 1 Type WRITE CODEFILE1 (We'll use 1 to name the file, but any number between 1 and 10 can be used) and list the report codes with a space between each code. Like this: ★ WRITE CODEFILE1 DAR2 CNS358 CNS367 CNS368 CNS364 CNS373 CNS838 CNS857 CNS830

(Continued on page 40)

Install the Model III Business Graphics Package on Your 5 Meg Hard Disk

by Annette Zamberlin-Main



I've just run across, and then tested, a terrific little set of procedures for installing the Model III Business Graphics package (Cat. No. 26-1597) on the 5 Meg Hard Disk (Cat. No. 26-1130). The Business Graphics package has always been a powerful tool for anyone involved in the production and use of bar graphs, pie charts, scatter charts, and line charts. Installing the graphics package on your 5 Meg Hard Disk will make it even more convenient, not to mention much faster to use.

On the installation I tried, we used a 2-Drive Model III and one 5 Meg Hard Disk. You can, if you so desire, use up to a four hard drives and a four floppy drives configuration.

IS YOUR HD INITIALIZED?

The preliminary subset of procedures deals with the initializing or reinitializing of the system. Before any installation is made, your hard drive system must first be initialized. It may already be initialized so you may proceed to the next section of instructions. You may want to reinitialize your hard disk drive; in which case directions for reinitialization may be found in Appendix B of your Hard Disk Operating System manual (Cat. No. 26-1130).

Remember that it is during the initialization process that LDOS assigns the logical drive numbers to your hard drive(s) and floppy drive(s). In the configuration that we are using, our one hard drive contained four logical drive numbers (0-3) and two logical drive numbers for the two floppy drives (4-5) contained in the Model III:

- Logical Drive 0 Hard Drive (primary)
- Logical Drive 1 Hard Drive (primary)
- Logical Drive 2 Hard Drive (primary)
- Logical Drive 3 Hard Drive (primary)
- Logical Drive 4 Floppy Drive (first)
- Logical Drive 5 Floppy Drive (second)

Hard Disk Drive initialized? All right?

INSTALLATION OF THE PROPER DEVICE DRIVERS

Another member of the subset of preliminary procedures deals with the installation of the proper printer/plotter drivers. The Business Graphics chart diskettes are set up to produce output on the Line Printer VIII. If you are using a different output device, you must configure the chart diskettes for your printer or plotter before attempting to install any of the diskettes onto the HD. Please see Chapter 9 of the Business Graphics Analysis Pak for a further discussion of the procedures.

The Initialization Diskette which you used in the above process now becomes your Boot Diskette. Place your boot diskette in your first physical floppy drive, boot-up the system and let's proceed with the installation of the Business Graphics package.

LDOS CONV UTILITY

There are four floppy diskettes contained in your Business Graphics package that you will want to install on your hard disk system. There is a Model III diskette in the Business Graphics package for each type of chart or graph: line, bar, scatter, and pie.

It is now time to delve into the LDOS utilities for our next subset of procedures. The CONV utility will allow us to move files from a Model III TRSDOS diskette onto an LDOS formatted drive. Two drives are required. The syntax is:

```
CONV :s :d (parm,parm, . . . ,parm)
```

:s is the Logical Source Drive. It cannot be Logical Drive 0.

:d is the Logical Destination Drive.

The allowable parameters are as follows:

VIS Convert visible files.

INV Convert invisible files.

SYS Convert system files.

NEW Convert files only if they do not exist on the destination disk.

OLD Convert files only if they already exist on the destination disk.

QUERY Query each file before it is converted.

abbr: All parameters may be abbreviated to their first character.

For the purposes of our experiment, when each of the four diskettes from the Business Graphics package were inserted in Logical Drive 4, I typed:

CONV :4 :1 (NEW,Q=N)

The above means that all files on the TRSDOS disk, which is in Logical Drive 4 (the Source Disk), are converted onto Logical Drive 1 (the Destination Disk), only if they do not already exist on the Destination Disk. For the sake of expediency, Q=N is used so that I would not be asked before each separate file on the disk is moved.

After the first disk was converted and installed onto the destination drive, I called up the destination drive's DIRectory to check to see what files I now had on it. For my particular experiment, I had converted the Model III Business Graphics disk containing the Bar Chart programs. My DIRectory listing for Logical Drive 1 now included two new files TRSBAR and TRSCHART/CMD.

POSSIBLE PROBLEMS

Having worked with the Business Graphics package before, solely with floppy diskettes, I realized that every diskette in the package contained one user file called TRSCHART/CMD. I could see trouble brewing ahead if I continued on with the LDOS CONV utility and did not make some readjustments to file names.

What I would have on Logical Drive 1 if I continued on with the LDOS CONV utility and made no readjustments would be five files:

TRSBAR TRSPIE TRSSCT TRSLIN TRSCHART/CMD

The trouble now becomes extremely evident. During the execution of the CONV utility, TRSCHART/CMD was overwritten three separate times. All that would result would be four driver files (TRSBAR, TRSPIE, TRSSCT, and TRSLIN) and only one TRSCHART/CMD file which would load its partner driver file and provide the actual user interaction. Not a very good situation.

LDOS RENAME COMMAND

In order to circumvent this potential disaster, I utilized the LDOS library command of RENAME after each execution of the CONV utility. The LDOS RENAME command allows you to change the filename and extension of a given file.

The syntax is:

```
RENAME filespec1 TO filespec2
```

After the first execution of the CONV utility, when I installed TRSBAR and TRSCHART/CMD onto Logical Drive 1, I typed:

```
RENAME TRSCHART/CMD TO BARCHART/CMD
```

Please remember that this RENAME command must be utilized after the installation of each disk of the Business Graphics package in order to insure against any potential conflicts because of duplications of file names.

In total you would have typed:

```
RENAME TRSCHART/CMD TO BARCHART/CMD
RENAME TRSCHART/CMD TO PIECHART/CMD
RENAME TRSCHART/CMD TO LINCHART/CMD
RENAME TRSCHART/CMD TO SCTCHART/CMD
```

All of the four CONVs utilities and four RENAMEs commands completed, I now had fully installed the Business Graphics package onto my 5 Meg Hard Disk.

GETTING FANCY

If you want to get fancier with all of this, say install a user menu onto the hard disk which enables selection of one of the

four graphics packages, Profile III Plus HD (Cat. No. 26-1593) may be just what you are looking for. Check the documentation in the software package concerning "Defining User Menus" and simply follow the straight forward instructions. The menu you create will increase your Business Graphics files access efficiency and lend a professional tone to the entire selection process.

QUICK CHECKLIST

- Initialize or reinitialize HD.
- Install proper printer/plotter drivers onto chart floppy diskettes.
- Use CONV utility to move files from Model III TRSDOS diskette onto the LDOS formatted drive.
- Remember to RENAME TRSCHART/CMD each time the file is moved onto a drive.
- Optional: Add a selection menu using Profile III Plus HD.

MAGAZINES

Below are five magazines of special interest to TRS-80 owners that we believe have editorial content of high quality and will be of use to our customers.

Basic Computing—The TRS-80
User Journal (Name change for
80 US Journal—covers all TRS-80's)
3838 South Warner Street
Tacoma, WA 98409
(206)475-2219

Color Computer Magazine
Highland Hill
Camden, ME 04843
(207)236-9621

Color Computer Weekly
P.O. Box 1355
Boston, MA 02205

Rainbow (Covers the TRS-80 Color Computer)
P.O. Box 209
Prospect KY 40059
(502)228-4492

two/sixteen magazine
P.O. Box 1216
Lancaster, PA 17603
(717)397-3364

Codefile (From page 38)

Step 2 Press **(ENTER)**. The computer will confirm the report codes entered and stored in the codefile.

To access the codefile, type CODEFILE1 or CF1 at any *****. If more than one codefile has been created and must be accessed in sequence, the codefile numbers can be entered in a string. For example, type CF1 CF3 CF10 to display all the reports stored in those codefiles.

One of the major design principles inherent in AGRI-STAR is to make access to any one or many of the thousands of reports simple and fast. The codefile feature meets that objective and ought to be considered whenever reports must be accessed regularly.

The PC-2 RS-232 Interface—Part I

By Peter Levy

This is the first in a series of articles designed to introduce the reader to the PC-2's RS-232 Interface (26-3612). This month I plan to explain what an RS-232 is, roughly how it works, and go into a little detail about the particular RS-232 Radio Shack sells for the PC-2. Later articles will discuss specific applications. As the astute reader has probably noticed, the world is getting rather full of computers. The amount of information stored on all these machines is beyond reasonable description. It is useful for all these computers to be able to share information, since putting information into a computer from scratch is something of a pain. Hooking up a cable and then drinking coffee while the machines chat is a whole lot less grief than retyping some huge list of information.

The problem here is that if we just told all the computer manufacturers to build machinery with communications channels, there would be at least one distinct data encoding scheme for each computer designer in the industry. Communicating computers aren't too useful if one is saying "Who's there?" and the other replies "Wer ist da?". Some sort of standard is needed. Several such standards exist, but by far the most common standard for microcomputers is one named RS-232-C, which simply defines a set of conventions for computers to use while sending information over a wire. Although the actual RS-232 standard just defines pin locations and voltages and such, a fairly universal communications protocol has been established around it. Communications circuitry and devices that work according to these particular conventions are usually just called RS-232 devices. In fact, a general purpose RS-232-standard communications device is likely just to be called an RS-232. In the interest of simplicity, when I refer to an "RS-232" in this article, I'll actually be referring to the entire device. Also, I'm going to call a binary 1 bit (this is the same thing as a bit which is "set" or "true") a "true" bit.

The basic notion behind the RS-232 is simple. If a computer can send one bit—just one "true/false" unit of binary information—it can, given time, send enough bits to transfer any amount of information.

The RS-232 sends individual bits over a wire by changing the voltage present in the wire. The receiving RS-232 is built so that it can distinguish varying voltages and, according to its interpretation of those voltages, set or reset a data bit which is "visible" to the computer the RS-232 serves. Thus, one bit is transferred.

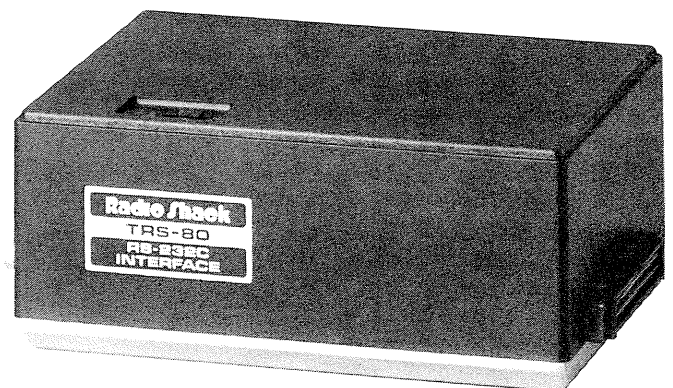
A single bit doesn't contain very much information, so computers (and programmers) seldom work with them. Microcomputers usually handle bits in groups of eight, in units called bytes. These are called eight-bit machines, of course. The industry is evolving toward sixteen-bit machines (Obviously, the more bits a computer can deal with at once,

the faster and more powerful it is.), but at the RS-232 level communications are still done with eight-bit (or less) protocols.

Communications would be a real pain if programmers had to work at the bit level, so the RS-232 is designed to handle bytes for us. We give the transmitting RS-232 a complete byte, and the RS-232 disassembles it into its component bits, sending each one in turn over the wire. The receiving RS-232 reads each bit in sequence, assembles the bits into a complete byte, and hands it to the computer for processing.

Since, when the RS-232 transfers a byte, it has to do it one bit at a time,—it transfers the bits of a byte one after another, in sequence, over a single wire—it is called a SERIAL device. This is opposed to PARALLEL communications where all the bits are sent at once over several parallel wires.

When it processes a byte, the RS-232 also throws in some bits of its own for control purposes and to make sure the other RS-232 knows what is going on. First, it sends one "true" bit before anything else, just to warn the other RS-232 that a new byte is starting. This is called a START BIT. After it has sent the data bits, it sends some more extra bits to tell the other RS-232 that this is the end of the byte. These are called STOP BITS. Between the last of the data and the stop bits, it may also send a PARITY BIT. A parity bit is set up so that the total number of "true" bits after the start bit and before the stop bit(s) is always either odd or even, which gives the receiving RS-232 a fairly good way of knowing if something went wrong. If we're using even parity (under which the number of true bits after the start bit and before the stop bit is always even) and the receiving RS-232 counts five "true" bits in the data bits plus a "true" parity bit, then the total number of "true" bits is even, and the byte has probably been transferred correctly. Parity can be odd, even, or none with no parity meaning that the parity bit is not sent at all. Some machines also allow an "ignore" parity, in which the parity bit is present but ignored.



Since the RS-232 is sending start and stop bits, timing (between bytes) isn't critical, and there is no need to keep up a continuous data stream between bytes. Yes, two RS-232's must stay synchronized with each other during the transmission of a single byte from start bit to stop bits, but at the working level—the byte level—they (and the programmer) don't need to worry about timing considerations. This makes the RS-232 an **ASYNCHRONOUS** device. The opposite of this, a **SYNCHRONOUS** device, is exemplified by a cassette drive, which is timing-critical and cannot pause between bits, or even bytes, of information.

Now, let's consider what can go wrong during the transmission of a byte via an RS-232.

Well, first, a bit can get scrambled somehow (commonly caused by a noisy telephone line). This will probably (but not certainly) cause the parity of the received byte to be wrong (assuming parity is in use) and will therefore be detectable to the receiving RS-232 as a **PARITY ERROR**. The next possibility is that the receiving RS-232 might somehow lose synchronicity with the sending device. It winds up being sort of "out of phase" with reality and can't figure out which bits are start, data, parity, or stop information. This condition is called a **FRAMING ERROR**. Finally, if the RS-232 we're using can only store one complete received byte and a new byte is received before the old one has been asked for by the computer the RS-232 serves, the RS-232 throws away the old byte and saves the new one. The lost byte has been overrun by the new one, and this is called an **OVERRUN ERROR**.

The RS-232 can detect any of these errors, and via circuitry provided for that purpose, it can tell the computer what is wrong. It is up to the computer, or actually its communications program, to detect the RS-232's error signal and somehow notify the operator.

Now, when using this wonderful device, we need to consider a few things about the nature of the information we're sending. If we're sending it over a phone line, which is a fairly noisy medium, it should be sent slowly and carefully so that each received bit can be examined carefully. If the two computers are near each other with a high-quality cable in use, we can use a much higher speed. If we're just sending text we can do it with only seven data bits per byte, speeding our transmission by one-eighth, and for some purposes we may only need as few as five data bits. We may or may not want to use parity bits. Fortunately, the RS-232 lets us select which of these different **PROTOCOLS** is to be used, within limits.

The first protocol is speed. Serial transmission speed is measured in bits per second, called **BAUD**. Most phone lines are only good for about 300 baud with ordinary equipment. RS-232's generally allow speeds from 50 to 19,200 baud, with some going to 38,400. Although the baud rate actually only defines the rate at which bits within a byte are sent, this ultimately limits the speed with which a byte can be processed. At 300 baud about 30 characters of text per second are transmitted.

Next is **WORD LENGTH**, which is the number of data bits used per byte (a byte is sometimes called a word). This can be from five to eight bits. Next is the parity convention—odd, even, or none. Last is the number of stop bits, usually one or two. Yes, you in the back, there is such a thing as 1.5 stop bits. No, I don't want to explain it just now.

All this flexibility creates one problem: namely, that for two RS-232's to communicate, they must both use exactly the

same protocol. If they're set differently then all sorts of confusion will result over which bits are start, data, parity, and stop bits. The RS-232 is designed so that these protocols can be set by the user, and this is generally done for him by the computer via a command provided for the purpose. In the TRS-80 world this command is usually called **SETCOM** (SET Communications parameters).

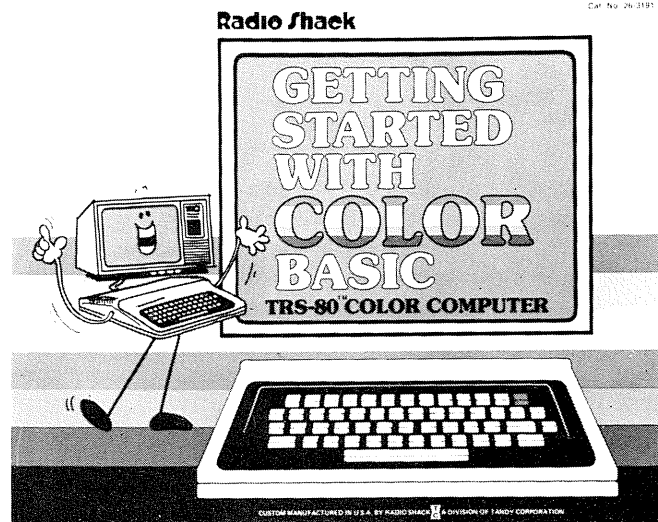
The PC-2 RS-232 accessory includes complete RS-232 communications hardware as described above, and so gives the PC-2 the ability to communicate with any other RS-232 device (at least in theory). But it is also a lot more than just a communications board. It includes an 8K ROM (8092-byte permanent memory) which contains machine-language instructions for PC-2 BASIC telling it what to do with the added hardware, how to behave as a terminal, and how to recognize and process all the new commands the PC-2 needs to work with the RS-232. There is some additional circuitry that provides sort of a translation service between the computer and the RS-232 itself so they can talk via the expansion bus. The case also contains a set of rechargeable batteries, to keep the whole thing portable, and all the plugs and jacks and connectors and stuff needed to hook it all together.

In later articles, I'll go into some specific applications of the PC-2 RS-232, including a talk with CompuServe (next month), how to work with serial printers, and whatever else comes to mind later (requests welcome).

IF/THEN/ELSE Statements

Ray B. Blessum
5657 West Rim Circle
Riverside, CA 92509

I have noticed that neither of the Getting Started books for the Color Computer list **ELSE** as an acceptable BASIC word; however, I have found that the machine accepts **IF/THEN/ELSE** sequences. This is useful to know in translating Model I programs to run on the Color Computer. Such translations make many programs written for Model I available. The chief change is substituting **PRINT#-2**, for **LPRINT** and getting around the **DEFINT**, etc., statements.



Notes on Previous Issues

NOVEMBER 1981

Reading Profile II Files from BASIC

David Boing
500 S. Main St.
Akron, OH 44318

The procedure suggested in the November, 1981, *Microcomputer News* works fine for reading Profile KEY files.

However, if you are creating a new KEY file from BASIC to be used by Profile, then a problem exists if all three sub-records of a physical record are not written.

The problem is the FIELD statement. The buffer is 256 bytes long. The FIELD statement will handle this correctly for the third sub-record but will put garbage into the second and third or third sub-record when writing the first or second sub-record respectively.

The following is my solution:

```
PR=INT((LR-1)/3)+1 : SR=LR-3*(PR-1)-1
FIELD x,(85*SR) AS XX$,6 AS A$,.....,((2-SR)*85)
AS XZ$
LSET A$=.....:LSET XZ$=STRING$((2-SR)*85,32):PUT
x,PR
```

FEBRUARY 1982

3-D Color Graphics

Douglas W. Evans
577 Carter Ave.
Woodbury, NJ 08096

I discovered a way to shorten the 3-D Color Graphics program and to save some memory using the following changes:

```
130 CLS
: INPUT"RX,RY,RZ";R(1),R(2),R(3)
: INPUT "HOW MANY MOVES";M
150 R(1)=R(1)*PI/180
: R(2)=R(2)*PI/180
: R(3)=R(3)*PI/180
Delete lines 220-370
192 FOR Q=1 TO 7
194 READ Y,Z,N
195 A(X,NY)=N1
: A(X,NZ)=N2
200 N1=A(X,Y)
: N2=A(X,Z)
: T=R(N)
210 GOSUB 1000
212 NEXT Q
213 RESTORE
: NY=Y
: NZ=Z
215 GOTO 390
2080 INPUT "WHICH --";Q
: IF Q<1 OR Q>6 THEN 2000
5000 DATA 1,2,3,4,5,3,1,3,2,4,6,2,3,2,1,6,5,1,0,0,1
```

Chris Petit
Rt. 3 135 Davis Drive
Luling, LA 70070

Thanks to Mark Granger's 3-D Color Graphics program, I realized one could save any graphic picture to tape by knowing the starting and ending memory locations. Everything went well with the command CSAVEM "FILENAME",1536,7679,0 for PMODES 3 and 4 (4 graphic pages), but after purchasing the Color Computer disk drive,

the command SAVEM "FILENAME",1536,7679,0 seemed to work until trying to retrieve the picture. After many I/O errors, I decided to look through some old issues of *Microcomputer News*, and I ran across the article "Answers to Interesting Questions" (March 1982). It provided the clues I needed to come up with SAVEM "FILENAME",3584,9727,0. These numbers work in PMODE 3 and 4 without changing the RAM structure after power up. I've also tried Hex numbers but retrieval of the picture seems to go slower. To retrieve the picture simply type or include in your BASIC program CLOADM"FILENAME" or LOADM"FILENAME".

Stunt Racer

Steve Rapp, Jr.
150 Dogwood Land
Bloomfield, Indiana 47424

Here is a hall of fame for high scorers addition to the "Stunt Racer" program written by Y. Maksik. (The REM statements could be taken out to conserve memory.)

```
55 IF A<25 OR A>1000 THEN 50
Add to line 280 :GOTO 50000
50000 IF SC+S1<375 THEN END '375 COULD BE CHANGED TO
A HIGHER OR LOWER NUMBER
50001 PRINT
: PRINT "YOU QUALIFY FOR"
50002 PRINT "THE STUNT RACER"
: PRINT "HALL OF FAME"
: PRINT"*****"
50003 PRINT
: PRINT "ENTER YOUR INITIALS BELOW."
50004 C$=INKEY$
: IF C$="" THEN 50004
50005 D$=INKEY$
: IF D$="" THEN 50005
50006 E$=INKEY$
: IF E$="" THEN 50006
50007 CLS
: PRINT
50008 PRINT "STUNT RACER *HALL OF FAME*"
: REM BY STEVE RAPP, JR.
50009 PRINT "*****"
: REM COULD USE STRING$
50010 PRINT
: PRINT
50011 N$=C$+D$+E$
50012 PRINT " ";N$;"-----";SC+S1
```

In lines 50013-50020 you can put your own permanent high scores using "PRINT" statements. Note: This score is non-permanent.

```
50050 PRINT@ 979, "PLAY AGAIN?";
50052 I$=INKEY$
: IF I$="" THEN 50052
50053 IF I$="Y" THEN RUN ELSE STOP
```

For people who want a challenge, delete line 90 and RUN it. Type a '1' for length of obstacles.

MARCH 1982

Relocating Machine Language Programs

Michael J. Brady
864 Carr Ave.
Santa Rosa, CA 95404

I recently made use of your March 1982 article regarding the moving of tape programs to disk. I successfully moved tape SCRIPSIT (Ver.1.0) using the information in the article, but could not get it to execute. The only way I could get it to work was to use the following procedure (this process works best with TRSDOS 2.3):

- 1) LOAD SCRIPS/CMD (DO NOT execute!)
- 2) Get into Level II (**BREAK**)-RESET from 2.7 or BASIC2 from 2.3)
- 3) POKE 14308,0 (To set the cassette port to 1. I use the EI ports)
- 4) SYSTEM and at the *?/38608 (the decimal equivalent of the move-up routine's address)

If there is some text in the buffer, I have been successful with RESEting out of SCRIPSIT to do something like changing character sets, then restarting with a SYSTEM and /.

Randall J. Britto
4209 Sherwood Ave.
Decatur, GA 30035

The article about relocating machine language programs was a life saver. In the article you said that "... if all these procedures outlined are followed exactly, there shouldn't be any trouble." And you were right with Scripsit and several of the others, until I got to EDTASM (Series I).

With this one I had problems. I used a utility program that told me the loading, entry, and ending addresses of a machine language tape. I found that my EDTASM (Series I) started at address 4646 not the 4AEE as stated. Using address 4646, it loaded the first time.

If there are others that can't seem to get it to work, you might want to try that value for S1 and S2.

Just an added note for LDOS users. None of these relocations seem to work with LDOS yet (gives us something to work on).

APRIL 1982

Perpetual Calendar

A. Arnold Weiss
Apt. 1626 Kennedy House
1901 J. F. Kennedy Blvd.
Philadelphia, PA 19103

The Perpetual Calendar program has a bug. If run as printed, you get an "OD" error in line 1340. This can be corrected by changing the "HDT\$" in line 1345 to "HOL\$". The following changes also help clean up the program:

```
In line 190, change "9999" to "10000"
In line 9999, change "STOP" to "RETURN"
Add the following lines.
5 PCLEAR1
2041 M=H/10
    : IF M=FIX(M) THEN K=1 ELSE K=0
2042 IF K=1 THEN GOSUB 11000
10000 END
11000 PRINT " Press any key to continue"
11010 R$=INKEY$
    : IF R$="" THEN 11010
11020 RETURN
```

The changes in lines 190 and 9999 prevent the program from coming to a grinding halt. Line 10000 is just an "END" statement. Lines 2041, 2042, 11000, 11010 and 11020 prevent the Holidays from scrolling off the screen. Line 5 clears as much memory as possible.

Ken Heberle
116 Goodrich
Erie, PA 16508

Here are some enhancements for my Perpetual Calendar program.

- 1) To make it run faster add the following lines:


```
35 POKE 65495,0
415 R$=""
```

The POKE instruction above will cause the program to execute much faster, but you will not be able to use any tape I/O functions until you press RESET. Change the following lines to read:

```
1080 IF R$="" THEN DD=0 ELSE GOSUB 1300
2080 IF R$="" THEN 1050
```

Delete line 961

- 2) To keep Holidays/etc. from scrolling off the screen if you have more than 12 in a month add the following lines:

```
2039 HC=0
2043 SS=1
2047 HT=INSTR(SS,HOL$(H),CHR$(13))
2048 IF HT<>0 THEN HC=HC+1
    : SS=HT+1
    : GOTO 2047
2049 HC=HC+1
    : IF HC>12 THEN GOSUB 8500
8500 PRINT "PRESS ANY KEY TO CONTINUE";
8510 R$=INKEY$
    : IF R$="" THEN 8510
8520 PRINT
    : HC=0
    : RETURN
```

- 3) To add Daylight Savings Time add the lines below:

```
3620 IF D1>4 THEN HX=36-D1 ELSE HX=29-D1
3630 HOL$(HX)="BEGIN STANDARD TIME"+CHR$(13)+
    " (** SET CLOCKS BACK **)"
3650 RETURN
3982 IF MO<>4 THEN 3990
3984 HX=29-D1
3986 IF D1=6 THEN HX=30
3988 HOL$(HX)="BEGIN DAYLIGHT SAVINGS TIME"
    +CHR$(13)+ " (** SET CLOCKS AHEAD **)"
```

MAY 1982

Ultra Precision Multiplication

Jack Watts
1925 Kalakaua Ave., Apt. 2703
Honolulu, HI 96815

Use of the following modifications will enable the Ultra Precision Multiplication program to handle many digit decimal numbers. The F is a flag used to eliminate zeros before the decimal or first non-zero number.

Note the change in line 310: DP+1.

Line 321 retains zeros after the first non-zero digit is printed. Without this line, zeros embedded in the middle or end of the string would be omitted.

Line 322 eliminates zeros before the decimal or first non-zero numeral is printed.

The changes are:

```
276 F=0
310 FOR T=DP+1 TO 1 STEP -1
320 IF T=(D1+D2) THEN PRINT ". ";
    : F=1
    : GOTO 330
321 IF A(T)<>0 THEN F=1
322 IF T>=(D1+D2) AND A(T)=0 AND F=0 THEN 340
```

Verifying Programs and Data Files

Ben H. Nation
P.O. Box 391
Fairfield IL 62837

Line 25 of this program should be changed to read:
 INPUT "ENTER LOGICAL RECORD LENGTH (LRL)";L

JUNE 1982

Concentric Circles

Steve Havens
67 North Sable St., #15
Keeseville, NY 12944

There are mistakes in lines 70 and 90 of my program, "Concentric Circles," as it appeared in the June issue. Change lines 70 and 90 to read as follows:

```
70 I=63+Y*(2.6667*SIN(A))
   : N=23+Y*COS(A)
90 NEXT A
   : Y=Y-1-Z
   : IF Y<0 THEN 100 ELSE 60
```

Graphs for the PC-2

Randy Klindt

If you enter eight items in the line graph section of this program, after it finishes printing all eight it will draw a line down to the bottom. The problem is in line 410. Change line 410 to read:

```
410 FOR J=1 TO I-1
```

JULY/AUGUST 1982

Accounts Payable (26-4505 Ver. 1.0/2.0)

Christopher G. Hardin
P. O. Box 847
Arleta, CA 91331

The information regarding the Print Checks option (line 2410) is correct. But please be advised that Trinity Forms Company (advertised in the back of the Accounts Payable Manual) supplies two different formats for their Accounts Payable checks. If you are running your Accounts Payable program with Trinity Accounts Payable checks which have a 1/2" left side ringer, the program and checks are just fine. But, if you are using their checks that have a 3/4" left hand side ringer, the following correction is required. Change line 2410 to read:

```
2410 LPRINT L1$
   : LPRINT TAB(5);NA$
   : LPRINT L1$
   : LPRINT L1$
   : RETURN
```

The necessary TAB statement can be any number between 3-10, depending upon the number of characters in your name string.

SEPTEMBER 1982

General Electric ASCII Sequential Files

John M. Price
P. O. Box 194
White Sulphur Springs, MT 59645

In General Electric ASCII Sequential Files, Mr. Halloran's method is very useful when there are many lines of data. However, both Mr. Halloran and Michael Guerard (in an update in the April, 1983 issue) used the MID\$ statement to separate the line number from the first datum.

I have found that it is easier to put a comma between the line number and the first datum, when creating the data file. Then, in the INPUT statement of the actual program, I make the first variable a "dummy" variable. This way the line number is assigned to the "dummy" variable and is conveniently out of the way.

I use a descriptive variable name called SKIP for the

"dummy" variable. This way, when I'm reviewing the program or looking for errors, I don't forget that it is a "dummy" variable. Also, if the INPUT statement is within a FOR loop, be sure that you don't array the "dummy" variable and waste memory.

By using this method, two problems are eliminated. First, the MID\$ statement is no longer needed and second, you don't have to worry about the length of the line number.

OCTOBER 1982

Coded Message

Harris Bockover
1001 S. 27th
Fort Dodge, IA 50501

I found Peter L. Vogel's program that solved David Snider's Hex/ASCII problem to be an excellent program. However, there is one small problem. It didn't solve the whole coded message. I added the following lines to Mr. Vogel's program (and ran it on my TRS-80 Color Computer.)

```
155 FOR T=1 TO 8
295 NEXT T
305 DATA 2020594F5527524520534D4152544552
310 DATA 205448414E20594F55204C4F4B2120
320 DATA 205448415427532050524F4241424C59
330 DATA 2057485920594F552052454144205241
340 DATA 44494F20534841434B204D4943524F43
350 DATA 4F4D5055544552204E4557534C455454
360 DATA 45522E
```

DECEMBER 1982

Christmas Eve

Chris Petit
Route 3, 135 Davis Drive
Luling, LA 70070

I have added the following lines to Todd Day's joystick routine.

```
9000 SCREEN 1,0
   : H=JOYSTK(0)*4
   : V=JOYSTK(1)*3
   : C=PPOINT(H+1,V+1)
   : PSET(H,V,0)
   : IF X1<>H OR Y1<>V THEN PSET(X1,Y1,C)
   : X1=H
   : Y1=V
9010 P=PEEK(65280)
   : IF P=126 OR P=254 THEN 9020 ELSE 9000
9020 IF X=H AND Y=V THEN 9000 ELSE Z1=Z1+1
9030 IF Z1=2 THEN LPRINT #-2,X;Y,H;V
   : PRINT X;Y,H;V
   : STOP
9040 X=H
   : Y=V
   : GOTO 9000
```

These changes allow a single dot to be displayed and moved around until just the right coordinates are found. To save the first set simply push the fire button, then move the dot to the second location and do the same. Both sets of coordinates will be printed on the screen and on the printer. (Optional—to delete this option, remove "PRINT#-2" in line 9030.) I put in the printer option since some of the lines, when editing, cause the coordinates to scroll off the screen before you can use them. The extra SCREEN command in line 9000 allows you to RUN 9000 again for a quick second set of numbers.

JANUARY 1983

Non-Reset of the Random Number Generator

David A. Levine
Hda. Zotoluca #404
Echegaray, Edo. de Mex. 53300
Mexico

This program doesn't work on my 16K Extended BASIC Color Computer. It just takes out the first number of a sequence, but the others are exactly the same. It seems to work by adding the following to the program:

```
XXX A$=INKEY$
   : ZZ=RND(TIMER)
   : IF A$="" THEN XXX
```

(where XXX is the line number)

You can also enter the following as a direct statement.

```
FOR X=1 TO RND(TIMER):ZZ=RND(TIMER):NEXT
```

If this command takes too much time, you can **BREAK** it and then CLOAD or type in your program.

Day of the Week or Monthly Calendar

David A. Levine
Hda. Zotoluca #404
Echegaray, Edo. de Mex. 53300
Mexico

The Day of the Week or Monthly Calendar program didn't find the correct day of the week on which I was born. I discovered that line 430 must be deleted.

John Barach
Box 99
Sexsmith, AB
T0H 3C0 Canada

After reading George Quellhorst's program, I made the following addition to determine the elusive date of Easter Sunday. Except for the formula in line 550, it is just a simple addition.

```
7 REM EASTER ADDITION BY JOHN BARACH
225 GOSUB 550
550 F=1+Y-((INT(Y/19))*19)
555 G=475
560 IF F=1 AND M=4 GOSUB 760
   : PRINT@ G,"14TH";
570 IF F=2 AND M=3 GOSUB 760
   : PRINT@ G,"3RD";
580 IF F=3 AND M=3 GOSUB 760
   : PRINT@ G,"23RD";
590 IF F=4 AND M=4 GOSUB 760
   : PRINT@ G,"11TH";
600 IF F=5 AND M=3 GOSUB 760
   : PRINT@ G,"31ST";
610 IF F=6 AND M=4 GOSUB 760
   : PRINT@ G,"18TH";
620 IF F=7 AND M=4 GOSUB 760
   : PRINT@ G,"8TH";
630 IF F=8 AND M=3 GOSUB 760
   : PRINT@ G,"28TH";
640 IF F=9 AND M=4 GOSUB 760
   : PRINT@ G,"16TH";
650 IF F=10 AND M=4 GOSUB 760
   : PRINT@ G,"5TH";
660 IF F=11 AND M=3 GOSUB 760
   : PRINT@ G,"25TH";
670 IF F=12 AND M=4 GOSUB 760
   : PRINT@ G,"13TH";
680 IF F=13 AND M=4 GOSUB 760
   : PRINT@ G,"2ND";
690 IF F=14 AND M=3 GOSUB 760
   : PRINT@ G,"22ND";
700 IF F=15 AND M=4 GOSUB 760
   : PRINT@ G,"10TH";
```

```
710 IF F=16 AND M=3 GOSUB 760
   : PRINT@ G,"30TH";
720 IF F=17 AND M=4 GOSUB 760
   : PRINT@ G,"17TH";
730 IF F=18 AND M=4 GOSUB 760
   : PRINT@ G,"7TH";
740 IF F=19 AND M=3 GOSUB 760
   : PRINT@ G,"27TH";
750 RETURN
760 PRINT@ 457,"EASTER--SUN AFTER";
   : RETURN
```

Also, by adding the following line you can switch directly from Day of the Week to a calendar of that month.

```
535 IF P$="K" THEN D=1
   : GOSUB 300
   : GOTO 100
```

FEBRUARY 1983

Variable Swapping

G. Kinun
P. O. Box 2033
Glendora, CA 91740

I would like to submit an improved version of the Variable Swapping Program by Steven Kaiser and a programming tip for other novice programmers like myself. The variable swap routine is mostly used in 'SORT' routines. Using one less variable for the swap would be helpful, and less confusing, especially if data in two or three columns must be swapped in unison. Given 'A' and 'B' as the variables to be swapped, the simpler method is:

```
10 C=A
20 A=B
30 B=C
```

The variables are now swapped. In my programs I keep the swap to one line, and use one line for each column to be simultaneously swapped, i.e.:

```
40 C=A : A=B : B=C
50 C$=A$ : A$=B$ : B$=C$
```

Leon Bryant
1005 Johnson St.
High Point, NC 27262

To conserve memory space make the following changes:

Delete line 30 and change line 40 to A = B.

Labels and Renumbering on the PC-2

Bernard Clark
25151 Brookpark Rd., Apt. 1108
North Olmsted, OH 44070

The renumbering program by Fabio Marzocca works great—when the PC-2 is not supporting additional memory. If the program is executed on a PC-2 equipped with either a 4K or 8K RAM module, the renumbering program will not perform its task and may cause the PC-2 to "act" weird and possibly "lock up" (this happened to me on one such occasion).

The installation of the RAM module will not only change the number of program steps available but will also change the starting address of the program and the ending address of the variable storage area.

The program will work nonetheless with only one change as outlined below. Although this has been tried on a PC-2 with an 8K RAM module, the same approach ought to work with the 4K RAM module as well.

1. If there is any program currently in memory, save it to cassette so there is no program in memory. Type in **(NEW) (O) (ENTER)** to clear memory.
2. Find the program memory starting address by typing **(S) (T) (A) (.) (2) (.) (1) (ENTER)**. Keep a record of this address for future reference.
3. Load the program to be renumbered as well as the renumbering program itself.
4. On line 60000 of the renumbering program, replace the 16581 with the figure from step 2. This will provide the start point for the renumbering program to act upon.

The above steps ought to make it possible to use the renumbering program with any ROM module installed in the PC-2.

Periods to Commas in Data Statements

W. R. Ham, Jr.
3007 Sansom Court
Milton, WV 25541

Please try the following:

```

1 'ROUTINE TO CHANGE PERIODS TO COMMAS IN DATA
  STATEMENTS
2 'ORIGINAL BY EDWARD M. ROBERTS
3 'TRS-80 MICROCOMPUTER NEWS. FEBRUARY 1983
4 'MODIFIED BY BILL HAM AND CHRIS GUNLOCK
5 '2701 1/2 FIFTH AVE., HUNTINGTON, WV 25702
6 'LINE 40, AS SHOWN, IS FOR DISC BASIC
7 'FOR NON-DISC USE
  : "FOR Z=17129 TO VARPTR(Z)"
8 'L10-30 IS TEST. L30-90 IS ROUTINE.
  SAVE BEFORE RUN
10 '.....
20 '.....
30 DATA 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15
40 FOR Z=PEEK(16548)+256*PEEK(16549)
  TO VARPTR(Z)
50 IF PEEK(Z)=136 Y=Z
  : Z=VARPTR(Z)
  : NEXT ELSE NEXT
60 FOR X=Y TO Z
70 IF PEEK(X)=46 POKE X,44
80 NEXT
90 CLS
  : LIST

```

Merge BASIC Programs on the Color Computer

Cyrus Dadgar
1018 Westridge Cir.
Lafayette, IN 47905

I have found that, on 32K computers, this program cuts memory down to a little more than 8K. I use the following line changes to correct this problem:

```

30 CLEAR 200,32364
40 FOR A=32365 TO 32381

```

Also, change the following steps of the procedure:

Step 5. EXEC 32365

Step 8. EXEC 32376

All other steps should be followed according to instructions.

Clifford L. Siebert
7460 Brightwood
STL, MO 63123

Here is a listing for a 32K version of the MERGE program sent in by George Fraser.

```

10 ' "MERGE FOR THE COLOR COMPUTER
20 ' BY G. FRASER MARCH 1982
21 ' MODIFIED FOR 32K BY C.L. SIEBERT FEB.'83

```

```

30 CLEAR 200,32747
40 FOR A=32748 TO 32764
50 READ B : POKE A,B : NEXT
60 DATA 158,25,175,140,12,158,27,48,30
70 DATA 32,3,174,140,3,159,25,57

```

Also, change the following steps of the procedure:

Step 5. EXEC 32748

Step 8. EXEC 32759

MARCH 1983

Renumbering on the Models I and III

Ronald Schneider, M.D., P.C.
1030 President Avenue
Fall River, MA 02720

I thought you might be interested in knowing that under TRSDOS BASIC there is a renumbering program called "NAME" that not only renumbers lines but also renumbers GOTO's and GOSUB's. I didn't discover "NAME" until after typing in the program and trying to find the best way to add a REMARK line and MERGE it with resident programs.

Matthew Belmonte
2120 Marlboro Drive
Alexandria, VA 22304

The renumbering program apparently does not work as stated. I have received many letters and telephone calls, and I have heard of only one Model III user who has been able to get the program to run. He used the following change.

Delete NEXT Y in 63998

He got the wrong line numbers above 255. To remedy this problem, try the following changes:

Line 63996 change A=Y/255 to A=Y/256

Line 63997 change IF Y>255 THEN Y=Y-255 to
IF Y>255 THEN Y=Y-256

I have no guarantee that these changes will work. My apologies to Mr. Dunbrack, whose letter I did not have time to answer.

William P. York
2885 Tanglefoot Lane, #8
Bettendorf, IO 52722

I believe that there is a bug in the Renumbering on the Models I and III program. Attached is my program, which takes a different approach.

```

65000 INPUT "NEWLINE, STARTLINE, INCREMENT";Y,S,I
  : X=16548
  : Z=INT(Y/256)
  : Y=Y-Z*256
65010 FOR A=0 TO S
  : X=PEEK(X)+256*PEEK(X+1)
  : A=PEEK(X+2)+256*PEEK(X+3)
  : NEXT A
65020 FOR A=0 TO 65000
  : POKE X+2,Y
  : POKE X+3,Z
  : Y=Y+I+256*Z
  : Z=INT(Y/256)
  : Y=Y-Z*256
  : X=PEEK(X)+256*PEEK(X+1)
  : A=PEEK(X+2)+256*PEEK(X+3)
  : NEXT A

```

High Res Screen Dump for the CGP-115

David Andrew Palmer
P.O. Box 814
Deep River, Ont
K0J 1P0 Canada

There is an error in your listing of my program. Line

11000 should be changed to read:

```
11000 PRINT#-2,"J1,0"  
: PRINT#-2,"R-1,0"
```

A deeper second density can be made by changing line 10000 to:

```
10000 PRINT#-2,"J2, 0, 0, -2, -2, 0, 0, 2, 2, 0, 0,  
-2, -2, 0, 0, 2"
```

This will cause the plotter to form two boxes, one over the other, for each set pixel.

Additional PMODE3 Color Set

Cameron Parker
11421 30th Pl., SW
Seattle, WA 98146

I found an error in the Additional PMODE3 Color Set program. I have also known about the color set, but lines 10 and 20 should read:

```
10 PMODE3  
: SCREEN 1,1  
20 PMODE4
```

Music Program Pak (26-3151)

Stanley Davis
2882 Instone Ct.
Westlake Village, CA 91361

Several months ago I wrote to you about some features that I found in the MUSIC Program Pak for the Color Computer. (Published in the March issue) However, there was a typographical error. Instead of pressing the letter "I" to insert a note, you should press the letter "I" (for insert).

Also, in order to stop a composition to insert a note, you press this sequence \square HHP. The right and left arrows can be used to advance or backspace one note at a time to the exact spot at which you wish to make corrections.

Plotter Cassette Interface Owner's Manual (26-3605)

Joseph J. Sokolosky
4L Quiet Stream Ct.
Timonium, MD 21093

I found one of the corrections to the documentation to be in error.

The error concerns the insertion of a comma on the PRINT# - 1 and the INPUT# - 1 statements. The correction as listed suggests that the comma should not be included after the - 1 on these statements. This is not the case. If the comma is not included after the - 1, an ERROR 1 (syntax) is returned by the BASIC interpreter. However, if a comma is included after the # on the INPUT# or PRINT# commands, ERROR 21 (variable not in expression) results.

To summarize these findings, the formats for the INPUT# and PRINT# commands are listed below:

```
INPUT#"filename";variable, . . .  
PRINT#"filename";variable, . . .  
INPUT# - 1, "filename";variable . . .  
PRINT# - 1, "filename";variable . . .
```

Engineering Math II (26-3526)

Albert I. Hardy, Jr.
118 Roberts Avenue
Haddonfield, NJ 08033

You state that the last paragraph on page 8 of this manual should be deleted. There is no such paragraph on page 8.

(Editor's Note: The statement should have indicated the last paragraph on page 5.)

APRIL 1983

The Expressive, Expeditious, Exhilarating X-Pad

Bill Mueller
Box 5
Upham, ND 58789

I enjoyed reading about the X-Pad by Paul S. Hoffman. But seeing that the price was a little out of my range, I came up with the following program. When the screen goes to graphics, press **CLEAR** to get back to the text screen.

```
4 CLEAR  
5 CLS  
: INPUT "TYPE IN NUMBER OF TIMES";A  
8 DIM X(A)  
9 DIM Y(A)  
10 FOR B=1 TO A  
15 CLS  
: PRINT "YOU HAVE"A-B"TIMES LEFT"  
20 INPUT "TYPE IN X AXIS";X(B)  
30 INPUT "TYPE IN Y AXIS";Y(B)  
40 NEXT B  
47 DIM B(A)  
50 PMODE 4,1  
60 SCREEN 1,0  
70 PCLS  
80 FOR B=1 TO A  
90 PSET (X(B),Y(B))  
100 NEXT B  
110 A$=INKEY$  
120 IF A$="" THEN 110  
130 IF A$=CHR$(12) THEN 4
```

Sieve of Eratosthenes

Kenneth R. Weiss
3700 Wilshire Boulevard
Los Angeles, CA 90010

The prime number generator program by Dan Jeuch and Steve Hess contains two critical errors. Lines 100 and 510 should read:

```
100 PRINT X;  
: Y=Y+1  
: PM(Y)=X  
: GOTO 90  
510 PRINT PM(P)
```

In addition, the inclusion of the following line will significantly improve the speed of the program:

```
65 IF PM(T)>SQR(X) THEN 100
```

For BASIC interpreters lacking the SQR function, raise X to the .5 power ($x\wedge.5$). On my TRS-80 Model II, use of this line reduced the time to find all primes between 0 and 1800 from 8 minutes, 45 seconds to 4 minutes, 35 seconds, nearly a 50% improvement. The execution speed improvement should increase as the program searches for larger primes.

William F. Dossett
9102 Happy Trail
Austin, TX 78754

The program "PRIME" by Jeuch and Hess will not run as listed. The reason seems to be that the test for primeness in line 70, on which the program is based, quickly runs into a division by zero condition and the program crashes. The remedy is to put a line just before the test, say at line 65 that reads:

```
65 IF PM(T)=0 THEN 80
```

I've modified their program as shown below to take care of division by zero and to eliminate some surplus codes. (Incidentally, line 510 of the original program has an error.)

The test for primes, as written in their program, requires successive division by each integer from 1 to the target integer. Actually only the first half of this sequence of numbers is needed. Hence, some gain in efficiency can result by using Y/2 instead of Y, where Y is the upper limit of the testing loop (line 60 in the original program).

```

10 REM *** <PRIME2> ***
11 REM THE ORIGINAL PRIME PROGRAM BY
12 REM JEUCH AND HESS APPEARED IN TRS-80
13 REM MICROCOMPUTER NEWS, APR. '83, P.38
20 CLS
   : DEFINT A-Z
   : DIM PM(1000)
30 X=9
   : Y=4
40 PM(1)=2
   : PM(2)=3
   : PM(3)=5
   : PM(4)=7
50 INPUT "N=";N
60 PRINT
100 PRINT
   : FOR P=1 TO Y
   : PRINT PM(P);
   : NEXT
110 X=X+1
   : IF X=N THEN END
120 FOR T=1 TO Y/2
130 IF PM(T)=0 THEN 150
140 IF INT(X/PM(T))=X/PM(T) THEN 110
150 NEXT
160 PRINT X;
   : Y=X+1
   : PM(Y)=X
   : GOTO 110

```

I'd like to offer an alternative approach for generating prime numbers. It is noticeably faster than the first program. Note that the only arithmetic operation is addition.

```

10 REM *** <ERASIEVE> ***
11 REM WFD APR. 83
20 DEFINT A-Z
   : DIM F(10000)
30 CLS
   : INPUT "N=";N
40 PRINT
100 FOR I=0 TO N
110 IF F(I)=1 THEN 200
120 P=I+I+3
   : IF P>N THEN 210
130 PRINT P;
140 K=I+P
150 IF K>N THEN 190
160 F(K)=1
170 K=K+P
180 GOTO 150
190 C=C+1
200 NEXT I
210 PRINT
   : PRINT
   : PRINT C;" PRIMES"

```

This program is a true sieve. Primes are generated directly by the sieve process. No test for primeness is made; none is necessary. I'll leave it to the interested reader to discern the generating principle. Analyze lines 120, 140 and 170. Look especially carefully at the embedded loop formed by lines 150-180. Hint: F() is a flag; when its value is 1 there is a hole in the sieve. Take some small value for N, say N = 15, calculate and trace each line of the program manually, but do it slowly and carefully. I think you will find it illuminating and interesting.

MAY 1983

Disk Editor Disassembler

Ashok Basargekar
1423 N. Cleveland St.
Orange, CA 92667

Please note that there is a slight error in the Disk Editor Disassembler program. The statement IF PEEK(PI)=18 in line number 51 should be corrected to:

```
IF PEEK(PI)=17
```

PC-2 INPUT Statements

Lloyd E. Scott
1780 Miller Rd.
Castleton, NY 12033

I would like to propose an additional syntax not given in the PC-2 Input Statements article. I think this statement will put a little extra "class" in PC-2 programs.

```
Pause "INPUTX = ";:BEEP 1:INPUT X:CLS
```

This syntax prints the input prompt, gives a beep and displays the "?" when ready to accept the data input.

```
eg., INPUT X = (BEEP)?
```

The ? gets lost with the syntax input "INPUT X";X.

Disk Directory

William L. Harris
P.O. Box 143
Bourbonnais, IL 60914

Here is my Directory-To-Printer program, which is somewhat shorter than Berry Rinaldo's Disk Directory program, being only one line long. The program is just as effective in the immediate mode as it is as a program.

```

10 POKE 111,254
   : DIR
   : PRINT#-2
   : PRINT#-2,FREE(0)"GRANULES FREE"

```

The only limitation I am aware of is that the POKE and DIR must be combined in one statement for it to work.

JUNE 1983

USA Flag for the Color Computer and MC-10

Kent Jakway
4861-CR7
Garrett, Ind. 46738

I have added a repeat feature (press "R") and a music start feature (press "M").

```

122 A$=INKEY$
124 IF A$="M" THEN 130
126 GOTO 122
172 A$=INKEY$
174 IF A$="R" THEN 130
176 GOTO 172

```

Document Listing for the Model II

Dave Zimmerman
1115 Byron Avenue
Elizabeth, NH 07208

I saw two or three flaws in this program. One concerns an ?RG error (RETURN w/o GOSUB) in line 1152 that you can get if you get an error other than 52 or 53 on line 20. To remedy this, change these lines to read as follows:

```

20 ON ERROR GOTO 1140
   : ML=1
   : CLOSE
   : OPEN "D",2,"PROGLIST/PRT"
   : ON ERROR GOTO 0

```

```

: CLOSE
: ML=0
1140 IF ERR=54 THEN RESUME NEXT ELSE GOSUB 1150
: IF ML THEN RESUME NEXT ELSE RESUME 1100

```

The other error concerns the indentation inside of a FOR-NEXT loop. The indentation for the loop spaces 12 positions and at the end of the loop only brings it back five. Another cause of strange indentation in a printed document is that the program doesn't recognize a multiple NEXT, as in NEXT M,L. To fix the 5/12 space problem, change line 4188 to read:

```
4188 IF C=129 THEN NF=NF+5
```

Bargraph

Norman Moulton, Jr.
Eagle Hollow Rd.
Corinth, VT 05039

I was thoroughly impressed with the graphics display of the program by Dennis L. Hargens. However, I would like to suggest a better way of entering the vertical axis label. This routine allows you to type it in all at once instead of entering it one letter at a time and then entering 'XX'.

First type: DEL 70-150

Then replace it with this:

```

70 PRINT@ 224,"WHAT ARE THE UNITS OF THE GRAPH?";
80 INPUT UN$
90 FOR Y=1 TO LEN(UN$)
100 UN$(Y)=MID$(UN$,Y,1)
110 NEXT Y

```

Grid for the CGP-115

Ervin A. Madera
843 Santa Dorotea Circle
Rohnert Park, CA 94928

I tried the program as indicated (change all the LPRINTs to PRINT#-2,) on my BASIC 16K TRS Color Computer and the needle fairly well flew off the machine with the program as written. The following line changes achieved what I thought the program was designed to do.

(Editor's Note: We are listing the corrected lines with LPRINTs in order to standardize with the original program. To run on the Color Computer, all LPRINTs, including those listed below, should be changed to PRINT#-2,.)

```

40 PRINT#-2,"J440,0"
50 PRINT#-2,"R-440,-24"
100 PRINT#-2,"R425,0"
: PRINT#-2,"J0,-240"
: PRINT#-2,"H"
101 PRINT#-2,"J0,10":PRINT#-2,"S0"
105 PRINT#-2,CHR$(17)

```

Variable Swapping Routine

Frank Romanelli
711 Lincon Ave.
Ridgefield Park, NJ 07660

The Variable Swapping Routine by Dennis Lee Bieber will work on the Extended BASIC Color Computer if the function is changed to a single variable. Change the following lines to read:

```

10 DEF FNX(A)=(A AND NOT B) OR (NOT A AND B)
40 A=FNX(A)
: B=FNX(A)
: A=FNX(A)

```

Walt Nolan
283 38th St. Dr. SE #7
Cedar Rapids, IA 52403

Here is a simpler routine which swaps two variables with-

out using temporaries, or having to first define a function.

```

10 INPUT A,B
20 A=A+B
: B=A-B
: A=A-B
30 PRINT A,B

```

A Tribute to Columbia

Arnold E. van Beverhoudt, Jr.
P. O. Box 56
St. Thomas, V.I. 00801

I was pleased to see my program, "A Tribute to Columbia," printed in the June 1983 issue. However, I did notice that one line in the program listing was incomplete. Please inform your readers that line 40070 should read as follows;

```

40070 IF POINT(X,Y)=-1 THEN LPRINT CHR$(18);
CHR$(28);CHR$(3);CHR$(128); ELSE LPRINT
CHR$(18); CHR$(28);CHR$(3);CHR$(255);

```

This correction should solve any problems in getting a screen print of the graphic pictures of the Space Shuttle.

Danny Sherman
8112 S. Greenwood Ave.
Columbia, MO 62501

The program "A Tribute to Columbia" by Arnold E. van Beverhoudt, Jr. was very nicely done, but the printed version contained two errors. The most important mistake was in line 240. The line should be changed to read:

```

240 H$(14)=CHR$(143) + STRING$(3,128) + CHR$(143) +
STRING$(2,140) + ...etc.

```

The last segment read STRING\$(3,140) in the printed version. The second error appeared in line 40070. The parenthesis at the end of this line was not included. Once these changes were made the program ran perfectly.

JULY 1983

Record Chess Play

Dieter W. Koch
2883 Concord Blvd.
Concord, CA 94519

Thank you for publishing my "Record Chess Play" program. Unfortunately, five program lines have been reproduced with errors in them and could cause frustration in someone trying to make the program work properly.

In line 13110 and 13240: E = 15

Line 13210 must include: S=3

In line 13270: E = 12

In line 13670: B = 6

Stephen T. Whitney
Whitney Tobacconist
Mill Pond Shopping Center
Cos Cob, CT 06807

There are several errors in Record Chess Play by Dieter W. Koch. The PC-2 has some unique features and the program is a complex one so even an experienced user would have trouble. I found the following corrections necessary.

Change lines 13110 and 13240 from E + 15 to E = 15

Change line 17320 to GOSUB 37410

The routine for printing a King only prints cross-bar to right. On line 13670 try: R = 15, B = 6, E = 12, S = 6.

Invasion

Fred Truncale
10 Vista Way
Springfield, NJ 07081

There was one small problem with Invasion. Every time you killed a ship in the top row, you got an illegal function call in 700. I fixed this by changing line 700 to read:

```
700 IF X<128+1 THEN GOTO 1000
      : PRINT@X+1-128,""
```

And by adding these lines to the program:

```
1000 PRINT@ X,""
1010 GOTO 710
```

Robert Sisco
127 S. Travis St.
Lindenhurst, NY 11757

Here are some changes for the Invasion program for the Model III, using the special character set which makes the game better. The changes and additions are listed below.

Changes:

```
320 C$=CHR$(214)
340 A$=CHR$(234)
350 B$=CHR$(255)
450 IF X>Y THEN PRINT@Y,CHR$(238) ELSE 510
500 INPUT "ENTER TO PLAY AGAIN ";M
      : PRINT CHR$(21)
      : GOTO 110
680 PRINT@ V-64+1,C$
      : PRINT@V-64+1," "
690 IF V=X THEN PRINT@X,CHR$(238) ELSE 730
```

Add the following lines.

```
90 PRINT CHR$(21)
115 PRINT CHR$(21)
455 FOR I=1 TO 8:NEXT I
      : PRINT@Y,CHR$(239)
      : FOR I=1 TO 8
      : NEXT I
      : PRINT@Y,CHR$(240)
      : FOR I=1 TO 8
      : NEXT I
      : PRINT@Y," "
695 FOR I=1 TO 8
      : NEXT I
      : PRINT@X,CHR$(239)
      : FOR I=1 TO 8
      : NEXT I
      : PRINT@X,CHR$(240)
      : FOR I=1 TO 8
      : NEXT I
      : PRINT@X," "
```

Also, to get the program to run without an FC Error in line 700, change line 700 as follows:

```
700 PRINT@X+128-1,""
```

I hope these changes will make the game a little more interesting. It won't leave the screen messy and will make both the alien and your ship look like they are exploding.

Bar, Line and Scatter Graphs for the Model III

A. F. Bell
680 Kirkwood Dr.
Sudbury, Ont P3E 1X3
Canada

This program is written in BASICG for a TRS-80 Model III with a High Resolution Graphics Board in place. With it a user can get hard copies of Bar, Line, or Scatter graphs. Up to three sets of data can be graphed.

The program is broken into blocks :

Lines 100-180 initialize the program and allow choice of graph. XL and XW set the left side of the graph (the place of the Y axis) and the width of the graph respectively. YT and YH set the top of the graph and its height. These are the only two variables which need to be changed if graphs of different size or location are preferred. XG and YG set the gradations on the X and Y axes, and XD and YD are the resulting distances between the gradations.

Lines 200-290 allow input of the parameters for the graph. Lines 200-210 allow input of titles and center them. TG is the flag for the type of graph, and if TG = 1 (signifying a bar graph) then lines 225 and 230 set the X axis gradations. In line 280 and 285, XF! and YF! are the factors with which the coordinates are calculated.

After all that fuss, lines 300-345 suffice to allow input of the raw values of the coordinates. Once this is finished in line 340, the screen is cleared and converted to graphics mode.

Lines 400-490 draw the axes, their titles, and their gradations. If the graph is a bar graph using names along the X axis, then line 450 prints out the names. If necessary lines 475-490 identify the point, line, or graphic block belonging to each set of data using subroutines 2100, 2200, or 2300.

Lines 500-525 together with subroutines 1100, 1200, and 1300 are enough to add the points, lines, or blocks for each set of data. Line 515 is required to provide a start for the lines.

Lines 600-620 allow a choice between ending the program, repeating the options, or printing out the graph.

```
100 'BLSGRAPH/GRA 10/07/83
105 'Written in Model 3 Disk BASICG using about 6.5K by
110 'Dr AFJ Bell, 680 Kirkwood Dr,Sudbury,Ontario P3E 1X3
115 CLS
      : PRINT@ 348,"Initializing...";
120 CLEAR 500
      : DEFINT A-Z
      : SCREEN 1
      : CLR
125 XL=60
      : XW=560
      : XR=XL+XW
      : XG=10
      : XD=XW/XG
130 YT=10
      : YH=200
      : YB=YT+YH
      : YG=10
      : YD=YH/YG
135 DIM PT!(3,50,2),MV(2),C$(2),P$(3),L(3),
      T$(3),N$(3),NP(3),XN$(31)
140 ID$="Invalid data"
      : C$(1)=" X ="
      : C$(2)=" Y ="
145 P$(1)="+"
      : P$(2)="-"
      : P$(3)="*"
```

```

150 L(1)=&HFFFF
   : L(2)=&H8888
   : L(3)=3855
155 T$(1)="D"
   : T$(2)=CHR$(&H55) +CHR$(&HAA)
   : T$(3)="Z"
160 PRINT@ 344,"Graphs available";
   : PRINT@ 472,"A Bar Graph";
165 PRINT@ 536,"B Line Graph";
   : PRINT@ 600,"C Scatter Graph";
170 PRINT@ 720,"What is your choice (A, B, or C) ?";
   : GOSUB 1000
175 CLS
   : TG=ASC(A$)-64
180 :
200 INPUT"MAIN TITLE";MT$
   : LMT=LEN(MT$)
   : IMT=(XW-LMT*10)/2
205 INPUT"X Axis Title";XT$
   : LXT=LEN(XT$)
   : IXT=(XW-LXT*10)/2
210 INPUT"Y Axis Title";YT$
   : LYT=LEN(YT$)
   : IYT=(YH-LYT*8)/2
215 INPUT "How many sets of data (1-3)";NS
   : IH=640/(NS+1)
220 IF NS<1 OR NS>3 THEN PRINT ID$
   : GOTO 215
225 IF TG>1 GOTO 255 ELSE INPUT
   "How many bars in each set ";XG
   : IF NS*XG>30 THEN PRINT ID$
   : GOTO 215
   : ELSE XD=XW/XG
   : BW=XD/(NS+.5)
   : FOR L=1 TO NS
   : NP(L)=XG
   : NEXT
230 PRINT "X axis divided by number(A) or name(B)?"
   : GOSUB 1000
   : IF A$="C" GOTO 230 ELSE XA=ASC(A$)-64
   : CLS
   : IF XA=2 THEN LN=(XW/10)/XG-1 ELSE GOTO 255
235 FOR L=1 TO XG
240 PRINT "Name of group ";L; (" (<;LN;"chars");
   : INPUT XN$(L)
245 IF LEN(XN$(L))>LN PRINT ID$
   : GOTO 240
250 NEXT
   : PRINT
255 FOR L=1 TO NS
260 PRINT "Name of set";L;
   : INPUT N$(L)
   : IF TG=1 GOTO 275
265 INPUT"How many points in set (1-50) ";NP(L)
270 IF NP(L)<1 OR NP(L)>50 THEN PRINT ID$
   : GOTO 265
275 NEXT
   : PRINT
   : IF XA=2 MV(1)=100
   : XF=XW/MV(1)
   : GOTO 285
280 INPUT"Maximum value of X ";MV(1)
   : XF=XW/MV(1)
285 INPUT"Maximum value of Y ";MV(2)
   : YF=YH/MV(2)
   : PRINT
290 :
300 FOR J=1 TO NS 'Inputs coordinates
305 PRINT "Please enter coordinates for set ";J
310 FOR K=1 TO NP(J)
315 FOR L=1 TO 2
320 IF L=1 AND TG=1 PT!(J,K,1)=(K-1)*MV(1)/XG
   : GOTO 335
325 PRINT "Value for point";K;C$(L);
   : INPUT V!
330 IF V!>=0 AND V!<=MV(L) THEN PT!(J,K,L)=V!
   : ELSE PRINT ID$
   : GOTO 325
335 NEXT L
   : PRINT
340 NEXT K,J
   : CLS
   : SCREEN 0
345 :
400 GLOCATE(XL+IMT,5),0
   : PRINT#-3,MT$ 'Prints titles & axes
405 LINE (XL,YT)-(XL,YB)
   : LINE -(XR,YB)
410 FOR L=YG TO 0 STEP-1 'Numbers Y Axis
415 LINE(XL,YT+L*YD)-(XL+2,YT+L*YD)
420 NY=(YG-L)*MV(2)/YG
425 IF NY>99 THEN PX=50 ELSE IF NY>9
   THEN PX=40 ELSE PX=30
430 GLOCATE(XL-PX,L*YD+8),0
   : PRINT#-3,NY
435 NEXT
   : GLOCATE(0,YB-IYT),3
   : PRINT#-3,YT$
440 FOR L=0 TO XG 'Numbers X Axis
445 LINE(XL+L*XD,YB)-(XL+L*XD,YB-2)
450 IF XA=2GLOCATE(XL+L*XD,YB+3),0
   : PRINT#-3,XN$(L+1)
   : GOTO 470
455 NX=L*MV(1)/XG
460 IF NX>99 THEN PX=40 ELSE IF NX>9
   THEN PX=45 ELSE PX=50
465 GLOCATE(PX+L*XD,YB+3),0
   : PRINT#-3,NX
470 NEXT
   : GLOCATE(XL+IXT,YB+12),0
   : PRINT#-3,XT$
   : IF NS=1 GOTO 500
475 FOR L=1 TO NS 'Identifies data sets
480 XH=L*IH
   : GLOCATE(XH-40,YB+21),0
   : PRINT#-3,N$(L)
485 ON TG GOSUB 2100 ,2200 ,2300
490 NEXT
   : XA=0
495 :
500 FOR J=1 TO NS 'Plots points
505 FOR K=1 TO NP(J)
510 X=XL+PT!(J,K,1)*XF!
   : Y=YB-PT!(J,K,2)*YF!
515 IF TG=2 AND K=1 THEN CIRCLE(X,Y),1,0
   : GOTO 525
520 ON TG GOSUB 1100 ,1200 ,1300
525 NEXT K,J
530 :
600 A$=INKEY$
605 IF A$="" OR (A$<"E" AND A$<"P" AND A$<"R") GOTO 600
610 IF A$="E" THEN CLS
   : END
615 IF A$="R" THEN CLR
   : CLS
   : GOTO 160
620 LPRINT CHR$(27) CHR$(14)
   : CMD"I","GPRINT"
625 :
1000 A$=INKEY$
   : IF A$="" OR A$<"A" OR A$>"C" GOTO 1000
1010 PRINT A$
   : RETURN
1020 :
1100 X=X+(J-1)*BW
   : XE=X+BW 'Draws bars
1110 LINE(X,Y)-(XE,YB),1,B
1120 IF YB-Y<1 THEN RETURN
1130 PAINT(X+1,Y+1),T$(J),1
   : RETURN
1140 :
1200 LINE-(X,Y),1,,L(J)
   : RETURN 'Draws lines
1210 :
1300 GLOCATE(X,Y),0
   : PRINT#-3,P$(J)
   : RETURN 'Plots points
1310 :
2100 LINE(XH+40,YB+20)-(XH+80,YB+28),1,B
2110 PAINT(XH+41,YB+21),T$(L),1
   : RETURN
2120 :
2200 LINE(XH+40,YB+23)-(XH+80,YB+23),1,,L(L)
   : RETURN
2210 :
2300 GLOCATE(XH+40,YB+21),0
   : PRINT#-3,P$(L)
   : RETURN

```

The CGP-220 Ink Jet Printer

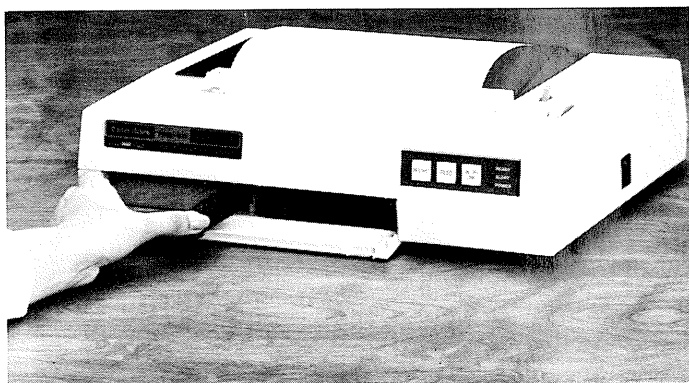
By Linda Miller

The first time that I saw the terrific color printouts created by an ink jet printer, I wondered when an affordable Radio Shack version would be available for TRS-80 computers. Such a printer now exists in the CGP-220 Ink Jet Printer (Catalog number 26-1268, suggested retail price \$699.00).



THE CGP-220 IS COLORFUL!

There are seven colors available (red, yellow, blue, green, cyan, magenta, and black). Ink is neatly contained in two up-to-four million character ink cartridges, one black and the other tricolor. There's no mess in handling the cartridges. They're easily removed and installed.



The CGP-220 uses the YMC method to produce different color combinations. With this method, the colors Yellow, Magenta, and Cyan are mixed in different combinations to produce other colors.

Although the Color Computer uses a RGB method for mixing colors, it is possible to print graphic displays from a color monitor because the CGP-220 is equipped with built-in software that allows conversion from the RGB method to the YMC method.

TECHNOLOGY, SPEED AND PAPER

The technologically advanced CGP-220 uses a drop-on-demand ink jet system and a piezoelectric-type print head. It has a one line print buffer, a printing speed of thirty-seven characters per second, bi-directional line scanning, and uses either plain single sheet paper or an 8 1/2" by 135' paper roll.

SERIAL AND PARALLEL INTERFACING

The CGP-220 has both a serial and an eight bit parallel interface. It can be connected to a standard TRS-80 parallel port or a Color Computer serial port.

MORE FEATURES

The CGP-220 is a very versatile printer with many features including variable line feeds, elongated characters, dot pitch selection, and print head positioning.

With variable line feed capability, the distance of a line feed can be changed from 1/6" (normal) to 1/8" (three quarter). The following program and sample printout illustrate the difference between the normal and three quarter line feeds. (For TRS-80s other than the Color Computer, change the "PRINT#-2," to "LPRINT" in all the program lines.) In line 30 CHR\$(27); CHR\$(56) changes the line feed from 1/6" to 1/8". CHR\$(27); CHR\$(54) in line 50 resets the line feed back to normal again.

```
10 PRINT#-2, "LINE FEED PITCH TEST"
20 PRINT#-2, "THIS IS NORMAL";
30 PRINT#-2, CHR$(27);CHR$(56)
40 PRINT#-2, "THIS IS 1/8 INCH"
50 PRINT#-2, CHR$(27);CHR$(54)
```

```
LINE FEED PITCH TEST
THIS IS NORMAL
THIS IS 1/8 INCH
```

The distance between the first line and the second line of the printout is the result of the 1/6" line feed while the distance between the second and third line is the result of the 1/8" line feed.

PRINT#-2, CHR\$(27);CHR\$(14) switches to printing elongated characters.

```
ELONGATED elongated
```

PRINT#-2, CHR\$(27);CHR\$(15) switches back to normal characters.

Characters are usually set in a ratio of 4:3 meaning the characters are slightly taller (4) than they are wide (3). It is possible with the CGP-220 to print characters in a 1:1 ratio where they are as wide as they are tall. PRINT#-2, CHR\$(27);CHR\$(78) or PRINT#-2, CHR\$(27);"N" changes the ratio to 1:1. "N" is the alternate instruction for CHR\$(78).

```
THIS IS A 1:1 RATIO
THIS IS THE 4:3 RATIO
```


PRINT#-2, CHR\$(27);CHR\$(80); or PRINT#-2, CHR\$(27);"P" switches back to the 4:3 ratio setting.

The CGP-220 print head can be positioned in any of 640 dot columns in each of the three print modes (Text, Bit Image, and Color Scan) with a BASIC statement like the following.

```
PRINT#-2,CHR$(27);CHR$(16);CHR$(n1);CHR$(n2)
```

CHR\$(27);CHR\$(16) first tells the printer to get ready to position the print head. CHR\$(n1) will indicate a range on the line. Each line is divided into three ranges. The first range is 0, 1 is the second range and 2 is the third range. The "n2" in CHR\$(n2) indicates the dot position within range 0,1, or 2.

Table 1 indicates the possible values of n1 and n2.

| If you wish to specify dot column | n1 (range) must be | n2 must be |
|-----------------------------------|--------------------|------------|
| 0 - 255 | 0 | 0 - 255 |
| 256 - 511 | 1 | 0 - 255 |
| 512 - 640 | 2 | 0 - 129 |

Table 1.

PRINTING CHARACTERS MULTIPLE TIMES

A character can be printed repeatedly up to 256 times. PRINT#-2, CHR\$(28);CHR\$(n1);CHR\$(n2) tells the printer that the character whose ASCII code is "n2" will be printed "n1" times. The character to be printed enclosed in quotes could be substituted for CHR\$(n2) as in the following example where the asterisk (*) would be printed five times.

```
PRINT#-2,CHR$(28);CHR$(5);"*"
```

COLOR SELECTION

Colors can be selected from the seven available colors and white which is non-printing.

```
PRINT#-2,CHR$(27);CHR$(84);CHR$(n)
```

CHR\$(27);CHR\$(84); says "I want a new color." "n" is the number for the selected color.

| If you want to use color | n must be |
|--------------------------|-----------|
| Black | 48 |
| Red | 49 |
| Green | 50 |
| Yellow | 51 |
| Blue | 52 |
| Magenta | 53 |
| Violet | 54 |
| White (not printing) | 55 |

Table 2.

THREE PRINTER MODES

The CGP-220 has three separate printer modes: Text, Bit Image and Color Scan.

The text mode is the default mode of the CGP-220 and is used for printing documents, letters, etc. The characters are formed in a 5 x 7 dot matrix and may be printed in any of the seven available colors. The CGP-220 has a full 128 character ASCII character set including upper and lower case, numbers, punctuation, European symbols, and other special characters.

The graphic capabilities of the CGP-220 are most impressive in the Bit Image and Color Scan modes. Both have bit addressable graphics with up to 640 dots per line of high resolution printing. In these modes all positioning and action of the print head is done through software. There are no pre-defined characters.

BIT IMAGE MODE

Up to 640 dots across by 7 dots **vertically** can be addressed in each line in the Bit Image mode. That's 640 dot columns and seven vertical dots in each dot column (640 x 7 = 4480). That's a lot of dots in a single line!

Consider this short BASIC program.

```
10 PRINT#-2, CHR$(18)
20 PRINT#-2, CHR$(27);CHR$(16);CHR$(0);CHR$(100);CHR$(255)
```

CHR\$(18) in line 10 puts the printer in the Bit Image mode. CHR\$(27);CHR\$(16) tells the printer to get ready to position the print head. CHR\$(0) indicates the first range (0) of the line, and CHR\$(100) indicates the one hundredth column position in range 0. CHR\$(255) causes all seven dots in the dot column to be printed. How did CHR\$(255) do that? Consider the chart below.

| Dot position in column | Dot # value | Number used to print the dot |
|------------------------|-------------|------------------------------|
| 1 | 1 | 129 |
| 2 | 2 | 130 |
| 3 | 4 | 132 |
| 4 | 8 | 136 |
| 5 | 16 | 144 |
| 6 | 32 | 160 |
| 7 | 64 | 192 |

Table 3.

To print a dot at position 5 in the column, substitute CHR\$(144) for CHR\$(255) in line 10. From the chart it's easy to see how each single dot can be printed simply by using its number, but the question of how to print more than one dot in a single column still remains. This is where the numbers in the Dot# value column come in handy. To print dots 1, 5, 6, and 7 in the column, the dot # values of the dots to be printed are added together (1 + 16 + 32 + 64 = 113). The result (113) is added to the number 128 which results in 241. By substituting 241 for 255 in line 20, dots 1, 5, 6, and 7 are printed. CHR\$(255) caused all seven of the dots in the vertical column to be printed because 255 is the sum of all the Dot # values (1 + 2 + 4 + 8 + 16 + 32 + 64 = 127) plus 128 (127 + 128 = 255).

Elongation and dot pitch selection as well as color are available in the Bit Image Mode.

THE COLOR SCAN MODE

The Color Scan Mode allows for greater manipulation of colors and even greater detail in graphics creation. Unlike the Bit Image mode which addressed vertical columns of seven dots, the Color Scan mode addresses **horizontal** rows of eight dots with up to 80 dot rows for 640 dots across the line.

PRINT#-2,CHR\$(27);CHR\$(67) puts the CGP-220 in the Color Scan mode. Next CHR\$(n) is sent to the printer to indicate an "n" number of eight dot rows to be printed out of a possible eighty. The dots in each eight dot row are numbered

in dot positions from right to left with the dot # values shown in the chart below.

| | | | | | | | | |
|--------------|-----|----|----|----|---|---|---|---|
| Dot column # | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Dot # value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Table 4.

In a manner similar to the Bit_Image mode, specific dots or combinations of dots are printed by their corresponding number or sum of corresponding numbers. To print the dots in columns 7, 5, 4, 2, and 1, the sum arrived at would be 64 + 16 + 8 + 2 + 1 or 91. Unlike the Bit_Image mode, no other number is added to the sum.



All the colors (black, red, green, yellow, blue, magenta, violet, and white) are available in the Color Scan mode. Each dot in every eight dot row may be printed in a specific color.

After entering the Color Scan mode (line 10 below) the number of dot rows to be printed is sent to the printer (line 20), followed by the red data + green data + blue data (lines 30-50).

```

10 PRINT #2,CHR$(27);CHR$(67); 'ENTER COLOR SCAN MODE
20 PRINT#-2,CHR$(3); 'NUMBER OF ROWS = 3
30 PRINT#-2,CHR$(128); 'RED DATA - COL 8
   CHR$(128);CHR$(128); 'ONLY IN EACH OF THE
   'THREE ROWS
40 PRINT#-2,CHR$(16);CHR$(16); 'GREEN DATA - COL 5
   CHR$(16); 'ONLY IN EACH OF THE
   'THREE ROWS
50 PRINT#-2,CHR$(1);CHR$(1); 'BLUE DATA - COL 1
   CHR$(1); 'ONLY IN EACH OF THE
   'THREE ROWS

```

Line 30 contains the red data. The three CHR\$(128)s indicate that the left most dots (column 8) will be printed in red

ink in each of the three rows of eight dots. The three CHR\$(16)s in line 40 mean that fourth dot from the left (column 4) in each of the three rows of eight dots will be printed in green ink. The three CHR\$(1)s in line 50 indicate that the right most dot (column 1) in each of the three rows will be printed in blue ink. Where no color is indicated, the dots will be printed in black ink. Dots that have color data indicated for all three of the colors will be non printed or white.

The colors yellow, magenta, and violet are formed by mixing colors.

| |
|-----------------------|
| Red + Green = Yellow |
| Red + Blue = Magenta |
| Green + Blue = Violet |

Table 5.

Change lines 30, 40, and 50 in the program above to read like the ones below.

```

30 PRINT#-2,CHR$(195);CHR$(195);CHR$(195);
40 PRINT#-2,CHR$(51);CHR$(51);CHR$(51);
50 PRINT#-2,CHR$(60);CHR$(60);CHR$(60);

```

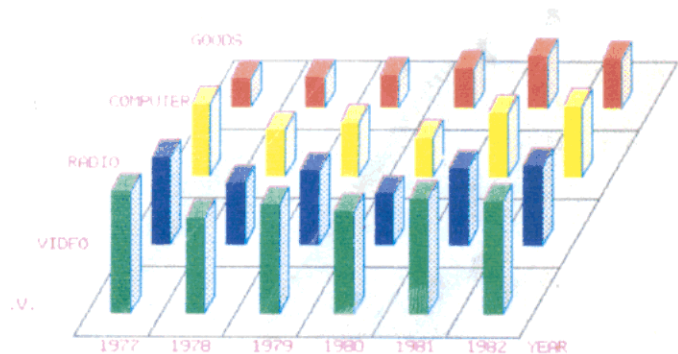
Table 6 graphically illustrates the dots to be printed and the color data with which they will be printed.

| Column # | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Dot Total |
|------------|-----|----|----|----|---|---|---|---|-----------|
| Red data | 128 | 64 | | | | | 2 | 1 | 195 |
| Green data | | | 32 | 16 | | | 2 | 1 | 51 |
| Blue Data | | | 32 | 16 | 8 | 4 | | | 60 |

Table 6.

Dots 128 and 64 are only used for red data so they will print red. Dots 32 and 16 are used for green and blue data so they will print violet. Dots 8 and 4 are used for blue data only so they will print blue, and dots 2 and 1 are used for both red and green data so they will print yellow.

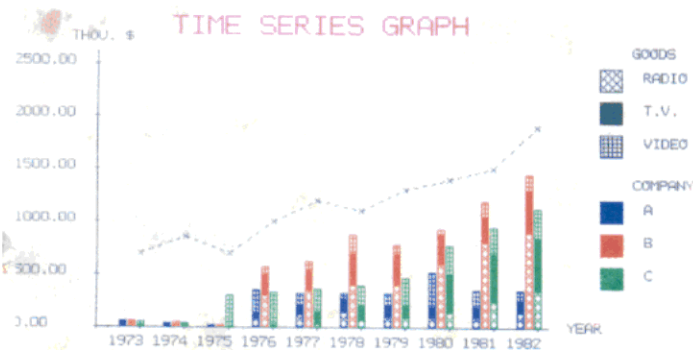
ANNUAL SALES RESULTS



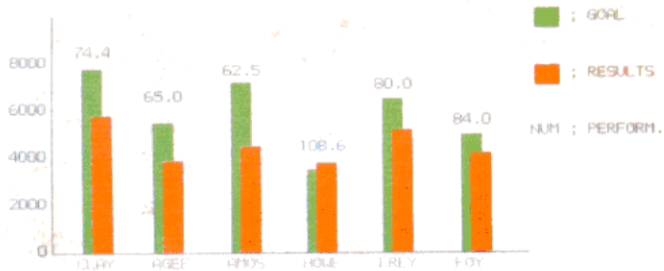
Graphic pictures can be created in meticulous detail using the Color Scan mode.

COLOR COMPUTER SCREEN DUMP

With a 16K Color Computer, there is a faster (and simpler) way to get a color printout. The Color Computer Screen Dump program can be used to reproduce the four color graphic printouts which appear on the Color Computer screen on the CGP-220. At this writing the Color Computer Screen Dump program is not available due to changes that are being made in the program. It will be available at some future date.



INDIVIDUAL SALES RESULTS



| | CLAY | AGEE | RHOS | HOME | FREY | FOY | UNIT ; \$ |
|---------|------|------|------|------|------|------|-----------|
| GOAL | 7800 | 6000 | 7200 | 3500 | 6500 | 5000 | |
| RESULTS | 5800 | 3900 | 4500 | 3300 | 5200 | 4200 | |

With Micro Painter (26-3077) pictures can be created on the screen and dumped to tape. After loading the screen dump program into memory using the instructions in the manual, the picture tape created using Micro Painter can be loaded with a CLOADM and checked using the BASIC program below.

```

10 PCLEAR 0
20 PMODE 4:1
30 SCREEN 1:1
40 CLOADM
50 GOTO 50
    
```

Once the picture has been verified, **(BREAK)** the program, press the **(-)**, and **(G)**, **(W)** or **(B)** (for a green, white, or black background color). The screen will begin dumping to the CGP-220.

Radio Shack®

TRS-80 Microcomputer News
 P.O. Box 2910
 Fort Worth, Texas 76113-2910

**BULK RATE
 U.S. POSTAGE
 PAID
 Radio Shack
 A Div. of Tandy Corp.**

ADDRESS CHANGE

Remove from list

Change as shown

Please detach address label and mail to address shown above

THIS ONE'S GOT IT ALL

The CGP-220 has so much - a large ASCII character set, three printer modes including tremendous graphics capabilities, color, and many other features. All totaled up the CGP-220 is a unique, affordable, and very useful printer.

