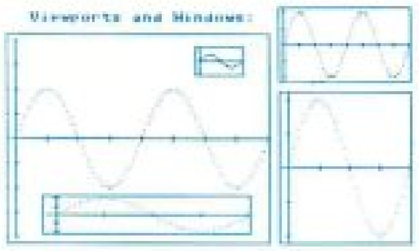
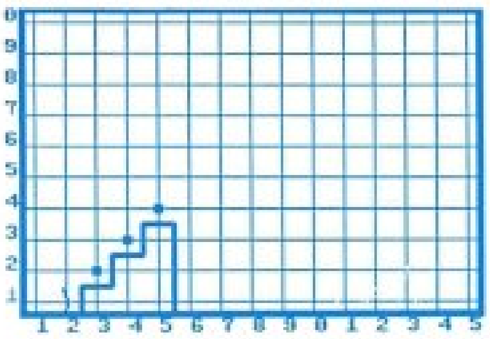
COMAL TODAY 9



From program 'VIEWPORT' on COMAL TODAY Disk #9



Learn COMAL 2.0 With Rod the Roadman
An Article and program by
Borge Christensen

COMAL TODAY
COMAL USERS GROUP, U.S.A., LIMITED
6041 Monona Drive, Madison, WI 53716

BULK RATE
U.S.POSTAGE
PAID
MADISON, WI
PERMIT 2981

IF YOUR LABEL SAYS
LAST ISSUE: 9
YOU MUST RENEW NOW.
USE CARD INSIDE

COMAL TODAY ISSUE 9 September 30, 1985

EDITORS

Len Lindsay
Colin Thompson

ASSISTANTS

Debra Junk
David Stidolph
Maria Lindsay
Denise Bernstein
Geoffrey Turney

PHOTOS / ART

Wayne Schmidt
Bill Nissley

CONTRIBUTORS

Phil Bacon
Phyrne Bacon
Doug Bittinger
Ray Carter
Borge Christensen
Glen Colbert
Bob Hoerter
Daniel Horowitz
Tom Kuiper
Mike Lawrence
Len Lindsay
Ian McPhedron
Michael Mercurio
Franco Pavase
Raymond Saller
Nicholas Seachord
David Skinner
David Stidolph
David Tamkin
Colin Thompson
Craig VanDegrift
Birrell Walsh

PUBLISHER

COMAL Users Group,
USA, Limited
6041 Monona Drive
Madison, WI 53716
608-222-4432

From the Editor's Disk

Welcome to COMAL. Welcome to COMAL TODAY. It is my privilege to be the editor of this newsletter. I also was the founding editor of the PET GAZETTE in 1978. It started its first issue as one sheet of paper folded in half. The PET GAZETTE is still around, only now it's called COMPUTE! The same bright future is in store for COMAL TODAY, though we are starting with a bit more than one sheet of paper. I hope you will help us grow. Everyone has something to share, even just a one line piece of advice. When a tip or note about COMAL comes to mind, jot it down and send it off to COMAL TODAY.

That is how COMAL TODAY issue #1 started. The first four issues were packed full of great COMAL information. Unfortunately they have been out of print for nearly a year. But we had so many requests for copies that we decided to reprint them. Not just parts of them. Not just the best. **Everything**! Even spiral bound! This is a limited quantity reprint, so get your copy now. Only $10.95 plus $2 shipping to subscribers. The other back issues are also available to subscribers for only $2.95 plus $1 shipping each. You now can have a complete collection of COMAL TODAY.

You will notice a few changes in this issue of COMAL TODAY. We now have a laser printer connected to our C64. It printed nearly everything in this issue. Take a peek at the list of best selling COMAL books (page 23) for a preview of its typeset quality. We also are adding to our staff. On the next page you will find none other than **Colin Thompson**. He left the ideal weather of California and moved to Madison to help us. His first week here we had a terrific thunderstorm and tornado warnings. Welcome Colin!

The father of COMAL, **Borge Christensen**, just finished his first visit to America. All who met him agree: he brings COMAL to life. I can almost **feel** the energy as he talks about COMAL. And you can too. By the time you read this we should have a video tape of his talk in California. Right now they are adding titles to it (using COMAL 2.0 large lettering of course). Call for information on how to buy or rent a copy. Perfect for schools and user groups!

Schools! Do you realize that COMAL was designed **just for you!** We are reprinting the article COMAL IN SCHOOL from the first issue of COMAL TODAY. We also are offering the SCHOOL COMAL PACKAGE again - at the same price as in January 1984! But with twice as much included! For just $170.35 (plus $7.20 shipping) a school gets 12 books, 12 disks, and a COMAL TODAY subscription. Ideal for reviewing COMAL.

COMAL ONLINE! Most national networks now support COMAL. You can find COMALites to 'talk' with Tuesdays on People Link and Thursdays on PlayNet. Delphi soon may allow you to electronically scan and selectively seach past issues of COMAL TODAY. Quantum Link may be offering a variety of services tailored to COMALites, including a complete section of the network just for us. Even CompuServe supports COMAL.

They're **beautiful. Marvelous.** And you can be too! In royal blue! Yes, the COMALite shirts are back and better than ever, featuring Calvin the COMAL turtle on the front and a bold COMALite lettering on the back. Introduced at the MARCA computer show, blue shirts were seen everywhere. Wear your COMALite shirt to your user group meetings. Then just look around for other blue shirts when you want to talk about COMAL. ■

COMALites Unite!

by Colin Thompson

Three new disks emerge this month, assembled from a mountain of top-notch COMAL programs submitted to us. A trend, predicted for months, has finally appeared. It seems that the most prolific COMAL programmers have switched from COMAL 0.14 to COMAL 2.0.

Since our goal is to distribute the best programs available, some changes have been made. Starting with disk #11, User Group disks will include the excess, first rate 2.0 programs that for one reason or another, didn't make it on the COMAL TODAY disk.

We are also planning to release some specialized COMAL 2.0 disks soon. Some authors (like myself) write **systems** that fill up a disk. Due to their size, these large-scale programs can only be published on a disk of their own.

The first of these specialized disks is called the **FONT DISK**, and features 31 different COMAL 2.0 fonts. Also included is an improved font editor, a font viewer, and a font convertor (BASIC to COMAL).

Other disks on tap for release this fall are **MATH & SCIENCE**, **THE GRAPHICS TOOLBOX**, **TYPING TUTOR**, and the COMAL 0.14 **GRAPHICS EDITOR**.

This month we are featuring an extraordinary program called **PROG'RAM** by Glen Colbert (see page 73). I used his program to solve a sticky little problem, by turning one of my COMAL 2.0 programs into a PACKAGE. My program is a 30 block MENU, which let's you select one of 14 different programs to be CHAINed in. Mind you, the menu program is all COMAL code - no ML.

I first PROTECTed it to reduce it's size (by 667 bytes), and then ran Glen's PROG'RAM against it. The result is LINKable disk file called PKG.MENU. To use it, I **LINK** the program instead of LOAD it. It's set up to RUN on LINK. At this point, it looks and behaves just like it did when it was a normal PRG file.

When a new file is CHAINed in, MENU goes away and the new program is executed. Now the fun: When the second program is finished, (Press Q for QUIT), it executes a USE MENU. That brings back the MENU, which had been hidden away under the cartridge. You could issue the USE command in the immediate mode if you like.

The point is, you can now have two COMAL programs in memory at once. One as a package and the other a normal COMAL program. The package program is treated as though it is ROMmed (see page 18 of COMAL 2.0 Packages), so it won't be SAVEd along with any other program.

You can store away up to 16K of normal COMAL code this way without using up a single byte of your RAM workspace! If you don't like the idea of LINKing a program, write a short COMAL boot program that executes a **BATCH FILE FROM MEMORY** (see COMAL TODAY #7, page 32). The only command to execute would be LINK "PROGNAME".

As the new school year begins, I think it's important to remember those poor, deprived children, all across America, that will **not** be learning COMAL in their classrooms. Captain COMAL is making a determined effort to reduce the suffering in Computer Science Labs by offering a special introductory package to schools. See the Captain's remarks on page one. ■

How To Type in COMAL Programs

Line numbers are irrelevant to a running COMAL program. COMAL only provides line numbers for your benefit in editing the program. Thus most magazines do not use line numbers when listing a COMAL program. It is up to YOU to provide the line numbers. But of course, COMAL can do it for you quite easily. Just follow these steps to type in a COMAL program:

1) Enter command: NEW
2) Enter command: AUTO
3) Type in the program
4) When done:
   Version 0.14: Hit <RETURN> key twice
   Version 2.0 : Hit <STOP> key

Remember - use unshifted letters thoughout entering the program. If letters are capitalized in the listing it does not mean to use SHIFT with those letters. They are capitalized merely to be easy to read. The only place to use SHIFTED letters is inside quotes. Also, you don't have to type leading spaces in a line. They are listed only to emphasize structures. You DO have to type a space between COMAL words in the program.

LONG PROGRAM LINES: We are continuing to print COMAL TODAY with two columns per page, printed in elite (12 pitch) with 42 characters maximum per line. This makes it easiest to read. However, some program listings have program lines that extend beyond the 42 character limit. We decided to list these lines in the same manner that COMAL uses when listing long lines on a 40 column screen. We simply break the line right at the 42 character spot, and continue it on the next line, indenting it properly to keep the program structures obvious. These are called wrap lines. To draw your attention to these continued lines we add a //wrap line comment to the end of the line. Whenever you see this make sure you type both lines as one continuous program line! The following example includes a line with more than 42 characters that we must list on two lines, but you must type in as one long program line:

```
IF CURRENT'NAME$<>"FINISH" THEN PRINT'LABE
L(CURRENT'NAME$,PHONE$) //wrap line
```

If you type in this long program line as two shorter program lines, COMAL will not object (although sometimes it will)! But, the program will not work unless it is entered as one long line. The procedure name PRINT'LABEL is split onto two lines in the listing, but the //wrap line draws your attention to this fact.

Letters

Dear Len- You may recall my visit with you a year ago when I wanted information about starting COMAL classes in our high school. We did in fact start teaching COMAL last year, and have been very hapy with the results. We introduce it at the 7th grade level and teach some advanced work later on. Our director, however, is a bit uneasy about our progress. He wonders if we made the right choice, citing the fact that we were and still are the only 7-12 school in this area that teaches COMAL. His questions are centered around "Why aren't more schools catching on to COMAL?" My explanations that they are all missing out on a good thing don't seem to appease him much. I need help! - **William R Carlson, Duluth Cathedral High School, 1215 Rice Lake Road, Duluth, MN 55811.**

Ed note: TEACHERS UNITE! By using COMAL now, you are giving your students a couple years head start on the rest. But being first is often lonely, and the fact that FIVE coutries in Europe have adopted COMAL as the language to teach their students isn't much help. Keep in contact with each other! Jim Ventola, 328 Poplar Street, Roslindale, MA 02131 has offered to collect a list of interested teachers. Plus, how about a few teachers writing to Mr. Carlson? He can use your letters to appease his director. Meanwhile, the lack of an APPLE COMAL is what is slowing the use of COMAL in schools here. Writing an APPLE COMAL interpreter or compiler would make a wonderful graduate project. MIT had LOGO. Who will get COMAL? We offer assistance to anyone undertaking writing APPLE COMAL - just contact us. Meanwhile, remember that COMAL prepares your students for the future. BASIC only becomes a stumbling block. Keep your students best interest in mind. Use COMAL - and try to get neighboring schools to use it as well.

Subscribe Today!

To The Southeastern United States fastest growing Commodore Users Group tabloid newspaper

SPARKPLUG!

Published monthly by the Spartanburg Commodore Users Group (SPARCUG)

☐ $12.00 Per Year U.S.

☐ $1.00 For Sample Copy

☐ $20.00 Per Year U.S. SPARCUG Associate Membership

NAME: ____________________

ADDRESS: ____________________

CITY: ____________ STATE: ______ ZIP: ______

Send Check or Money Order To:

SPARCUG
P. O. Box 319
Spartanburg, S.C. 29304.

Dear Len- I thoroughly enjoyed your EASY SPRITES article in COMAL TODAY 8. However, the MOVING procedure on page 19 doesn't seem to work with my 2.0 Cartridge. Help!

You caught me! I wrote the article for COMAL 0.14 and then adapted it for use with COMAL 2.0. The MOVING procedure is correctly listed on pages 20 and 21. But on page 19 I forgot to remove the word CLOSED in the header. The first line should be:

```
PROC MOVING(NUM)
```

When the word CLOSED is added at the end of the line, all sprite keywords are unknown to the procedure. Thanks for bringing this to my attention.

Questions and Answers

QUESTION: I tried the first example from a listing and got an error. Why do you put the two 00 in front of the number 10? It comes out as just 10 when shown on the screen.

ANSWER: It sounds like you are still in BASIC. Line number 10 in BASIC is listed as 10. In COMAL all line numbers are always 4 digits, leading zero's are used as fillers. You may enter a line number as 10 but it will always list as 0010. Since you were still in BASIC, you can expect to get errors while typing in a COMAL program. First LOAD COMAL from disk. Then try the programs.

QUESTION: I have two printers. One needs a linefeed added to each carriage return but the other doesn't. Can COMAL handle both ways?

ANSWER: Good news. You can tell the COMAL 0.14 system whether or not you want a linefeed tacked on the end of each line sent to the printer. Use the following command:

```
LINEFEED +     means add a linefeed
LINEFEED -     means don't add a linefeed
```

COMAL 2.0 also can be set either way. But it is done differently:

```
USE SYSTEM
SETPRINTER("u5:/a+/l+/t+/s7/d-")
```

That would set printer device 5 to include the extra linefeed. Here is what the command said: Unit 5 will have ASCII translation ON, linefeed ON, timeouts as in IEEE, secondary address of 7, and it is NOT a disk file.

FILENAME CONVENTIONS

QUESTION: Why do I see periods in C64 COMAL file names? I thought file extensions were part of CP/M and MS-DOS. Does COMAL add them to the C64?

ANSWER: No, COMAL does not add the extension feature to Commodore file names. However, the COMAL Users Group has set up some conventions for naming files to help users identify the type of file by its name. Also, the COMAL 2.0 Cartridge allows pattern matching with the DIR and CAT commands. Thus, using these conventions it is easy to find all your batch files on a disk:

```
CAT "BAT.*"
```

Here are the conventions we use and recommend you use as well. It will help you keep track of your files, as well as give meaning to your file names if others use your disk.

0.14	2.0	MEANING
NAME	NAME	COMAL program file
NAME.L	LST.NAME	Program listed to disk
NAME.PROC	PROC.NAME	PROC listed to disk
NAME.FUNC	FUNC.NAME	FUNC listed to disk
NAME.DAT	DAT.NAME	Data file
NAME.TXT	TXT.NAME	Text file
NAME.DOC	DOC.NAME	Documentation file
	EXT.NAME	External PROC/FUNC
	SHAP.NAME	Sprite shape file
	FONT.NAME	Font file
	PKG.NAME	Package file
	BAT.NAME	Batch file
	SNG.NAME	Song file
	HRG.NAME	Color COMAL picture
NAME.HRG		Black/white bitmap

More ►

Questions and Answers

QUESTION: Will there be a COMAL version for the 128 mode of the C128?

ANSWER: No. The COMAL 2.0 cartridge already works in the C128, in the 64 mode.

QUESTION: I can't get the COMAL 2.0 cartridge to work reliably in my new C128. It works for about half an hour, then the computer locks up and the screen looks funny. If I turn the computer off and try it again later, everything works. What's up? Do I have a bad cartridge?

ANSWER: No, the cartridge is probably fine. Once again, Commodore has given us a shiny new computer with a dull old power supply. The C128's power supply can barely support the unexpanded machine. When you plug in another cartridge, you may find that <u>your</u> power supply is a bit weaker than most. This has always been a sore spot with C64 owners, and now the C128 continues the legacy.

The answer, as it has been for C64 users, is to replace the power supply, with <u>one not made by Commodore</u>. If Commodore included a good power supply, there would be no need for 'after-market' units.

QUESTION: I just got the COMAL 0.14 system - it's fantastic! I'm trying to LIST a program to my printer in upper/lower case and I just can't figure out how to do it. Please help.

ANSWER: Here's the proper syntax:

```
OPEN FILE 255,"", UNIT 4,7, WRITE
SELECT "LP:"
```

File 255 is reserved by COMAL for the printer. The double quotes mean <u>no filename</u> is necessary to talk to the printer. UNIT 4 indicates the device number (the printer usually is 4) and 7 is the secondary address. After you issue the command: <u>SELECT "LP:"</u>, all further PRINT statements will be sent to the printer in upper/lower case. Remember to <u>SELECT "DS:"</u> after the printing stops. (COMAL 2.0 uses upper/lower case mode automatically).

QUESTION: I've tried ENTERing some of the procedures found on a COMAL TODAY disk but they don't seem to do anything. After I RUN them, it says "END AT 9120". What good is a procedure if you can't RUN it?

ANSWER: A procedure is not a program, but a <u>part</u> of a program. You can RUN a program and expect it to do something, but not a procedure. The procedures we distribute, either on disk or printed here in COMAL TODAY, are meant to be merged into another program. A procedure is executed only if it is 'called'. You can call it in the immediate mode if you type RUN, then the name of the procedure.

To use a procedure in a running program, you call it by placing it's name on a line in the program. If the procedure requires a <u>parameter</u>, then that information must be passed along when it is called. Example:

```
0010 initialize // a simple PROC call
0020 box (10) //   make 10 boxes
                   (10) is a parameter
```

QUESTION: I want to make a printed listing of a disk directory. I've tried SELECT "LP:", then CAT, but it doesn't work.

ANSWER: You had the right idea (and in COMAL 2.0 it works), but COMAL 0.14 won't let you do it. Instead, use a small program called <u>dir'lister</u>, found on <u>Utilities #1</u> disk or <u>TODAY DISK #1</u>. ■

Control A - Ooops - Bug Fixes

Also see pages 4, 46, 63, 79

The CUSTOM ERROR MESSAGES program printed on pages 66-68 of COMAL TODAY #8 does indeed create a 2.0 package of error messages, but the output of the program is a memory image rather than the format COMAL expects to see with the LINK command.

To use the CUSTOM ERROR MESSAGE file with LINK requires an extra step - plus the following small change to one line:

Original line:
```
OPEN FILE 1,name$,WRITE
```

Revised line:
```
OPEN FILE 1,name$+",p",WRITE
```

SAVE this corrected version on your disk:

```
SAVE "make'err'correct"
```

Now, to create your own custom error message file just follow these steps:

1) LOAD "make'err'correct"
2) RUN
3) Type in <u>new'errors</u> as file name.
4) The new file is created.
5) From TODAY DISK #7:
   LOAD "make'object'file"
   This program listing in COMAL TODAY #7 has one typo. See correction below.
6) RUN
7) Type <u>new'errors</u> as first filename
8) Type <u>pkg.new'errors</u> as second name
9) An error message package file is made.

You now have two files on your disk. Delete the first file named <u>new'errors</u>. Keep the file named <u>pkg.new'errors</u>. Now to use it just enter this command:

```
LINK "pkg.new'errors"
USE new'errors
```

Sadly, the "up arrow" bug has struck again. This time in Ian MacPhredran's program in COMAL TODAY #7 called MAKE'OBJECT'FILE. The up arrow (to the power of) was printed as the number one (1). This is a problem with PaperClip that has since been discovered and corrected.

To make the corrections, turn to page 62 of CT#7 and look for two lines that end with MOD 216. Change the 1 in 216 to an up arrow. That will correct the error.

◆◆

In COMAL TODAY #8 the procedure TOGGLE'KEYS was published. The procedure works fine from immediate mode, but will not work within a running program. The following procedure should be used instead.

```
PROC graph'keys(state)
  POKE 49763,state
ENDPROC graph'keys
```

◆◆

In COMAL TODAY #7 Jim White wrote about using TURBODISK to load COMAL 0.14. While the program that ended the article was completely accurate, the text that described the operation was misleading. Page 25, paragraph 2 should read:

"Replacing the name TURBODISK.OBJ with TURBO 64 and SYS 49152, on line 60, with the proper address of TURBO 64 should load and initialize it."

However, you don't need TURBODISK to fast load COMAL. Elsewhere in this issue you will find ML'SIZZLE, a COMAL fast loader. Plus, COMAL QUICK is available as part of UTILITIES SET #2.

◆◆

The program ILLUSION which was published in COMAL TODAY #8 had a line missing in it. This line sets up the graphics screen.

```
//
init'graph //add this line
block2 // main  program
```
■

Comal in Schools

by Len Lindsay

This article is reprinted from the first issue of COMAL TODAY. Due to MANY requests for back issues you now can get the first four issues as a spiral bound book for $14.95 ($10.95 to subscribers).

My daughter, Rhianon, has her own computer. She likes to play some of the games and type on the keyboard spelling words on the screen. She now goes to school to learn reading, writing and arithmetic. Plus, while at school she also will learn about computers.

A few years ago, I was in charge of the committee formed to choose a new MATH series for the primary grades in the school where I was teaching. We spent alot of time making our decision. That is the way it should be. I wouldn't want my daughter being taught MATH from a text book that was used solely because it was given to the school as a free gift.

The same amount of care should go into chosing a computer programming language. First decide what the programming course is supposed to accomplish. Then identify the best way to meet the goals. Buying a dozen computers and using the language that comes free with them is not a responsible method. Here are a few of the many points that should be considered:

1) **The computer system should be affordable**. Some students may wish to get a similar computer to use at home. One computer for each student allows all students to simultaneously try an exercise presented by the teacher. The cost of two dozen computers can be too high for the school if an expensive computer is used. The Commodore 64 computer is an inexpensive yet powerful computer and special systems are available now that allow one disk drive and printer to be shared by several computers, further helping to keep costs down, and reduce the number of diskettes needed. Disk Loaded C64 COMAL is available.

2) **The programming language should be easy to learn and use**. Some students may not be naturally adept at programming, and the language should not scare them away. Avoid having to perform many steps just to run a simple program. A compiler requires extra steps and thus should be avoided. Find a language with a built in editor and command language, allowing everything to be done from within the language. C64 COMAL is ideal. It is an easy language with a built in full screen editor PLUS the popular <u>LOGO</u> turtle graphics and an easy to use FILL (paint) command.

3) **The programming language should adequately prepare the more advanced students for future programming courses**, most likely to include a structured language with modular programming. COMAL is a natural lead in to Pascal, Ada, PL/I, Modula, and C.

4) **The programming language should support general educational objectives**. This includes problem solving and communicating ideas. COMAL has no problem in this area.

5) **A program listing should be readable by each student in the class**, not just the one who wrote it. Each student can then learn from the work of the others. This one of COMAL's specialties. You will not find a better language in this area, encouraging sharing of ideas and programs.

6) **The programming language should have a clear definition**. This would answer any questions about what the language really

More ►

is. COMAL is well defined by the COMAL KERNAL, which is printed in the COMAL Handbook.

7) **There should be a choice of text books or teaching aids available.** Schools will be happy with COMAL here. They can get Beginning COMAL as an introduction text, Foundations in Computer Studies With COMAL and Captain COMAL Gets Organized for intermediate texts, and Structured Programming With COMAL for advanced studies. The COMAL Handbook is an excellent reference book. 14 books are available now. More books are being written. A matching disk containing the programs from the books is also available as an aid to the teacher. Plus an excellent COMAL TODAY newsletter. All are available from COMAL Users Group USA.

COMAL was originally designed to be an educational programming language, thus it is ideal for schools. COMAL may be a new language to American schools, but it has been around since 1974, and is the official language taught in schools in Denmark, Ireland, Norway, and Sweden (and soon England and Germany). It is a three pass interpreter, run-time compiler. It is easy to learn and use and supports general educational objectives. It's structures prepare students for future studies. Program listings are very readable, with all structures automatically indented. It has a clear definition, the COMAL KERNAL, and includes the power of Pascal. Best of all, it is affordable. The disk for the Commodore 64 is only $7 and a school system can make as many copies of the disk as they need (try that legally with LOGO!). Many text books and disks are available as well.

SUMMARY: It would be advisable to have all interested students exposed to several different programming languages. This would be a definite advantage in their future studies. They will see that there can be more than one acceptable way to solve a problem. A person who can speak three foreign languages in addition to ENGLISH has an advantage in many situations, especially in Europe where a variety of languages are spoken in a relatively small area.

A reasonable course of action would be to teach one good general purpose language first (**COMAL**), and present additional languages to interested students after that (BASIC, PILOT, and LOGO). The Commodore 64 supports all these languages, making it an ideal school computer.

Schools should take a close look at COMAL and the Commodore 64. They can buy five of these for the price of one APPLE. And the APPLE can't run COMAL without an extra Z-80 board and an expensive ($235) COMAL disk (and it doesn't have sprites or turtle graphics like C64 COMAL).

Once a school makes the move to COMAL, they will not be alone. We are compiling a list of COMAL schools. COMAL Users Group, U.S.A., Limited offers each school a special COMAL School price list with substantial discounts on COMAL text books and disks. Plus a special introductory SCHOOL COMAL PACKAGE offer: one of everything: COMAL System Disk, 3 Different COMAL text books, 1 COMAL reference book, 1 CAPTAIN COMAL book, 1 Turtle book, 1 COMAL newsletter subscription, and the 6 matching disks, all for only $170.35.

[Ed note: this package is now available again - same price but about twice as much! See announcements in this issue.] ■

Modem Fun with Comal 2.0

(reprinted from the Clark County COMAL Club newsletter)

Excerpts from a telephone conversation on May 28, 1985 between David Skinner and Clay Ratliff.

David: I have made a few adjustments to your scan'joy procedure. It should work better now.

Clay: Well, that is good news. I suppose Alan and I can come by later to pick it up. Say, why don't you try to send it over the modem line?

David: Okay. We'll need to use the same modem program and it will have to have upload and download capabilities.

Clay: You better pick one that is real easy to operate. This is a brand new modem and I don't have much experience using it.

David: I'm in the same shape here. I just wish we could tell COMAL to send it over the phone direct without using a modem program.

Clay: That's an incredible idea! I'm switching off my machine now to connect my modem.

David: Uh, what's the idea?

Clay: You can type LIST "sp:" and I will type ENTER "sp:" and it should work!

David: Now wait a minute! I am now switching off to connect the modem. We are going to have to time this just right. We will type the commands on the screen, but we will not press RETURN right away. Wait for the carrier detect light to come on, wait five seconds, and press RETURN. I will wait for fifteen seconds and then I will LIST the program to the serial port.

Clay: Wait until I get setup with a disk in my drive. Also I need to use a different phone for the modem.

David: That's okay, I haven't loaded the program yet.

Clay: I am now ready to ENTER the program.

David: I am ready to LIST the program, connect your modem!

(60 seconds of buzzing)

David: Clay, are you there?

Clay: I have it! I have it! I have got the program! This is so easy, this is great! This is better than a modem program, COMAL 2.0 **is** the modem program! I can't wait to send this to Alan. This is really great.

David: I am just thinking of all the time and gasoline we're going to save. You realize that we don't have any error correction protocal?

Clay: That's okay. This is just for local use. If there's an error, COMAL will tell us.

David: I wonder what we could do with SELECT INPUT "sp:" or SELECT OUTPUT "sp:"? We should think of some ways of testing it.

Clay: Well, okay. I have to go now, so I'll see you later.

David: Bye.

Excerpts from telephone conversation on May 28, 1985 between Clay Ratliff and Alan Tackett.

More ►

Clay: David just sent me some changes on our program, I thought you might like for me to send them on to you.

Alan: Send them? How are you going to do that?

Clay: Just plug in your modem and after you get a carrier detect signal type SELECT INPUT "sp:".

Alan: Are you sure this is going to work?

Clay: Don't worry, I know what I'm doing.

Alan: That's what I was afraid of. Okay, I'm ready, let's do it.

(five minutes of buzzing)

Clay: Did you get anything?

Alan: What did you do? This disk drive went crazy, I didn't have a disk in it when you typed CAT. And why did you change the TIME in my machine?

Clay: It worked! It worked! I took complete control of your machine. I knew it would work!

Alan: It didn't work all that well. The machine quit when you tried to CAT a disk that wasn't there, and it dropped out after you typed in the program. It also seemed to quit some time after you made my machine SELECT OUTPUT "sp:".

Clay: Well, that's okay, it still works. Anything we can do on the COMAL cartridge we can send over the phone lines and never bother with modem programs!

Alan: I bet it won't work on long distance, and I bet it won't work on bulletin boards. The only people it will work with are the ones with COMAL 2.0.

Clay: Okay, so maybe it won't do everything, but we can send programs across town.

Alan: Have you told Gene Harrelson about this?

Clay: No, not yet.

Alan: Well, bye Clay, I have to go now. I have this real important phone call to make. See you at school tomorrow.

Excerpts from telephone conversation on May 28, 1985 between David Skinner and Gene Harrelson.

Gene: Hello, David? This is Gene Harrelson.

David: Sure, what can I do for you?

Gene: I think you better warn the club members about the COMAL 2.0 Cartridge.

David: Why, what is it that's gone wrong?

Gene: Alan called me on the telephone and had me type SELECT INPUT "sp:" and the next thing I knew, he had taken control of my computer and had it drawing strange pictures on the graphics screen.

David: Oh, my goodness, I will see that everyone hears about this. You can be sure of it.

Gene: I don't mind Alan so much, but if someone wants you to SELECT INPUT "sp:" it had better be someone you really trust. ■

Connect Your C64 to An IBM PC

by Tom Kuiper

A few months ago, I obtained a demo version of MYTECH COMAL for the IBM PC. Not owning such a machine, I took the PC COMAL to work, where IBM PC's abound. Having COMAL at work then tempted me to do some quick calculations. Alas, one grew in size until it surpassed the memory limit of COMAL demo, but also assumed an importance which demanded more additions. I did not relish retyping 105 lines of code into my C64. David Stidolph's COMAL 2.0 MODEM USE in COMAL TODAY 5 pointed the way. The result is a communications program (DIRECT'COM on TODAY DISK 9) to transfer files between the C64 and the IBM PC, the latter using SmartCom II, a PC communication program for the Smartmodem. DIRECT'CON can easily be adapted for more general communication purposes.

My C64 also uses a Smartmodem, connected with a VIC-1011A RS232C adapter. DIRECT'CON is set up for a direct modem-modem hookup using a modular telephone cord. SmartCom on the PC must be configured. (The remainder of this paragraph assumes that you are reasonably familiar with SmartCom.) I defined a communication set which differs from the "Standard Values" set as follows:

Name of Set:	300-baud direct
Connection Type:	Bell 300
Receive Time-out:	255
Send Time-out:	255
Answer on Ring:	0
Phone Number:	(one blank)

It would be possible to change other parameters to facilitate communication and file transfer, but I expect to use this program with minimal change to talk to other computers as well. So, I placed the burden of conforming to the C64.

DIRECT'CON uses the special function keys in the following way:

f1 - turns character echo on/off. The default is off, which means that the C64 does not echo each character it receives. It should be turned on if the other computer has initiated the call in full-duplex and the C64 has answered (at least if you want the other guy to see what he types on his screen).

f2 - adds a linefeed for every carriage return sent out. This is normally off but can be turned on if the other computer needs it.

f3 - turns ASCII code display on/off. This is normally off. When turned on, the program displays the PetASCII code of each incoming character, in parentheses and a different color, after printing it to the screen (or acting on it if it is a control code). Incoming characters that the program cannot interpret are always displayed this way, regardless.

f4 - sends the command to the C64's Smartmodem to connect. This can be tailored easily for answering, dialing stored phone numbers, etc. It is set up to connect without dialing a number, for a direct modem-modem connection.

f5 - opens a file to receive incoming data. The program has a buffer of 100 128-character lines. When the buffer is full, the stop character (X-off = DC3) is sent several times to the other computer. When the buffer has been written to disk, transmission is resumed by sending a start character (X-on = DC1). There is enough memory left to add more lines. Also, memory could be used more efficiently, since most lines have much less than 128

More ►

characters. However, the present choice works well enough.

f6 - shows the disk directory; useful if you need to find or choose a file name.

f7 - opens a file to send data to the other computer. This is pretty slow. The problem is that the other computer can be expected to echo each character, so that there is the possibility of conflict on the modem line. Therefore, the program is set up to wait for the returned character. The current wait value is 3 jiffies. There is still an occasional incoming character that is messed up, but the outgoing data seem to be fine. A possible solution would be to have the other computer go into half-duplex mode. I haven't tried that.

f8 - closes an open file. You would use this when the other computer has stopped sending a file. If you are communicating over the telephone, you'll need a way of knowing that the transfer is complete. You could have the other operator type "DONE" or something equivalent, or send CTRL-G's to beep your computer. If you use it while a file is coming in, you'll close the C64 file but the incoming data will continue to be displayed on your screen. You may also use this to terminate data going out from a file, for example if you see a problem developing.

There are some oddities in COMAL's ASCII to PetASCII conversion (see the accompanying article COMAL 2.0 ASCII CONVERSION). In this program, brackets [] have been turned back to brackets. Left and right brace (curly bracket) are represented by C=W (Commodore and W key together) and C=Q respectively. A vertical bar is obtained by shift-minus. A tilde (wavy line) is equated to C=Y. An 'accent grave' (backwards apostrophe) has been turned into C=V. A backslash is C=B. Some of these graphic symbols look only a little like what they should represent. Will someone design a proper font?

I think that this can be quite a general communications program, but I haven't the time to explore all the possibilities. As I use it for various purposes, it will continue to improve. In the meantime, this version is contributed to encourage others to work on it. How about terminal emulation, or XMODEM and KERMIT file transfer protocols? Please describe any modifications clearly and submit them to COMAL TODAY.

One word of caution. Timing is critical. It doesn't take much to make the basic loop time longer than the interval between incoming characters. This will be pretty obvious when you start losing data. Conflict on the line will generate garbage characters. When either of those things happens, think clearly about the possible causes. An amateur "try something and see if it works" can quickly lead you from bad to worse. Good luck!

More ►

PLEASE READ

We are celebrating the release of our first COMAL 2.0 User Group disk (#11). Get it free when you buy our User Group Disk #10.

We also are celebrating the reprinting of the first issues of COMAL TODAY newsletter. Get our very first TODAY DISK free when you order TODAY DISK #2.

Two ways to spend only $9.75 for one disk and get two. ■

Direct'con - Link to The IBM PC

```
// delete  "0:direct'con"
// save    "0:direct'con"
// direct connect communications
// (c) 1985 Tom Kuiper - (213) 202-0917
// this program may be copied and modified
// for personal use. it may not be sold in
// any form without written consent of the
//author. it may be included on user group
// distribution disks. the author requests
// a copy of such disks: 10517 northvale
// road, los angeles, ca 90064
DIM rec$ OF 1, send$ OF 1, file'name$ OF
 16, c$ OF 1 //wrap line
DIM rcv$(255) OF 1, snd$(255) OF 1
DIM line$(100) OF 128, state$(2) OF 3, e
rror'message$ OF 40 //wrap line
state$(1):="off"; state$(2):="on"
// PetASCII special codes
DIM delet$ OF 1, clear'screen$ OF 1
DIM set'lower'case$ OF 1, lock'case$ OF
1, unlock'case$ OF 1 //wrap line
DIM down$ OF 1, up$ OF 1, left$ OF 1
DIM red$ OF 1, green$ OF 1, d'grey$ OF 1
DIM m'grey$ OF 1, blue$ OF 1
DIM rev'on$ OF 1, rev'off$ OF 1
DIM f1$ OF 1, f2$ OF 1, f3$ OF 1
DIM f4$ OF 1, f5$ OF 1, f6$ OF 1
DIM f7$ OF 1, f8$ OF 1
delet$:=CHR$(20); clear'screen$:=CHR$(147)
set'lower'case$:=CHR$(14); lock'case$:=C
HR$(8); unlock'case$:=CHR$(9) //wrap line
down$:=CHR$(17); up$:=CHR$(145); left$:=
CHR$(157) //wrap line
red$:=CHR$(28); green$:=CHR$(30); d'grey
$:=CHR$(151); m'grey$:=CHR$(152)//wrapline
blue$:=CHR$(31)
rev'on$:=CHR$(18); rev'off$:=CHR$(146)
f1$:=CHR$(133); f2$:=CHR$(137)
f3$:=CHR$(134); f4$:=CHR$(138)
f5$:=CHR$(135); f6$:=CHR$(139)
f7$:=CHR$(136); f8$:=CHR$(140)
// Normal ASCII control codes
DIM nul$ OF 1, bs$ OF 1, lf$ OF 1,cr$ OF 1
DIM xon$ OF 1, xoff$ OF 1, quote$ OF 1
nul$:=CHR$(0); bs$:=CHR$(8); lf$:=CHR$(10)
cr$:=CHR$(13); quote$:=CHR$(34)
xon$:=CHR$(17); xoff$:=CHR$(19)
// COMAL constants and vectors
chrin'vec'lo:=PEEK($0324)
chrin'vec'hi:=PEEK($0325)
//
modem:=22
disk'file:=33
USE system
PRINT set'lower'case$,lock'case$,
textcolors(1,1,11)
PRINT "Building conversion tables..."
build'rcv'tbl(rcv$())
build'snd'tbl(snd$())
ech#:=0
PRINT red$,"Echo is";state$(ech#+1);"(f1
 to change)",d'grey$ //wrap line
add'lf#:=0
PRINT red$,"Add LF is";state$(add'lf#+1)
;"(f2 to change)",d'grey$ //wrap line
ascii#:=0
PRINT red$,"Show ASCII is";state$(ascii#
+1);"(f3 to change)",d'grey$ //wrap line
clear'to'send#:=TRUE
request'to'send#:=TRUE
read'file#:=FALSE
write'file#:=FALSE
//  300 baud, 8 bits, no parity, 1 stop
OPEN FILE modem,"sp:b300d8s1pn"
TRAP ESC-
lin#:=1
REPEAT
  received#:=0
  REPEAT
    rec$:=modem'get$(modem,asc#)
    IF asc# THEN
      IF received#>128 THEN pause'modem
      received#:+1
      IF asc#<32 THEN
        process'control'code
      ELSE
        PRINT rec$,
        IF asc#=34 THEN POKE $d4,0
        //avoids entering quote mode
        line$(lin#):=line$(lin#)+rec$
      ENDIF
      IF ech# THEN PRINT FILE modem: rec$,
```

More ►

```
      // echos to sender
      IF ascii# THEN PRINT m'grey$,"(",as
      c#,")",d'grey$,//show ASC//wrapline
      TIME 0 // reset timer
    ENDIF
  UNTIL asc#=0 AND TIME>3 // check that
  nothing else is coming in //wrap line
  resume'modem
  received#:=0
  PRINT rev'on$," ",left$,rev'off$,
  send$:=KEY$
  IF send$<>nul$ THEN
    process'key
    IF ech#=1 THEN // avoid quote mode
      PRINT green$,send$,d'grey$,
      IF send$=quote$ THEN POKE $d4,0
    ENDIF
    TIME 0 //reset timer to wait for echo
  ELIF read'file#=TRUE AND clear'to'send
  #=TRUE THEN //wrap line
    IF EOF(disk'file)=FALSE THEN
      send$:=GET$(disk'file,1)
      PRINT FILE modem: send$,
      TIME 0 // reset timer-wait for echo
    ELSE
      read'file#:=FALSE
      CLOSE FILE disk'file
      PRINT
      PRINT green$,file'name$;"closed",d
      'grey$ //wrap line
    ENDIF
  ENDIF
  PRINT " ",left$,
UNTIL ESC
CLOSE
PRINT unlock'case$,
END
//
PROC process'control'code
  CASE asc# OF
  WHEN 7
    bell(1)
  WHEN 8,10,12,127 //these incoming codes
    PRINT rec$,//print corrected character
  WHEN 13
    IF NOT add'lf# THEN PRINT up$,
    PRINT rec$,
    IF ech# THEN PRINT FILE modem: lf$,
    // LF before CR to other unit
    IF write'file# THEN
      lin#:+1
      IF lin#>100 THEN
        pause'modem
        FOR i:=1 TO 100 DO
          PRINT FILE disk'file: line$(i)
          line$(i):=""
        ENDFOR i
        lin#:=1
        resume'modem
      ENDIF
    ENDIF
  WHEN 17
    clear'to'send#:=TRUE
  WHEN 19
    clear'to'send#:=FALSE
  OTHERWISE
    PRINT m'grey$,"(",asc#,")",d'grey$,
  ENDCASE
ENDPROC process'control'code
//
FUNC modem'get$(f'num,REF asc#)
  TRAP
    POKE $0324,PEEK($032a)
    POKE $0325,PEEK($032b)
    c$:=GET$(f'num,1)
    POKE $0324,chrin'vec'lo
    POKE $0325,chrin'vec'hi
    IF c$=nul$ THEN
      asc#:=0
      RETURN c$
    ELSE
      asc#:=ORD(c$)
      RETURN rcv$(asc#)
    ENDIF
  HANDLER
    POKE $0324,chrin'vec'lo
    POKE $0325,chrin'vec'hi
    REPORT // pass error outside
    asc#:=-1
    RETURN ""
  ENDTRAP // of this routine.
ENDFUNC modem'get$
```

More ►

```
PROC build'rcv'tbl(REF rcv'tbl$())
  // correct ASCII conersion
  FOR i:=1 TO 255 DO
    CASE i OF
    WHEN 8 // backspace=cursor left
      rcv'tbl$(i):=left$
    WHEN 10 // linefeed=cursor down
      rcv'tbl$(i):=down$
    WHEN 12 // vertical tab=clear screen
      rcv'tbl$(i):=clear'screen$
    WHEN 91 // left brace = [W]
      rcv'tbl$(i):=CHR$(179)
    WHEN 92 // vertical bar = (-)
      rcv'tbl$(i):=CHR$(221)
    WHEN 93 // right brace = [Q]
      rcv'tbl$(i):=CHR$(171)
    WHEN 96 // accent grave = [V]
      rcv'tbl$(i):=CHR$(190)
    WHEN 126 // tilde = [Y]
      rcv'tbl$(i):=CHR$(183)
    WHEN 127 // delete
      rcv'tbl$(i):=CHR$(20)
    WHEN 219 // left bracket
      rcv'tbl$(i):=CHR$(91)
    WHEN 220 // back slash = [B]
      rcv'tbl$(i):=CHR$(191)
    WHEN 221 // right bracket
      rcv'tbl$(i):=CHR$(93)
    OTHERWISE
      rcv'tbl$(i):=CHR$(i)
    ENDCASE
  ENDFOR i
ENDPROC build'rcv'tbl
//
FUNC open'file(f#,fn$,op#,REF em$) CLOSED
  TRAP
    CASE op# OF
    WHEN -1
      OPEN FILE f#,fn$,READ
    WHEN -2
      OPEN FILE f#,fn$,WRITE
    WHEN -3
      OPEN FILE f#,fn$,APPEND
    WHEN op#>0
      OPEN FILE f#,fn$,RANDOM op#
    OTHERWISE
      PRINT "Invalid OPEN'FILE operation"
      RETURN FALSE
    ENDCASE
    em$:=STATUS$
    IF em$(1:2)="00" THEN
      RETURN TRUE
    ELSE
      RETURN FALSE
    ENDIF
  HANDLER
    em$:=ERRTEXT$
    RETURN FALSE
  ENDTRAP
ENDFUNC open'file
//
PROC process'key
  CASE send$ OF
  WHEN cr$
    PRINT FILE modem: lf$
  WHEN f1$ // toggle echo
    ech#:=1-ech#
    PRINT red$,"Echo is";state$(ech#+1);
    "(f1 to change)",d'grey$ //wrap line
  WHEN f3$ // toggle show-ASCII
    ascii#:=1-ascii#
    PRINT red$,"Show ASCII is";state$(asci
    i#+1);"(f3 to change)",d'grey$//wrapln
  WHEN f5$ // receive file
    INPUT AT 0,0,16: blue$+"Store to filen
    am? "+d'grey$: file'name$ //wrap line
    IF open'file(disk'file,file'name$,-2,
    error'message$)=FALSE THEN//wrap line
      PRINT red$,error'message$,d'grey$
      IF em$(1:2)="63" THEN
        INPUT blue$+"Append,New name,Quit
        (A,N,Q)? "+d'grey$: an$//wrap line
        IF an$ IN "aA" THEN
          IF open'file(disk'file,file'name
          $,-3,error'message$)=FALSE THEN
          //two lines above are wrap line
            PRINT red$,error'message$,d'
            grey$ //wrap line
            write'file#:=FALSE
          ELSE
            write'file#:=TRUE
```

More ►

```
        ENDIF
      ELIF an$ IN "nN" THEN
        IF open'file(disk'file,file'name
        $,-2,error'message$)=FALSE THEN
        //two lines above are wrap line
          PRINT red$,error'message$,d'
          grey$ //wrap line
          write'file#:=FALSE
        ELSE
          write'file#:=TRUE
        ENDIF
      ELSE
        write'file#:=FALSE
      ENDIF
    ELSE
      write'file#:=FALSE
    ENDIF
  ELSE
    write'file#:=TRUE
  ENDIF
  IF write'file#=TRUE THEN
    PRINT green$,file'name$;"opened",b
    lue$ //wrap line
    PRINT "Hit f8 when transfer is don
    e",d'grey$ //wrap line
  ENDIF
  FOR i:=1 TO 100 DO
    line$(i):=""
  ENDFOR i
WHEN f7$ // send file
  INPUT AT 0,0,16: blue$+"Send filename?
  "+d'grey$: file'name$ //wrap line
  IF open'file(disk'file,file'name$,-1,
  error'message$)=FALSE THEN//wrap line
    PRINT red$,error'message$,d'grey$
  ELSE
    PRINT green$,file'name$;"opened",d
    'grey$ //wrap line
    read'file#:=TRUE
    PRINT blue$,"If necessary, send CT
    RL-Q from other" //wrap line
    PRINT "computer to start transfer"
    ,d'grey$ //wrap line
  ENDIF
WHEN f2$ // toggle add-line-feed
  add'lf#:=1-add'lf#
    PRINT red$,"Add LF is";state$(add'lf#+
    1);"(f2 to change)",d'grey$//wrap line
  WHEN f4$ // connect Hayes modem
    PRINT blue$,"Start other modem to an
    swer." //wrap line
    PRINT "When the tone is heard, hit a
    ny key." //wrap line
    PRINT "Wait for CONNECT, then hit RE
    TURN and" //wrap line
    PRINT "wait for other computer to re
    spond",d'grey$ //wrap line
    WHILE KEY$="" DO NULL
    PRINT FILE modem: "ATD"
  WHEN f6$ // show directory
    DIR
  WHEN f8$ // close file
    IF write'file# THEN
      pause'modem
      FOR i:=1 TO lin# DO
        PRINT FILE disk'file: line$(i)
        line$(i):=""
      ENDFOR i
      write'file#:=FALSE
      resume'modem
    ENDIF
    CLOSE FILE disk'file
    PRINT green$,file'name$;"closed",d'g
    rey$ //wrap line
    read'file#:=FALSE
  OTHERWISE
    PRINT FILE modem: snd$(ORD(send$)),
  ENDCASE
ENDPROC process'key
//
PROC pause'modem
  IF request'to'send#=TRUE THEN
    PRINT FILE modem: xoff$,xoff$,
    PRINT FILE modem: xoff$,xoff$,
    request'to'send#:=FALSE
  ENDIF
ENDPROC pause'modem
//
PROC resume'modem
  IF request'to'send#=FALSE THEN
    request'to'send#:=TRUE
    PRINT FILE modem: xon$,
```

More ►

```
  ENDIF
ENDPROC resume'modem
//
PROC build'snd'tbl(REF snd'tbl$())
  //correct ASCII conversion
  FOR i:=1 TO 255 DO
    CASE i OF
    WHEN 157 // backspace=cursor left
      snd'tbl$(i):=bs$
    WHEN 17 // linefeed=cursor down
      snd'tbl$(i):=lf$
    WHEN 147 // vertical tab=clear scree
    n //wrap line
      snd'tbl$(i):=CHR$(12)
    WHEN 179 // left brace = [W]
      snd'tbl$(i):=CHR$(91)
    WHEN 221 // vertical bar = (-)
      snd'tbl$(i):=CHR$(92)
    WHEN 171 // right brace = [Q]
      snd'tbl$(i):=CHR$(93)
    WHEN 190 // accent grave = [V]
      snd'tbl$(i):=CHR$(91)
    WHEN 183 // tilde = [Y]
      snd'tbl$(i):=CHR$(126)
    WHEN 20 // delete
      snd'tbl$(i):=CHR$(127)
    WHEN 91 // left bracket
      snd'tbl$(i):=CHR$(219)
    WHEN 191 // back slash = [B]
      snd'tbl$(i):=CHR$(220)
    WHEN 93 // right bracket
      snd'tbl$(i):=CHR$(221)
    OTHERWISE
      snd'tbl$(i):=CHR$(i)
    ENDCASE
  ENDFOR i
ENDPROC build'snd'tbl ■
```

Questions

QUESTION: How can a running C64 COMAL 0.14 program tell if the output is going to the printer or to the screen?

ANSWER: COMAL 0.14 uses memory location 152 to keep track of where the output is going. The two line below show how you can check this:

```
IF PEEK(152):=0 THEN PRINT "SCREEN"
IF PEEK(152) THEN PRINT "PRINTER"
```

QUESTION: Why are there always 2 or 3 strange lines at the start of your programs. They seem to be commands, but are preceded by // which means remarks.

ANSWER: By including the DELETE and SAVE commands you use to save the program to disk at the top of the line, it makes it easy for you to save your program each time you modify it. The lines start with // so the commands will not be executed when the program is RUN. To use them just:

```
LIST-33
0010 // DELETE "0:NAME1"
0020 //  BY CAPTAIN COMAL
0030 //   SAVE "0:NAME3"
```

The number after the name is the version number. To save your program, update the version number in each line. Here, change the 1 to a 2 and the 3 to a 4:

```
0010 // DELETE "0:NAME2"
0020 //  BY CAPTAIN COMAL
0030 //   SAVE "0:NAME4"
```

Now cursor up to line 10. Delete <u>0010 //</u> by pressing the space bar 7 times. Hit <RETURN>. The old file is deleted. The cursor is now on line 30. Type 7 spaces again. Press <RETURN>. The updated program is saved with the new version number.

Notice that the version being saved is 2 higher than the one being deleted. That allows the latest two version to be on the disk at all times. Thus if you really mess up your program <u>AND</u> save it - you still can get back the previous version. An extra benefit is that your program will always have the version number at the top for easy reference.■

ASCII Conversion in Comal 2.0

by Tom Kuiper

When assigning the serial port (sp:) and other devices, you have the option of selecting conversion to and from standard ASCII. The question which arises if you want to write a communications program is "How **standard** is the conversion?" The program described in CONNECT YOUR C64 TO AN IBM PC (page 12) gave me the opportunity to investigate this. The table below shows the results. The notation in the table is the same as that used in the article DISPLAYED CODES IN COMAL 2.0 on the following page.

	Standard	COMAL 2.0	
ASC	Key	Key	ORD
0	c @		
1	c A	c A	1
:	:	:	:
5	c E	c E, C 2	5
:	:	:	:
8	c H, bkspc	c H	8
9	c I, tab	c I	9
10	c J, LF	c J	10
:	:	:	:
13	c M, return	c M, return	13
14	c N	c N, C s	14
:	:	:	:
17	c Q	c Q, crsr dn	17
18	c R	c R, C 9	18
19	c S	c S, home	19
20	c T	c T, del	20
:	:	:	:
27	c [, esc		27
28	c bkslash	c 3, Eng.lb	28
29	c]	crsr right	29
30	c caret	c 6	30
31	c -	c 7	31
32	space	space	32
33	!	!	33
34	"	"	34
35	#	#	35
36	$	$	36
37	%	%	37
38	&	&	38
39	'	'	39
40	(	(	40
41	)	)	41
42	*	*	42
43	+	+	43
44	,	,	44
45	-	-	45
46	.	.	46
47	/	/	47
48	0	0	48
:	:	:	:
57	9	9	57
58	colon	colon	58
59	;	;	59
60	<	<	60
61	=	=	61
62	>	>	62
63	?	?	63
64	@	@	64
65	A	A	193
:	:	:	:
90	Z	Z	218
91	[	s +	219
92	bkslash	C -	220
93	]	s -	221
94	caret	uparrow	94
95	underscore	underscore	164
96	acc. grave	s *	96
97	a	a	65
:	:	:	:
122	z	z	90
123	lft brace	[	91
124	vert. bar	Eng.lb	92
125	rt brace	]	93
126	tilde	C +	126
127	del, c bksp		

Notes:

brace=curly brackets	C=Commodore key
acc. grave=reverse '	c=control key
	s=shift key ■

Displayed Codes in Comal 2.0

by Tom Kuiper

The tables in Appendices E and F of the C64 User's Guide drive me bananas. They don't easily tell me what ASCII code to expect from a given key. Table F is also set up for character set 1 only, whereas COMAL 2.0 primarily uses set 2 (with some changes). I finally resorted to producing my own table, which I think may be useful to others.

The column Cdr contains the Commodore ASCII code, the column Key the keystroke which returns that code, Dis/ASC the character which appears on the screen and the true ASCII code for that character, and Effect the effect of non-displayed codes when printed to the screen. For the keystrokes:

c means CTRL
s means SHIFT
C means Commodore key

A display of [ ] means the graphic symbol shown on the left of the key inside the brackets. ( ) means the graphic symbol on the right.

Cdr	Key	Dis/ASC		Effect
0				
1	c A			note 3
2	c B			note 3
3	c C			stops program
4	c D			note 3
5	c E			white textcolor
"	C 2			"
6	c F			note 3
"	c bkarrw			
7	c G			
8	c H			case change on
9	c I			case change off
10	c J			
11	c K			note 4
12	c L			note 4
13	c M			carriage return
14	c N			lower case
"	C shift			"
15	c O			
16	c P			note 3
17	c Q			cursor down
18	c R			reverse on
19	c S			home
"	home			"
20	c T			delete
21	c U			note 3
22	c V			note 3
23	c W			note 3
24	c X			note 3
25	c Y			note 3
26	c Z			note 3
27				
28	c 3			red textcolor
"	C Eng.lb			"
29	crsr right			cursor right
30	c 6			green textcolor
31	c 7			blue textcolor
"	C =			"
32	space	space	32	
33	s 1	!	33	
34	s 2	"	34	
35	s 3	#	35	
36	s 4	$	36	
37	s 5	%	37	
38	s 6	&	38	
39	s 7	'	39	
40	s 8	(	40	
41	s 9	)	41	
"	C 9	)	41	
42	*	*	42	
43	+	+	43	
44	,	,	44	
45	-	-	45	
46	.	.	46	
47	/	/	47	
48	0	0	48	
"	C 0	0	48	
49	1	1	49	
:	:	:	:	
57	9	9	57	
58	colon	colon	58	

More ►

Displayed Codes in COMAL 2.0 - continued

Cdr	Key	Dis/ASC		Effect
59	;	;	59	
60	s ,	<	60	
"	C ,	<	60	
61	=	=	61	
"	s =	=	61	
"	C =	=	61	
62	s .	>	62	
"	C .	>	62	
63	s /	?	63	
"	C /	?	63	
64	@	@	64	
65	A	a	97	
:	:	:	:	
90	Z	z	122	
91	s :	[	91	
"	C :	[	91	
92	Eng.lb	Eng.lb		
93	s ;	]	93	
"	C ;	]	93	
94	uparrw	uparrw	94	
95		bkarrw	95	

Codes 96 to 126 give the same display as those 96 higher (e.g. 96=192) but are not returned by any keystrokes.

127		note 1		
128		space		
129	C 1			orange text
130			32	carriage return
131	s STOP			RUN "*"
132				
133	f1			
134	f3			
135	f5			
136	f7			
137	s f1			
"	C f1			
138	s f3			
"	C f3			
139	s f5			
"	C f5			
140	s f7			
"	C f7			
141				return
142				upper case

Cdr	Key	Dis/ASC	Effect
143			
144	c 1		black textcolor
145	up CRSR		cursor up
146	c 0		reverse off
147	s HOME		clear screen
148	s DEL		insert
149	C 2		brown textcolor
150	C 3		lt. red text
151	C 4		dark grey text
152	C 5		med. grey text
153	C 6		lt. green text
154	C 7		lt. blue text
155	C 8		lt. grey text
156	c 5		purple text
157	lf CRSR		cursor left
158	c 8		yellow text
159	c 4		cyan textcolor
160		space	
161	C K	[K]	
162	C I	[I]	
163	C T	[T]	
164	bkarrow	[@]	
"	C @	[@]	
165	C G	[G]	
166	C +	[+]	
167	C M	[M]	
168		[Eng.lb]	
169	s Eng.lb	note 2	
170	C N	[N]	
171	C Q	[Q]	
172	C D	[D]	
173	C Z	[Z]	
174	C S	[S]	
175	C P	[P]	
176	C A	[A]	
177	C E	[E]	
178	C R	[R]	
179	C W	[W]	
180	C H	[H]	
181	C J	[J]	
182	C L	[L]	
183	C Y	[Y]	
184	C U	[U]	
185	C O	[O]	
186	s @	check	
187	C F	[F]	

More ►

Cdr	Key	Dis/ASC		Effect
188	C C	[C]		
189	C X	[X]		
190	C V	[V]		
191	C B	[B]		
192	s *	(*)		
193	s A	A	65	
:	:	:	:	
218	s Z	Z	90	
219	s +	(+)		
220	C -	[-]		
221	s -	(-)		
222	s uparrw	rvrs[+]		
223		note 1		

Codes 224 to 254 display the same as those 64 lower (e.g. 224=160) but are not returned by any keystrokes.

255		(+)		

Note 1 - This character is a block consisting of diagonal lines going from upper left to lower right.
Note 2 - This character consists of diagonal lines going from upper right to lower left.
Note 3 - These codes have special meaning in C64 COMAL 2.0 command mode. See Cartridge Graphics and Sound, page 63.
Note 4 - These codes also have special meaning which are recognized by many terminals as well as C64 COMAL 2.0. ■

'FREE' COMAL DISKS

Did you ever stop to ask yourself, "Where do all those great programs in COMAL TODAY come from?" Most of them come from programmers just like you.

If you send us an original program, on disk, with some instructions, we will mail back to you any stock COMAL disk, like the COMAL TODAY series. ■

DISPLAY KEY VALUES

Doug Bittinger came up with the following simple program to find the display codes of certain key combinations that weren't listed in the handbooks.

```
PRINT CHR$(147) // clear screen
TRAP ESC- // disable stop key
REPEAT
  PRINT CHR$(19) // home cursor
  PRINT "Press any key combination"
  c:=ORD(KEY$) // code of keypress
  IF c>0 THEN PRINT "CODE=",c,"  "
UNTIL ESC // stop key pressed
TRAP ESC+ // enable stop key
```

When you run the program, try pressing the CTRL key and another key. The Shift and Commodore keys will also work in combination with other keys. The values of the colors, function keys and cursor keys are easily revealed. Even the code for the STOP key is displayed. ■

Question: I have the Master Composer program from Access Software to create and load Interrupt driven music files. These files can be loaded directly from BASIC and started with a SYS 30120. The SYS address can be relocated.

I have not been able to LOAD and RUN these files under COMAL. If you know how to do it, please let me know. If not, please publish my question.

Answer: James, we don't know how to do it either. Any attempt to OBJ'LOAD the file and then SYS to 30120 results in a crashed C64. If anyone has a solution, let us know. There is a wealth of music files just waiting to be tapped by the clever COMAL programmer. ■

Top 5 Best Selling Comal Books

Beginning with issue #3 we have reported on the best selling COMAL books. Since last issue did not include the report, we are presenting the top 5 books from May through August 1985 now.

The Amazing Adventures of Captain COMAL series of books is doing very well indeed. It's nice to know that you don't have to have a big well known publisher to market your books. The Captain COMAL books are not yet available in stores, but are usually in stock here at COMAL Users Group USA. The five of them that made the top 5 list are:

COMAL Workbook
COMAL 2.0 Packages
COMAL From A To Z
Cartridge Graphics and Sound
COMAL Quick and Utilities #2

The authors of the Captain COMAL books spend alot of time preparing their manuscripts. We hope you will use their books when you need more information.

More Captain COMAL books are already in the works. **Watch for the announcement in next issue.** And if you are looking for someone to publish your COMAL book, contact us. **We pay the same royalties as the big publishers!**

◆◆

<u>May 1985</u>

#1 - **COMAL Workbook**
by Gordon Shigley
#2 - **COMAL 2.0 Packages**
by Jesse Knight
#3 - **COMAL Handbook**
by Len Lindsay
#4 - **COMAL From A To Z**
by Borge Christensen
#5 - **Cartridge Graphics and Sound**
by Captain COMAL's Friends

<u>June 1985</u>

#1 - **Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
#2 - **COMAL Handbook**
by Len Lindsay
#3 - **Cartridge Graphics and Sound**
by Captain COMAL's Friends
#4 - **Beginning COMAL**
by Borge Christensen
#5 - **COMAL Workbook**
by Gordon Shigley

<u>July 1985</u>

#1 - **COMAL From A To Z**
by Borge Christensen
#2 - **Cartridge Graphics and Sound**
by Captain COMAL's Friends
#3 - **Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
#4 - **COMAL Handbook**
by Len Lindsay
#5 - **Beginning COMAL**
by Borge Christensen

<u>August 1985</u>

#1 - **COMAL From A To Z**
by Borge Christensen
#2 - **COMAL Workbook**
by Gordon Shigley
#3 - **Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
#4 - **Cartridge Graphics and Sound**
by Captain COMAL's Friends
#5 - **COMAL Quick and Utilities #2**
compiled by Jesse Knight ■

The Joy of Trap

by Birrell Walsh

It would be fun to start a "What can you do with TRAP?" contest. I bet a lot of other people have discovered things to do with it.

Hmmm...maybe for prizes:

First Prize: A week on the set of Trapper John.

Second Prize: A week with the Trappist monks at Gethsemane, Kentucky.

Third Prize: Two weeks on the set of Trapper John.

The Joy of Trap
or
Who Was That Masked Array, Anyway?

plus some convenient statistical programs

by Birrell Walsh

It's always a joy to come back to COMAL after trying to do something in Pascal.

In this case, I was trying to do statistics. A lot of statistics involve the summing and averaging of a series of numbers. A series of numbers is most easily stuck into a one-dimensional array. That is how these series usually arrive --as an array.

So far, so good. But these arrays must be dimensioned. In Pascal, the dimension must be a constant, which can drive you mad. You must decide at the beginning of the program how long your series of numbers will be, and then stick with that maximum number throughout the program. Your procedures use that constant to set an upper bound for such things as FOR loops.

COMAL is much better, in that the DIMension can be a variable. That allows the program to ask the user, interactively through the terminal, how much space is needed and then to allot that amount. COMAL adds a whole new flexibility.

Alas, even in COMAL, one must dimension each global array only once. Once you make your choice, you're stuck with it.

The following programs circumvent the issue of array size (which is what the DIM statement sets) by using COMAL's TRAP...HANDLER...ENDTRAP device.

TRAP says that all the program errors that happen between TRAP and HANDLER will have a special handling. What is that special handling? Whatever you decide to put in the program between HANDLER and ENDTRAP.

If one programmer, George, is always trying to divide by zero, you can write this code:

```
INPUT divisor
TRAP
 PRINT 10/divisor
HANDLER
 PRINT "George, don't divide by zero!"
ENDTRAP
```

Now if there is an error between line 20 (TRAP) and line 40 (HANDLER), the HANDLER section will execute and George will get his message. It is almost certain that the only error that will occur in line 30 is division by zero, so the error message will be appropriate.

You can use the HANDLER sequence for more than personalized error messages. If you write code that asks for item (5), when the array only goes from item (1) to item (4), you get an error message:

More ►

at (linenumber): index out of range

Since this is an error, you can catch it in a TRAP.

Why would you want to? Because COMAL lets you pass arrays to procedures and functions without specifying the dimension of the array. You can say:

```
20 examine (items())
```

The procedure EXAMINE will then work on the array named ITEMS. It needs to know that ITEMS is an array; you tell it that with the empty parentheses following the variable name: ITEMS(). [More details about parameter passing appear in the section "Proc" of THE COMAL HANDBOOK, second edition, page 230.]

You find that to do much with the array ITEMS, you need to know its dimension. In Pascal, you would have to rewrite the procedure for each application. In COMAL, TRAP can be used to make array processors that discover the dimension of an array for you.

Function LAST, the first item in the listing, shows how. It sets a counter variable, COUNT, equal to zero. Then it starts working its way up the array from array(1). DUMMY is just that -- a meaningless operation that looks into the array element ARRAY(COUNT). All goes well until COUNT goes over the number actually allotted to ARRAY. Then trying to inspect ARRAY(COUNT) gives an error.

When the error happens, you know that COUNT is too high. By how much? By exactly 1, because if COUNT were 2 or more too high, it would have produced an error earlier. So the function returns COUNT-1 as the number of elements.

Are there traps in this use of TRAP to find the dimension of an array? Yes. The biggest trap is that an array may have a dimension statement that does not start with (1). For instance, an array that has the statement:

```
10 dim test(5:8)
```

will completely befuddle this routine. This is because the program assumes it is appropriate to start counting with (1).

You must be careful that you are working with a one-dimensional array. Versions of this program could be designed to measure the sizes of multidimensional arrays, but this function as listed works only on an array of one dimension.

You must be sure that the only kind of error that is going to happen within your TRAP...HANDLER section is the one you want to trap. If you don't, you may trap an error other than "**index out of range,**" and get a false reading.

But the advantage of this set of functions is that you can use the same functions, unmodified, on various numerical arrays without having any idea what their dimensions may be. I used it successfully on an array with 2 elements and one with 2000. The functions are transportable and make a nice library.

Anyone want to buy a used Pascal book?

[Ed. Note: Amazing. Like ESP, IBM PC COMAL includes LAST as a built-in function. It works with multi-dimension arrays and the starting index need not be 1. IBM PC COMAL also includes FIRST which returns the first index. MYTECH PC COMAL includes MAX'INDEX and MIN'INDEX.]

More ►

```
FUNC last(REF array()) CLOSED
  // Counts the elements in a one
  // dimensional array whose lowest
  // element is (1).
  TRAP
    count:=0
    LOOP
      count:+1
      dummy:=array(count)
    ENDLOOP
  HANDLER
    RETURN count-1
  ENDTRAP
ENDFUNC last
//
FUNC sigma(REF array()) CLOSED
  // Gives the sum of the elements of
  // any size one-dimensional array,
  // whose lowest element is
  // numbered (1).
  TRAP
    sum:=0
    count:=0
    LOOP
      count:+1
      sum:+array(count)
    ENDLOOP
  HANDLER
    RETURN sum
  ENDTRAP
ENDFUNC sigma
//
FUNC mean(REF array()) CLOSED
  IMPORT sigma,last
  // Returns the mean value of the
  // elements of any numerical
  // array whose lowest element
  // is numbered (1).
  RETURN sigma(array())/last(array())
ENDFUNC mean
//
FUNC sdev(REF array()) CLOSED
  IMPORT last,mean
  // Calculates the standard
  // deviation of the elements in
  // an array whose lowest element
  // is numbered (1).
  // SDEV calculates sample, not
  // population standard-deviation.
  n:=last(array())
  average:=mean(array())
  sum:=0
  FOR i:=1 TO n DO
    sum:+(array(i)-average) ↑ 2
  ENDFOR i
  RETURN SQR(sum/(n-1))
ENDFUNC sdev
//
FUNC rms(REF array()) CLOSED
  // Does a root-mean-square
  // calcuation on an array whose
  // lowest element is numbered (1).
  TRAP
    count:=0
    sum:=0
    LOOP
      count:+1
      sum:+SQR(array(count))
    ENDLOOP
  HANDLER
    RETURN (sum/(count-1)) ↑ 2
  ENDTRAP
ENDFUNC rms ■
```

TRACE THOSE BUGS WITH TRACE

COMAL 2.0 has a little used keyword that can be a big help in de-bugging your programs: **TRACE**. Whenever your program stops, for any reason - bugs, file problems, the STOP key, anything - just type in TRACE and COMAL will report which line the program stopped in. It will also list all the proc calls made to get to that line. Yes, you can even send the results to the printer like this: **TRACE "LP:"**.

Remember that **TRACE** may only be used in the immediate mode, not from a running program.

New Package

by David Stidolph

Two new commands in IBM COMAL are FIRST and LAST. The following COMAL 2.0 package for the Commodore 64 will emulate them.

FIRST'REAL, FIRST'INT, FIRST'STR

The functions FIRST'REAL, FIRST'INT, and FIRST'STR will return the number of the first element of an array. For instance, if you have DIMensioned an array:

```
DIM number(5:100)
```

In this example the first element is numbered 5. The function FIRST'REAL would be used to find this out, since NUMBER is an array of real numbers. If NUMBER had been NUMBER# (an integer array) then FIRST'INT would be used. FIRST'STR is used for NUMBER$ (a string array).

LAST'REAL, LAST'INT, and LAST'STR are functions used to find the number of the last element of an array. In the DIM statement above, the last element is numbered 100.

Each function is called in the same way, passing the array name as a parameter:

```
f:=first'real(number())
f:=first'int(number#())
f:=first'str(number$())
l:=last'real(number())
l:=last'int(number#())
l:=last'str(number$())
```

The name of this package is **FIRST'LAST**. Both the source code and package file are on TODAY DISK #9. Before you can use any of the functions in this package you must issue the command:

```
USE FIRST'LAST
```

■

Iconmaker

by Michael Mercurio

[Editor's Note: The program ICONMAKER is not listed here due to its length, and use of a machine code package. The program is on the back side of TODAY DISK #9.]

This project, like many before it, began small, but quickly mushroomed. The way COMAL 2.0 allows me to interface my machine code is nothing short of fantastic.

ICONMAKER is a small graphics editor in COMAL for creating graphics to be used in 'Print Shop'. Any 3 block long Print Shop file can be loaded, altered, saved under a new name, or created from scratch.

[Print Shop saves a 3 block file if a non-Commodore printer is selected. A two block file is saved if a 1525 printer is used. If you use a 1525 with Print Shop, your pictures will not be useable with ICONMAKER.]

A joystick or trackball is used in port 2, and H can be pressed anytime for a help screen. Three different images can be worked on at one time, or be blended together. The image being edited can be finely scrolled up, down, left or right, with a wrap mode similar to COMAL's WRAP command.

There is a FILL command with a YES/NO option afterwards that allows you to restore the screen to its state before the FILL. Also, MIRROR and INVERT commands change images to face other directions.

All of these commands execute with the speed of machine code, and make it a professional grade graphics editor.■

Using A Two Sided Disk

Will a double sided diskette, such as a COMAL TODAY Disk, work in the Commodore 1541 disk drive? We've received cards and letters from several new COMALites around the country who weren't too sure.

Last spring, we began distributing disks with programs written on BOTH sides. While this practice was viewed as a boon by most of the recipients of the disks, some wondered how the single sided 1541 disk drive could read a double sided diskette.

The 1541 cannot read a double sided disk, of course, without a bit of help. You, the user, must provide the help by flipping the diskette over. Here's how it works.

For an example, let's use the TODAY DISK #9. It has COMAL 0.14 programs written on the front side of the diskette. To use the programs, simply insert the diskette the usual way, with the label **up**, and follow the label's instructions for loading.

The tricky part comes when COMAL 2.0 cartridge owners want to use the 2.0 programs that are written on the back side of the diskette. The 1541 drive has only one **read/write head**, not two, so you **must** take the diskette out of the drive and **flip it over**. Then insert the diskette in the drive with the label **down**.

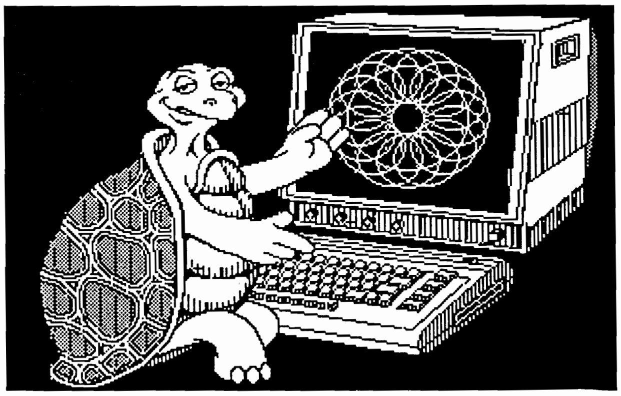
After you've done this, the 1541's read/write head can read the back side of the diskette. The 2.0 programs can then be loaded normally.

By recording programs on both sides of the disk you get twice as many programs as you would if we only used the front side. We use the back side merely to transport these extra programs to you. We use only special disks manufactured to be used on both sides by flipping the disk over.

We expect you to copy the programs on both sides to two other disks, and then file away the original disk as a Master.

This is what you should do with all disks you receive from us (and other software distributors). The disk you receive is considered your MASTER DISK. Never use a master disk. Use only copies of it. Then if anything happens to the disk, you need only make another copy. This is especially true with Commodore format disks, since the 1541, 4040, and MSD drives are read compatible - but NOT write compatible. Our master disks are created on a 4040 drive and are not copy protected.

Master disks should never, never be written on. To insure that you don't accidently write on one of your Masters, most of the disks we use have no write protect notch. That makes them **READ ONLY** disks. ■

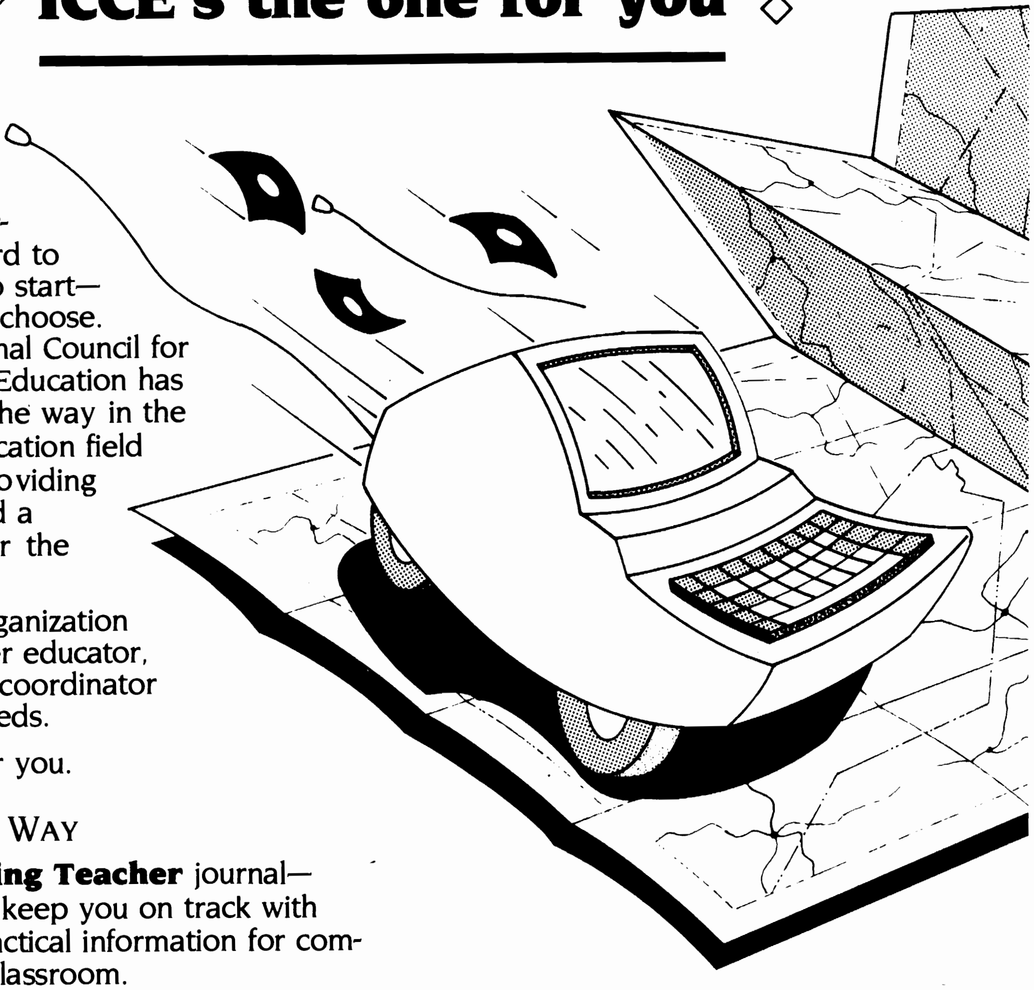


Calvin just made copies of his double sided Master disk.

Ed. Note: But where's the disk drive?

Take Off With Us

◇ ICCE's the one for you ◇



With today's profusion of computer information, it's hard to know where to start—which road to choose. The International Council for Computers in Education has been guiding the way in the computer education field since 1979, providing leadership and a ground plan for the future.

It's the one organization every computer educator, administrator, coordinator or librarian needs.

It's the one for you.

GUIDING THE WAY

The Computing Teacher journal—guaranteed to keep you on track with up-to-date, practical information for computers in the classroom.

Special Interest Groups—share information to help your special interest area grow. SIGs include computer coordinators, teacher educators, administrators and special educators, and are planned for advanced placement in computer science, community colleges and videodisc users. The quarterly **SIG Bulletin** serves as a forum for SIG information.

Booklets and Monographs point to additional information on specific topics. **ICCE Packets** provide you with teacher training materials. Members receive a 10% discount on all three.

ICCE Committees address a variety of ethical and practical issues important to you as a computer-using educator.

ICCE participates in computer education conferences throughout the world, supporting our state- and region-wide member organizations.

INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION ICCE

Join the One for You.

The Quick Change Sprite

by Daniel Horowitz

The **METAMORPHOSE** procedure allows you to change one sprite image into another as slowly or as quickly as you like, even while it is moving. Originally I thought it could be used in a game for a dramatic or funny effect, but after I got it to work I found myself running it over and over again, using different parameters, just to watch the transformations. It's kind of addictive!

The METAMORPHOSE procedure requires 12 parameters and calls another procedure, GET'IMAGE. The sprite definitions and the data statements (lines 410-840) are included only for demonstration. The real idea is to use it in a running program which already has sprite images defined and to change the image of one of those sprites into another (slowly...) after a collision, score, win, loss, disaster, or whatever. In that case, of course, most or all of the parameters would be variables determined by the state of the game at the moment the procedure was called. Another possible use of METAMORPHOSE would be simply as part of a program designed to do pretty, amusing, artistic, interesting things on the graphics screen. Try changing the values in line 650 or using other shapes. Here is how the procedure works:

Line 20: Sorry about the small variable names, but there was no room! The 12 parameters are:

O'IM(old image): the starting image definition number.

N'IM(new image): the ending image definition number.

SPD(speed): the delay between intermediate images. The higher the number, the slower the metamorphosis (must be at least 1).

STP(steps): the number of intermediate images (up to 63) between the old image and the new one. The more, the merrier (up to a point...)

SPR(sprite): the changing images are identified with this sprite (0-7).

COL(color): the sprites color (0-15).

XO,YO(x-old, y-old): the starting location

XN,YN(x-new, y-new): the final location. For no movement use the starting location values.

DUB'HGT,DUB'WID(double height, double width): expand sprite size? (true or false).

Lines 30 and 40: gets the old and new sprite images from the memory locations where they reside.

Line 50: in'between$ is the constantly changing sprite definition string. Originally it is the same as the old'sprite'image$.

Lines 80-100: makes the current image a little closer to the final image. It must go through this loop STP number of times before the image reaches it final shape.

Lines 120-150 and 300: show the sprite.

Lines 160-290: move sprite in equal steps.

Line 320: the pause between images.

More ►

```
0010 //save "metamorphose"//LONG line next
0020 proc metamorphose(o'im,n'im,spd,stp,s
     pr,col,xo,yo,xn,yn,dub'hgt,dub'wid)
0030  get'image(old'sprite'image$,o'im)
0040  get'image(new'sprite'image$,n'im)
0050  in'between$:=old'sprite'image$
0060  n=1; counter:=1; x'pos:=xo; y'pos:=yo
0070  repeat
0080   repeat
0090    in'between$(n)=new'sprite'image$(n)
0100    n:+stp
0110   until n>63
0120   define 31,in'between$
0130   identify spr,31
0140   spritecolor spr,col
0150   spritesize spr,dub'hgt,dub'wid
0160   if x'pos<>xn then
0170    x'increment:=int((xn-x'pos)/(stp+1-
        counter)) // wrap line
0180   else
0190    x'increment:=0
0200   endif
0210   if y'pos<>yn then
0220    y'increment:=int((yn-y'pos)/(stp+1-
        counter)) // wrap line
0230   else
0240    y'increment:=0
0250   endif
0260   x'pos:+x'increment; y'pos:+y'increme
       nt // wrap line
0270   if counter=stp then
0280    x'pos:=xn; y'pos:=yn
0290   endif
0300   spritepos spr,x'pos,y'pos
0310   counter:+1; n:=counter
0320   for delay:=1 to spd do null
0330  until counter=stp+1
0340 endproc metamorphose
0350 //
0360 proc get'image(ref old'or'new$,image'n
     umber) // wrap line
0370  for n:=0 to 63 do
0380   old'or'new$(n+1):=chr$(peek(49152+im
       age'number*64+n)) // wrap line
0390  endfor n
0400 endproc get'image
0410 //
0420 // create sprite images for this
0430 //        demonstration
0440 //
0450 dim old'sprite'image$ of 64
0460 dim new'sprite'image$ of 64
0470 dim in'between$ of 64
0480 background 0
0490 border 0
0500 setgraphic 0
0510 hideturtle
0520 spriteback 11,2
0530 dim image$ of 64, image2$ of 64
0540 //
0550 for i:=1 to 64 do
0560  read value
0570  image$(i):=chr$(value)
0580 endfor i
0590 for i:=1 to 64 do
0600  read value
0610  image2$(i):=chr$(value)
0620 endfor i
0630 define 0,image$
0640 define 1,image2$
0650 metamorphose(1,0,100,20,0,7,230,40,50,
     190,1,1) // wrap line
0660 metamorphose(0,1,100,20,0,7,50,190,230
     ,40,1,1) // wrap line
0670 // pitfall harry
0680 data 0,60,0,0,52,0,0
0690 data 20,0,0,16,0,0,40,0
0700 data 0,40,64,0,170,128,2,170
0710 data 0,1,40,0,0,60,0,0
0720 data 42,128,3,40,128,15,160,192
0730 data 0,0,240,0,0,0,0,0
0740 data 0,0,0,0,0,0,0,0
0750 data 0,0,0,0,0,0,0,0,1
0760 // tractor
0770 data 0,0,128,0,0,128,0,0
0780 data 128,12,0,128,12,0,170,63
0790 data 2,170,189,138,170,188,42,170
0800 data 191,170,106,170,169,90,170,165
0810 data 86,170,165,86,170,149,149,170
0820 data 150,165,150,22,165,85,22,165
0830 data 105,21,149,105,21,84,105,5
0840 data 84,85,1,80,20,0,64,1 ■
```

Function Keys for MSD Dual

by Colin Thompson

Here's a couple of tricks for those with an MSD dual disk drive and the COMAL cartridge. First, type:

```
USE SYSTEM
SHOWKEYS
```

Then cursor up and make the list of DEFKEY definitions look exactly as they appear at the end of this article. Hit RETURN on each line as you go.

The sequence of f1,f3,f5 will print drive zero's directory on your printer. For drive one, press f1,f4,f5.

f3 clears the screen and brings in the directory from drive 0.

f4 clears the screen and brings in the directory from drive 1.

f2 will print the DUPlicate command, but without the ""13"" to execute it.

f6 will display the key definitions.

f7, the RUN button, I left alone.

f8 is to be used only with a directory on the screen. Put the cursor on the line (of the directory) that has the file you want to copy. Press f8 and the file will be copied over to the formatted disk on drive one. It's a one button file copier.

Did you ever want to DELETE many files from a disk, but leave others? You can redefine the f6 key to make it an automatic DELETE key.

CAREFUL! One mistake here and you will lose some data.

```
defkey(6,"DELETE"13""11""13"")
```

Use this newly made DELETE key just as you would the COPY key or the RUN key: directly from the directory.

After you've finished DELETEing files with this key, remember to **return it to its previous setting.**

NOTE: The "13" in these lines is equivalent to hitting the RETURN key at that point. The "11" means erase to end of line.

FUNCTION KEYS FOR THE MSD DUAL DRIVE

```
defkey(1,"select""lp:"""13"")
defkey(2,"PASS ""d1=0""")
defkey(3,""147""13"dir""0:"""13"")
defkey(4,""147""13"dir""1:"""13"")
defkey(5,"select""ds:"""13"")
defkey(6,"showkeys"13"")
defkey(7,"RUN  "13""11""13"")
defkey(8,"copy "13",""1:*"13"") ■
```

SOLVE THOSE ROUNDING ERRORS

by Nicholas L. Seachord

A problem arises when working with floating point numbers and comparing two variables. After many calculations the numbers may become corrupted (sightly off), thus when you manually input a variable, it stays the same, but another one used in a calculation can change slightly. Then, when comparing numbers as in the case of "IF a=b THEN ...", a-b may actually be .0000007 instead of 0. The following function will eliminate any numbers beyond the hundredths place and works with either version of COMAL.

```
FUNC trunc(num)
  RETURN INT(num*100)/100
ENDFUNC trunc ■
```

Function Keys

by Mike Lawrence

Here is a simple way to save your own custom sets of function key definitions for the 2.0 cart:

1) Type: USE SYSTEM and hit return.
2) Type: SHOWKEYS and hit return.
3) Cursor up and make the changes you want to the key definitions.

Once you have your key definitions the way you like them, you can save them to disk by first directing output to a disk file. For example:

4) Type: SELECT OUTPUT "my'keys" <return>
5) Type: PRINT "USE SYSTEM" <return>
6) Type: SHOWKEYS <return>
7) Type: SELECT OUTPUT "ds:" <return>

Now, anytime you want to define the keys, just pop that disk in your drive, and type:

```
SELECT INPUT "my'keys" <return>
```

and your keys will be defined just the way you like them.

You can have any number of these batch files on the disk. Remember to use a filename that describes the keys so you can remember what they do. ■

MORE FUNCTION KEYS

by Mike Lawrence

The SELECT INPUT statement can be used in a COMAL program, but will not execute until you actually leave the program. This is fine as long as the batch file is the last thing you want your program to do, but suppose you want it to continue after calling the batch file? The answer is quite simple. In your program put:

```
SELECT INPUT "0:your'file"
STOP
```

And make the last line in your batch file read:

```
    CON
```

Then, as long as your batch file doesn't do anything that makes the program uncontinuable (like modifying the program) your program will continue after the batch file is executed.

FUNCTION KEYS REVISITED

COMAL can redefine the function keys using the DEFKEY command, but there is a limit of 32 characters allowed in the command string. Here is an example:

```
DEFKEY (1,"CAT")
```

What happens when you need more than 32 characters in the string? David Tamkin provides this (obvious) solution. (Well it seems obvious once you see it).

```
DEFKEY (1,"SELECT INPUT ""bat.fn""//")
```

When function key 1 is pressed, the command string is printed on the screen like this:

```
SELECT INPUT "bat.fn"//
```

When the return key is pressed, the batch file called BAT.FN is opened and the commands contained in the batch file are brought into the computer and executed.

This assumes you have prepared, in advance, a batch file called BAT.FN and have placed it on the disk. There is no limit to the number of commands a batch file may contain. The remark (//) at the end of the command string is included so it can be used at the beginning of any line. Any text on the screen following the // will be ignored. ■

1541 Disk Alignment Update

by Craig Van Degrift & Franco Pavase

Here is a more recent version of my COMAL 0.14 disk alignment program and a COMAL 2.0 version of that program (both are included on TODAY DISK #9).

The 0.14 version that you published must have migrated to you from one of our local users groups. It was improved to nearly the present form a day after I did an alignment demonstration at one of our users groups. The changes are minor: a clear command for the graph and improved help text.

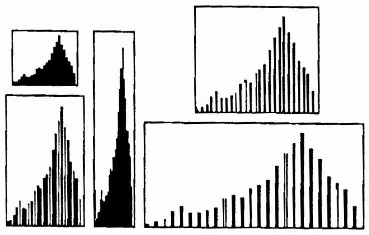
I wanted to add a few more whistles and bells to the COMAL 2.0 version before submitting it to you, but it is perhaps best to pass it on now. But if you and other COMALites keep me informed regarding additions and enhancements to the program and types of drive failures that confuse the alignment process, I could provide an improved version.

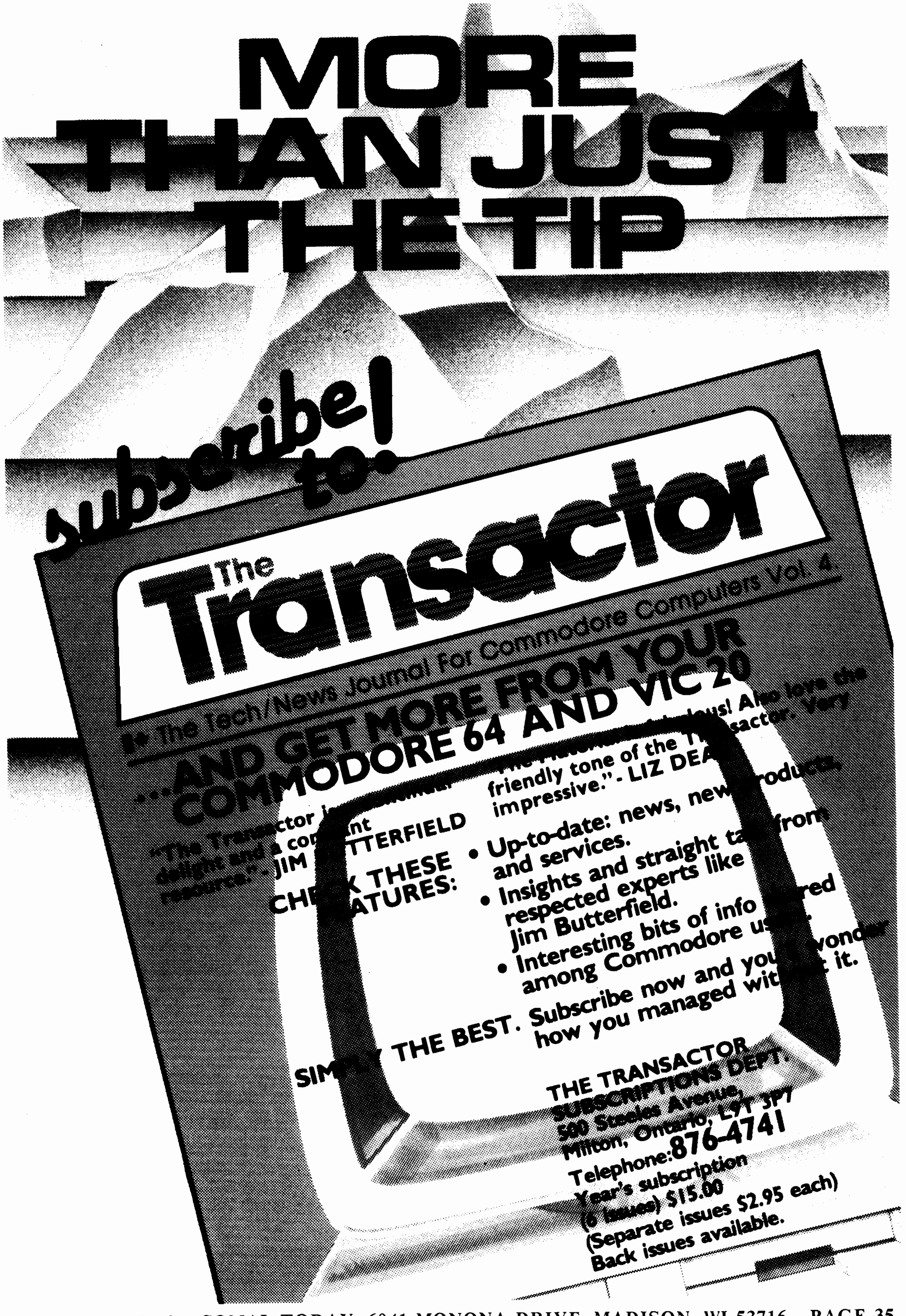
I especially want the by-line for 1541 DISK ALIGNMENT to include Franco Pavese, my collaborator in this endeavor since it was the combination of an alignment failure of my disk drive and his visit to my laboratory at the National Bureau of Standards that led to the fascinating idea of using the paddle ports as the measuring instrument during alignment. Franco wanted to use COMAL to monitor the waveform of the sound generator and I wrote a short machine code routine to transfer paddle data directly to the graphic screen.

Even after doing that, I didn't realize that the paddle inputs were actually A/D converters as Franco had thought, but rather expected that they were relaxation oscillators. Experiments revealed that Franco was right. We then determined the circuit necessary to monitor the head signal in the 1541 disk drive.

The most difficult part, however, was not the mechanics of manipulating the drive or monitoring the voltages, but rather the creation of the help advice. It is extremely difficult to write clear instructions! A most useful procedure was to find some brave soul with little electronics experience and a bad drive and quietly note his mistakes while trying to use the program. No instructions can be made fool-proof, but the final result is perhaps fool-resistant. I strongly urge this folklore to be passed on through demonstrations at user groups by fully competent electronics experts. It seems that only about a third of 1541 drive problems are cured by an alignment. Unfortunately, the person doing the alignment can be frustrated by an unreliable stepping motor or head.

Even in a polished form, the 0.14 version is somewhat awkward to use because of its CHAINing. When the drive is on the operating table, it is not always available to supply programs! An attempt was made to make chaining unnecessary during the actual surgery, but a better solution is to use the COMAL cartridge. With it's added space the entire program easily fits in memory at once. ■





...AND GET MORE FROM YOUR COMMODORE 64 AND VIC 20

"The Transactor i... delight and a con...nt resource." - JIM ...TTERFIELD

"...friendly tone of the T...sactor. Very impressive." - LIZ DEA...

CHE...K THESE ...ATURES:

- Up-to-date: news, new ...oducts, and services.
- Insights and straight ta... from respected experts like Jim Butterfield.
- Interesting bits of info ...red among Commodore us...

SIM... THE BEST. Subscribe now and you... wonder how you managed wit... it.

THE TRANSACTOR
SUBSCRIPTIONS DEPT.
500 Steeles Avenue,
Milton, Ontario, L9T 3P7
Telephone: 876-4741
Year's subscription
(6 issues) $15.00
(Separate issues $2.95 each)
Back issues available.

Detecting The Drive Type

by David Stidolph

Many programs ask the user what type of disk drive is being used. Now your program can automatically detect what type of drive is being used. The function DRIVE'TYPE returns one of three values:

1	means	MSD single disk drive
2	means	MSD dual disk drive
1541	means	Commodore 1541 disk drive

Any other type of disk drive (such as Indus) will return 1541. The function can be easily modified to return other values if necessary, such as 0 for the default 1541 drive. The function as listed works with COMAL 2.0. To use it with COMAL 0.14 simply omit the IMPORT line (which is printed in bold face).

```
FUNC drive'type CLOSED
  IMPORT disk'read,disk'write
  IF disk'read(49153)=46 THEN
    old'byte:=disk'read(20480)
    disk'write(20480,(old'byte+1) MOD 256)
    new'byte:=disk'read(20480)
    disk'write(20480,old'byte)
    IF old'byte=new'byte THEN
      RETURN 1
    ELSE
      RETURN 2
    ENDIF
  ELSE
    RETURN 1541
  ENDIF
ENDFUNC drive'type
//
FUNC disk'read(addr) CLOSED
  DIM com$ OF 20
  com$:="m-r"+CHR$(addr MOD 256)+CHR$(ad
  dr DIV 256)+CHR$(1) // wrap line
  PASS com$
  num:=ORD(STATUS$)
  RETURN num
ENDFUNC disk'read
PROC disk'write(addr,num) CLOSED
  DIM com$ OF 20
  com$:="m-w"+CHR$(addr MOD 256)+CHR$(ad
  dr DIV 256)+CHR$(1)+CHR$(num)//wraplne
  PASS com$
  num:=ORD(STATUS$)
ENDPROC disk'write ■
```

COMAL NIGHT ON PLAYNET

COMAL Users Group USA is using PlayNET as a COMAL support network. Every Thursday night, from 9:30 p.m. until midnight Eastern Time, you can sign on to discuss COMAL with others from around the country. It is hosted by Captain COMAL.

Two discussion rooms are available: COMAL and Beginning COMAL. Other COMAL rooms may open up as well, so check the listing of existing rooms to see the current set. For example, IBM COMAL was a room for part of one Thursday night.

COMAL BULLETIN BOARD

We also are maintaining a COMAL bulletin board within the PlayNET bulletin board system. You may post questions, answers, or COMAL information on this board. Captain COMAL periodically checks the board to answer any pending questions. Of course, anyone else may answer posted questions as well.

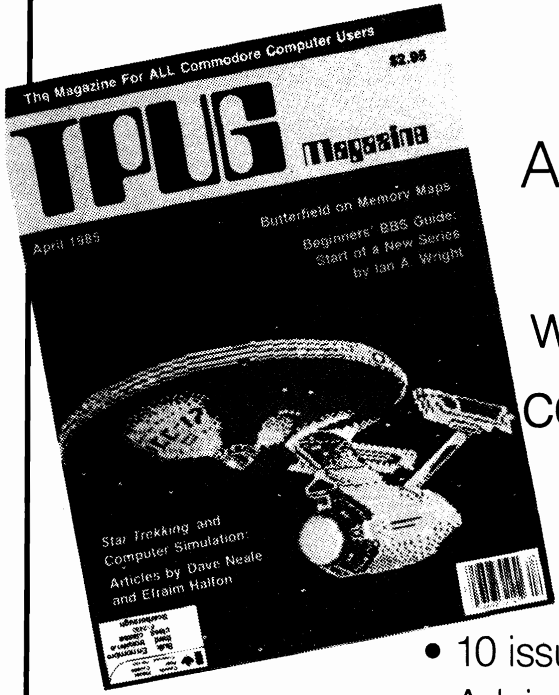
QUESTIONS?

Any questions about the above COMAL services can be directed to COMAL Users Group USA, Limited.

Any questions about the PlayNet system, billing information, or how to use PlayNET should be directed to PlayNET, Inc., at 1-800-752-9638. ■

TPUG

MORE THAN JUST ANOTHER COMPUTER MAGAZINE!


The Magazine For ALL Commodore Computer Users
$2.95
TPUG Magazine
April 1985
Butterfield on Memory Maps
Beginners' BBS Guide: Start of a New Series by Ian A. Wright
Star Trekking and Computer Simulation: Articles by Dave Neale and Efraim Halfon

The Magazine For ALL Commodore Computer Users
TPUG Magazine
TPUG Past, Present and Future
by Michael Bonnycastle, Louise Redgers and Garry Gold
May 1985
Fake Flight Sims Grounded by Ken Cox
Butterfield: In Praise Of VIC 20
WORLD of TPUG

A membership in the world's largest computer club will provide you with:

- 10 issues of TPUG magazine
- Advice from experts like Jim Butterfield and Elizabeth Deal
- Access to our huge library of public domain software
- Access to our online help services
- All for only $25/yr.

TPUG
101 Duncan Mill
Suite CT
Don Mills, Ontario
CANADA
M3B 1Z3

JOIN US NOW!!

I want to join TPUG as an associate member.

Name ______________________ Home phone (____) __________

Address ______________________ Work phone (____) __________

City/Town ______________________ ☐ Cheque enclosed

Prov/State ______________________ ☐ Money order enclosed

Postal/Zip Code __________ Country __________ ☐ Visa ☐ MasterCard

My computer equipment is: ______________________ Card# __________

______________________ Expiry __________

______________________ Signature __________

Bitmaps in Comal

by Colin Thompson

Bitmap graphics is an all encompassing term usually meaning the pixel by pixel representation of a picture on the computer's high resolution screen. Two kinds of pictures may be displayed on your C64's screen: color or black and white.

BITMAP GRAPHICS

If the colors were removed from the picture, we would be left with a black and white "sketch" which is called a BITMAP. You've seen printouts of these bitmaps adorn the cover of many issues of COMAL Today. Two disks full of bitmaps, called Slide Show #1 & 2 are available from the COMAL User Group. If you are a "collector" of public domain programs, you undoubtedly have some BASIC bitmaps on some of your disks.

Since thousands of bitmap pictures are available, let's look at how we could get the pictures off the diskette and onto the C64's hi-res screen using COMAL.

Programs 1 and 2 are written in COMAL 2.0, and will not work without the COMAL Cartridge.

Programs 3 and 4 are written in COMAL 0.14 and will only work with the disk-loaded COMAL.

The first three programs will get a standard bitmap picture from the disk and display it on the screen. Program 4 will convert a BASIC bitmap into a COMAL readable bitmap.

A "standard" bitmap picture as written on a disk, is a 32 or 33 block PRoGram file. The file's first two bytes are the Load Address. This is the place in memory where the picture will LOAD. It will correspond to the hi-res screen's address.

COMAL BITMAPS VS. BASIC BITMAPS

A COMAL bitmap has a Load Address of $E000, has 8002 bytes, is 32 blocks in length, and its filename ends in ".hrg".

A BASIC bitmap can have any one of four different Load Addresses, is usually 8192 bytes, is usually 33 blocks long and will not have ".hrg" in the filename.

Don't try to load a BASIC bitmap with programs 2 or 3. If you try it, your computer will probably crash because the extra 192 bytes will overwrite some vital system vectors at the top of memory.

Note: If you see a filename that BEGINS with "hrg.", it is a color COMAL picture, not a bitmap.

COMAL's screen is located at $E000, directly under the Kernal. This "hidden screen" presents some challenges to the programmer. In order to load a picture there, the kernal must be banked out in order to write the bitmap data in the blank 8K under it. Unfortunately, when the kernal is banked out, all communications with the disk drive are suspended.

Two solutions to the problem are presented below. The first solution is to get all 8000 bytes of data from the disk file and store it into a string variable. Once this is done, the bitmap data can be poked directly to the hi-res screen. Program 1 does this.

I've included line numbers for easy reference. Let's see how Program 1 works.

One package is needed, SYSTEM, so it is

More ►

called at the beginning of the program. Next the string MAP$ is DIMmed to hold the picture. Remember the Load Address bytes at the beginning of the disk file? DUMMY$ will get those two bytes and discard them. This lets us bring in any bitmap, whether it was made with COMAL or BASIC.

Line 60 calls the main PROC that does all the work. PROC load'bitmap begins by asking for the filename of the bitmap picture. Lines 110-150 set up the hi-res screen. All disk access is TRAPped for errors. That means that if an error occurs between the TRAP and HANDLER statements, the program flow will branch to line 320, and those instructions will be carried out.

The real work begins in line 180 where the file is opened. Next the load address bytes are taken from the file and discarded. Look closely at line 190. Those two bytes are retrieved by simply declaring DUMMY$ equal to two bytes taken in from the open file (5). That's all it takes to get data into a string.

The same technique is used in line 200 to retrieve the 8000 bytes of bitmap data. Line 200 takes 25 seconds to execute. Line 210 closes the open file.

We can POKE the bitmap data from map$ directly onto the screen. That's done in lines 250-270. This operation takes about 26 seconds. When complete, you will see the picture on the screen.

Now let's jump over the HANDLER statements to lines 370-380. These lines clear the keyboard buffer and wait for you to press any key. That will cause the textscreen to be displayed, and the program will END. Press the f5 key to see the picture again.

As mentioned earlier, this program will retrieve and display any bitmap, regardless of Load Address. (Remember that we took in those two bytes and threw them away). This whole process takes about 52 seconds, which is quite slow, but we still have the bitmap in memory, so we could now write it back to the disk with a Load Address that COMAL could use. ($E000, or 57344 Decimal).

If you wanted to write the bitmap back to the disk as a COMAL bitmap, just remove the remark (//) from line 70, before you run the program. That will cause the PROC write'comal'bitmap to be executed. Let's examine the PROC.

Lines 430-470 determine the length of the filename. If it is longer than 12 characters, it is truncated and the suffix ".hrg" is added. If not, the suffix is simply added.

Line 490 opens a WRITE file, with the new filename. Lines 500 and 510 write the new Load Address in lo byte/hi byte format. Line 520 writes the data that was stored in map$, then the file is closed.

All we have really done to the original file is to change the Load Address, change the filename, and limit the length of the file to 8002 bytes.

Once the picture has been rewritten back to the disk with this PROC, the picture is now an "official" COMAL bitmap, one that Programs 2 and 3 can read and display. That's because the filename has been renamed to have ".hrg" tacked onto the end, and the Load Address is now $E000.

More ►

```
 *******   PROGRAM ONE   ********
  (LOAD and display any bitmap)

0010 USE graphics
0020
0030 DIM map$ OF 8000
0040 DIM dummy$ OF 2
0050
0060 load'bitmap
0070 // write'comal'bitmap
0080
0090 PROC load'bitmap
0100   INPUT "Picture name? ": name$
0110   graphicscreen(0)
0120   pencolor(1)
0130   background(0)
0140   border(0)
0150   clearscreen
0160
0170   TRAP
0180     OPEN FILE (5),name$+",p",READ
0190     dummy$:=GET$(5,2)
0200     map$:=GET$(5,8000)
0210     CLOSE
0220
0230
0240
0250     FOR c#:=1 TO 8000 DO
0260       POKE $dfff+c#,ORD(map$(c#))
0270     ENDFOR c#
0280
0290
0300
0310   HANDLER
0320     PAGE
0330     PRINT AT 12,8: ERRTEXT$
0340     END
0350   ENDTRAP
0360
0370   WHILE KEY$<>CHR$(0) DO NULL
0380   WHILE KEY$=CHR$(0) DO NULL
0390   textscreen
0400 ENDPROC load'bitmap
0410
0420 PROC write'comal'bitmap
0430  IF LEN(name$)>12 THEN
0440   name$=name$(1:12)+".hrg"
0450  ELSE
0460   name$=name$+".hrg"
0470  ENDIF
0480
0490  OPEN FILE 5,name$+",p",WRITE
0500  PRINT FILE 5: CHR$(0),  // 00
0510  PRINT FILE 5: CHR$(224),// E0
0520  PRINT FILE 5: map$,
0530  CLOSE
0540 ENDPROC write'comal'bitmap
```

Programs 2 and 3 only work with bitmaps that have a load address of $E000. Program 2 looks very much like program 1, with two differences. Program 2 uses some machine code written by Phyrne Bacon to bring in the picture and display it. This PROC (load'obj) is called in line 130. This program only reads in and displays the picture, but does it in 24 seconds.

```
***** Program Two *********

0010 USE graphics
0020 load'bitmap
0030
0040 PROC load'bitmap
0050   INPUT "Picture name? ": name$
0060   graphicscreen(0)
0070   pencolor(1)
0080   background(0)
0090   border(0)
0100   clearscreen
0110
0120   TRAP
0130     load'obj(name$)
0140   HANDLER
0150     PAGE
0160     PRINT AT 12,8: ERRTEXT$
0170     END
0180   ENDTRAP
0190
```

More ►

```
0200   WHILE KEY$<>chr$(0) DO NULL
0210   WHILE KEY$=chr$(0) DO NULL
0220   textscreen
0230 ENDPROC load'bitmap
0240
0250 PROC load'obj(name$) CLOSED
0260
0270   POKE 858,169
0280   POKE 859,0
0290   POKE 860,76
0300   POKE 861,213
0310   POKE 862,255
0320   OPEN FILE 78,name$+",p",READ
0330   SYS 858
0340   CLOSE FILE 78
0350 ENDPROC load'obj
```

The PROC load'obj is COMAL's way to load a program file to the load address found in the first two bytes. In BASIC it would look like this:

```
LOAD"NAME",8,1
```

Now: let's do the same thing using COMAL 0.14. Once again we will ask for the picture's name and prepare the hi-res screen. Lines 110-140 are different because COMAL 0.14 isn't quite as smart as its big brother, the COMAL 2.0 Cart. Here we must inform color memory of our intention to write some white characters on a black screen. This FOR-ENDFOR loop need only be executed once. It simply draws some horizontal lines on the hi-res screen, and in doing so, prepares color memory. If this wasn't done, the picture would not be seen.

After the lines are drawn, the picture is loaded with the load'obj PROC. Line 220 is meant to be a comment line. The few people with COMAL 0.14 cartridges need to add the line.

```
****** Program Three ********

0010 dim name$ of 16
0020 load'bitmap
0030 //
0040 proc load'bitmap
0050  input "picture name? ": name$
0060  setgraphic (0)
0070  pencolor (1)
0080  background (0)
0090  border (0)
0100  hideturtle
0110  for i#:=0 to 192 step 8 do
0120   moveto 2,i#
0130   drawto 316,i#
0140  endfor i#
0150  load'obj(name$)
0160  while key$<>chr$(0) do null
0170  while key$=chr$(0) do null
0180  fullscreen
0190 endproc load'bitmap
0200 //
0210 proc load'obj(name$) closed
0220  // poke 157,0 // 0.14 cart only
0230  poke 858,169
0240  poke 859,0
0250  poke 860,76
0260  poke 861,213
0270  poke 862,255
0280  open file 78,name$+",p",read
0290  sys 858
0300  close file 78
0310 endproc load'obj
```

Program 1 converted any kind of standard bitmap into a COMAL bitmap, using the COMAL 2.0 Cart. Now let's do the same thing in COMAL 0.14. This is a little tricky, since 0.14 has only 9902 bytes of workspace and we will need 8000 of them to hold the bitmap in an array. That leaves very little room for our program.

Program 4 uses the disk'get PROCs to read the data into a string array. MAP$ has 40 elements of 200 bytes each. (40 times 200

More ►

equals 8000). This array is necessary because disk'get can get no more than 256 characters at once, so we tell it to get 200 characters forty times. Once the picture is in memory, you are asked to change disks, if you like, and then give the new file a different name. The WRITE file is opened using the new name and the Load Address is written. Then the array is written and the file is closed.

You will not see the picture on the screen. It takes one minute and 25 seconds to read in a picture and another 25 seconds to write it back.

Note: In lines 150 and 330, the "Q" at the end of the print statement is really a cursor up.

******* PROGRAM 4 *********

```
0010 dim map$(40) of 200
0020 dim name$ of 16
0030 read'write'bitmap
0040 proc read'write'bitmap
0050  print chr$(147)
0060  for q:=1 to 9 do print
0070  input "filename? ": name$
0080  open file 5,name$+",p",read
0090  disk'get'init
0100  file'end:=false
0110  disk'get'skip(2,5,file'end)
0120  for j:=1 to 40 do
0130   file'end:=false
0140   disk'get'string(map$(j),200,5,fi
       le'end) // wrap line
0150   print j*200;"bytes read inQ"
0160  endfor j
0170  close file 5
0180  file'end:=true
0190  print
0200  print "change disks now"
0210  print "new filename? (12 chars)"
0220  input "-> ": name$
0230  if len(name$)>12 then
0240   name$:=name$(1:12)+".hrg"
0250  else
0260   name$:=name$+".hrg"
0270  endif
0280  open file 5,name$+",p",write
0290  print file 5: chr$(0),    // 00
0300  print file 5: chr$(224), // E0
0310  for k:=1 to 40 do
0320   print file 5: map$(k),
0330   print k*200;"bytes writtenQ"
0340  endfor k
0350  close file 5
0360 endproc read'write'bitmap
0370 //
0380 proc disk'get(file'num,count,ref f
     ile'end) closed // wrap line
0390  poke 2025,file'num
0400  poke 2039,(count mod 256)
0410  sys 2024
0420  file'end:=peek(144)
0430 endproc disk'get
0440 //
0450 proc disk'get'skip(count,file'num,
     ref file'end) closed // wrap line
0460  disk'get(file'num,count,file'end)
0470 endproc disk'get'skip
0480 //
0490 proc disk'get'string(ref item$,cou
     nt,file'num,ref file'end) closed
     // wrap line
0500  item$:=""
0510  disk'get(file'num,count,file'end)
0520  for x#:=1 to count do item$(x#):=
      chr$(peek(1023+x#)) // wrap line
0530 endproc disk'get'string
0540 //
0550 proc disk'get'init closed
0560  for loc#:=2024 to 2045 do
0570   read v
0580   poke loc#,v
0590  endfor loc#
0600  data 162,0,32,198,255,162,0
0610  data 32,207,255,157,0,4,232
0620  data 224,0,208,245,32,204,255,96
0630 endproc disk'get'init ▣
```

Sizzle - Comal 0.14 Fastload

by Phil and Phyrne Bacon

If you have a Commodore 1541 disk drive, you can now load BOOT C64 COMAL and C64 COMAL 0.14 three times faster using our fastloader named ML.SIZZLE.

To fast load COMAL 0.14 with SIZZLE add the following lines to the beginning of BOOT C64 COMAL:

```
PRINT CHR$(147)
PRINT"LOADING ML.SIZZLE"
IF A=0 THEN A=1: LOAD"ML.SIZZLE",8,1
SYS 49152:CLR:PRINT CHR$(147);
```

Another way of making use of the SIZZLE routine is to type in the following BASIC program, save it to disk under the name FAST'BOOT and simply use it to boot up COMAL. The program makes sure that the disk drive is a 1541 (SIZZLE will not work with other drives such as MSD).

```
NEW

10 IF S=1 THEN 80
20 OPEN 15,8,15
30 PRINT#15,"M-R";CHR$(1);CHR$(192);
40 PRINT#15,CHR$(1)
50 GET#15,A$:D=ASC(A$+CHR$(0)):CLOSE15
60 IF D<>170 THEN 90
70 IF S=0 THEN S=1:LOAD"ML.SIZZLE",8,1
80 SYS 49152:CLR
90 LOAD"BOOT C64 COMAL",8

SAVE "FAST'BOOT",8
```

After you have altered the BASIC boot program (or added FAST'BOOT to the disk) you should add the following lines to the beginning of the COMAL HI program.

```
POKE 816,165
POKE 817,244
```

These two POKEs reset the LOAD vector to its usual COMAL setting. Otherwise, certain programs which use the LOAD'OBJ procedure could crash.

The modified BOOT C64 COMAL will load SIZZLE in five seconds, and COMAL in fifteen seconds. Using FAST'BOOT would add a couple seconds.

To use SIZZLE to load programs in BASIC, type:

```
LOAD"ML.SIZZLE",8,1
SYS 49152:NEW
```

and then load your program in the usual way, and it will load much faster.

ML.SIZZLE is a public domain machine language program (written by Phil Bacon) for fast loading PRG type files with the Commodore 1541 disk drive. Unlike many similar programs, SIZZLE does **not** blank the screen while loading programs.

SIZZLE occupies memory locations $C000 to $C0F6 and alters the LOAD vector at $0330 to point to SIZZLE instead of the standard LOAD routine in the Kernal ROM. To change the LOAD vector back to the usual BASIC and COMAL 0.14 value, use either SYS 49155 or the two POKEs above. SIZZLE will load C64 COMAL 0.14 quickly, but does not load COMAL programs any faster, because COMAL's LOAD command does not refer to the standard LOAD vector. SIZZLE can load pictures or other machine code faster though. Refer to the LOAD'SIZZLE procedures listed on page 46.

The following program will create the ML.SIZZLE file on the disk in the drive. Follow these steps:

More ►

1) Type in the program
   (or load it from TODAY DISK #9)
2) Insert the disk you want to put the
   ML.SIZZLE program on into the drive
3) RUN the program

If you have mistyped any of the DATA statements, the program will tell you.

```
// delete "0:create'sizzle"
//  by phil bacon
// save   "0:create'sizzle"
//
print chr$(147),chr$(14),
print "Checking DATA statements..."
total:=0
while not eod do
 read num
 total:+num
endwhile
if total<>107607 then
 print "Error in data statements"
 stop
endif
dim ds$ of 40
print "opening file, please wait"
open file 2,"ml.sizzle,prg",write
ds$:=status$
if ds$(1:2)<>"00" then
 print ds$
 close
 stop
endif
restore
while not eod do
 read num
 print file 2: chr$(num),
endwhile
close
end
//
// data statements for ml.sizzle
data 0,192,24,144,24,169,165,141
data 48,3,169,244,141,49,3,160
data 0,185,41,192,240,6,32,22
data 231,200,208,245,96,169,66,141
data 48,3,169,192,141,49,3,160
data 13,208,230,13,83,73,90,90
data 76,69,32,79,70,70,13,0
data 13,83,73,90,90,76,69,32
data 79,78,13,0,133,147,165,147
data 208,30,160,0,177,187,201,36
data 240,22,162,16,169,160,157,201
data 195,202,16,250,177,187,153,201
data 195,200,196,183,144,246,176,11
data 165,147,76,165,244,77,45,87
data 0,0,32,169,16,133,255,169
data 0,133,251,169,194,133,252,169
data 0,133,253,169,5,133,254,165
data 186,32,177,255,169,111,32,147
data 255,165,253,164,254,141,110,192
data 140,111,192,160,0,185,107,192
data 32,168,255,200,192,6,208,245
data 160,0,177,251,32,168,255,200
data 192,32,144,246,165,251,105,31
data 133,251,165,252,105,0,133,252
data 165,253,105,32,133,253,165,254
data 105,0,133,254,32,174,255,198
data 255,208,180,165,186,32,177,255
data 169,111,32,147,255,169,85,32
data 168,255,169,67,32,168,255,32
data 174,255,173,17,208,41,7,24
data 105,46,133,251,173,0,221,41
data 7,133,252,9,32,133,254,169
data 255,162,4,69,252,42,42,202
data 208,249,42,133,253,120,32,132
data 193,44,0,196,48,78,164,195
data 166,196,165,185,240,6,172,2
data 196,174,3,196,132,174,134,175
data 162,4,173,0,196,240,21,32
data 108,193,32,132,193,173,0,196
data 48,45,240,6,32,106,193,24
data 144,240,162,2,160,0,189,0
data 196,145,174,200,232,236,1,196
data 144,244,189,0,196,145,174,200
data 32,119,193,24,72,104,166,174
data 164,175,88,96,169,4,44,169
data 0,56,176,240,162,2,160,0
data 189,0,196,145,174,200,232,208
data 247,24,152,101,174,133,174,165
data 175,105,0,133,175,96,160,0
data 165,254,141,0,221,44,0,221
```

More ►

```
data 112,251,56,173,18,208,229,251
data 144,6,41,7,201,2,144,242
data 165,252,141,0,221,234,234,234
data 234,234,165,253,77,0,221,42
data 42,234,77,0,221,42,42,234
data 234,234,234,77,0,221,42,42
data 234,77,0,221,42,42,42,153
data 0,196,200,208,187,96,0,0
data 0,0,0,0,0,0,0,0
data 0,0,0,0,0,0,0,0
data 0,0,0,0,0,0,0,0
data 0,0,0,0,0,0,0,0
data 0,0,0,0,0,0,0,0
data 0,0,0,0,0,0,0,0
data 0,0,32,66,208,32,24,193
data 120,169,21,141,7,28,169,18
data 160,1,141,0,3,140,1,3
data 172,1,3,32,234,5,169,3
data 133,60,162,7,189,96,5,133
data 59,160,0,177,59,201,130,208
data 24,160,3,185,198,6,201,42
data 240,48,201,63,240,4,209,59
data 208,7,200,192,18,208,236,240
data 33,202,16,216,173,0,3,208
data 199,169,255,141,0,3,32,144
data 5,169,58,141,7,28,88,76
data 69,217,226,194,162,130,98,66
data 34,2,230,59,160,0,177,59
data 141,0,3,200,177,59,141,1
data 3,32,234,5,32,144,5,173
data 0,3,208,245,169,247,45,0
data 28,141,0,28,169,58,141,7
data 28,96,160,0,185,0,3,74
data 74,74,74,170,189,218,5,170
data 169,1,44,0,24,240,251,169
data 8,141,0,24,169,1,44,0
data 24,208,251,142,0,24,138,10
data 41,15,141,0,24,185,0,3
data 41,15,170,189,218,5,141,0
data 24,10,41,15,234,141,0,24
data 234,234,234,169,0,141,0,24
data 200,208,185,96,0,4,1,5
data 8,12,9,13,2,6,3,7
data 10,14,11,15,172,1,3,132
data 7,173,0,3,197,6,8,133
data 6,40,240,16,169,176,133,0
data 88,36,0,48,252,120,165,0
data 201,1,208,78,169,238,141,12
data 28,169,6,133,50,169,0,133
data 51,133,48,169,3,133,49,32
data 111,6,80,254,184,173,1,28
data 153,0,3,200,208,244,160,186
data 80,254,184,173,1,28,153,0
data 1,200,208,244,32,224,248,165
data 56,197,71,240,4,169,34,208
data 20,32,233,245,197,58,240,4
data 169,35,208,9,169,236,141,12
data 28,96,24,105,24,133,68,169
data 255,141,0,3,32,144,5,169
data 58,141,7,28,165,68,76,200
data 193,32,117,6,76,177,6,165
data 18,133,22,165,19,133,23,165
data 6,133,24,165,7,133,25,169
data 0,69,22,69,23,69,24,69
data 25,133,26,32,52,249,162,90
data 32,177,6,80,254,184,173,1
data 28,217,36,0,208,6,200,192
data 8,208,240,96,202,208,233,169
data 32,208,170,169,208,141,5,24
data 169,33,44,5,24,16,158,44
data 0,28,48,246,173,1,28,184
data 160,0,96,160,160,160,160,160
data 160,160,160,160,160,160,160,160
data 160,160,160 ■
```

BRIDGE, ANYONE?

by Bob Hoerter

If there are any bridge players out there I would like to know if you would be interested in working on a tutorial or a computer bridge game. As of now I have one that will deal out the deck into four hands, sort each hand by suit, calculate high card points and distributional points, and suggest which partnership might have a game or a slam bid. I can be contacted at:
92 Union St, Crystal Lake, IL 60014. ■

Fastload A Comal Bitmap Picture

by Phillip Bacon and Friends

The same machine code routine that fast loads COMAL can be used from COMAL 0.14 to fast load bitmap pictures or machine code object files. Since COMAL 0.14 will deactivate sizzle's machine code after the language is loaded, the following set of procedures can be used to load and activate the ml.sizzle file. The procedure LOAD'SIZZLE should be called at the beginning of the program, and does not need to be called again.

```
proc load'sizzle
 pass "m-r"+chr$(1)+chr$(192)+chr$(1)
 if ord(status$)=170 then
  if file'exists("ml.sizzle") then
   load'obj("ml.sizzle")
   sys 49152
  endif
 endif
endproc load'sizzle
//
func file'exists(file'name$) closed
 dim ds$ of 2
 open file 78,file'name$,read
 ds$:=status$
 close file 78
 if ds$="00" then
  return true
 else
  return false
 endif
endfunc file'exists
//
proc load'obj(name$) closed
 poke 858,169
 poke 859,0
 poke 860,76
 poke 861,213
 poke 862,255
 open file 78,name$+",p",read
 sys 858
 close file 78
endproc load'obj ■
```

Speedscript Fix

by Phyrne Bacon

Versions of SPEED'TO'SEQ and SEQ'TO'SPEED on TODAY DISK #8, have errors. Updated versions are on TODAY DISK #9.

In the 0.14 versions, these errors can be corrected by moving the line POKE'MACHINE'LANGUAGE to immediately before SYS 828. Thus, in each program the command POKE'MACHINE'LANGUAGE should be inside of the procedure CONVERT'FILE instead of near the beginning.

In the 2.0 version of SEQ'TO'SPEED, the error can be corrected by:

1) inside procedure OPENFILE delete line:

   MOUNT

2) add the line:

   MOUNT

   inside CONVERT'FILE before the line:

   OPENFILE(2,INPUTNAME$)

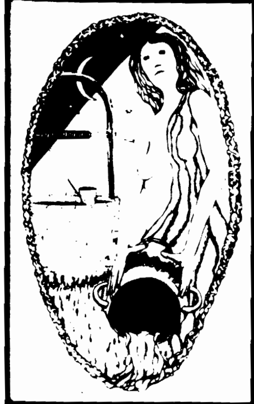
3) add the line:

   MOUNT

   inside CONVERT'FILE before the line:

   OPENFILE(3,OUTPUTNAME$)

In addition to the necessary corrections listed above, the revised programs also now accept 18 characters for <unit:><filename>. Also, the M/L parts have been changed to remove two extraneous bytes from the beginning of each sequential file. These bytes correspond to the load address of the Speedscript PRG type file. ■



Aquarian Software Introduces . . .

Powerful Programs at Affordable Prices

Mike J. Henry's

FAST BOOT!

- Works with **Both** 1541 and MSD Drives!
- Multiple loading options for increased compatibility.
- Disk-Based & Non-Protected

Only $14.95

Modem Master

The Fastest Commodore BBS Available!

- Works with 1541 or MSD Dual Drive
- 300/1200 Baud Operation
- New Punter File Transfer Protocol
- Fully Menu Driven
- Over 25 Commands
- Public Message Base
- Private E-Mail Base
- E-Mail Check at Sign-On
- Sub-Directories for File Transfer
- 250 User Capacity
- Accurate Clock/Calendar
- Printer Output
- Information Files
- "Old" E-Mail Deleted After One Week
- User Survey/Poll
- Set Up in Only 10 Minutes!

Only $29.95

Graphic Screen Exporter 64

NEW!

- A Universal Graphic Converter
- Converts Anything to Anything — Including:
  Koala Pad Flexidraw
  Doodle Print Shop
  And Many More!
- Complete On-Screen Editor

The Most Versatile Graphics Utility Ever Released for the Commodore 64 !

Only $29.95

The Cataloger

The Ultimate Disk Cataloging System for the 64!

Features of The Cataloger V3.5A Include:

★ Easy loading of information. Loads directly from the disk itself.
★ Ability to change name of entry.
★ Ability to change any information instantly — uses relative files exclusively.
★ Search, Sort and Print by any of 12 fields.
★ Capacity for 1100 different programs (or disks) per data disk.
★ FAST — all machine language.
★ Menu driven — very easy to use.
★ Works with ALL printers.
★ Works with one or two drives.
★ Allows duplicate ID's.
★ Allows selection of items to be entered.

Only $24.95

TURBO CALC/64

A Great Beginner's Spreadsheet At an UNBELIEVABLE Price!!

- 100% Memory Resident
- 100% Machine Code and P-Code
- 100% Menu Driven
- Over 13 K RAM for Data
- Large Matrix, 100 Rows by 26 Columns
- On-Line Help Screens
- On-Screen Menu at All Times
- Full Printer Support
- Built-in Functions (SUM, AVG, MIN, MAX)

At last, a powerful, yet affordable spreadsheet!

ONLY $17.95

I.E.A. Instant Editor Assembler

for the Commodore 64

★ Written 100% in machine language
★ Assembles 17K source code in 5 seconds!
★ Co-resident editor assembler monitor
★ Compatible with HES MON
★ Slow-motion code analyzer
★ Technical assistance number

"I would have to recommend the IEA Instant Editor Assembler, the BEST VALUE-FOR-PRICE development package on the market!"

Jim Gracely,
Technical Editor
Commodore Microcomputer Magazine

"IEA is a nice, simple assembler, good for beginners at an UNBELIEVABLE PRICE!"

Jim Butterfield

Now Only $17.95

Nova Decoder 64

Version 3.0

A 6502/6510 Disassembler Designed for the Commodore User

A MUST for anyone just learning Machine Language.
A TREAT for those examining the operation of the C-64.
A BLESSING to machine language programmers trying to de-bug their programs.

★ Choose Screen or Printer Options
★ Addresses Printed in Either Hex or Decimal
★ Complete Calculations of Branch Commands
★ Brief Explanations of Each Command Encountered
★ Generates Tables of Zero Page Addresses Used, Branch Addresses, All Jump Routines, and More . . .

Only $19.95

Disk-Lock V2.1

At Last! A professional copy protection system is available to YOU. Now you can protect your programs from most copy programs!

- Disk-Lock is Designed to Prevent Full Disk Copies.
- Stops Over 90% of All Copiers
- Offers Multiple Protection Schemes
- Place Protection on Any Track From 1 to 40!
- Fully Menu Driven

Only $39.95

Canadian Best-Sellers NOW Available in the U.S.

The Gold Disk Series

Quality Software for Your 64 at an Affordable Price!

Only $14.95 Per Disk*

* Plus Shipping and Handling

Each Disk Contains:

- The Feature Program
- A Home Business Program
- Two High-Quality Games
- Tutorials on Programming
- And MUCH MORE !!

Volume 10 Now Available — Featuring an Advanced Sound Synthesizer

Volumes 1 through 9 Are Also Available

Dealers Welcome ! — Call (503) 654-2641 For Details.

The Complete Thriller Collection

All six thrilling adventures now together on one disk! Plus — as a bonus — **Three Hours To Live!** Each is a full-length machine language text adventures that will take you weeks to complete. They feature full-sentence input (not just two words!) with color and sound.

Includes:

- Night of the Walking Dead
- Perils of Darkest Africa
- Revenge of The Moon Goddess
- Son of Ali Baba
- Frankenstein's Legacy
- Shipwrecked!
- Plus Three Hours to Live

Only $24.95
For All SEVEN !

Aquarian Software
P.O. Box 22184
Portland, OR 97222

To Order, Call: (503) 654-2641
VISA and MasterCard Accepted

Dealer Inquiries Welcome!

Add $3.00 S & H Per Order
Add $5.00 for COD
Canadian Orders: Add $10.00 S & H-U.S. Funds Only, Please.
Allow 4 to 5 weeks for delivery.

User Group Disk #10

All the programs on this disk are written in COMAL 0.14. The programs are not copyrighted, so you may give them to your friends. So many COMAL 0.14 programs have been written that we cannot publish all the good ones on the TODAY DISK series. The User Group disks have many programs that you may find fun or useful, that would not have been published otherwise.

Many of the programs on this disk require other programs or data files to operate. These 11 auxiliary files have been grouped together under the banner of:

```
---------------
  DATA FILES
---------------
 !DON'T LOAD!
---------------
```

Do not try to load, or use these files. There are five PROCedures and a FUNCtion on the disk. These are for programmers and should not be used by those without the skills necessary to use them.

The COMAL programs are separated into four catagories:

Applications, Utilities, Graphics Fun, and Plotter programs.

******* PROCEDURES/FUNCTIONS *******

Larry Phillips provided a complete set of PROCs to control the 1520 plotter, and also two programs that use them: Formatter and Sunflake. Larry also rewrote the DISK'GET routines and VAL.

COLR'BAR.PROC is from the Transactor, and TWO'TONE.PROC is simple noise that may be merged into your program.

********** APPLICATIONS ************

CODE'TRAINER, by Mike Erskine. Ham radio. You type in a message and hear it played back in dots and dashes.

DRUNKARDWALK, by H. Schlosser. A COMAL .14 version of the old problem.

INVOICER, by Harris Oliff. A simple invoice printer. Requires novice programming skills to change the company's name.

LINEAR'REGRESSION, by H. Schlosser. A curve fitting program for math wizards.

NURSERY'LIBS and RHYMING'SPELLER, by Judy Nahman. Simple educational programs for children.

PHONE'LOG, by Bruce Spell. Keep track of your phone calls and print the results. You must create a file called phone data for your own use. This file is on the disk as an example.

PULSE'RATE. Check your pulse with the computer.

SIEVE'ERATOSTHE, by Ed Cox. The old favorite benchmark.

SIM'EQUATIONS, by Ian MacPhedran. Solves simultaneous equations.

SUNDIAL. Calibrate your sundial.

TERMINAL.14, by Robert Shingledecker. This a unique protocol, and may be used for 64 to 64 communications. An instruction file called TERM.14 may be read with your wordprocessor.

More ►

*********** UTILITIES *************

FIND'LOAD'ADDR, by Valerie Kramer. Returns the LOAD ADDRESS of any program (PRG) file, in decimal.

LOAD'COLOR'SCRN, by David Stidolph and Colin Thompson. This demo program may be used by anyone, and will load a COMAL 2.0 color picture called HRG.AIRFORCE from the disk and display it on the Hi-Res screen. Programmers should read the remarks. The ML will also allow you to SAVE a color COMAL 0.14 screen in COMAL 2.0 SAVESCREEN format.

P1090'CHAR'EDTR, by Mike Erskine. This is a character (or FONT) editor that works with the Panasonic P1090 printer. You can generate a new character set for downloading into the printer. This may work on other Epson compatable printers, also.

MIRROW'WRITER, by Mike Erskine. This will download the font MIRROW.DAT to the printer. For use with the P1090'CHAR'EDTR.

PRINT'SHOW.NEC, by Colin Thompson. A modified version of the Slide Show software. This will print all the pictures on a Slide Show disk to an NEC 8023A or Prowriter printer. Programmers may easily adapt their own screen dump routines to the program. Requires " dump.nec". A Seq file, " directory", containing the filenames of the pictures on the disk must be on the picture disk.

********** GRAPHICS FUN ***********

SUNFLAKE, **YARN'ART'3**, and **ART** are turtle graphics programs.

DOTS.GAME, by Craig Veiner. A one or two player game for all ages. If you play against the computer, expect to win. This is a "complete the box" game, well done.

QUEENS, By UniCOMAL Aps. A COMAL 0.14 version of the famous math problem that finds all the solutions. The object is to place 8 Queens on a chessboard and have none of them in check. This is pretty to watch, even if you don't know what's happening. Complete running time: 2 hours+.

KOALA'DOODLER, by Len Lindsay. Plug in your Koala Pad and have some COMAL fun!

SPIROGRAPH, by Bill Raecke. This is the best COMAL version of programs like Spirolizer. Very well done, and much better than the original idea.

SON'O'SPIROGRAPH, by Bill Raecke. This is an adaptation of Jesse Knight's ART program and uses Kevin Quiggle's Gutenberg lettering program.

****** 1520 PLOTTER PROGRAMS *******

FORMATTER.1520, by Larry Phillips. This is the standard Formatter program, with the output directed to a 1520 plotter. You must LIST a program to the disk for this program to read and print it.

SUNFLAKE.1520, by Larry Phillips. Plots a pretty pattern on the plotter. ■

COMAL
USERS GROUP
USA

Comal 2.0 Internal Structure

by Ian McPhedron

Those who have read John H. McCoy's excellent article in COMAL Today #5 know that COMAL 0.14 stores its programs in tokenized form. So, it will come as no great surprise that COMAL 2.0 also stores programs as tokenized files.

For those unfamiliar with tokens, **a token is a code that COMAL uses to identify keywords**. This code is only one or two bytes long. As most COMAL keywords are much longer than this, it can be easily seen that this is a memory saving feature.

Each line entered from the keyboard (or from disk via ENTER) is broken down into tokens and stored sequentially. The line, as stored in memory, is composed of two parts: the header and the body.

The header is 3 bytes long. The first two bytes are the line number in a high byte/ low byte format. It is interesting to note that COMAL 2.0 stores its line numbers as 10000 plus the displayed program line number (i.e. line 10 of the program is stored as 10010). Changing these line numbers causes some interesting things to occur. Renumbering the internal lines below 10000, or above 19999 causes the program to disappear! It will not be displayed when listed. This can be used to hide certain lines from prying eyes. Note that a RENUM command will bring back lines numbered over 19999. The third byte of the header is the length of the line. This length is the total length of the line (body and header) in bytes. As the length is represented by one byte, the length cannot exceed 255 bytes (including the header).

The body of the line is composed of tokenized keywords, constants and names.

As Mr. McCoy has noted, the internal structure of COMAL is based on a Reverse Polish Logic notation. This is the notation used by efficient, stack orientated languages such as FORTH, as well as some electronic calculators. In this notation, the operands are entered first, then the operations. For example:

```
2+3-4 is entered as 2 3 + 4 -
SIN(PI) is entered as PI SIN
LEN(a$) is entered as a$ LEN
TRUE OR FALSE is entered as TRUE FALSE OR
```

The keywords used by COMAL are broken into one or two byte tokens according to the formula presented in the list at the end of this article. In two byte tokens, the first byte (a flag byte) is $FF.

COMAL also goes to the trouble of "tokenizing" all constants and variables, as well as keywords. This tokenization is not the same as for keywords, but is used for the same purpose, lessening the memory usage and increasing speed. Numerical constants less than or equal to 255 are stored as 1 byte, preceeded by the token $CE (hex). Numbers greater than 255, but less than 32768 are stored as a two byte integer preceeded by the token $02. Numbers larger than this, or numbers with a fractional part, are stored as floating point numbers (5 bytes long) preceeded by $01. String constants one byte long are preceeded by a token $CD, and if longer, they are preceeded by the token $03 and two other bytes representing the length.

Names are represented by their position in the variable table. This position is given as two bytes (lo byte, hi byte form) which are the offset from the beginning of the variable table to the entry for that variable name. This applies to all names, whether they are for variables, PROCs,

More ►

FUNCs, or packages. A more detailed description of the variable table will be given later. The start of this table is pointed to by the vector at $18).

Structured statements have internal pointers. These usually point to the next portion of the structure. This speeds up the process of finding the end of a conditional block.

Now that the basic rules have been set out, two examples will be given. The LISTed COMAL line is shown first, then the tokenized line below it:

```
1000 DIM a$ OF 20
2A F8 0B 8F 00 00 00 CE 14 93 8C

1010 b:=$14 // $14=20
2B 02 11 D9 00 14 FA 05 00
      00 20 24 31 34 3d 32 30

1020 FOR i#:=1 TO b DO
2B 0C 10 83 0A 00 46 08 CE 01 84
      04 05 00  85 87

1030  a$(i#):=CHR$(RND(32,127))
2B 16 12 0C 00 00 08 0A 00 1D CE
      20 CE 7F 55 54 4B 3C

1040 ENDFOR
2B 20 08 8B 0A 00 2C 08

1050 PRINT a$
2B 2A 09 5F 06 00 00 65 60
2B 2A 00 9F
```

This example demonstrates the type of tokens COMAL uses. The tables following this article show some of the tokens used by COMAL 2.0. NOTE: much of the structure of COMAL 2.0 is the same as that used in version 0.14. Many of the same tokens are also used.

The lines translate as follows:

```
 LINE 1000
2A F8 line number (11000)
0B    line length (bytes)
8F    DIM (for string variable)
00 00 variable a$ (zero byte offset)
00    no dimensions in array
CE 14 one byte constant (decimal 20)
93    string length
8C    end of dimensioning

 LINE 1010
2B 02 line number
11    line length
D9 00 14 hexadecimal constant $14
FA    assign to real variable
05 00 variable b (offset five bytes)
00    remark
20 24 31 34 3d 32 30
      text: " $14=20"

 LINE 1020
2B 0C line number
10    line length
83    FOR (integer variable)
0a 00 variable i# (ten byte offset)
46 08 address of ENDFOR ($0846)
CE 01 one byte constant (1)
84    FROM
04    real source
05 00 variable b
85    TO
87    DO

 LINE 1030
2B 16 line number
12    line length
0C    string array destination
00 00 variable a$
08    integer source
0A 00 variable i#
1D    substring parenethes
CE 20 one byte constant (32)
CE 7F one byte constant (127)
55    comma for RND
```

More ►

```
54    RND (2 parameter) function
4B    CHR$ function
3C    assign to string array
```

LINE 1040

```
2B 20 line number
08    length
8B    ENDFOR integer variable
0A 00 variable i#
2C 08 address of FOR ($082C)
```

LINE 1050

```
2B 2A line number
09    length
5F    start of PRINT
00 00 variable a$
65    PRINT string
60    PRINT carriage return
```

HIDDEN LINE

```
2B 2A line number (same as last line)
00    last line (zero length)
9F    END
```

The next example is a PROC. While this demo does nothing, it shows the basic structure of the PROC (or FUNC) header.

0010 PROC a(b) CLOSED

```
27 1A 13 70 00 00 19 08 00 00 01
      72 05 00 7F 00 00 00 00
```

0020 ENDPROC a

```
27 24 06 0E 00 00
```

The lines translate as follows:

LINE 0010

```
27 1A Line number 10010
13    Line length (19 bytes)
70    PROC token
00 00 PROC name (offset)
19 08 Address of 1st after ENDPROC
00 00 Purpose unknown
01    Number of parameters
72    Parameter type (real)
05 00 Parameter name (offset)
7F    CLOSED token (if open 7E)
00 00 Address of start of first
      embedded PROC
00 00 Purpose unknown (only in
      a CLOSED PROC)
```

LINE 0020

```
27 24 Line number 10020
06    Line length (6 bytes)
0E    ENDPROC
00 00 PROC name (offset)
```

The variable table is immediately after the program storage. This table contains information about each name used in a program. It will also contain names created by spelling errors, so it is a good idea to clear out the name table by LISTing the program to disk, and then ENTERing it again before SAVEing the final version.

The variable table is organized as a series of separate descriptions for each name. These have the form below:

1) a one byte length for the description

2) one byte signifying the type of name - this could be one of the following:

$10	real variable
$11	integer variable
$12	string variable
$13	label
$14	PROC
$15	real FUNC
$16	integer FUNC
$17	string FUNC
$18	package (as in USE package)
$90	real array
$91	integer array
$92	string array

More ►

3) a two byte address to the start of the field reserved for the contents of the variable (or start of PROc, etc.) this is in a lo byte / hi byte format.

4) a series of bytes consisting of the PETASCII representation of the name.

I hope that this article has helped some people to become more aware of what COMAL does to their programs. I also hope that the following tables are of some help to programmers. Experimentation is the only way to become familiar with these tokens. It would be advisable for the readers to acquire a copy of SMON.COMAL which is on the disk accompanying Jesse Knight's superb book, COMAL 2.0 Packages.

REFERENCES
1) The COMAL Handbook, Second Ed., by Len Lindsay, Reston Publishing, 1984
2) "How COMAL Statements are Stored", by John H. McCoy, COMAL Today #5, COMAL Users Group USA, December 1984
3) COMAL 2.0 Packages, by Jesse Knight, COMAL Users Group USA, 1985
4) "Another Look at COMAL 0.14 Tokens", by Ian MacPhedran, COMAL Today #6, COMAL Users Group USA, March 1985 ■

CANADIANS -- PLEASE NOTE

When you order COMAL-related products, please pay with US currency. Make sure that your cheques or Postal Money Orders are payable in US dollars.

FOREIGN

We accept orders from outside North America only if paid in US Dollars and extra shipping is added. Add $30 per 6 issue subscription, $3 per disk, $10 per book. VISA / MasterCard are OK. ■

Symbolic Differentiation

by Tom Kuiper

Cartridge Demo Disk #2 has a program which, for some readers, is by itself reason enough to buy the cartridge. (You mean you don't have one yet?) The program is called DIFFERENTIATION. It does symbolic differentiation of functions coded in "computerese". For example, the derivative of sin(x) is cos(x). For that you don't need a computer. But how about the derivative of exp(-4*ln(2)*((x-a)/w)!2) with respect to w? (For ! read up-arrow) Indeed, that would take me 20 minutes of algebraic scribbling to get it wrong!

I've modified the original program to facilitate interactive use. College freshmen should NOT use it to do their calculus homework. (You might have to do a derivative sometime without your Commodore. Learn how!) The program comes up with an instruction screen. After you hit any key, you will be asked to input a function. Use only single letter lower case variables. Next you are asked for the variable with respect to which you want to differentiate. The result is displayed shortly. The cursor is positioned back at the variable name input. You can specify the variable with which the RESULT is to be differentiated. You can repeat this until the result string overflows. If you respond with a blank, then the cursor goes back up for another function input. You might redo your original function with respect to another variable, edit your original function, or type in a completely new one. If you enter a blank line the progam ends. ■

Comal 2.0 Token Table

COMAL 2.0 TOKENS
compiled by Ian MacPhedran

Hex value:	
00	Remark (//)
01	Real Constant
02	Integer Constant
03	String Constant
04	Real Source
05	Integer Source
06	String Source
07	Statement Separator (;) Also Real Destination
08	Integer Destination
09	String Destination
0A	
0B	
0C	
0D	
0E	ENDPROC
0F	
10	
11	
12	
13	
14	RETURN string
15	
16	
17	
18	
19	Substring Parentheses
1A	Array Destination Parentheses
1B	
1C	
1D	String array parentheses
1E	
1F	Substring Colon
20	Positive Flag (+)
21	Negative Flag (-)
22	Exponentiate (
23	Division (/)
24	Multplication (*)
25	DIV
26	MOD
27	Addition (+)
28	String Concatination (+)
29	Subtraction (-)
2A	Numerical LessThan (<)
2B	String LessThan (<)
2C	Numerical Equal (=)
2D	String Equal (=)
2E	Numerical LessEqual (<=)
2F	String LessEqual (<=)
30	Numerical GreaterThan (>)
31	String GreaterThan (>)
32	Numerical NotEqual (<>)
33	String NotEqual (<>)
34	Numerical GreaterEqual (>=)
35	String GreaterEqual (>=)
36	IN
37	NOT
38	AND
39	OR
3A	
3B	
3C	assign string matrix
3D	AssignAdd Real (:+)
3E	AssignAdd Integer (:+)
3F	AssignAdd String (:+)
40	AssignSub Real (:-)
41	AssignSub Integer (:-)
42	
43	TRUE
44	FALSE
45	ZONE Read
46	ZONE Set
47	Parentheses ()
48	ABS
49	ORD
4A	ATN
4B	CHR$
4C	COS
4D	ESC
4E	EXP
4F	INT
50	
51	LEN
52	LOG
53	RND
54	RND (2 numbers)
55	function comma
56	SGN

More ►

COMAL 2.0 Token Table - continued

57	SIN
58	SPC$
59	SQR
5A	TAN
5B	TIME
5C	EOD
5D	EOF
5E	ERRFILE
5F	PRINT start
60	PRINT carriage return
61	PRINT zone
62	USING
63	TAB
64	PRINT number
65	PRINT string
66	
67	PRINT space
68	IF
69	THEN
6A	THEN (one line)
6B	LOOP
6C	EXIT
6D	ELIF
6E	ELSE
6F	ENDIF
70	PROC
71	NULL
72	real parameter
73	integer parameter
74	string parameter
75	REF real parm
76	REF integer parm
77	REF string parm
78	real matrix parm
79	integer matrix parm
7A	string matrix parm
7B	REF real mat parm
7C	REF integer mat parm
7D	REF string mat parm
7E	unCLOSED proc/func
7F	CLOSED proc/func
80	
81	EXEC
82	FOR real
83	FOR integer
84	(FOR) from
85	(FOR) TO
86	(FOR) STEP
87	(FOR) DO
88	(FOR) DO (one line)
89	ENDFOR (one line)
8A	ENDFOR real
8B	ENDFOR integer
8C	DIM
8D	DIM real
8E	DIM integer
8F	DIM string
90	DIM colon
91	DIM comma
92	DIM parentheses
93	DIM string length
94	DIM separate variables or end
95	REPEAT
96	REPEAT end
97	WHILE
98	(WHILE) DO
99	(WHILE) DO (one line)
9A	ENDWHILE (one line)
9B	ENDWHILE
9C	LABEL
9D	GOTO
9E	ENDLOOP
9F	END
A0	STOP
A1	CASE
A2	(CASE) OF
A3	
A4	WHEN
A5	
A6	CASE separator
A7	
A8	OTHERWISE
A9	ENDCASE
AA	DATA start
AB	DATA end
AC	
AD	
AE	
AF	READ start
B0	READ real
B1	READ integer
B2	READ string

More ►

COMAL 2.0 Token Table - continued

B3	READ end
B4	RESTORE
B5	INPUT start
B6	(ELIF) THEN
B7	INPUT prompt string
B8	INPUT real
B9	INPUT integer
BA	INPUT string
BB	INPUT end
BC	
BD	SELECT
BE	OUTPUT
BF	TRAP
C0	ESC
C1	Enable Trap Stop (+)
C2	Disable Trap Stop (-)
C3	WRITE
C4	WRITE number
C5	
C6	WRITE string
C7	
C8	(FILE I/O) offset
C9	(FILE I/O) record
CA	(FILE I/O) file
CB	
CC	(Select) INPUT
CD	1 Byte String Constant
CE	1 Byte Integer Constant
CF	Blank Line
D0	(When) EXIT
D1	WHEN (Exit)
D2	AND (Then)
D3	
D4	USE
D5	GET$
D6	2 Byte Hexadecimal Constant
D7	
D8	
D9	
DA	UNTIL
DB	INTERRUPT
DC	(Print) AT
DD	
DE	STATUS$
DF	RETURN real
E0	RETURN integer
E1	RETURN
E2	
E3	FUNC real
E4	FUNC integer
E5	FUNC string
E6	ENDFUNC real
E7	ENDFUNC integer
E8	ENDFUNC string
E9	VAL
EA	STR$
EB	INPORT
EC	BITAND
ED	BITOR
EE	BITXOR
EF	(IMPORT) end
F0	RESTORE name
F1	(And) THEN
F2	OR (Else)
F3	CURSOR
F4	(Or) ELSE
F5	
F6	
F7	
F8	Assign String (:=)
F9	LOOP
FA	Assign Real (:=)
FB	Assign Integer (:=)
FC	HANDLER
FD	
FE	ERR
FF	**token flag**

PARTIAL TABLE OF 2.0 TOKENS WITH $FF FLAG

FF + 00
FF + 01
FF + 02
FF + 03
FF + 04
FF + 05
FF + 06
FF + 07
FF + 08
FF + 09
FF + 0A
FF + 0B

More ►

COMAL 2.0 Token Table - continued

FF + 0C	
FF + 0D	
FF + 0E	
FF + 0F	
FF + 10	
FF + 11	
FF + 12	
FF + 13	
FF + 14	PI
FF + 15	PAGE
FF + 16	
FF + 17	ERRTEXT$
FF + 18	CLOSE
FF + 19	CLOSE file
FF + 1A	
FF + 1B	(OPEN) file
FF + 1C	CHAIN
FF + 1D	
FF + 1E	UNIT set
FF + 1F	KEY$
FF + 20	OPEN
FF + 21	
FF + 22	(OPEN) file with descriptors
FF + 23	READ descriptor
FF + 24	WRITE descriptor
FF + 25	APPEND descriptor
FF + 26	RANDOM descriptor
FF + 27	DIR
FF + 28	UNIT$ read
FF + 29	COPY
FF + 2A	
FF + 2B	POKE
FF + 2C	
FF + 2D	SYS
FF + 2E	
FF + 2F	
FF + 30	STOP with string
FF + 31	set TIME
FF + 32	
FF + 33	
FF + 34	
FF + 35	
FF + 36	MOUNT
FF + 37	
FF + 38	END with string
FF + 39	DELETE
FF + 3A	
FF + 3B	RANDOMIZE
FF + 3C	REPORT
FF + 3D	CREATE
FF + 3E	RENAME
FF + 3F ■	

LINKUP WITH PEOPLE/LINK

COMAL can now be found on another communications subscription service. People/Link has expanded its operation to include a COMAL section. The following announcement, received from Mr. Merrill Millman, President of People/Link explains what is available.

There will now be a COMAL club up and running twenty-four hours a day on American People/Link. The club will have fifteen sections with notice boards and upload/download libraries plus our own special conference area, where COMALites can meet and talk on line whenever they are in the club.

People/Link is a communications network accessible by all brands of computers and modems through Telenet and Tymnet, and you can call with any standard terminal program. You don't need special software.

If you are already on People/Link, send a letter there to user-ID JACQUARD to join the club. If you aren't you can sign up by calling People/Link Offices at 1-800-524-0100 (in Illinois call 1-312-870-5200). Tell them you are going to join the COMAL club and you can sign up for only $14.95 and get three free hours of connect time. ■

Learn Comal 2.0 with Rod The Roadman

by Borge Christensen

Ed. Note: Borge has created his own small universe with his ROD THE ROADMAN program. The program may be used by beginners quite easily. Since it uses a large grid and a visible robot (named ROD), it is ideal to use with a group of students. In addition to providing us with one of our best programs, Borge has thrown us a curve. His universe is created by a program that you can't list. But it allows you to merge your own additions to it! This is a fantastic feature. Now when you give the LIST command, you only will see the lines that YOU added. They are not mixed in with the lines that create the universe.

"How did Borge do that?", you may ask. Well, the answer is quite REVEALing. We leave you with that challenge. We'll even give you another hint: the procedure PROC.SHOWNAMES is very helpful. Next issue we will REVEAL the solution to you.

We think you can have hours of fun working with ROD THE ROADMAN. All you need is the COMAL 2.0 Cartridge and our TODAY DISK 9. Teachers and parents may find the system to be exactly what they were looking for. And why not? Borge is a teacher as well as the father of COMAL.

If you come up with problem screens you would like to share with our readers, send them in on a disk. We will collect them and make them available to others in the future. Watch future issues for an announcement.

Now, Borge tells us about ROD and his universe:

1. ROD AND HIS TOWN

Figure A shows a map of Rod's town. The horizontal lines are called roads, and the vertical ones are called streets. There are 10 roads and 15 streets. Rod is standing where 2nd street is crossing 2nd road. You can only see the peak of his cap. In Figure B, below, you can see Rod and the things in his world.

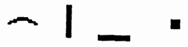


Figure B: Rod, Roadwall, Streetwall, Beeper

Then in Figure C you see what Rod's world may look like, when he has a house of his own built from streetwalls and roadwalls. Outside the house is a beeper. Rod is standing inside the house.

To get the computer to display Rod the Roadman and his world, insert the back side of TODAY DISK 9 into the drive and from COMAL 2.0 type:

```
RUN "ROD"
```

It takes a little while before Rod is ready. Then you get a picture like in Figure A. Now type these commands:

```
move
move
rt
move
lt
move
```

Here is a little explanation about the commands. As soon as Rod reads this one:

```
move
```

More ►

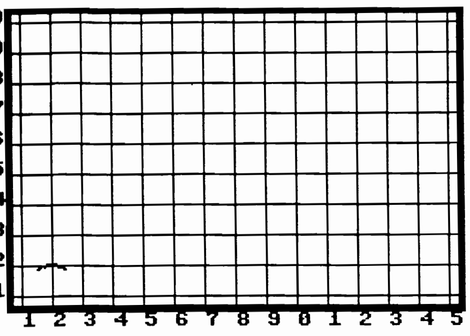


Figure A - Map of Rod's town.

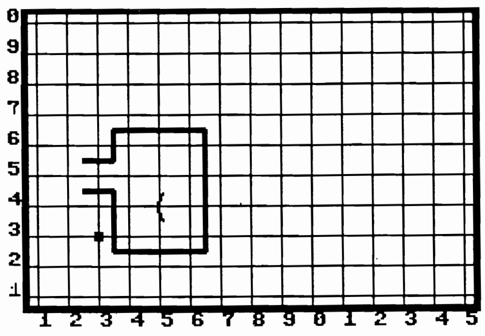


Figure C - Rod's house and a beeper.

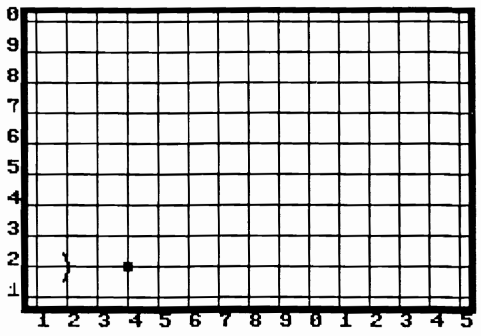


Figure 1 - Rod and beeper.

he moves forward to the next crossing. The command:

```
rt  (which can also be: right)
```

makes him turn to the right. The command:

```
lt  (which can also be: left)
```

makes him turn to the left.

Press <f7> to see the computer draw a fresh map of the town. Then enter this command line:

```
move;move;move;rt;move;move;move
```

Press <f7> again and enter the following command:

```
for i=1 to 10 do move;move;move;rt
```

Turn up the volume on the monitor and type:

```
putbeep
```

and watch Rod putting down a beeper. Then enter the commands:

```
move;rt;rt;move
```

to make him move away from the beeper and then back again. Note what happens as he comes back to the beeper. Now type the command:

```
pickbeep
```

and see what happens.

YOU NOW KNOW THAT:

move	moves Rod forward to next crossing
rt	turns Rod right 90 degrees
right	turns Rod right 90 degrees
lt	turns Rod left 90 degrees
left	turns Rod left 90 degrees
putbeep	makes Rod put a beeper down
pickpeep	makes Rod pick up a beeper

If Rod gets to a beeper, he can hear it. So he knows it is there.

EXERCISE 1.1

Shift to the textscreen by pressing <f8>. Then enter the command:

```
merge "problems"
```

to get some procedures loaded from the disk. Press <f7> to get a fresh map of the town. When it has been drawn type the command:

```
fig'1
```

You should now have a picture like Figure 1 on the screen. Now type a command line that makes Rod move the beeper to end up with a situation like the one shown in Figure 1B.

2. ROD THE ROADMAN MEETS BEN THE BRICKLAYER

In Rod's world there is a very clever fellow called Ben. He is a bricklayer. He decides where all the roadwalls and streetwalls are to be built. Sometimes he teases poor Rod by not telling him about the walls he sets up.

More ►

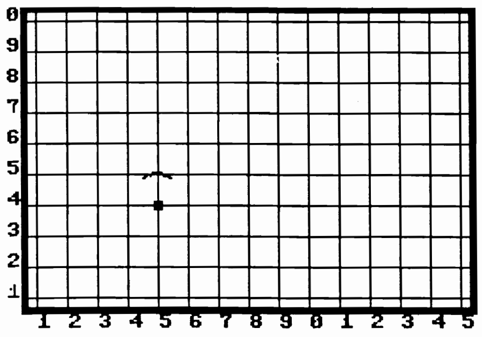


Figure 1B - Result for exercise 1.1

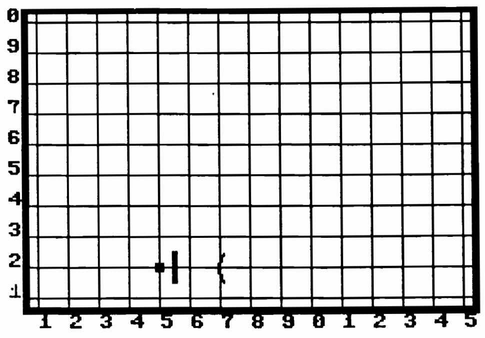


Figure 2 - Wall blocking Rod's path.

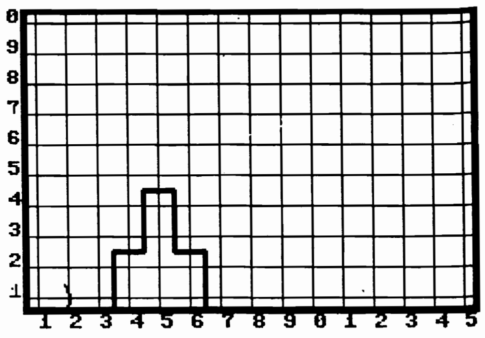


Figure 3 - Exercise 2.2

EXERCISE 2.1

Look at Figure 2. Press <f7> and enter the command:

```
fig'2
```

to get a similar picture on your screen. This is how the story behind it goes: Rod has put down a beeper where 5th street crosses 2nd road earlier that day. He is going back to pick it up again. But in the meantime Ben has built a roadwall across 2nd road between 5th street and 6th street. Now try to control Rod by using this command line:

```
move;move
```

Rod cannot pass the wall! The road is blocked! Help him to get around it by using the proper commands. Do not forget that in the end he has to pick up the beeper.

Rod cannot walk through or climb over the walls set up by Ben. He must make a detour. If he hits a wall, he laments in a most pitiful manner, because he has hurt himself.

It is not very often Ben teases Rod. Most of the time they help each other.

EXERCISE 2.2

Look at Figure 3. Ben has blocked an area of the town, because some dangerous remnants of gas have been found in the ground. He asks Rod to put a beeper where 5th street crosses 5th road, i.e. the crossing (5,5). The path Rod should follow is shown in Figure 3B.

Press <f7> and enter the command:

```
fig'3
```

Then type in some commands - preferably on one line - that make Rod do the job.

Sometimes it pays for Rod to memorize some commands, because he has to do them very often. This can be done by writing procedures for Rod to follow.

EXERCISE 2.3

Look at Figure 4. It shows a house that Ben has built for Rod. He is in his house now. Outside is a beeper with a lamp to light up the number of the house. Each morning this beeper is taken into the house, and each evening it has to be put out again.

Press <f7> and enter the command:

```
fig'4
```

Write some lines with commands to make Rod take the beeper into the house and put it in its upper right corner.

Press <f8> to get to the textscreen. Then press <f3> and watch COMAL writing a line number and the word:

```
proc
```

Add the name of the procedure to make the whole line look like this:

```
proc takein
```

Now press <RETURN> and enter these lines:

```
rt; move; move
lt; move; move
lt; move; move
lt; move
pickbeep
```

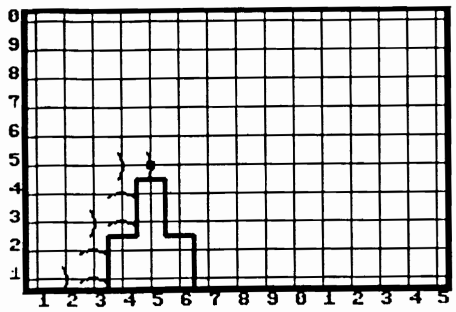


Figure 3B - Solution for exercise 2.2

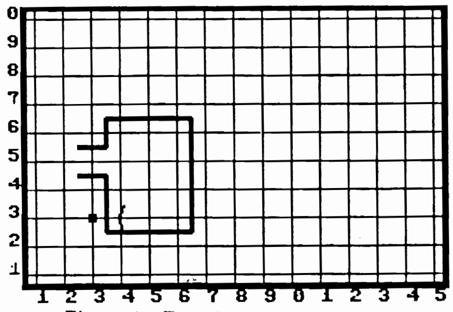


Figure 4 - Exercise 2.3

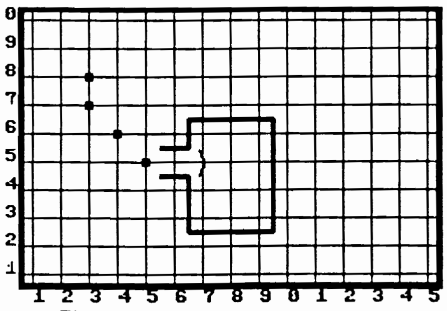


Figure 5 - Exercise 2.4

```
rt; rt; move
rt; move; move
rt; move; move
move; move; move
putbeep
```

Press <f4> to make COMAL finish the procedure with the statement:

```
endproc
```

NOTE: If two of you are working at the computer, it is best if one dictates while the other one does the typing. If you are working alone then use a ruler or the like to mark a line as you are typing it in.

Now press <f7> and type the commands

```
fig'4
takein
```

As you can see, the procedure is not in order. Rod was meant to put the beeper in the corner (6,6) and walk back to the place where he stood before. Look carefully at the picture (Figure 4) and the procedure in order to plan the necessary corrections and additions. Then press <f8> to get to the textscreen and correct the procedure. Clear the screen (press <SHIFT> with <CLR/HOME>). You can get a list of the present procedure by typing this command:

```
list takein
```

Having finished the corrections, press <f7> to get a fresh picture and enter the commands:

```
fig'4
takein
```

When the procedure is in order, write another one to make Rod take the beeper back to its place outside the house and then go back in again. Call the procedure takeout.

To test both procedures, press <f7> and use the commands:

```
fig'4
takein
takeout
```

EXERCISE 2.4

Look up Figure 5. Rod has got into his house but finds out that he has lost some of his precious little beepers on his way home. You should help him to pick them up and get back to his house again.

Shift to the textscreen by pressing <f8>. Press <f3> and start writing the procedure needed to make Rod pick up his lost beepers and then go back into his house again. Press <f4> to finish.

Press <f7> and enter the command

```
fig'5
```

Now test your procedure.

EXERCISE 2.5

Look up Figure 6. Rod only knows what right, left, and move mean. He knows nothing about up or down. Therefore he also do not know what stairs are all about. Ben has set up some walls to make him understand that. Rod is supposed to pick up the beepers on the stairs and end up on top of it all as shown in Figure 6B.

Write a procedure to help Rod do what he is supposed to. Use <f3> to start the procedure and <f4> to finish it. To test it, press <f7> and enter the command:

More ►

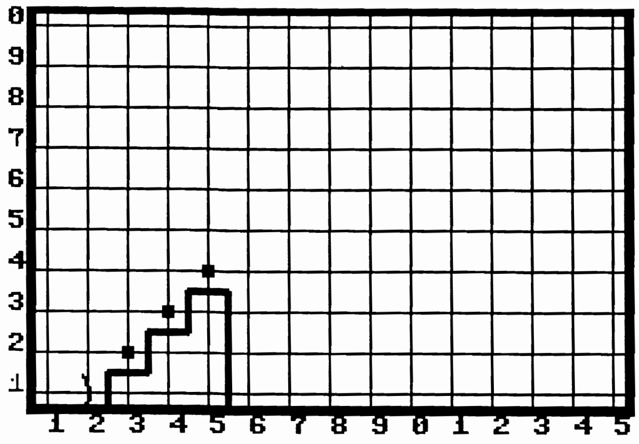


Figure 6 - Exercise 2.5

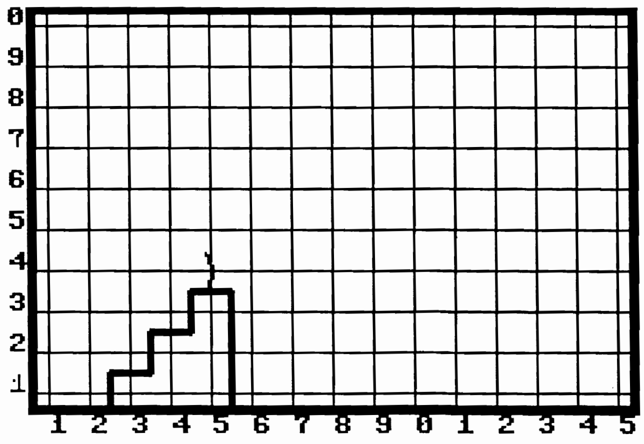


Figure 6B - Result for exercise 2.5

```
 fig'6
```

to get the stairs displayed.

3. THERE IS A SYSTEM IN WHAT ROD DOES.

We have tried to make Rod follow a path that reminds you of some stairs. Let us take a closer look at what Rod does to pick up the beepers on each of the stairs.

EXERCISE 3.1

Press <f7>. When the town has been drawn type:

```
 fig'6
```

and check that a picture like Figure 6 is coming up on the screen. We shall try to make Rod do the same as in exercise 2.5.

Press <f8> to get the textscreen. Then press <f3> to start a procedure and type:

```
 proc one'step
   lt; move
   rt; move
   pickbeep
```

Press <f4> to finish the procedure.

Press <f7> to get a map of the town and enter the command

```
 fig'6
```

Now type the command:

```
 one'step
```

and watch what happens. Go ahead using the same command until a picture like Figure 6B is on the screen.

Press <f8> to get the textscreen and enter a new procedure stairs that will make Rod pick up all the beepers as he did before. Test it by pressing <f7> and typing the commands:

```
 fig'6
 stairs
```

Very often we can help Rod best by giving the problems a second thought, before trying to have them solved systematically. As a rule it is not a good idea to start chasing Rod around without pondering first to find out what we really want him to do.

Ed. Note: Ah, yes. While Borge was in Madison, he left us a challenge. Can you write a procedure the will get ROD out of any house no matter where he is? Those who heard his talk at the MARCA computer show in July already saw the solution - but can you do it on your own now? Send your solution to us right away! We will print the names of all who can solve Borge's challenge. To do this you should use the scanners that ROD always carries with him. CLEAR'AHEAD will return TRUE if there is no obstacle ahead of ROD. Likewise, CLEAR'LEFT and CLEAR'RIGHT will check on either side. ■

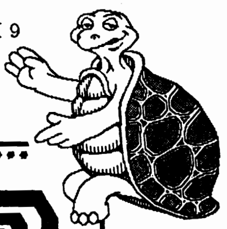
2.0 Scrolling Text

On page 59 of COMAL TODAY 8 two procedures contained a small error. The correct versions are shown below with the changes underlined:

```
PROC window'down(bottom,top) CLOSED
 USE system
 DIM screen$ OF 1505, char$ OF 3
 IF bottom>0 AND bottom<top AND top<26 THEN
  b:=(bottom-1)*60; t:=(top-1)*60
  getscreen(screen$)
  screen$(66+b:t+65):=screen$(6+b:t+6)
  back'color:=ORD(screen$(2))
  char$="  "+CHR$(back'color+back'color*16)
  FOR row#:=6+b TO 63+b STEP 3 DO screen$(r
  ow#:row#+2):=char$ // wrap line
  setscreen(screen$)
 ENDIF
ENDPROC window'down
```

```
PROC window'up(bottom,top) CLOSED
 USE system
 DIM screen$ OF 1505, char$ OF 3
 IF bottom>0 AND bottom<top AND top<26 THEN
  b:=(bottom-1)*60; t:=(top-1)*60
  getscreen(screen$)
  screen$(b+6:t+6):=screen$(66+b:t+65)
  back'color:=ORD(screen$(2))
  char$="  "+CHR$(back'color+back'color*16)
  FOR row#:=6+t TO 63+t STEP 3 DO screen$(r
  ow#:row#+2):=char$ // wrap line
  setscreen(screen$)
 ENDIF
ENDPROC window'up ■
```

Hi. I'm Calvin the COMAL Turtle. Davids Directory Designer is so easy that even turtles can use it. I used it to organize TODAY DISK 9 and it looks great. Get the disk and see for yourself. Also look below to read all about it.



DAVID'S
DIRECTORY
DESIGNER

"If this can't do it to your 1541 disk directory -
IT PROBABLY SHOULDN'T BE !"

Alphabetize all or part.	Read all files.
Self documenting.	Fast and true.
Scratch, Rename or move.	Print directory.
Add or delete outside quotes.	Lock or unlock files.

Built-in disk monitor - Whole block w/ translation on screen at one time.
Put your own comments in directory without loss of disk space.
MUCH MORE !! (It also works!)

SPONSORED by the Catalina Commodore Computer Club, Inc. of Tucson, AZ

This $15.00 program, written by David Lovelock, has been donated to raise funds for the purchase of a special free use computer dedicated to search and rescue. (The author gets none of the funds.)

Send your $15 check to "CCCC Search & Rescue Fund, P.O. Box 32548, Tucson, AZ 85750. (Allow 4 weeks but we usually get them out in 2 days.)

The program alone is worth much more, but you will also be supporting a worthwhile project. Incidentally the purchase is tax deductible. (IRS # 86-6004791)

Help us prove that Computer Hackers can be productive members of our society! Then when you hear of a computer assisted rescue you'll know you helped!

Tanks & The Animate Keyword

by Bob Hoerter

This program demonstrates sprite animation with two sound effects - the beginning of my journey into the land of sprites. After a little study of the ANIMATE commands the light came on and I discovered that only ONE sprite was being used, but that multiple images were assigned to that sprite by the ANIMATE command. The other numbers in the command refer to how long the image is 'on'. For example:

```
ANIMATE(sprite#, image1, time'image1'on,
image2, time'image2'on)
```

In a program listing this might appear as:

```
0100 ANIMATE(1,""1""30""2""30"")
```

This would animate sprite number 1 using image1 for 30/60ths of a second and image2 for 30/60ths of a second.

The use of sound in these procedures is controlled by the SPRITEX function in the sprites package. This illustrates how easy a sprites postion can be used to control another action.

[Ed. note: Bob's program is filled with useful PROCs. Do you need a machine gun sound for your programs? The PROC is already done. The program shows clearly how to use the ANIMATE keyword.]

```
// delete "tank'animation"
// save   "tank'animation"
//  by Bob Hoerter
USE system
USE sprites
USE sound
USE graphics
DIM sprite$ OF 64
PAGE
background(15)
pencolor(0)
border(12)
graphicscreen(1)
tank
WHILE KEY$<>""0"" DO NULL
WHILE KEY$=""0"" DO NULL

PROC tank
  read'data
  setup'tank
  hide'tank
  animate'tank
  move'tank
ENDPROC tank

PROC setup'tank
  spriteback(8,5)
  FOR s#:=1 TO 4 DO
    spritecolor(s#,9)
    spritesize(s#,TRUE,FALSE)
    spritepos(s#,-48,30)
    priority(s#,TRUE)
    identify(s#,s#); showsprite(s#)
  ENDFOR s#
ENDPROC setup'tank

PROC animate'tank
  y:=30; x:=272
  animate(1,""1""7""2""7"")
  animate(3,""3""10""4""10"")
  movesprite(1,x,y,800,1)
  movesprite(4,x,y,800,1)
  movesprite(2,x,y,800,1)
  movesprite(3,x,y,800,1)
ENDPROC animate'tank

PROC move'tank
  startsprites
  WHILE moving(1) DO
    IF spritex(1) MOD 30=0 THEN
      bang
    ELIF spritex(1) MOD 51=0 THEN
      machine'gun'sound
    ENDIF
  ENDWHILE
ENDPROC move'tank
```

More ►

```
PROC bang
  volume(15)
  adsr(1,2,4,3,3); soundtype(1,4)
  adsr(2,2,4,3,3); soundtype(2,4)
  adsr(3,0,6,5,4); soundtype(3,4)
  note(1,"c4")
  note(2,"g7")
  note(3,"f5")
  FOR v:=1 TO 3 DO gate(v,TRUE)
  FOR dd:=1 TO 35 DO
    IF dd>20 THEN volume(35-dd)
  ENDFOR dd
  FOR v:=1 TO 3 DO gate(v,FALSE)
ENDPROC bang

PROC hide'tank
  viewport(0,170,0,50)
  window(0,50,0,100)
  pencolor(0)
  paint(0,5)
  textstyle(1,5,0,0)
  FOR x#:=0 TO 50 STEP 10 DO
    plottext(x#-10,10,STR$(x#))
  ENDFOR x#
ENDPROC hide'tank

PROC machine'gun'sound
  adsr(1,0,2,0,0); note(1,"g3")
  adsr(2,0,2,0,0); note(2,"c2")
  adsr(3,1,4,6,10); note(3,"f3")
  FOR v:=1 TO 3 DO soundtype(v,4)
  volume(15)
  FOR ratatattat:=1 TO 15 DO
    FOR v:=1 TO 3 DO gate(v,TRUE)
    FOR wait:=1 TO 50 DO NULL
    FOR v:=1 TO 3 DO gate(v,FALSE)
  ENDFOR ratatattat
  volume(0)
ENDPROC machine'gun'sound

PROC read'data
  FOR s:=1 TO 4 DO
    sprite$:=""
    FOR n:=1 TO 64 DO
      READ num
      sprite$:+CHR$(num)
    ENDFOR n
    define(s,sprite$)
  ENDFOR s
  // shape tb 2
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,5,84,0
  DATA 26,169,104,90,170,144,153,153
  DATA 148,110,186,217,174,186,238,122
  DATA 235,185,186,235,152,25,153,144
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,1
  // shape tb 3
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,5,84,0
  DATA 26,169,104,90,170,144,102,102
  DATA 100,186,235,149,122,235,186,174
  DATA 186,237,110,186,212,38,102,96
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,1
  // shape turr 1
  DATA 0,0,0,0,0,0,2,160
  DATA 0,14,173,64,14,165,0,14
  DATA 148,0,14,172,0,5,84,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,1
  // shape turr 2
  DATA 0,0,0,0,0,0,2,160
  DATA 0,14,108,0,14,108,0,14
  DATA 108,0,14,172,0,5,84,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,1
ENDPROC read'data
```

Epson / Cardco +G Procs - Update

by Raymond E. Saller

COMAL TODAY #6, page 65, listed a set of procedures for the Epson FX-80 printer codes. As they stand there may be some problems.

First of all they do not take into account the interface that must be used in conjuction with a non-Commodore printer, the MOST POPULAR being the Cardco +G, which emulates a Commodore printer such as the 1525.

To the Cardco interface a CHR$(18) is a 'turn on REVERSE MODE', not uncondense. Also the code used to turn on the EXPANDED MODE will be cancelled by a line feed automatically. Using letters such as "E", "F", etc. will cause problems depending on whether or not you are in the lower case or uppercase/graphics mode.

Below is a listing of all of the most popular control codes used by the FX and RX Epsons. They have been tried and tested using the Cardprint +G interface and an RX-80ft printer. The procedures will work with either version of COMAL.

```
// delete "proc.epson'cardg"
// list   "proc.epson'cardg"
//  by Ray Saller
//
PROC beep
  PRINT CHR$(7),
ENDPROC beep
//
PROC backspace
  PRINT CHR$(8),
ENDPROC backspace
//
PROC enlarge
  PRINT CHR$(27),CHR$(87),CHR$(1),
ENDPROC enlarge
//
PROC unenlarge
  PRINT CHR$(27),CHR$(87),CHR$(0),
ENDPROC unenlarge
//
PROC condense
  PRINT CHR$(27),CHR$(20),
ENDPROC condense
//
PROC uncondense
  PRINT CHR$(27),CHR$(18),CHR$(146),
ENDPROC uncondense
//
PROC underline
  PRINT CHR$(27),CHR$(45),CHR$(1),
ENDPROC underline
//
PROC underline'off
  PRINT CHR$(27),CHR$(45),CHR$(0),
ENDPROC underline'off
//
PROC emphasize
  PRINT CHR$(27),CHR$(69),
ENDPROC emphasize
//
PROC emphasize'off
  PRINT CHR$(27),CHR$(70),
ENDPROC emphasize'off
//
PROC bold
  PRINT CHR$(27),CHR$(71),
ENDPROC bold
//
PROC bold'off
  PRINT CHR$(27),CHR$(72),
ENDPROC bold'off
//
PROC elite
  PRINT CHR$(27),CHR$(77),
ENDPROC elite
//
PROC pica
  PRINT CHR$(27),CHR$(80),
ENDPROC pica
//
```

More ►

```
PROC superscript
  PRINT CHR$(27),CHR$(83),CHR$(0),
ENDPROC superscript
//
PROC endscript
  PRINT CHR$(27),CHR$(84),
ENDPROC endscript
//
PROC subscript
  PRINT CHR$(27),CHR$(83),CHR$(1),
ENDPROC subscript
//
PROC reverse
  PRINT CHR$(18),
ENDPROC reverse
//
PROC reverse'off
  PRINT CHR$(146),
ENDPROC reverse'off
//
PROC italics
  PRINT CHR$(27),CHR$(52),
ENDPROC italics
//
PROC italics'off
  PRINT CHR$(27),CHR$(53),
ENDPROC italics'off
//
PROC spacing'one'sixth // default
  PRINT CHR$(27),CHR$(50),
ENDPROC spacing'one'sixth
//
PROC spacing'7'72
  PRINT CHR$(27),CHR$(49),
ENDPROC spacing'7'72
//
PROC formfeed
  PRINT CHR$(12),
ENDPROC formfeed
```

[Ed. Note: The CARDCO +G printer interface has several print modes. Transparent mode would probably work with the regular codes which are presented in the EPSON printer manual.] ■

Gemini 10X 'Color' Screendump

by Ray Carter

This is a program written entirely in COMAL which will dump any multi-color graphics screen to a GEMINI-10X printer (minor modifications should enable use with other printers).

The first part of the program generates a color screen consisting of vertical bars which will be dumped. Replace this part with whatever you like. The dump routine is contained in PROC dumpscreen. Each multicolor pixel (which is defined by a pair of bits) is converted to a 2 by 2 block of dots on the printer, and each color has a unique pattern. Also, each color has a density reflected by its gray-color group.

A little study of the main portion of this routine will reveal much about the way the color graphics screen is organized. In particular, note the SETPAGE(6) and SETPAGE(0). The screen memory is in RAM underneath the color nybble RAM. Each pair of bits in the bitmap determines whether the corresponding pixel on the screen takes its color from the background color, or color RAM, or the high or low nybble of the screen memory.

You will also notice that the image is rotated sideways on the page. Each pass of the printhead across the page corresponds to scanning up one column of the screen. This orientation was chosen since the GEMINI produces nearly the same aspect ratio as the screen. ■

Show The Comal 2.0 Name Table

by Mike Lawrence

The following two procedures can be used from within a program, or in immediate mode - after a SCAN is done.

```
PROC showtable(skip) CLOSED
  // if skip=0 it prints unknown names
  PRINT "Current symbol table "
  PRINT
  point:=PEEK($18)+256*PEEK($19)
  st:=point; total:=0; unknown:=0
  REPEAT
    jump:=PEEK(point)
    type:=PEEK(point+1)
    IF NOT type THEN unknown:+1
    IF NOT (skip AND NOT type) THEN
      total:+1
      printtype(type)
      PRINT ": ",TAB(24),
      FOR i:=point+4 TO point+jump-1 DO
        PRINT CHR$(PEEK(i)),
      ENDFOR i
      PRINT
    ENDIF
    point:+jump
  UNTIL PEEK(point)=0
  PRINT
  PRINT "number printed:";total
  PRINT "number unknown:";unknown
  PRINT "table size:";point-st;"bytes"
  PRINT "End of table"
  //
  PROC printtype(type)
    CASE type OF
    WHEN 0
      PRINT "unknown",
    WHEN 16
      PRINT "simple real",
    WHEN 17
      PRINT "simple integer",
    WHEN 18
      PRINT "simple string",
    WHEN 20
      PRINT "procedure",
    WHEN 21
      PRINT "real function",
    WHEN 22
      PRINT "integer function",
    WHEN 23
      PRINT "string function",
    WHEN 24
      PRINT "package name",
    WHEN 48
      PRINT "real parameter",
    WHEN 49
      PRINT "integer parameter",
    WHEN 50
      PRINT "string parameter",
    WHEN 80
      PRINT "real ref parameter",
    WHEN 81
      PRINT "integer ref parameter",
    WHEN 82
      PRINT "string ref parameter",
    WHEN 144
      PRINT "real array",
    WHEN 145
      PRINT "integer array",
    WHEN 146
      PRINT "string array",
    OTHERWISE
      PRINT "unknown type";type,
    ENDCASE
  ENDPROC printtype
ENDPROC showtable
//
PROC showproc CLOSED
  DIM name$(20:24) OF 17
  DIM word$ OF 80
  name$(20):="Procedures"
  name$(21):="Real Functions"
  name$(22):="Integer Functions"
  name$(23):="String Functions"
  name$(24):="Packages"
  PRINT "Procedures and Functions"
  PRINT
  FOR find:=20 TO 24 DO
    point:=PEEK($18)+256*PEEK($19)
    PRINT name$(find)
    PRINT
    total:=0
```

More ►

Show the COMAL 2.0 Name Table - continued

```
    REPEAT
      jump:=PEEK(point)
      type:=PEEK(point+1)
      IF type=find THEN
        PRINT TAB(5),
        word$:=""
        FOR i:=point+4 TO point+jump-1
        DO // wrap line
          word$:+CHR$(PEEK(i))
        ENDFOR i
        IF word$<>"showproc" THEN
          PRINT word$,
          total:+1
          CASE type OF
          WHEN 22
            PRINT "#"
          WHEN 23
            PRINT "$"
          OTHERWISE
            PRINT
          ENDCASE
        ENDIF
      ENDIF
      IF KEY$=" " THEN
        REPEAT
        UNTIL KEY$<>" "
        REPEAT
        UNTIL KEY$=" "
      ENDIF
      point:+jump
    UNTIL PEEK(point)=0
    PRINT
    PRINT "number of";name$(find);":";
    total // wrap line
    PRINT "--------------------------
    ------------" // wrap line
    PRINT
  ENDFOR find
  PRINT "End of table"
ENDPROC showproc
```

Comments

by Colin Thompson

Wasn't it Lee Iacocca that said "It's over, over there?" He certainly couldn't have been referring to COMAL programming. Last month we featured some Dutch programs, and now we have not one, but two first rate educational programs from "over there".

Borge Christensen's **Rod the Roadman** can provide endless hours fun while learning about procedures. It's quite **revealing** how he took a simple concept and turned education into fun.

On the other end of the spectrum, comes a gem from England which will teach you all about structured programming from a different angle: that of the Computer Scientist. Did you ever wonder what goes on in a Computer Science class? Well, you can finally find out by turning over the COMAL TODAY disk #9 to the COMAL 0.14 side. (Yes, I know that most of you Cartridge owners haven't seen a COMAL 0.14 program since you got your precious toy, but this program is worth it).

A.C. Millest, with help from Brian Grainger, has written a five part tutorial which instructs in the fine art of designing a program **before** you actually sit down and begin writing it. This is not the way most of us do it. Maybe we're doing it wrong... Anyway, the filename to LOAD is: **STRUCTURE'PRG'1**

The lessons take you through planning, structure, flowcharting, then final coding and testing of a simple game program. This program reminded me of Roy Atherton's Structured Programming with COMAL. If you are new to COMAL or structured languages, this program will teach you some worthwhile discipline. Old A.C.'s done up a jolly good piece of work, what?

Using Strings in Comal

by David Stidolph

Most people think of computers as "souped-up calculators" (which in a sense is true), but in reality, what separates them is the ability to handle words as well as numbers. To the computer they are the same thing, but in a high level language like COMAL they are very different. Any word or name can be broken down into individual letters, and that is how they are stored in the computer. The term **string** is used to describe a word as a string of characters. For example:

```
10 PRINT "Hello, I am a string!"
```

The above line is valid in either Basic or COMAL. When the line is executed by the language, it will print what is inside of the quotes to the screen. The example above would print:

```
Hello, I am a string!
```

This first string example is referred to as a **string constant**, because it remains constant, and cannot be changed while the program runs (to change it you would need to stop the program).

In order to manipulate names, addresses, ect., you need to know about **string variables**. These are places in memory you call by name where the computer stores words. You don't have to know where in memory they are stored, only that you refer to it by name. First you have to tell the computer how much space to set aside for the information you want (the MAXIMUM amount). For example, if you need to know someones first name, and the maximum length you need is 10 characters, you might use this line:

```
20 DIM first'name$ of 10
```

The statement **DIM** stands for dimension, and tells the computer that it needs to set aside room in memory for a variable. The name **first'name$** is the variable name. Note the dollar sign at the end of the name. This dollar sign identifies it as a string variable. The word **OF** tells the computer to look for the length of the string next, which in our example is ten. The following line shows how you assign a string variable a value (from a string constant):

```
30 first'name$:="I am a string"
```

If you ran this program, the string variable "first'name$" would be created, and assigned a value. If you typed:

```
PRINT first'name$
```

you would get the following:

```
I am a str
```

Notice - only the first ten letters were stored by COMAL. The reason for this is that we dimensioned the string to a length of 10 characters, and we tried to assign 13 characters to it. The last three letters are lost. COMAL takes the characters that will fit into the variable (the first 10), assigns them to the variable, and ignores the rest. If used properly, this can be a bonus. Example:

```
 10 DIM reply$ OF 1
 20 REPEAT
 30  INPUT "Enter first number: ":a
 40  INPUT "Enter second number: ":b
 50  PRINT "The sum of";a;"and";b;"is";
 60  PRINT a+b
 70  PRINT
 80  INPUT "Continue (y/n): ":reply$
 90 UNTIL reply$="n" OR reply$="N"
100 PRINT "This is the end."
```

More ►

The program dimensions a string variable to a length of one. Line number 80 INPUT's the string, but no matter how many characters you type, only the first character will be assigned to the string. Line number 90 tests the string to see if it is "n" or "N". The loop would end if you typed No, no, nah, nope, or any word beginning with "n" or "N".

Let's suppose you want to INPUT the entire response, but need to test only the first character. COMAL allows you to look at any part of the string easily. You do this by putting the particular character posistion you want in parentheses after the string variable name. In the above example we could change lines 10 and 90 to read:

```
10 DIM reply$ OF 25
90 UNTIL reply$(1)="n" OR reply$(1)="N"
```

Now we would have the complete response of the user, but only test the first character for particular values. Be careful though. If you specify a string position that is greater than the length of the string (the number of characters assigned, NOT how many it was diminsioned to), you will get a substring error message. To find the length of a string, you use the LEN function, which will be explained shortly.

If you need to test more than one position, don't worry, you can. COMAL allows you to specify any part of a string, just by putting the starting and ending positions (separated with a colon ':') within parentheses after the string variable name. For example, you could take the string **name$** and print out the first six characters like this:

```
PRINT name$(1:6)
```

You can also assign a value to a part of a string. In the following example, assume the variables have already been dimensioned properly.

```
first$:="David"
second$:="Stidolph"
message$:=first$+" "+second$
PRINT message$
message$(1:6):="=>"
PRINT message$
```

The example above shows how to concatenate strings. The third line sets **message$** equal to the first string variable and the second string variable with a space in the middle. This would print:

```
David Stidolph
```

The fifth line will wipe out the first six characters (1 through 6), and replace them with =>. You'll notice that I am replacing six characters with two. This in not a problem, because COMAL merely adds on spaces as needed. The last line prints out the new value of message$, which is:

```
=>    Stidolph
```

As I said before, to find the current length of a string, COMAL has the LEN function. The example below shows how to use this function:

```
PRINT LEN(a$)
```

The PRINT statement would output the length of the string, which is 14. This function is not especially fast (with either version of COMAL), so if you need to the same value often, set a variable equal to the length of the string, and use the variable instead (remember to update the variable when the string variable is changed).

More ►

Suppose you need to search one string for a certain character or word. COMAL has a very fast string function called "IN" to do this. The way you use it is shown in the following program:

```
DIM a$ OF 1
DIM b$ OF 40
a$:="@"
b$:="This is a test. '@*+-'"
WHILE a$ IN b$ do
 position:=a$ IN b$
 b$(position):="#"
ENDWHILE
PRINT b$
END
```

This program starts by assigning the string:

"This is a test. '@*-'"

to the varible b$. The WHILE loop tests for the character "@" in the variable b$, and if it is found, the code within the loop is executed. In this example, the condition a$ IN b$ is being tested, and in this case, a value of 17 is returned, so the code within the loop IS executed.

If the character "@" was NOT in b$, a value of zero (0) is returned, and the code within the loop would NOT be executed. One use of the IN function would be to have a very long string containing all the words in a dictionary, and use IN to test each word to make sure it is spelled properly.

Everything in this article, so far, has been true for both versions of COMAL on the Commodore 64. In the 2.0 cartridge version, though, if you do not dimension a string before you assign it something, COMAL will dimension it for you to a default length of 40. It is a good idea, though, to **always** dimension string variables, so that the program stays compatible with most versions of COMAL, and to remind yourself how many characters the string variable can hold.

In closing, here is where you can find more information on string handling in COMAL:

COMAL from A to Z: pages 56-57

The COMAL Handbook (first edition): pages 17, 34, 35, 82, 90, 100, 103, 104, 143, 149, 180, 190, 208, 237, 264

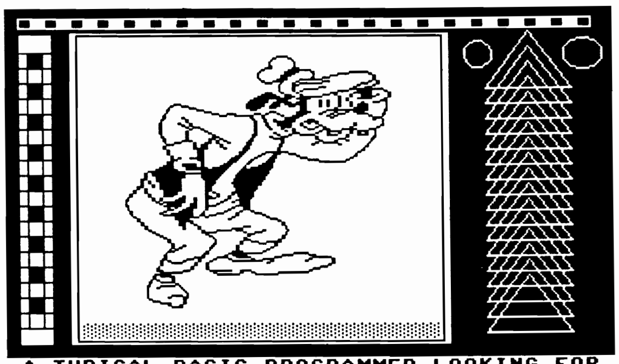
The COMAL Handbook (second edition): pages 68, 157, 174, 175, 230, 234, 240, 354-357, 431, 435

Structured Programming With COMAL: pages 18, 27-30

Beginning COMAL: pages 70-84

Foundations in Computer Studies Using COMAL: pages 7, 10, 30, 31, 50, 55, 63, 80, 128, 131, 137-140, 157, 165, 198, 208, 209, 229

COMAL Today number six: page 73 ■


A TYPICAL BASIC PROGRAMMER LOOKING FOR A BETTER LANGUAGE.... COMAL ?? YES !!!

Make Your Program A Package
For Intermediate and Advanced Programmers

by Glen Colbert

Do you use one COMAL 2.0 program a lot? Do you spend time looking for the disk and waiting for it to load? Now PROG'RAM can change that COMAL program into a package file. You then can LINK it in from disk and it hides under the cartridge. Or burn it into an EPROM and insert it into the empty socket in your black COMAL 2.0 Cartridge. Either way you have instant access to the program at all times. Your COMAL program is actually changed into a package. Commands NEW, DISCARD, and LOAD will not affect it. Start it like this:

```
USE <name> // name is the name of package
```

The program will then be moved into the COMAL workspace area from the package area and executed.

PROGRAM OPTIONS

The PROG'RAM program creates one disk file which contains the binary image of the package. This file can be used to make an EPROM. It must be burned into the EPROM with the correct offset to be located at the position it was written to reside at. If the LINKable file option is selected, another file is created with the same name as the binary image file, but prefixed by PKG. This file may be LINKed directly from disk and the package will remain in place until the computer is turned off or BASIC is selected.

CHANGING YOUR PROGRAM INTO A PACKAGE

1) IMPORTANT: Make sure your program is FINAL and WORKING.

2) To save space in the package you may wish to clean up the name table. To do this issue these two commands:

```
LIST "LST.TEMP"
ENTER "LST.TEMP"
```

Be careful. You also loose any sprite images, special fonts, or packages that were attached to the program.

3) SAVE the program to disk (we will use PROGRAM for a name in this example):

```
SAVE "PROGRAM"
```

4) To save even more space in the package you also may wish to PROTECT the program now using the PROTECT64 program on CARTRIDGE DEMO DISK #3:

```
RUN "PROTECT64"
```

Specify the correct name of the program file to be protected. Make sure you have another copy of the program. There is no UNprotect option. Make sure there is lots of free space on the disk.

5) LOAD the package maker program from TODAY DISK #9:

```
LOAD "PROG'RAM"
```

6) Insert your program disk again. Then issue the RUN command to start PROG'RAM.

7) Choose what page of memory to use:

REPLY: 5 if EPROM is final destination
REPLY: 0 if disk package is wanted

8) Choose memory location. Number may be entered in decimal or hex (preceded by $):

For an EPROM type package use one of these:

$8000, $9000, $A000, or $B000

16K is maximum size for an EPROM type

More ►

package. If the program file is over 60 blocks on the disk, it is too big. The package may NOT exceed $BFFF.

For a disk file package use any number between <u>$6000</u> and <u>$BFFF</u>. Note: package cannot use $8000 thru $8009 <u>unless</u> it starts at <u>$8000</u>. <u>NOTE</u>: Using memory below $8000 will reduce the free memory for programs accordingly.

9) **Choose whether you want the program to AUTOSTART or not.** If you reply <u>N</u> the program will be available but require a <u>USE</u> command to start.

10) **Do you want the program to delete itself** from program memory when it finishes? This might be handy as a program protection tool. It always is in the package area, even when deleted from the program area.

If you created a disk file package you retrieve it like this:

```
LINK "PKG.PROGRAM"
```

Start it like this:

```
USE PROGRAM
```

If your program is destined to be an EPROM package, you now have a file ready to be burned into an EPROM. This process is not simple. Fortunately, it is not extremely complicated either. Follow the usual EPROM programmer procedures. Very few of us have our own EPROM programmers. However, if you check with your local user's group, the odds are good that someone in the group has one.

Make sure the EPROM is burned from your file with no offset so it stays in the same memory locations. Once your program is burned in, the only way to make changes is to erase the chip and start over. Unless you have your own EPROM programmer, this could be quite an inconvenience.

The black COMAL 2.0 Cartridge is set up to accept a 27128 EPROM. By cutting the trace on the back of the board and soldering the other link closed, you could use a 27256 EPROM.

To open the cartridge case (which voids the warranty), first remove the small phillips screw on the underside with the correct size screwdriver. The screw head strips very easily.

The four slots on the bottom of the cartridge provide access to the plastic case's catches. These catches hook the two halves of the cartridge together. Slide a SMALL flat edged screwdriver all the way into the slot and twist slightly while pulling to separate the top half of the case from the bottom. All four catches must be released to open the case.

After the case has been opened, plug the EPROM into the socket (make sure it's notch is facing the same as the other two chips). Snap the cover back on and replace the screw. The cartridge is now ready to fire up with your program running! Start it with the command:

```
USE PROGRAM
```

Ed. note: Glen would like to hear from you if you add options to this system: Glen Colbert, 2352 So. 1440 W., Salt Lake City, UT 84119. Glen asked us to release only a protected version of this program, since even slight modifications could seriously affect the dependability of the program. ■

Beginners -

Going through the first four issues of COMAL TODAY preparing our reprint was interesting. Most of the material is still applicable. New beginners missed quite alot of information. We were numbering our notes - and they were in the hundreds! For the few of you who missed those issues and didn't immediately order the reprint, here are just a few of the notes for beginners from the first issues of COMAL TODAY:

UNSHIFTED LETTERS

Remember, use **unshifted letters** throughout entering a program. If letters are capitalized in the printed listing it does not mean to use SHIFT with those letters. They are capitalized merely for a pleasant, easy to read, look.

LINE NUMBERS REQUIRED BUT NOT LISTED

Line numbers ARE used by COMAL, but they are not significant while a program is running. Thus, whether a line is numbered 0010 or 5480 is not going to change anything. Only the sequence of numbers matters. So - COMAL programs will often be listed without any line numbers. But COMAL can supply them for you. Just use the AUTO command.

GRAPHICS MODE

To go into hi-res graphics mode the first time enter the following command:

```
SETGRAPHIC 0
```

For multi-color graphics use:

```
SETGRAPHIC 1
```

This first time you must specify either 0 or 1 with SETGRAPHIC because you are initializing the graphics system. Once the system is initialized you may reenter it with just:

```
SETGRAPHIC
```

Once the graphics screen has been initialized three function keys are available for use from ready mode (whenever the cursor is blinking, no program running, or during an INPUT statement). <f1> is the same as **SETTEXT** to return to the text screen. <f3> is the same as **SETGRAPHIC** and **SPLITSCREEN**. <f5> is the same as **SETGRAPHIC** and **FULLSCREEN**.

WHAT DOES THE ' APOSTROPHE MEAN

Those used to BASIC and its restrictions, wonder what the ' or the <- (left arrow) means when seen in a variable name. COMAL, being of European origin, allows several 'extra' characters in addition to the usual alphnumerics. As most of you know, many European languages have more than 26 characters in their alphabet. Thus the apostrophe ('), square brackets ([) and (]), the left arrow (<-), and the backslash (\) are allowed to be used as part of an identifier (ie, names of variables, procedures, functions). Printers that can print the special characters of the European languages (such as ø in Danish) can be designed to print these special letters in place of the special symbols. The left arrow is converted to an underline character by Commodore COMAL (COMAL running on computers with an underline key allow its direct use). It and the apostrophe allow a multi word variable name to be readable:

SUMOFSIDES can become **SUM'OF'SIDES**

Much better. Why didn't BASIC think of that? ■

The Real Julian Day

by Tom Kuiper

Captain COMAL has given a handy set of routines for converting between calendar date and day-of-the-year (abbreviated DOY), which he called Julian date. I hope the Captain will forgive a picky astronomer for setting the record straight, as long as the pill is sweetened with another software contribution.

Anyone who wants the full story should locate a copy of that classic introductory astronomy text by George Abell, Exploration of the Universe. Basically, the modern calendar traces its history to Julius Caesar, who in 46 BC decreed a calendar devised by the Alexandrian astronomer Sosigenes. This calender is very similar to our own. In 1582, Pope Gregory XIII instituted some amendments, which were not accepted in the British Empire until 1752. (Thus, George Washington was born on Feb. 11 by the American calendar of the time.)

The complexities of the various calendars makes astronomical calculations hopelessly difficult. For example, how do you calculate the date of closest passage of a comet that comes by every 76 years? Therefore the Julian day (J.D.) was invented which is simply the number of days since noon on January 1, 4713 BC, a date which is supposed to have something to do with the creation of the world!

The following routines are included:

DAY'OF'YEAR(year#,month#,day) converts Gregorian calendar date to DOY.

LEAP'YEAR(year#) returns TRUE if the year is a leap year in the Gregorian calendar.

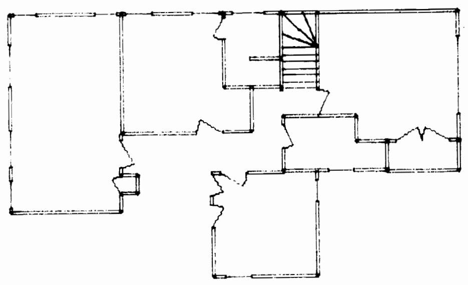
JULIAN'DAY(year#,day'of'year) computes the (true) Julian day. For example, the start of Christmas 1985 is JD 2445328.5!

DAY'YEAR(jul'day,REF current'year#,REF day'of'year) converts JD back again.

DATE(doy#,year#,REF month#,REF day#) converts DOY back to calendar date.

DAY'OF'WEEK(month#,day#,year#) gives the day of the week (Sunday=1, Saturday=7).

To explore further the potential use of Julian Days, refer to an article called "Generalized Equations for Julian Day Numbers and Calendar Dates" by D. A. Hacker in the Quarterly Journal of the Royal Astronomical Society. Besides the equations, which seem relatively easy to code, it contains coefficients for 16 calendars. If any reader wants to code this and cannot locate the journal, write to me via the Captain for a copy of the article. ■



TOM'S DREAM HOUSE

```
user group 10
boot comal
c64 comal 0.14
hi
menu
!-------------!
!  data files  !
!-------------!
! !don't load! !
!-------------!
-error-messages-
 directory
 dump.nec
 spi-main
alpha1.dat
e & jim v.hrg
hrg.air force
information.dat
load/save.mem
mirrow.dat
phone data
txt.term14.inst
!-------------!
!--procedures--!
!-------------!
disk'get'improv
plotter.procs
two'tone.proc
colr'bar.proc
val.lp.proc
min'max.func
!-------------!
! applications !
!-------------!
code trainer
drunkardwalk
invoicer
linear'regressn
nursery libs
phone log
pulse rate
rhyming speller
sieve'eratosthe
sim'equations
sundial
terminal.14
!-------------!
!  utilities   !
!-------------!
find'load'addr
load'color'scrn
mirrow writer
p1090 char edtr
print show.nec
!-------------!
! graphics fun !
!-------------!
art
dots.game
koala doodler
queens
son o spirograph
spirograph
sunflake
yarn'art'3
!-------------!
!the following !
! requires the !
! 1520 plotter !
!-------------!
formatter.1520
sunflake.1520
```

```
ct#9-version 2.0
hi
!-------------!
!   programs   !
!-------------!
1541'alignment
clue
convert'num
dates&julian
direct'con
gemini'colordump
icon'maker
infantry
magic'paint
make'package
num'to'word
oki92'test
program'1
program'2
rod
roderigue
roll'over
seq'to'speed
show'errors
single'file'copy
speed'to'seq
tank'animate
viewport
waves'demo
yahtzee
!-------------!
!  functions   !
!     and      !
!  procedures  !
!-------------!
func.decimal
func.drive'type
func.last
func.mean
func.random'size
func.rms
func.sdev
func.sigma
proc.1520plotter
proc.convert1
proc.convert2
proc.convert3
proc.epson'cardg
proc.graph'keys
proc.problems
proc.show'names
proc.trunc
proc.window'down
proc.window'up
!-------------!
!  data file   !
!-------------!
dat.information
!-------------!
!   packages   !
!-------------!
pkg.first'last
pkg.oki92
src.first'last
src.oki92
!-------------!
! shape files  !
!-------------!
shap.down'rod
shap.lt'rod
shap.rt'rod
shap.up'rod
```

```
ct#9-version .14
boot c64 comal
c64 comal 0.14
hi
menu
ml.sizzle
comalerrors
!-------------!
!copy the above!
!files together!
!to other disks!
!for the sizzle!
!loader        !
!-------------!
>---programs---<
1520flag'day.14
1520max'print.14
1520polygons.14
1541'alignment
1541'align'1
1541'align'2
create'sizzle
metamorphose
program'3
program'4
seq'to'speed
speed'to'seq
structure'prg'1
structure'prg'2
structure'prg'3
structure'prg'4
structure'prg'5
structure'prg'6
waves'keybd'demo
!-------------!
!  procedures  !
!     and      !
!  functions   !
!-------------!
1520/drv.proc
decimal.func
drive'type.func
dump'1525.proc
dump1520.proc
epson'cardg.proc
load'sizzle.proc
read'errors.proc
showtable.proc
zerotable.proc
!-------------!
!  data file   !
!-------------!
information.dat
!-------------!
!basic program !
!-------------!
fast'boot.bas
```

```
utility disk #2
    side 1
bootquick
load1
load2
0.14
-error-messages-
hi
---------------
1541'alignment
1541'align'1
1541'align'2
boot'dir'editor
comal'keypad.14
dir'manipulator
directory'editor
disk'edit/protct
disk'editor
display'seq'file
find
find'string/fast
find'string/full
ml'setup
names'printout
print'2'col'dir
removecomments
sd2copier
sd2'copy&label
seq'to'speed
speed'to'seq
sprite'converter
sprite'editor
text'dum'ctl-p
>--procedures--<
buffer.proc
cat'.proc
joystick.proc
load'obj.proc
loadshape.proc
ml'procs
paddle.proc
plot'char.proc
repeat'key.proc
restore'lbl.proc
saveshape.proc
tod.proc
wait'n'go.proc
>--data-files--<
dir'editor.mem
```

```
utility disk #2
    side 2
>     1525      <
dump'1525
dumpscreen'1525
pretty'printer
>     epson     <
dual'epson'dump
fx-80'cmds.proc
>      nec      <
dump'nec8023a
nec'ml'dump
ml'dump.obj
nec'comal'dump
bigdump'nec.src
dir'print'nec
>   cbm 8023    <
8023p'options
>    okidata    <
oki92'hi
oki92'screen'io
oki92.dump.obj
>    gemini     <
gem10x'lister
print'calendar
bit'map'print.l
>      imp      <
imp'dump
>  prowriter    <
dump'prowriter
>     1520      <
1520/0.14demo1
1520/0.14demo2
1520/0.14demo3
1520/0.14demo4
1520/0.14demo5
1520/0.14demo6
1520/0.14demo7
1520/0.14demo8
1520/0.14demo9
1520/0.14demo10
1520/0.14demo11
1520/0.14demo12
1520/0.14demo13
1520/0.14demo14
1520'driver.proc
```

COMAL

COMAL TODAY DISK #9
2.0 SIDE SEPT '85

1525 ML Screen Dump from Comal

by Mike Lawrence

The following two procedures provide an 8 bit FAST screen dump to the Commodore 1525 or MPS-801 printer. These programs can be added to any of your programs needing a graphics screen dump.

```
proc dump'1525 closed
 oldzone:=zone
 zone 0
 open file 255,"",unit 4,6,write
 print file 255: chr$(28),
 close file 255
 if peek(49171)<>165 then mlpoker
 open file 255,"",unit 4,0,write
 print file 255: chr$(13)+chr$(10),
 print file 255: chr$(8),
 sys 49171
 open file 255,"",unit 4,0,write
 print file 255: chr$(15),
 close file 255
 open file 255,"",unit 4,6,write
 print file 255: chr$(30),
 close file 255
 zone oldzone
endproc dump'1525
//
proc mlpoker closed
 s:=0
 for i:=49152 to 49444 do
  read dat
  poke i,dat
  s:=s+dat
 endfor i
 if s<>30314 then
  print "error in data lines"
  end
 endif
 //
 data 0,0,0,0,224,0,0,0,0,1,2,4,8
 data 16,32,64,128,0,0,165,2
 data 72,165,3,72,169
 data 0,141,6,192,162,255,32
 data 201,255,169,0,141,1
 data 192,141,2,192,169,0
 data 141,8,192,141,7,192,173
 data 6,192,24,109,7,192
 data 141,0,192,32,154,192,240
 data 13,174,7,192,189,9
 data 192,24,109,8,192,141,8
 data 192,238,7,192,173,7
 data 192,201,7,208,218,173,8
 data 192,24,105,128,32,210
 data 255,173,1,192,24,105,1
 data 144,3,238,2,192,141,1
 data 192,201,64,208,183,173
 data 2,192,201,1,208,176,169
 data 13,32,210,255,173,6
 data 192,24,105,7,141,6,192
 data 201,203,240,3,76,35
 data 192,32,231,255,104,133
 data 3,104,133,2,96,169,199
 data 205,0,192,176,3,169,0
 data 96,173,3,192,133,2
 data 173,4,192,133,3,173,0
 data 192,74,74,74,170,240
 data 17,165,2,24,105,64
 data 144,2,230,3,133,2,230
 data 3,202,76,181,192,173
 data 2,192,74,173,1,192,106
 data 74,74,170,240,15,165
 data 2,24,105,8,144,2,230
 data 3,133,2,202,76,211
 data 192,173,0,192,41,7,24
 data 101,2,144,2,230,3,133
 data 2,173,1,192,41,7
 data 141,5,192,169,7,56
 data 237,5,192,170,189,9,192
 data 32,11,193,45,18,192,96
 data 72,165,1,141,17,192
 data 120,73,7,133,1,162,0
 data 161,2,141,18,192,173
 data 17,192,133,1,88,104,96
endproc mlpoker ■
```

Price Protection Plan

Every COMAL TODAY subscriber is protected. If you buy any COMAL item from us - and the price goes down within 1 month - you automatically are entitled to a credit for the difference. Now you can buy with confidence. We wonder how many other computer companies will be willing to follow our lead with this policy. Look at the CREDITS AVAILABLE listing below if you bought any COMAL items after July 1, 1985 at the higher price and the price dropped within a month. Note: special prices offered at a show, conference, or similar event do not count as a price drop.

YOU ARE THE DEALER

All COMAL TODAY subscribers now automatically get a discount on nearly everything they buy from us. The price on most items allows for a dealer markup. Since most COMAL items are not sold through dealers, we decided to give our subscribers a dealer price. We will continue our special quantity prices to schools and groups as well.

THE NEW PRICE STRUCTURE

Keep it simple. What good is a discount if it takes you an hour to calculate? This is how our SUBSCRIBER DISCOUNT works:

$1 off each CHEATSHEET keyboard overlay
$1 off each back issue of COMAL TODAY
$2 off every book (now includes all books)
plus only $4.95 for its matching disk
$4 off every Captain COMAL book/disk set
$4 off spiral bound COMAL TODAY issues 1-4
$4 off COMALite shirts
$5 off each COMAL 2.0 Deluxe Cartridge Pak
All other disks are $9.75 each
$10 off each McPen Lightpen
$15 off the set of 4 Cartridge Demo Disks
$45 off IBM PC COMAL package

CREDITS AVAILABLE - PROTECTION PLAN

To claim your credit you must return your original invoice showing your purchase at the higher price. Circle the items and clearly mark the credit amount. The credit can be applied to any future order. Possible credits at this time are listed below:

CREDIT	ITEM	OLD	NEW
$2.00	CAPT COMAL GET ORGNZD	12.95	10.95
$2.00	LIBRARY FUNC & PROC	12.95	10.95
$2.00	GRAPHICS PRIMER	12.95	10.95
$2.00	COMAL 2.0 PACKAGES	12.95	10.95
$2.00	COMAL FROM A TO Z	6.95	4.95
$2.00	COMAL WORKBOOK	6.95	4.95
$5.00	CART GRAPHICS & SOUND	9.95	4.95
$3.00	COMAL HANDBOOK	18.95	16.95
$5.00	BEGINNING COMAL	22.95	17.95
$3.00	FOUNDATIONS W COMAL	19.95	17.95
$1.00	Cheatsheet	3.95	2.95
10.00	McPen Light Pen	49.95	39.95
20.00	Disk Subscription	49.95	29.95 ■

◆◆◆

Duane Baade provides a fix to the program called MORGAGE, on TODAY DISK #8:

I found that three lines of code had been left out of the program on the disk. The missing lines should be placed between lines 150 and 160. These are the missing lines of code:

```
IF interest>1 THEN
  interest:=interest/100
ENDIF
```

I suggest modifying line 150 to allow you to enter an interest rate of, for example, 12.875. You then can use a current percentage rates without having to round it off. Make line 150:

```
INPUT AT 6,2,6: "interest rate per year:":
interest // wrap line ■
```

Subscriber's Specials

Here are this issues subscriber only specials. If you subscribe now, you can take advantage of these specials at the same time. If you are a current subscriber you MUST include your subscriber number with your order, or we will not honor the special prices. Your subscriber number is printed on the newsletter mailing label. New subscribers get these prices automatically and a number will be assigned with the order.

EVERYTHING

As a subscriber you now get a special price on nearly everything! The specials continue in addition to the regular discount.

DISK SUBSCRIPTION ONLY $29.95

We hate to have our readers type in the programs in the newsletter. Right from our first issue of COMAL TODAY a matching disk has been available. Now it is affordable as well. We could not find a lower priced disk subscription offer anywhere. And our disks are full of programs on BOTH sides. That is better than half price.

FREE DISK OFFER

Buy User Group Disk #10 for $9.75 and get User Group Disk #11 free.

Buy TODAY DISK #2 for $9.75 and get TODAY DISK #1 free.

Join Us Today

NO GROUP OFFERS SO MUCH

UNITED STATES COMMODORE USERS GROUP

COMMODORE 64 & VIC-20 USERS

Join the largest VIC-20 / COMMODORE 64 users group in the United States.

MEMBERS RECEIVE: • Subscription to "Command Performance"
- Access to hundreds of VIC-20 and C64 public domain programs
- Technical assistance • Informative reviews • Contests
- Consumer assistance bureau • Product purchasing assistance

35+ UTILITY PROGRAMS ON DISK

Individual and chapter support

12 month membership:
$25.00 - U.S.A.
U.S. $30.00 - Foreign surface
U.S. $40.00 - Foreign air mail

#1

UNITED STATES COMMODORE USERS GROUP
P.O. BOX 2310, Roseburg, OR 97470 USA

(503) 673-3321

Order Form

Name:______________________________ SUBSCRIBER NUMBER:______ October 1985
(required for reduced prices)
Street:______________________________ Pay by check/MoneyOrder in US Dollars
Canada Postal US Dollar Money Order is OK
City/St/Zip:______________________________ VISA / MasterCard print card#/exp date:

VISA/MC #:______________________________exp date:______Signature:__________________

Qnty Price List/Subscriber price - Item Description (all disks Commodore 1541 format) **Prices subject to change**

SYSTEMS: (two disks may be supplied as a double sided disk)
[ ] _____ $89.95/**$84.95** Deluxe Cartridge Pak (2 books/1 disk/1 cartridge)-(shipping add $4)
[ ] _____ $14.95/**$10.95** COMAL Quick 0.14 (fastloaded) with Utility Disk #2 & book (shipping add $2)
[ ] _____ $11.00/**$11.00** C64 COMAL 0.14 **$11 Special** AutoRun/Tutorial disk, COMAL From A To Z
[ ] _____ $295/**$250** IBM Denmark PC COMAL (Danish manual/COMAL Handbook)-(shipping add $5)

SUBSCRIPTIONS:
[ ] _____ **COMAL TODAY newsletter-> How many issues? ____ Start with: #9 / #10 <-Circle one**
($14.95 first 6; $2 each added issue; >>> Canada add $1 per issue; >>> overseas add $5 per issue)
[ ] _____ $14.95/**$10.95 First 4 issues COMAL TODAY spiral bound (shipping add $2)**
[ ] _____ $3.95/**$2.95 COMAL Today backissue: circle issues wanted-> 5 6 7 8 -(shipping add $1 each)**
[ ] _____ **TODAY DISK subscription >>>>>>>>> How many disks? ______ Start with disk#_____**
($49.95/$29.95 for first 6 disks - $5 each added disk)-(no extra shipping charge)

BOOKS: (optional matching disks are $14.95/**$4.95**)-(**>>>>Canada shipping add $1 more per book<<<<**)
[ ] _____ $18.95/**$16.95** COMAL Handbook - optional disk [___] (shipping add $3)
[ ] _____ $15.95/**$13.95 Starting With COMAL (matching disk not available yet)-(shipping add $3)**
[ ] _____ $19.95/**$17.95** Foundations With COMAL - optional disk [___] (shipping add $3)
[ ] _____ $28.95/**$26.95** Structured Programming With COMAL - optional disk [___] (shipping add $3)
[ ] _____ $20.95/**$18.95** Beginning COMAL - optional disk [___] (shipping add $3)
[ ] _____ $17.95/**$15.95** C64 Graphics With COMAL 0.14 (disk not available yet)-(shipping add $3)
[ ] _____ $6.95/**$4.95** COMAL From A To Z (part of $11 special)-(shipping add $2)
[ ] _____ $14.95/**$10.95** Captain COMAL Gets Organized (includes disk)-(shipping add $2)
[ ] _____ $6.95/**$4.95** COMAL Workbook (perfect companion to Tutorial Disk)-(shipping add $2)
[ ] _____ $14.95/**$10.95** COMAL Library of Functions & Procedures (includes disk)-(shipping add $2)
[ ] _____ $14.95/**$10.95** Graphics Primer (includes disk) (for COMAL 0.14)-(shipping add $2)
[ ] _____ $6.95/**$4.95** Cartridge Graphics & Sound (part of Deluxe Cartridge Pak)-(shipping add $2)
[ ] _____ $14.95/**$10.95** COMAL 2.0 Packages (disk includes C64SYMB & Monitor)-(shipping add $2)
[ ] _____ $20/**$15** Cartridge Tutorial Binder (with disk)(part of Deluxe Cartridge Pak)(shipping add $3)

DISKS:
[ ] _____ $14.95/**$9.75 Font Disk for COMAL 2.0 (30 different fonts plus support programs)**
[ ] _____ $94.05/**$89.95** 19 Disk Set (about 1000 programs for COMAL 0.14)-(shipping add $3)
[ ] _____ $14.95/**$9.75** Best of COMAL 0.14 (includes Disk Data Base)
[ ] _____ $7/**$7** Auto RUN Demo and Tutorial Disk (part of $11 Special)(perfect with COMAL Workbook)
[ ] _____ $14.95/**$9.75** Bricks Tutorials (2 sided BEGINNERS disk!)
[ ] _____ $14.95/**$9.75** Modem Disk (for COMAL 0.14 & 2.0).
[ ] _____ $14.95/**$9.75** Utility Disk #1 for COMAL 0.14
[ ] _____ $14.95/**$9.75** TODAY Disk-**Circle disks wanted: 1 2 3 4 5 6 7 8 9 (or see subscription above)**
[ ] _____ $10/**$9.75** User Group Disks-**Circle disks wanted: 1 2 3 4 5 6 7 8 9 10 11**
[ ] _____ $14.95/**$9.75** Cartridge Demo Disks-**Circle disks wanted: 1 2 3 4 (or see next line)**
[ ] _____ $29.95/**$14.95 Set of all Cartridge Demo Disks (#1 #2 #3 #4)**

OTHER:
[ ] _____ OTHER: ____
[ ] _____ $3.95/**$2.95** Keyboard Overlay for C64 COMAL 0.14 - Cheatsheet (shipping add $1)
[ ] _____ $49.95/**$39.95** McPen Light Pen (with COMAL 2.0 Demo Disk)-(shipping add $3)
[ ] _____ $9.95/**$5.95** Royal Blue COMALite Shirt-circle **ADULT size: S / M / L / XL)-(shipping add $2)**
[ ] _____ **$170.35 School COMAL Package - 12 different books and 12 different disks (shipping add $7.20)**
=====
Total_____ + _____ Shipping (minimum $2 per order) = **Total Paid US$**________ (WI add 5% sales tax)
Mail To: COMAL Users Group USA, 6041 Monona Drive, Madison, WI 53716 or call 608-222-4432

THE SHADOW KNOWS

99.9%

EFFECTIVE

Shadow is a new and revolutionary way to duplicate even your most protected software. It encompases all the latest advances in software, as well as a highly sophisticated piece of hardware. This is absolutely **the best** utility available today. "It will even copy the other copy programs." Because of Shadow's unique abilities, we feel DOS protection is a thing of the past.

By the time you place your order we expect the Shadow to copy 100% — that's right, 100% — of **all** software available for the C-64.

Order by phone 24 hrs./7 days or send cashier's check/money order payable to Megasoft. Visa, MasterCard include card # and exp. date. Add $3.50 shipping/handling for continental U.S., $5.50 for UPS air. CODs add $7.50, Canada add $10.00 Other foreign orders add $15.00 and remit certified U.S. funds only. Distributors invited and supported.

MegaSoft LTD.

INTRODUCTORY OFFER **$89.95**

P.O. Box 1080 Battle Ground, Washington 98604
Phone (206) 687-5116 • BBS 687-5205 After-Hours Computer-to-Computer Ordering